

---

# **temci Documentation**

***Release 0.8.5***

**Johannes Bechberger**

**Jul 07, 2022**



# CONTENTS

<b>1</b>	<b>Why should you use temci?</b>	<b>3</b>
<b>2</b>	<b>Usage</b>	<b>5</b>
<b>3</b>	<b>Installation</b>	<b>7</b>
3.1	Auto completion . . . . .	7
<b>4</b>	<b>Using temci to set up a benchmarking environment</b>	<b>9</b>
<b>5</b>	<b>Why is temci called temci?</b>	<b>11</b>
<b>6</b>	<b>Contributing</b>	<b>13</b>
<b>7</b>	<b>Contents of this documentation</b>	<b>15</b>
7.1	Installation . . . . .	15
7.2	temci build . . . . .	17
7.3	temci exec . . . . .	19
7.4	temci shell . . . . .	38
7.5	temci report . . . . .	38
7.6	temci init . . . . .	45
7.7	temci format . . . . .	45
7.8	OS Support . . . . .	47
7.9	Extending temci . . . . .	48
7.10	Contributing . . . . .	51
7.11	Changelog . . . . .	52
7.12	License . . . . .	53
7.13	API Documentation . . . . .	62
7.14	Resources . . . . .	290
	<b>Python Module Index</b>	<b>291</b>
	<b>Index</b>	<b>293</b>



An advanced benchmarking tool written in Python 3 that supports [setting up an environment for benchmarking](#) and the generation of [visually appealing reports](#).

It runs on Linux systems and (rudimentarily) on macOS.



## WHY SHOULD YOU USE TEMCI?

temci allows you to easily measure the execution time (and other things) of programs and compare them against each other resulting in a pretty HTML5 based report. Furthermore it can set up the environment to ensure benchmarking results with a low variance. The latter feature can be used without using temci for benchmarking by using [temci short shell](#).





## USAGE

The main commands of `temci` are `temci exec` and `temci report`.

Suppose you want to see whether grepping for the strings that consist of `a` and `b` in the current folder is slower than for strings that consist only of `a`.

First we have to install `temci` (using [Nix](#), see below for more instructions):

```
nix-env -f https://github.com/parttimenerd/temci/archive/master.tar.gz -i
```

After this, we can benchmark both commands with `temci`:

```
# benchmark both commands 20 times
temci short exec "grep '[ab]*' -R ." "grep 'a*' -R ." --runs 10

# append --watch to get report (in which you can move with the arrow keys and scroll)
# after every benchmark completed (use --watch_every to decrease interval)
temci short exec "grep '[ab]*' -R ." "grep 'a*' -R ." --runs 10 --watch

# if you want to improve the stability your benchmarks, run them with root privileges
# the benchmarked programs are run with your current privileges
temci short exec "grep '[ab]*' -R ." "grep 'a*' -R ." --runs 10 --sudo --preset usable
```

This results in a `run_output.yaml` file that should look like:

```
- attributes: {description: 'grep '[ab]*' -R .'}
  data:
    etime: [0.03, 0.02, 0.02, 0.03, 0.03, 0.03, 0.02, 0.03, 0.03, 0.02]
    ... # other properties
- attributes: {description: 'grep 'a*' -R .'}
  data:
    etime: [0.02, 0.03, 0.02, 0.03, 0.03, 0.02, 0.03, 0.03, 0.02, 0.02]
    ... # other properties
- property_descriptions: {etime: elapsed real (wall clock) time, ... }
```

For more information on the support measurement tools (like `perf stat` and `rusage`), the supported [plugins](#) for setting up the environment and more, see `temci exec`.

We can now create a report from these benchmarking results using `temci report`. We use the option `--properties` to include only the elapsed time in the report to keep the report simple:

```
> temci report run_output.yaml --properties etime
Report for single runs
grep '[ab]*' -R .    ( 10 single benchmarks)
```

(continues on next page)

(continued from previous page)

```
etime mean =      2(6).(000)m, deviation = 18.84223%  
grep 'a*' -R .      ( 10 single benchmarks)  
etime mean =      2(5).(000)m, deviation = 20.000000%  
  
Equal program blocks  
grep '[ab]*' -R .    grep 'a*' -R .  
etime confidence =      67%, speed up =      3.85%
```

We see that there is no significant difference between the two commands.

There are multiple reporters besides the default [console reporter](#). Another reporter is the [html2 reporter](#) that produces an HTML report, use it by adding the `--reporter html2` option:

## INSTALLATION

The simplest way is to use the [Nix package manager](#), after installing Nix, run:

```
nix-env -f https://github.com/parttimenerd/temci/archive/master.tar.gz -i
```

Using pip requiring at least Python 3.6:

```
sudo pip3 install temci
```

For more information see the [Installation](#) page.

### 3.1 Auto completion

Temci can generate auto completion files for bash and zsh. Add the following line to your *.bashrc* or *.zshrc*:

```
. `temci_completion $0`
```



## **USING TEMCI TO SET UP A BENCHMARKING ENVIRONMENT**

Use the `temci short shell COMMAND` to run a command (`sh` by default) in a shell that is inside the benchmarking environment. Most options of `temci short exec` are supported. For more information, see [temci shell](#).



## **WHY IS TEMCI CALLED TEMCI?**

The problem in naming programs is that most good program names are already taken. A good program or project name has (in my opinion) the following properties:

- it shouldn't be used on the relevant platforms (in this case: github and pypi)
- it should be short (no one wants to type long program names)
- it should be pronounceable
- it should have at least something to do with the program

temci is such a name. It's lojban for time (i.e. the time duration between two moments or events).





## CONTRIBUTING

Bug reports and code contributions are highly appreciated.  
For more information, see the [Contributing](#) page.



## CONTENTS OF THIS DOCUMENTATION

### 7.1 Installation

This page covers installing and updating temci.

#### 7.1.1 System Requirements

- Linux or macOS (see [Supported Operating Systems](#))
- Processor with an x86 or AMD64 architecture (although most features should work on ARM too)

#### 7.1.2 Using Nix

The simplest way is to use the [Nix package manager](#). After installing Nix, run:

```
nix-env -f https://github.com/parttimenerd/temci/archive/master.tar.gz -i
```

This method has the advantage that Nix downloads a suitable python3 interpreter and all packages like matplotlib that could otherwise cause problems. The Nix installation also runs all the test cases, to ensure that temci works properly on your system.

To install temci from source, run:

```
git clone https://github.com/parttimenerd/temci
cd temci
nix-env -i -f .
```

`nix-env -i -f .` can also be used to update your installation after updating the git repository. For a more convenient development environment, see also [Temporary Python environment with nix-shell](#).

#### 7.1.3 Using pip3

There is also the traditional way of using pip, requiring at least Python 3.6.

temci depends on the existence of some packages that cannot be installed properly using pip and have to be installed manually:

```
# on debian/ubuntu/...
time python3-pandas python3-cffi python3-cairo python3-cairocffi python3-matplotlib_
↪python3-numpy python3-scipy linux-tools-`uname -r`
# on fedora
time python3-pandas python3-cffi python3-cairo python3-cairocffi python3-matplotlib_
↪python3-numpy python3-scipy perf
# on OS X (using homebrew)
gnu-time
```

The Linux packages can be installed by calling the `install_packages.sh` script.

After installing these packages, temci can be installed by calling:

To use temci plugins that need super user privileges, e.g. `cpu sets`, install temci globally.

If there are problems with `click` (if you get an exception like `ImportError: cannot import name 'ParameterSource'`), try installing it directly from github:

```
pip3 install https://github.com/pallets/click/archive/
↪f537a208591088499b388b06b2aca4efd5445119.zip
```

To install temci from source, run:

```
git clone https://github.com/parttimenerd/temci
cd temci
pip3 install -e
```

## Post Installation

Run the following command after the installation to compile some binaries needed for the `rusage` runner or the disabling of caches:

```
temci setup
```

This requires `gcc` and `make` to be installed.

### 7.1.4 Optional Requirements

Requirements that aren't normally needed are the following:

- `kernel-devel` packages (for compiling the kernel module to disable caches)
- `pdflatex` (for pdf report generation)

Temci runs perfectly fine without them if you are not using the mentioned features.

## Auto Completion

Temci can generate auto completion files for bash and zsh. Add the following line to your `.bashrc` or `.zshrc`:

```
. `temci_completion $0`
```

## 7.2 temci build

Build programs before the actual benchmarks, can checkout specific git commits. This has the advantage of being able to configure the build for all benchmarked programs and to build these programs at once. This build config also contains the run config for each program. `temci build` compiles a run config and stores it into a file that can be directly used with `temci exec` (or other configured run drivers).

**For most cases using the builder capabilities of ``temci exec <temci_exec.html#building>`` should be enough. This also has the advantage of using a single command for all benchmarked programs, whether they need to build or not.**

### 7.2.1 Usage

```
Usage: temci build [OPTIONS] BUILD_FILE
```

Options:

<code>--tmp_dir TEXT</code>	Used temporary directory [default: /tmp/temci]
<code>--threads INTEGER</code>	Number of threads that build simultaneously [default: 1]
<code>--sudo</code>	Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the <code>command</code> line. [default: False]
<code>--sudo / --no-sudo</code>	Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the <code>command</code> line. [default: False]
<code>--settings TEXT</code>	Additional settings file [default: ]
<code>--out TEXT</code>	Resulting run config file [default: run.exec.yaml]
<code>--log_level [debug info warn error quiet]</code>	Logging level [default: info]
<code>--in TEXT</code>	Input file with the program blocks to build [default: build.yaml]
<code>--help</code>	Show this message and exit.

`in`, `out` and `threads` can also be set in the settings in the build block.

Be aware the parallel building or building multiple version of a program is still fragile.

## Example

A build config (build\_config.yaml) file for tool called `test` might look like this:

```
- attributes:
  description: 'test'
  run_config:
    run_cmd: 'sh test'
  build_config:
    build_cmd: 'echo "sleep 1" > test'
```

To build it, run `temci build build_config.yaml`, resulting in the following `run_config.yaml`:

```
- attributes:
  description: test
  tags: []
  run_config:
    cwd: ['.']
    run_cmd: sh test
```

## With temci exec

`temci exec` supports calling the builder directly, omitting the call to `temci build`. Just call `temci build` if you want to separate building and benchmarking.

## 7.2.2 File Format

`temci build` accepts a file that consists of a YAML list of the entries in the following format:

```
# Optional attributes that describe the block
attributes:
  description:          Optional(Str())

# Tags of this block
tags:                  ListOrTuple(Str())

# Build configuration for this program block
build_config:
  # Base directory that contains everything to build an run the program
  base_dir:             Either(DirName()|non existent)
                        default: .

  # Used version control system branch (default is the current branch)
  branch:               Either(Str()|non existent)

  # Command to build this program block, might randomize it
  cmd:                  Str()

  # Number of times to build this program
  number:               Either(Int()|non existent)
                        default: 1
```

(continues on next page)

(continued from previous page)

```

# Used version control system revision of the program (-1 is the current revision)
revision:      Either(Either(Str()|Int())|non existent)
               default: -1

# Working directory in which the build command is run
working_dir:   Either(DirName()|non existent)
               default: .

# Run configuration for this program block
run_config:    Dict(, keys=Any, values=Any, default = {})

```

## 7.2.3 VCS Support

Currently only Git is supported, but adding support for other version control systems is simple. The code for the VCS drivers is in the `temci.utils.vcs` module.

## 7.3 temci exec

This page explains `temci exec` and `temci short exec` that allow you to run the actual benchmarks.

The basic concept is that there are

### run drivers

that support a specific benchmarking concept (like benchmarking whole programs that can be executed in the shell), these run drivers use

### runners

for the actual benchmarking and

### plugins

to setup up the benchmarking environment

Currently only one run driver is implemented, the `exec` run driver that supports benchmarking programs executed in a shell.

The benchmarking process produces a YAML file with the benchmarking results.

There are multiple features that require root privileges. To use these features, call `temci` with the `--sudo` option. It will run only `temci` in super user mode, but not the benchmarked programs themselves. Notable features that require these rights are `cpu sets` (for separating the benchmarked programs from the rest of the system), disabling hyperthreading and setting the CPU governor.

### 7.3.1 temci short exec

Supports basic benchmarks, without creating a configuration file. It supports the same command line options as `temci exec`:

```

Usage: temci short exec [OPTIONS] COMMANDS

-wd, --without_description COMMAND:
    Benchmark the command and use
    itself as its description. Appends

```

(continues on next page)

(continued from previous page)

```

'$ARGUMENT' to the command if the string
isn't present. Use the '--argument' option
to set the value that this string is
replaced with.
-d, --with_description DESCRIPTION COMMAND:
Benchmark the command
and set its description attribute. Appends
'$ARGUMENT' to the command if the string
isn't present. Use the '--argument' option
to set the value that this string is
replaced with.
...
(options of temci exec)

```

## 7.3.2 Usage

Basic benchmarking of two programs using the *time*:

```

# compare the run times of two programs, running them each 20 times
> temci short exec "sleep 0.1" "sleep 0.2" --runs 20
Benchmark 20 times [#####] 100%
Report for single runs
sleep 0.1 ( 20 single benchmarks)
  avg_mem_usage mean = 0.000, deviation = 0.0
  avg_res_set mean = 0.000, deviation = 0.0
  etime mean = 100.00000m, deviation = 0.00000%
  max_res_set mean = 2.1800k, deviation = 3.86455%
  stime mean = 0.000, deviation = 0.0
  utime mean = 0.000, deviation = 0.0

sleep 0.2 ( 20 single benchmarks)
  avg_mem_usage mean = 0.000, deviation = 0.0
  avg_res_set mean = 0.000, deviation = 0.0
  etime mean = 200.00000m, deviation = 0.00000%
  max_res_set mean = 2.1968k, deviation = 3.82530%
  stime mean = 0.000, deviation = 0.0
  utime mean = 0.000, deviation = 0.0

```

Use *--watch* to display the report continuously, every *--watch\_every* (default is 1) seconds.

The produced *run\_output.yaml* file is:

```

- attributes: {__description: sleep 0.1, description: sleep 0.1}
  data:
    max_res_set: [2148.0, 2288.0, 2152.0, 2120.0, 2340.0, 2076.0, 2152.0, 2280.0,
      2080.0, 2276.0, 2124.0, 2120.0, 2136.0, 2156.0, 2272.0, 2280.0, 2284.0, 2060.0,
      2120.0, 2136.0]
    ...
- attributes: {__description: sleep 0.2, description: sleep 0.2}
  data:

```

(continues on next page)



(continued from previous page)

```

max_res_set: [2080.0, 2284.0, 2140.0, 2124.0, 2156.0, 2096.0, 2096.0, 2284.0,
              2288.0, 2120.0, 2284.0, 2280.0, 2284.0, 2272.0, 2272.0, 2152.0, 2152.0, 2328.0,
              2152.0, 2092.0]
...
- property_descriptions: {avg_mem_usage: average total mem usage (in K), ...}

```

More information on the format of the result file can be found in the documentation for [temci report](#).

This documentation focuses on `temci exec` and its input file and options.

## Presets

temci has the `--preset` option (and the setting `run/exec_misc/preset`) that enables a specific combination of plugins:

### none

no plugins are enabled, the default for non super user benchmarking

### all

Use all available plugins and render the system partially unusable by stopping all unnecessary processes etc., enables: `cpu_governor`, `disable_swap`, `sync`, `stop_start`, `other_nice`, `nice`, `disable_aslr`, `disable_ht`, `disable_intel_turbo`, `cpuset`

### usable

Use all plugins that do not affect other processes (besides restricting them to a single CPU), covers essentially the [benchmarking tips of the LLVM project](#) and enables: `cpu_governor`, `disable_swap`, `sync`, `nice`, `disable_aslr`, `disable_ht`, `cpuset`, `disable_intel_turbo`. This preset is used by default in super user mode (with `--sudo` option).

Important: These presets don't include the `sleep` plugin. Enable it via `--sleep` if needed.

An overview over all available plugins is given at [Overview](#).

## Runners

The runners are selected on the command line using the `--runner` option and the configuration file via `run/exec_misc/runner`. They obtain the actual measurements and are configured in the run configuration. Configuring them in `temci short exec` is currently not possible.

### time

Uses the GNU time utility to measure basic properties. This is the default runner. It is relatively imprecise but gives good ball park numbers for the performance.

### rusage

Uses the `getrusage` method and a small wrapper written in C (be sure to call `temci setup` if you install temci via pip, to build the wrapper).

### perf\_stat

Uses `perf stat` for measurements, might require root privileges. Allows measuring a wide range of properties

### output

This runner obtains the measurements by parsing the output of the benchmarked program and interpreting it as a YAML mapping of properties to measurements (`property: NUMBER` lines). It can be used in combination with the `time` and `perf_stat` runners (using the `--parse_output` option or setting `parse_output` to true in the run block config).

## Building

`temci exec` supports to build the programs that are then benchmarked. It supports the same format and the same options as `temci build`.

In the most basic case (and the case that is thoroughly tested), just supply a build command:

```
- attributes: ...
  run_config: ...
  build_config:
    cmd: make # a sample build command
```

Executing the file with `temci exec` runs all available build commands.

This can be configured using the following options (set in the run settings block):

### **no\_build**

Do not build, default is false

### **only\_build:**

Only build the build configs for all blocks, default is false

### **abort\_after\_build\_error**

default true

If building a block fails and `abort_after_build_error` is not true (e.g. `--no-abort_after_build_error` is passed), then `temci` produces an `EXEC_INPUT_FILE.erroneous.yaml` that contains the configurations of all failing blocks. This file can be used to execute the missing blocks again after the error is fixed. Use the `--append` option to append the benchmarks to the preexisting benchmark result file.

## Error Codes

0	no error
1	at least one benchmarked program failed
255	temci itself failed

### 7.3.3 File format

The input file for `temci exec` consists of a list of entries per run program block:

```
-
  # Optional build config to integrate the build step into the run step
  build_config:      Either(Dict(, keys=Any, values=Any, default = {})|non existent)

  # Optional attributes that describe the block
  attributes:
    description:      Optional(Str())

    # Tags of this block
    tags:             ListOrTuple(Str())

  run_config:
    # Command to benchmark, adds to run_cmd
    cmd:              Str()
```

(continues on next page)

(continued from previous page)

```

# Configuration per plugin
time:
    ...
    ...

# Command to append before the commands to benchmark
cmd_prefix:      List(Str())

# Execution directories for each command
cwd:             Either(List(Str())|Str())
                default: .

# Disable the address space layout randomization
disable_aslr:    Bool()

# Override all other max runspecifications if > -1
max_runs:        Int()
                default: -1

# Override all other min runspecifications if > -1
min_runs:        Int()
                default: -1

# Parse the program output as a YAML dictionary of that gives for a specific
↪property a
# measurement. Not all runners support it.
parse_output:    Bool()
                default: False

# Used revision (or revision number). -1 is the current revision, checks out the
↪revision
revision:        Either(Int())|Str()
                default: -1

# Commands to benchmark
run_cmd:         Either(List(Str())|Str())

# Used runner
runner:          ExactEither()
                default: time

# Override min run and max runspecifications if > -1
runs:            Int()
                default: -1

# Environment variables
env:             Dict(, keys=Str(), values=Any, default = {})

# Configuration for the output and return code validator
validator:
    # Program error output without ignoring line breaks and spaces at the beginning

```

(continues on next page)

(continued from previous page)

```

# and the end
expected_err_output:          Optional(Str())

# Strings that should be present in the program error output
expected_err_output_contains:      Either(List(Str())|Str())

# Program output without ignoring line breaks and spaces at the beginning
# and the end
expected_output:             Optional(Str())

# Strings that should be present in the program output
expected_output_contains:         Either(List(Str())|Str())

# Allowed return code(s)
expected_return_code:           Either(List(Int())|Int())

# Strings that shouldn't be present in the program output
unexpected_err_output_contains:    Either(List(Str())|Str())

# Strings that shouldn't be present in the program output
unexpected_output_contains:       Either(List(Str())|Str())

```

A basic config file looks like:

```

- run_config:
  run_cmd: sleep 0.1
- run_config:
  run_cmd: sleep 0.2

```

### 7.3.4 Common options

These options are passed in the run settings block (see [Settings API](#) or directly on the command line, flags are of the schema `--SETTING/--no-SETTING`):

```

# Append to the output file instead of overwriting by adding new run data blocks
append:          Bool()

# Disable the hyper threaded cores. Good for cpu bound programs.
disable_hyper_threading:      Bool()

# Discard all run data for the failing program on error
discard_all_data_for_block_on_error:    Bool()

# First n runs that are discarded
discarded_runs:          Int()
  default: 1

# Possible run drivers are 'exec' and 'shell'
driver:          ExactEither('exec'|'shell')
  default: exec

```

(continues on next page)

(continued from previous page)

```

# Input file with the program blocks to benchmark
in:          Str()
            default: input.exec.yaml

# List of included run blocks (all: include all)
# or their tag attribute or their number in the
# file (starting with 0), can be regular expressions
included_blocks: ListOrTuple(Str())
                default: [all]

# Maximum time one run block should take, -1 == no timeout,
# supports normal time span expressions
max_block_time: ValidTimespan()
                default: '-1'

# Maximum number of benchmarking runs
max_runs:      Int()
                default: 100

# Maximum time the whole benchmarking should take
#   -1 == no timeout
# supports normal time spans
# expressions
max_time:      ValidTimespan()
                default: '-1'

# Minimum number of benchmarking runs
min_runs:      Int()
                default: 20

# Output file for the benchmarking results
out:          Str()
            default: run_output.yaml

# Record the caught errors in the run_output file
record_errors_in_file: Bool()
                      default: true

# Number of benchmarking runs that are done together
run_block_size: Int()
                default: 1

# if != -1 sets max and min runs to its value
runs:          Int()
            default: -1

# Order in which the plugins are used, plugins that do not appear in this list are used
# before all others
plugin_order: ListOrTuple(Str())
              default: ["drop_fs_caches", "sync", "sleep", "preheat", "flush_cpu_caches"]

# If not empty, recipient of a mail after the benchmarking finished.

```

(continues on next page)

(continued from previous page)

```

send_mail:          Str()

# Print console report if log_level=info
show_report:      Bool()
                    default: true

# Randomize the order in which the program blocks are benchmarked.
shuffle:          Bool()
                    default: true

# Store the result file after each set of blocks is benchmarked
store_often:      Bool()

# Show the report continuously
watch: false

# Update the screen nth run (less updates are better for benchmarks)
watch_every: 1

cpuset:
    # Use cpuset functionality?
    active:         Bool()

    # Number of cpu cores for the base (remaining part of the) system
    base_core_number: Int(range=range(0, NUMBER OF CPUS))
                    default: 1

    # 0: benchmark sequential
    # > 0: benchmark parallel with n instances
    # -1: determine n automatically (based on the number of cpu cores)
    parallel:       Int()

    # Number of cpu cores per parallel running program.
    sub_core_number: Int(range=range(0, NUMBER OF CPUS))
                    default: 1

    # Place temci in the same cpu set as the rest of the system?
    temci_in_base_set: Bool()
                    default: True

    # Maximum runs per tag (block attribute 'tag'), min('max_runs', 'per_tag') is used
max_runs_per_tag: Dict(, keys=Str(), values=Int(), default = {})

    # Minimum runs per tag (block attribute 'tag'), max('min_runs', 'per_tag') is used
min_runs_per_tag: Dict(, keys=Str(), values=Int(), default = {})

    # Runs per tag (block attribute 'tag'), max('runs', 'per_tag') is used
runs_per_tag:     Dict(, keys=Str(), values=Int(), default = {})

    # Do not build, the building process should not set the working directory
no_build: Bool()
                    default: False

```

(continues on next page)

(continued from previous page)

```
# Only build the build configs for all blocks
only_build: Bool()
            default: False

# Abort after the build error
abort_after_build_error: Bool()
            default: True
```

There also some exec run driver specific options:

```
# Parse the program output as a YAML dictionary of that gives for a specific property a
# measurement. Not all runners support it.
parse_output: Bool()

# Enable other plugins by default
preset: ExactEither('none'|'all'|'usable')
          default: none

# Pick a random command if more than one run command is passed.
random_cmd: Bool()
            default: true

# If not " overrides the runner setting for each program block
runner: ExactEither(''|'perf_stat'|'rusage'|'spec'|'spec.py'|'time'|'output')
```

## Number of runs

The number of runs per block is either fixed by the `runs` settings that apply or is between the applying `min_runs` and `max_runs` setting. In the latter case, the benchmarking of a program block is stopped early as soon as there is some significance in the benchmarking results compared to all other benchmarked programs.

## 7.3.5 Runners

The runners are selected on the command line using the `--runner` option and the configuration file via `run/exec_misc/runner`. They are configured in the run configuration file using the settings block named like the runner in each run block.

### time runner

Uses the GNU `time` tool and is mostly equivalent to the `rusage` runner but more user friendly.

The runner is configured by modifying the `time` property of a run configuration. This configuration has the following structure:

```
# Measured properties that are included in the benchmarking results
properties: ValidTimePropertyList()
            default: [utime, stime, etime, avg_mem_usage, max_res_set, avg_res_set]
```

The measurable properties are:

**utime**

user CPU time used (in seconds)

**stime**

system (kernel) CPU time used (in seconds)

**avg\_unshared\_data**

average unshared data size in K

**etime**

elapsed real (wall clock) time (in seconds)

**major\_page\_faults**

major page faults (required physical I/O)

**file\_system\_inputs**

blocks wrote in the file system

**avg\_mem\_usage**

average total mem usage (in K)

**max\_res\_set**

maximum resident set (not swapped out) size in K

**avg\_res\_set**

average resident set (not swapped out) size in K

**file\_system\_output**

blocks read from the file system

**cpu\_perc**

percent of CPU this job got (total cpu time / elapsed time)

**minor\_page\_faults**

minor page faults (reclaims; no physical I/O involved)

**times\_swapped\_out**

times swapped out

**avg\_shared\_text**

average amount of shared text in K

**page\_size**

page size

**invol\_context\_switches**

involuntary context switches

**vol\_context\_switches**

voluntary context switches

**signals\_delivered**

signals delivered

**avg\_unshared\_stack**

average unshared stack size in K

**socket\_msg\_rec**

socket messages received

**socket\_msg\_sent**

socket messages sent



This runner is implemented in the `TimeExecRunner` class.

Supports the `parse_output` option.

### rusage runner

Uses the `getrusage` method and a small wrapper written in C (be sure to call `temci setup` if you install temci via pip, to build the wrapper).

The runner is configured by modifying the `rusage` property of a run configuration. This configuration has the following structure:

```
# Measured properties that are stored in the benchmarking result
properties:      ValidRusagePropertyList()
                  default: [idrss, inblock, isrss, ixrss,
                             majflt, maxrss, minflt,
                             msgrcv, msgsnd, nivcs, nsignals,
                             nswap, nvcs, oubleck, stime, utime]
```

The measurable properties are:

#### **utime**

user CPU time used

#### **stime**

system CPU time used

#### **maxrss**

maximum resident set size

#### **ixrss**

integral shared memory size

#### **idrss**

integral unshared data size

#### **isrss**

integral unshared stack size

#### **nswap**

swaps

#### **minflt**

page reclaims (soft page faults)

#### **majflt**

page faults (hard page faults)

#### **inblock**

block input operations

#### **oubleck**

block output operations

#### **msgsnd**

IPC messages sent

#### **msgrcv**

IPC messages received

**nsignals**

signals received

**nvcs**

voluntary context switches

**nivcs**

involuntary context switches

This runner is implemented in the [RusageExecRunner](#) class.

**perf\_stat runner**

This runner uses the `perf stat` tool to obtain measurements. It might have to be installed separately (see *Installation* <[installation.html](#)>). `perf stat` allows measuring a myriad of properties but might require root privileges.

The runner is configured by modifying the `perf_stat` property of a run configuration. This configuration has the following structure:

```
# Limit measurements to CPU set, if cpusets are enabled
limit_to_cpuset:      Bool()
                        default: true

# Measured properties. The number of properties that can be measured at once is limited.
properties:         List(Str())
                        default: [wall-clock, cycles, cpu-clock, task-clock,
                                instructions, branch-misses, cache-references]

# If runner=perf_stat make measurements of the program repeated n times. Therefore scale_
↳ the number of
# times a program is benchmarked.
repeat:             Int()
                        default: 1
```

The measureable properties can be obtained by calling `perf list`. Common properties are given above, other notable properties are cache-misses and branch-misses. The wall-clock property is obtained by parsing the non-csv style output of `perf stat` which is fragile.

This runner is implemented in the [PerfStatExecRunner](#) class.

Supports the `parse_output` option.

**output runner**

This runner obtains the measurements by parsing the output of the benchmarked program and interpreting it as a YAML mapping of property to measurement (`property: NUMBER` lines).

It can be used in combination with the *time* and the *perf\_stat* runner, (using the `--parse_output` option), allowing benchmarking a command and parsing its result for additional measurements.

An example output is:

```
time: 10
load_time: 5
```

It also supports lists of values if the lists of all properties have the same number of elements. This can be used return the result of multiple measurements in one call of the benchmarked program:

```
time:      [11.0, 10.01, 8.5]
load_time: [5.0,  6.7,  4.8]
```

This runner is implemented in the `OutputExecRunner` class.

### spec runner

*This runner might not really work and is not really used.*

Runner for SPEC like single benchmarking suites. It works with resulting property files, in which the properties are colon separated from their values.

The runner is configured by modifying the `spec` property of a run configuration. This configuration has the following structure:

```
# Base property path that all other paths are relative to.
base_path:      Str()

# Code that is executed for each matched path.
# The code should evaluate to the actual measured value
# for the path. It can use the function get(sub_path: str = "")
# and the modules pytimeparse, numpy, math, random, datetime and time.
code:           Str()
                default: get()

# SPEC result file
file:           Str()

# Regexp matching the base property path for each measured property
path_regexp:    Str()
                default: .*
```

An example configuration is given in the following:

```
- attributes:
  description: spec
  run_config:
    runner: spec
    spec:
      file: "spec_like_result.yaml"
      base_path: "abc.cde.efg"
      path_regexp: 'bench\d'
      code: 'get(".min") * 60 + get(".sec") + random.random()'
- attributes:
  description: "spec2"
  run_config:
    runner: spec
    spec:
      file: "spec_like_result.yaml"
      base_path: "abc.cde.efg"
      path_regexp: 'bench\d'
      code: 'get(".min") * 60 + get(".sec") + 0.5 * random.random()'
```

This runner is implemented in the `SpecExecRunner` class.

### 7.3.6 Plugins

Plugins setup the benchmarking environment (e.g. set the CPU governor, ...). All their actions are reversible and are reversed if temci aborts or finishes.

The plugins are enabled via the command line option `--NAME`, in the configuration file via `run/exec_plugins/NAME_active` or by adding the name to set of active plugins in `run/exec_plugins/exec_active`. A collection of them can be activated using *Presets*.

All plugins are located in the `temci.run.run_driver_plugin` module.

#### Overview

New plugins can be added easily (see [Extending temci](#)) but there are multiple plugins already available:

*cpu\_governor*

Set the cpu governor

*cpuset*

Uses *CPUsets* to separate the CPUs used for benchmarking from the CPUs that the rest of the system runs on

*disable\_aslr*

Disable address space randomisation

*disable\_cpu\_caches*

Disables the L1 and L2 caches

*disable\_ht*

Disables hyper-threading

*disable\_intel\_turbo*

Disables the turbo mode on Intel CPUs

*disable\_swap*

Disables swapping data from the RAM into a backing hard drive

*discarded\_runs*

Discard the first runs (sets the `run/discarded_runs` setting)

*drop\_fs\_caches*

Drops file system caches

*env\_randomize*

Adds random environment variables to mitigate some cache alignment effects

*flush\_cpu\_caches*

Flush the CPU caches on x86 CPUs

*nice*

Increases the CPU and IO scheduling priorities of the benchmarked program

*other\_nice*

Decreases the CPU scheduling priority of all other programs

*preheat*

Preheats the system with a CPU bound task

*sleep*

Keeps the system idle for some time before the actual benchmarking

*stop\_start*

Stops almost all other processes (as far as possible)

***sync***

Synchronizes cached writes of the file system to a persistent storage

The order in which the plugins are used (and called) is defined by the `run/plugin_order`, see `common-options`.

**cpu\_governor**

Sets the CPU governor of all CPU cores.

The governor can be configured by either using the `--cpu_governor_governor GOVERNOR` option or by setting `run/exec_plugins/cpu_governor_misc/governor`.

The default governor is `performance` which is recommended for benchmarks.

The available governors can be obtained by calling

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
```

Requires root privileges.

**cpuset**

Uses cpusets to separate the CPUs used for benchmarking from the CPUs that the rest of the system runs on. For more information see *CPUSets*.

Requires root privileges.

**disable\_aslr**

Disables the address space randomisation which might lead to less variance in the benchmarks.

Requires root privileges.

**disable\_cpu\_caches**

Disables the L1 and L2 caches on x86 and x86-64 architectures. It uses a small custom kernel module (be sure to compile it via `temci setup --build_kernel_modules` after install the appropriate `kernel-devel` package, see *Installation*).

*Attention:* It will slow down your system by orders of magnitude, giving you essentially a Pentium I like processor. Only use it for demonstration purposes.

Requires root privileges.

**disable\_ht**

Disables hyper-threading, enabling it is equivalent to using the `disable_hyper_threading` option (see *Common options*).

It disable a number of CPU cores so that only one core per physical CPU core is active, thereby effectively disabling hyper-threading.

Requires root privileges.

### disable\_intel\_turbo

Disables the turbo mode on Intel CPUs. Might reduce the variance of benchmarks, as the CPUs cannot overclock partially.

Requires root privileges.

### disable\_swap

Disables swapping data from the RAM into a backing hard drive. Swapping during benchmarking sessions increases the variance as accessing data on a hard drive is significantly slower than accessing data in RAM.

Requires root privileges.

### discarded\_runs

Discard the first runs (sets the `run/discarded_runs` setting). As a result, the benchmark files should already be in the file system caches.

### drop\_fs\_caches

Drops the page cache, directory entries and inodes before every benchmarking run. This might improve the usability of the produced benchmarks for IO bound programs.

It can be either configured by using the `run/exec_plugins/drop_fs_caches_misc` block in the settings or by using the command line options of the same names prefixed by `--drop_fs_caches_`:

```
# Free dentries and inodes
free_dentries_inodes: true

# Free the page cache
free_pagecache: true
```

Requires root privileges.

### env\_randomize

Adds random environment variables before each benchmarking run. This causes the stack frames of the called program to be aligned differently. Can mitigate effects caused by a specific cache alignment.

It can be either configured by using the `run/exec_plugins/env_randomize_misc` block in the settings or by using the command line options of the same names prefixed by `--env_randomize_`:

```
# Maximum length of each random key
key_max: 4096

# Maximum number of added random environment variables
max: 4

# Minimum number of added random environment variables
min: 4
```

(continues on next page)

(continued from previous page)

```
# Maximum length of each random value
var_max: 4096
```

## flush\_cpu\_caches

Write back and flush Internal caches; initiate writing-back and flushing of external caches (see [WBINVD](#)).

It uses a small custom kernel module (be sure to compile it via `temci setup --build_kernel_modules` after install the appropriate `kernel-devel` package, see [Installation](#)).

## nice

Sets the `nice` and `ionice` values (and therefore the CPU and IO scheduler priorities) of the benchmarked program to a specific value.

It can be either configured by using the `run/exec_plugins/nice_misc` block in the settings or by using the command line options of the same names prefixed by `--nice_`:

```
# Specify the name or number of the scheduling class to use
# 0 for none
# 1 for realtime
# 2 for best-effort
# 3 for idle
io_nice: 1

# Niceness values range from -20 (most favorable to the process)
# to 19 (least favorable to the process).
nice: -15
```

`nice` values lower than -15 seem to cripple Linux systems.

Requires root privileges.

## other\_nice

Sets the `nice` value of processes other than the benchmarked one. Prioritises the benchmarked program over all other processes.

It can be either configured by using the `run/exec_plugins/other_nice_misc` block in the settings or by using the command line options of the same names prefixed by `--other_nice_`:

```
# Processes with lower nice values are ignored.
min_nice: -10

# Niceness values for other processes.
nice: 19
```

Requires root privileges.

## preheat

Preheats the system with a CPU bound task (calculating the inverse of a big random matrix with numpy on all CPU cores).

The length of the preheating can be configured by either using the `--preheat_time SECONDS` option or by setting `run/exec_plugins/preheat_misc/time`.

When the preheating takes place (before each run or at the beginning of the benchmarking) can be configured via `--preheat_when [before_each_run|at_setup]` or by setting `run/exec_plugins/preheat_misc/when` (accepts a list).

## sleep

Keep the system idle for some time before the actual benchmarking.

See [Gernot Heisers Systems Benchmarking Crimes](#):

Make sure that the system is really quiescent when starting an experiment, leave enough time to ensure all previous data is flushed out.

## stop\_start

Stops almost all other processes (as far as possible).

This plugin tries to stop most other processes on the system that aren't really needed. By default most processes that are children (or children's children, ...) of a process whose name ends with "dm" are stopped. This is a simple heuristic to stop all processes that are not vital (i.e. created by some sort of display manager). SSH and X11 are stopped too.

Advantages of this plugin (which is used via the command line flag `--stop_start`):

- No one can start other programs on the system (via ssh or the user interface)
- → fewer processes can interfere with the benchmarking
- Noisy processes like Firefox don't interfere with the benchmarking as they are stopped, this reduces the variance of benchmarks significantly

Disadvantages:

- You can't interact with the system (therefore use the `send_mail` option to get mails after the benchmarking finished)
- Not all processes that could be safely stopped are stopped as this decision is hard to make
- You can't stop the benchmarking as all keyboard interaction is disabled (by stopping X11)
- You might have to wait several minutes to be able to use your system after the benchmarking ended

Stopping a process here means to send a process a SIGSTOP signal and resume it by sending a SIGCONT signal later.

It can be either configured by using the `run/exec_plugins/stop_start_misc` block in the settings or by using the command line options of the same names prefixed by `--stop_start_`:

```
# Each process which name (lower cased) starts with one of the prefixes is not ignored.
# Overrides the decision based on the min_id.
comm_prefixes: [ssh, xorg, bluetoothd]

# Each process which name (lower cased) starts with one of the prefixes is ignored.
# It overrides the decisions based on comm_prefixes and min_id.
```

(continues on next page)



(continued from previous page)

```

comm_prefixes_ignored: [dbus, kworker]

# Just output the to be stopped processes but don't actually stop them?
dry_run: false

# Processes with lower id are ignored.
min_id: 1500

# Processes with lower nice values are ignored.
min_nice: -10

# Suffixes of processes names which are stopped.
subtree_suffixes: [dm, apache]

```

Requires root privileges.

## sync

Synchronizes cached writes of the file system to a persistent storage by calling `sync`.

### 7.3.7 CPUSets

The idea is to separate the benchmarked program from all other programs running on the system.

The usage of cpusets can be configured by using the following settings that are part of `run/cpuset` and can also be set using the options with the same names prefixed with `--cpuset_`:

```

# Use cpuset functionality?
active:          Bool()

# Number of cpu cores for the base (remaining part of the) system
base_core_number: Int(range=range(0, 8))
                  default: 1

# 0: benchmark sequential
# > 0: benchmark parallel with n instances
# -1: determine n automatically, based on the number of CPU cores
parallel:        Int()

# Number of cpu cores per parallel running program.
sub_core_number: Int(range=range(0, 8))
                  default: 1

# Place temci in the same cpu set as the rest of the system?
temci_in_base_set: Bool()
                  default: True

```

This functionality can also be enabling by using the `--cpuset` flag or by enabling the `cpuset` plugin.

## 7.4 temci shell

`temci short shell` opens a shell in a benchmarking environment. It allows to execute your own benchmarking suite in its own cpuset with disabled hyper threading, .... This command has the same options as `temci exec` (regarding presets and plugins).

For example running your own benchmarking suite `bench.sh` in a reasonably setup environment can be done via:

```
temci short shell ./bench.sh
```

The launched shell is interactive:

```
> temci short shell
>> echo 1
1
```

`temci shell` accepts an input file as its argument which has the following structure (see `ShellRunDriver`:

```
# Optional build config to integrate the build step into the run step
build_config:      Either(Dict(, keys=Any, values=Any, default = {})|non existent)

# Optional attributes that describe the block
attributes:
  description:      Optional(Str())

  # Tags of this block
  tags:             ListOrTuple(Str())

run_config:
  # Execution directory
  cwd:              Either(List(Str())|Str())
                  default: .

  # Command to run
  run_cmd:          Str()
                  default: sh

  # Environment variables
  env:              Dict(, keys=Str(), values=Any, default = {})
```

## 7.5 temci report

`temci report` supports the statistical evaluation of benchmarking runs. It processes the output file of `temci exec`. This page gives an overview over the different reporters and the expected format of the input file. The creation of a new reporter is explained in [Extending temci](#).

There are currently four different reporters:

### **console**

Outputs a summary of the benchmarks on the console, the default reporter

### **html2**

Creates a HTML based report with many graphics

**csv**

Outputs a configurable csv table

**codespeed**

Outputs JSON as expected by the `codespeed` tool

## 7.5.1 Usage

Using the `html2` reporter:

## 7.5.2 File format

The input file for `temci report` consists of list of entries per run program block:

```
-
# Optional attributes that describe the block
attributes:
    description:          Optional(Str())

    # Tags of this block
    tags:                ListOrTuple(Str())

data:
    property_1: List(Either(Int()|Float()))
    ...

# the run program aborted with an error
error:
    message: Str()
    return_code': Int()
    output: Str()
    error_output: Str()

# there was an internal error
internal_error:
    message: Str()

# only the error or the internal_error block can be present
# the recorded data is the data recorded till the error occurred

# optional property descriptions
- property_descriptions:
    property_1: long name of property_1
```

### 7.5.3 Common Options

These options are passed in the reporter settings block (see [Settings API](#) or directly on the command line (flags are of the schema `--SETTING/--no-SETTING`):

```
# Exclude all data sets that contain only NaNs.
exclude_invalid:      BoolOrNone()
                        default: true

# Properties that aren't shown in the report.
excluded_properties: ListOrTuple(Str())
                        default: [__ov-time]

# Files that contain the benchmarking results
in:                  Either(Str()|ListOrTuple(Str()))
                        default: run_output.yaml

# List of included run blocks (all: include all), identified by their description
# or tag attribute, can be regular expressions
included_blocks:     ListOrTuple(Str())
                        default: [all]

# Replace the property names in reports with longer more descriptive versions?
long_properties:     BoolOrNone()

# Possible reporter are 'console', 'html2', 'csv' and 'codespeed'
reporter:            ExactEither('console'|'html2'|'csv'|'codespeed')
                        default: console

# Produce xkcd like plots (requires the humor sans font to be installed)
xkcd_like_plots:     BoolOrNone()
```

Furthermore the formatting of numbers can be partially configured using the settings file block described in [temci format](#).

The statistical evaluation and the used properties can be configured via the stats settings block or with the unprefixed options of the same names:

```
# Properties to use for reporting and null hypothesis tests,
# can be regular expressions
properties:          ListOrTuple(Str())
                        default: [all]

# Possible testers are 't', 'ks' and 'anderson'
tester:               ExactEither('t'|'ks'|'anderson')
                        default: t

# Range of p values that allow no conclusion.
uncertainty_range:    Tuple(float, float)
                        default: [0.05, 0.15]
```

## 7.5.4 Console

A simple reporter that just outputs a basic analysis of the benchmarks on the command line. It works for large result files and can compute pair-wise statistical tests.

This reporter is either configured via the `report/console_misc` settings block or via the command line options of the same name (prefixed with `console_`):

```
# Matches the baseline block
baseline: ''

# Position of the baseline comparison:
# 'each': after each block
# 'after': after each cluster
# 'both': after each and after cluster
# 'instead': instead of the non baselined
baseline_position: each

# 'auto': report clusters (runs with the same description)
#           and singles (clusters with a single entry, combined) separately
# 'single': report all clusters together as one
# 'cluster': report all clusters separately
# 'both': append the output of 'cluster' to the output of 'single'
mode: auto

# Output file name or '-' (stdout)
out: '-'

# Report on the failing blocks
report_errors: true

# Print statistical tests for every property for every two programs
with_tester_results: true
```

Output for a simple benchmark (with `--properties utime`):

```
Report for single runs
sleep 0.5          (    2 single benchmarks)
  utime mean =      1.(211)m, deviation =   33.27828%

sleep 1            (    2 single benchmarks)
  utime mean =      1.(172)m, deviation =   29.91891%

Equal program blocks
  sleep 0.5    sleep 1
    utime confidence =      95%, speed up =      3.26%
```

Or using `sleep 0.5` as a baseline (`--console_baseline "sleep 0.5"`):

```
Report for single runs
sleep 0.5          (    5 single benchmarks)
  utime mean =      (1).(661)m, deviation =   18.91399%

sleep 1            (    5 single benchmarks)
```

(continues on next page)

(continued from previous page)

```

    utime mean =      (1).(138)m, deviation =   37.83985%

sleep 1          (    5) with baseline sleep 1          (    5)
    utime mean =      (68).(554)%, confidence =    9%, dev =   37.83985%,   18.91399%
geometric mean of relative mean =          68.554%

Uncertain program blocks
    sleep 0.5      sleep 1
        utime confidence =    9%, speed up =    31.45%

```

The sample run\_output.yaml was created via temci short exec 'sleep 0.5' 'sleep 1' --runs 5 --runner rusage:

```

- attributes:
  description: sleep 0.5
  data:
    utime: [0.00145, 0.001275, 0.001518, 0.002089, 0.001971]
    # ...
- attributes:
  description: sleep 1
  data:
    utime: [0.00174, 0.000736, 0.001581, 0.00085, 0.000785]

```

## 7.5.5 HTML2

Creates a report with many graphics (box-plots and bar-graphs) and tables that can be exported to TeX. The produced HTML page also contains many explanations. Viewing it requires an internet connection.

Output for the simple benchmark from above (with --properties utime --properties maxrss):

All images and tables are statically generated, this results in a large HTML file with many resources. It is therefore not recommended to use this reporter with a large number of benchmarking results (benchmarked programs and properties). Rule of thumb: Only use it to analyse results comparing less than eight programs.

This reporter is either configured via the report/html2\_misc settings block or via the command line options of the same name (prefixed with html2\_)

```

# Alpha value for confidence intervals
alpha: 0.05

# Height per run block for the big comparison box plots
boxplot_height: 2.0

# Width of all big plotted figures
fig_width_big: 25.0

# Width of all small plotted figures
fig_width_small: 15.0

# Format string used to format floats
float_format: '{:5.2e}'

```

(continues on next page)

(continued from previous page)

```

# Override the contents of the output directory if it already exists?
force_override: false

# Generate pdf versions of the plotted figures?
gen_pdf: false

# Generate simple latex versions of the plotted figures?
gen_tex: true

# Generate excel files for all tables
gen_xls: false

# Name of the HTML file
html_filename: report.html

# Show the mean related values in the big comparison table
mean_in_comparison_tables: true

# Show the minimum related values in the big comparison table
min_in_comparison_tables: false

# Output directory
out: report

# Format string used to format floats as percentages
percent_format: '{:5.2%}'

# Show zoomed out (x min = 0) figures in the extended summaries?
show_zoomed_out: false

```

### 7.5.6 CSV

A reporter that outputs the configurable csv table with rows for each run block. It can be used to access the benchmarking result for further processing in other tools without using temci as a library or creating a new reporter (see [Extending temci](#)).

This reporter is either configured via the `report/csv_misc` settings block or via the command line options of the same name (prefixed with `csv_`):

```

# List of valid column specs
# format is a comma separated list of 'PROPERTY[mod]' or 'ATTRIBUTE'
# mod is one of: mean, stddev, property, min, max and stddev per mean
# optionally a formatting option can be given via PROPERTY[mod|OPT1OPT2...]
# where the OPTs are one of the following:
#     % (format as percentage)
#     p (wrap insignificant digits in parentheses (+- 2 std dev))
#     s (use scientific notation, configured in report/number) and
#     o (wrap digits in the order of magnitude of 2 std devs in parentheses).
# PROPERTY can be either the description or the short version of the property.
# Configure the number formatting further via the number settings in the settings file
columns: [description]

```

(continues on next page)

(continued from previous page)

```
# Output file name or standard out (-)
out: '-'
```

Output for a simple benchmark (with `--csv_columns "utime[mean|p],utime[stddev],utime[max]"`, see *Console* <temci\_report.html#Console>):

```
utime[mean|p],utime[stddev],utime[max]
0.00(2),0.000,0.002
0.00(1),0.000,0.002
```

## 7.5.7 Codespeed

Reporter that outputs JSON as expected by `codespeed`. Branch name and commit ID are taken from the current directory. Use it like this:

```
temci report --reporter codespeed ... \
| curl --data-urlencode json@- http://localhost:8000/result/add/json/
```

This reporter is either configured via the `report/codespeed_misc` settings block or via the command line options of the same name (prefixed with `codespeed_`):

```
# Branch name reported to codespeed. Defaults to current branch or else 'master'.
branch: ''

# Commit ID reported to codespeed. Defaults to current commit.
commit_id: ''

# Environment name reported to codespeed. Defaults to current host name.
environment: ''

# Executable name reported to codespeed. Defaults to the project name.
executable: ''

# Project name reported to codespeed.
project: ''
```

Output for a simple benchmark (with `--properties utime`, see *Console* <#Console>):

```
[
  {
    "project": "",
    "executable": "",
    "environment": "i44pc17",
    "branch": "master",
    "commitid": null,
    "benchmark": "sleep 0.5: utime",
    "result_value": 0.0016606000000000004,
    "std_dev": 0.0003140857207833556,
    "min": 0.001275,
    "max": 0.002089
```

(continues on next page)



(continued from previous page)

```

    },
    {
      "project": "",
      "executable": "",
      "environment": "i44pc17",
      "branch": "master",
      "commitid": null,
      "benchmark": "sleep 1: utime",
      "result_value": 0.0011384,
      "std_dev": 0.00043076889395591227,
      "min": 0.000736,
      "max": 0.00174
    }
  ]

```

## 7.6 temci init

Commands to create documented sample config files. Accepts the option `--settings FILE` to configure a backing settings file.

### temci init settings

Creates a sample settings file with all the default (and currently applied) settings. Might be used to update a settings file for a new version of temci.

### temci init build\_config

Creates a sample build configuration file, for more information on the format see [temci build](#).

### temci init run\_config

Creates a sample exec configuration, for more information on the format see [temci exec](#)

## 7.7 temci format

```
temci format [OPTIONS] NUMBER [ABS_DEVIATION]
```

A small formatting utility, to format numbers with their standard deviation and si prefixes.

### 7.7.1 Usage Example

```

> temci format 1.0 0.5
1.(000)

> temci format 1.56 0.005
1.56(0)

> temci format 1560 --scientific_notation
1.560k

> temci format 1560 --no-scientific_notation_si_prefixes
1.560e3

```

This tool uses the number formatting module `temci.utils.number`. The therein defined method `format_number` can be used to format numbers and has the same options as the tool itself. Read [Usage as a Library](#) on how to use the module in a project other than temci.

## 7.7.2 Options

Usage: `temci format [OPTIONS] NUMBER [ABS_DEVIATION]`

Options:

```
--settings TEXT           Additional settings file [default: ]
--log_level [debug|info|warn|error|quiet]
                           Logging level [default: info]
--sigmas INTEGER          Number of standard deviation used for the
                           digit significance evaluation [default: 2]
--scientific_notation_si_prefixes
                           Use si prefixes instead of 'e...' [default:
                           True]
--scientific_notation_si_prefixes / --no-scientific_notation_si_prefixes
                           Use si prefixes instead of 'e...' [default:
                           True]
--scientific_notation      Use the exponential notation, i.e. '10e3'
                           for 1000 [default: True]
--scientific_notation / --no-scientific_notation
                           Use the exponential notation, i.e. '10e3'
                           for 1000 [default: True]
--percentages              Show as percentages [default: False]
--percentages / --no-percentages
                           Show as percentages [default: False]
--parentheses_mode [d|o]  Mode for showing the parentheses: either d
                           (Digits are considered significant if they
                           don't change if the number itself changes +=
                           $sigmas * std dev) or o (digits are
                           considered significant if they are bigger
                           than $sigmas * std dev) [default: o]
--parentheses              Show parentheses around non significant
                           digits? (If a std dev is given) [default:
                           True]
--parentheses / --no-parentheses
                           Show parentheses around non significant
                           digits? (If a std dev is given) [default:
                           True]
--omit_insignificant_decimal_places
                           Omit insignificant decimal places [default:
                           False]
--omit_insignificant_decimal_places / --no-omit_insignificant_decimal_places
                           Omit insignificant decimal places [default:
                           False]
--min_decimal_places INTEGER
                           The minimum number of shown decimal places
                           if decimal places are shown [default: 3]
--max_decimal_places INTEGER
                           The maximum number of decimal places
                           [default: 5]
--force_min_decimal_places
                           Don't omit the minimum number of decimal
```

(continues on next page)

(continued from previous page)

	places if insignificant? [default: True]
<code>--force_min_decimal_places / --no-force_min_decimal_places</code>	Don't omit the minimum number of decimal places if insignificant? [default: True]
<code>--help</code>	Show this message and exit.

These options can also be set in the settings file, under `report/number`.

## 7.8 OS Support

Linux is the main target for this tool. The support for other Unix like operating systems is limited. Most of the advanced environment setup functionality, like `cpu sets` or disabling hyper threading, is Linux specific.

### 7.8.1 What works and what does not

- **temci exec and temci short**
  - the `perf_stat` runner is Linux specific
  - all other runners should work, but it is uncertain whether the `rusage` runner works
  - the `time` runner requires the `gtime` program to be installed
  - most the environment setup code (i.e. the plugins) don't work, with the exception of `preheat` and `sleep` that are implemented in python
  - `--sudo` is only supported on Linux
- **temci shell**
  - see `temci exec` for the supported plugins
- **temci setup**
  - might not work
- **temci report, temci build, temci clean, temci completion, ...**
  - without any constraints

### 7.8.2 Other Unixes

Other Unix like operating systems aren't currently tested. But there is a chance that they might work as well.

### 7.8.3 Windows

Windows is currently not supported, but `temci report` might still work. The Linux subsystem in Windows might enable the usage of the features that work on Apples OS X.

## 7.9 Extending temci

Temci can be extended by either editing the code of temci directly or by placing the code in a file in your local `~/ .temci` folder or in a folder that is passed to temci via the `TEMCI_PLUGIN_PATH` variable.

This page documents how to implement new reporters, runners and run plugins and how to use temci directly as a library.

### 7.9.1 Usage as a Library

temci can be used in library mode by importing via

```
import temci.utils.library_init
```

### 7.9.2 New Reporter

New reporters can be added by creating a subclass of `AbstractReporter`. Adding a new reporter can be useful to integrate temci into other tools. It has the advantage over using temci as a library that it is directly integrated into the cli and the settings framework.

The following is an implementation of a sample reporter that outputs some benchmarking information as JSON. This reporter is based on the codespeed reporter:

```
@register(ReporterRegistry, "json", Dict({
    # define the settings for this reporter
    # currently every setting has to have a valid default value
    "project": Str() // Default("") // Description("Project name reported to codespeed.
    ↪"),
})) # the register call registers the reporter
class JSONReporter(AbstractReporter):
    """
    Outputs the benchmarking information with some meta data on the command line.
    """

    def report(self):
        """
        Create a report and output it as configured.
        """
        import json
        self.meta = {
            "project": self.misc["project"] # access the settings specific to this ↪
            ↪reporter
        }
        data = [self._report_prop(run, prop)
                # iterate overall recorded properties of all run programs
                for run in self.stats_helper.runs
                for prop in sorted(run.get_single_properties())]
        json.dump(data, sys.stdout)

    def _report_prop(self, run: RunData, prop: SingleProperty) -> dict:
        return {
            **self.meta,
```

(continues on next page)

(continued from previous page)

```

        "benchmark": "{}: {}".format(run.description(), prop.property),
        "result_value": prop.mean(),
        "std_dev": prop.stddev(),
        "min": prop.min(),
        "max": prop.max(),
    }

```

For more information, consider looking into the documentation of the [report module](#).

### 7.9.3 New Runner

Before implementing a new runner, you should consider whether using the output runner is enough. The output runner parses the output of the benchmarked programs as a list of `property: value` mappings, e.g. the output of a program could be `time: 10000.0`.

Implementing a new runner offers more flexibility, but is also slightly more work. A runner can be implemented by extending the `ExecRunner` class.

A good example is the `OutputRunner` itself, with some added documentation:

```

@ExecRunDriver.register_runner() # register the runner
class OutputExecRunner(ExecRunner):
    """
    Parses the output of the called command as YAML dictionary (or list of dictionaries)
    populate the benchmark results (string key and int or float value).
    For the simplest case, a program just outputs something like `time: 10000.0`.
    """

    name = "output" # name of the runner
    misc_options = Dict({})
    # settings of the runner, these can be set under `run/exec/NAME_misc` in the settings_
    ↪file

    def __init__(self, block: RunProgramBlock):
        """
        Creates an instance.

        :param block: run program block to measure
        """
        super().__init__(block)

    def setup_block(self, block: RunProgramBlock, cpuset: CPUSet = None, set_id: int = ↪
    ↪0):
        """
        Configure the passed copy of a run program block (e.g. the run command).

        The parts of the command between two `$SUDO$` occurrences is run with
        super user privileges if in `--sudo` mode.

        :param block: modified copy of a block
        :param cpuset: used CPUSet instance
        :param set_id: id of the cpu set the benchmarking takes place in

```

(continues on next page)

(continued from previous page)

```

    """
    pass

    def parse_result_impl(self, exec_res: ExecRunDriver.ExecResult,
                          res: BenchmarkingResultBlock = None) -> BenchmarkingResultBlock:
        """
        Parse the output of a program and turn it into benchmarking results.
        :param exec_res: program output
        :param res:      benchmarking result to which the extracted results should be
        ↪added
                        or None if they should be added to an empty one
        :return: the modified benchmarking result block
        """
        res = res or BenchmarkingResultBlock()
        # schema for the output of a program
        dict_type = Dict(key_type=Str(),
                        value_type=Either(Int(), Float(), List(Either(Int(), Float()))),
                        unknown_keys=True)
        output = yaml.safe_load(exec_res.stdout.strip())
        if isinstance(output, dict_type):
            res.add_run_data(dict(output))
        elif isinstance(output, List(dict_type)):
            for entry in list(output):
                res.add_run_data(entry)
        else:
            raise BenchmarkingError("Not a valid benchmarking program output: {}".
                                    .format(exec_res.stdout))
        return res

    def get_property_descriptions(self) -> t.Dict[str, str]:
        """
        Returns a dictionary that maps some properties to their short descriptions.
        """
        return {}

```

## 7.9.4 New exec Plugin

New plugins for setting up the benchmarking environment can be developed by extending the `AbstractRunDriverPlugin` class.

A simple example is the `DisableSwap` plugin:

```

# register the plugin and state the configuration
@register(ExecRunDriver, "disable_swap", Dict({}))
class DisableSwap(AbstractRunDriverPlugin):
    """
    Disables swapping on the system before the benchmarking and enables it after.
    """

    needs_root_privileges = True

```

(continues on next page)

(continued from previous page)

```
def setup(self): # called before the whole benchmarking starts
    self._exec_command("swapoff -a")

def teardown(self): # called after the benchmarking (and on abort)
    self._exec_command("swapon -a")
```

## 7.10 Contributing

Pull requests and issues are always welcomed.

### 7.10.1 Issues

Issues can be submitted at [GitHub](#) and should specify the used settings (and if possible the local `temci.yaml` configuration file).

### 7.10.2 New Features

New features, runners, reporters, ... are welcome. To learn how to extend temci, see [Extending temci](#). The code can be added to the appropriate places and should be tested with a few tests.

### 7.10.3 Coding Style

The code should use type annotations everywhere and use functionality of the `typecheck` module whenever there is uncertainty over the type of a variable (e.g. when reading from a YAML file). The currently used python version 3.6, all code should run in python 3.6 and above.

### 7.10.4 Documentation

Be sure to keep the documentation up to date and document your code. The code comments are written in `reStructuredText`.

### 7.10.5 Testing

The tests are located in the `tests` folder and roughly grouped by the temci subcommand they belong to. New features should be covered by tests.

There is also support for doctests that can be added into the documentation.

The tests are using the pytest framework and can be executed by simply calling

```
./test.sh
```

It is recommended to install the package `pytest-clarity` to improve the error output.

## 7.11 Changelog

### 7.11.1 0.8.5

- add *disable\_turbo\_boost* plugin, supporting amd and intel cpus
- fix #133 (not re-enabling hyper threading properly)

### 7.11.2 0.8.4

- Fix cpu set bug

### 7.11.3 0.8.3

- add *-watch*: prints console report continuously
- improve progressbar
- *perf* runner works on NixOS
- use new click version (temci is installable via pip click again)

### 7.11.4 0.8.2

- improve HTML2 reporter - fix typos - change “error” into “severe warning” - support disabling warnings altogether - clean up duplicates - further improve the summary section - support zoomed out graphs (make this the default) - use local copy of all JS and CSS (no works offline)
- record some information on the execution environment
- don’t build kernel modules by default
- remove meta analysis code

### 7.11.5 0.8.1

- fixed minor issues
- add new runner capabilities like output parsing or rusage

### 7.11.6 0.8.0

- removed the randomization features from the builder
- removed the html reporter (use the html2 reporter instead)



## 7.12 License

### GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<http://fsf.org/>> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

#### Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.

States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and

modification follow.

## TERMS AND CONDITIONS

### 0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

### 1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all

the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

## 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

## 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

## 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

## 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent

works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

## 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent

copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

## 7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

## 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

#### 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

#### 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

#### 11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the

patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

#### 12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

#### 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

#### 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future



versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different

permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

#### 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

#### 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

### END OF TERMS AND CONDITIONS

#### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WAR-
RANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under
certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

## 7.13 API Documentation

### 7.13.1 Subpackages

#### temci.build package

##### Submodules

#### temci.build.build\_processor module

```
class temci.build.build_processor.BuildProcessor(build_blocks: Optional[List[Dict[str, Any]]] =
                                                None)
```

Bases: object

Build programs

A block in the configuration has the following format:

```
# Optional attributes that describe the block
attributes:
    description:          Optional(Str())

    # Tags of this block
    tags:                 ListOrTuple(Str())

# Build configuration for this program block
build_config:
    # Base directory that contains everything to build an run the program
    base_dir:             Either(DirName()|non existent)
```

(continues on next page)

(continued from previous page)

```

        default: .

    # Used version control system branch (default is the current branch)
    branch:      Either(Str()|non existent)

    # Command to build this program block, might randomize it
    cmd:         Str()

    # Number of times to build this program
    number:      Either(Int(constraint=<function>)|non existent)
                default: 1

    # Used version control system revision of the program (-1 is the current_
    ↪revision)
    revision:    Either(Either(Str()|Int())|non existent)
                default: -1

    # Working directory in which the build command is run
    working_dir: Either(DirName()|non existent)
                default: .

    # Run configuration for this program block
    run_config:  Dict(, keys=Any, values=Any, default = {})

```

Creates a build processor for the passed build block configurations.

#### Parameters

**build\_blocks** – passed build block configurations

```

block_scheme = # Optional attributes that describe the block attributes:
description: Optional(Str()) # Tags of this block tags: ListOrTuple(Str()) # Build
configuration for this program block build_config: # Base directory that contains
everything to build an run the program base_dir: Either(DirName()|non existent)
default: . # Used version control system branch (default is the current branch)
branch: Either(Str()|non existent) # Command to build this program block, might
randomize it cmd: Str() # Number of times to build this program number:
Either(Int(constraint=<function>)|non existent) default: 1 # Used version control
system revision of the program (-1 is the current revision) revision:
Either(Either(Str()|Int())|non existent) default: -1 # Working directory in which
the build command is run working_dir: Either(DirName()|non existent) default: . #
Run configuration for this program block run_config: Dict(, keys=Any, values=Any,
default = {})

```

Type scheme of the program block configurations

#### build()

Build the configured programs.

#### out

Temporary directory in which the building takes place

**classmethod preprocess\_build\_blocks**(blocks: List[Dict[str, Any]]) → List[Dict[str, Any]]

Pre process and check build blocks

#### Returns

pre processed copy

```
classmethod store_example_config(file: str, comment_out_defaults: bool = False)
```

## temci.build.builder module

```
exception temci.build.builder.BuildError(thread: int, item: BuilderQueueItem, error: RecordedError)
```

Bases: Exception

log()

```
class temci.build.builder.Builder(id: int, build_dir: str, build_cmd: str, revision: Union[str, int], number: int, base_dir: str, branch: str)
```

Bases: object

Allows the building of a program configured by a program block configuration.

Creates a new builder for a program block.

### Parameters

- **build\_dir** – working directory in which the build command is run
- **build\_cmd** – command to build this program block
- **revision** – used version control systemrand revision of the program (-1 is the current revision)
- **number** – number of times to build this program
- **base\_dir** – base directory that contains everything to build an run the program
- **branch** – used version control system branch

```
build(thread_count: Optional[int] = None) → List[str]
```

Build the program block in parallel with at maximum *thread\_count* threads in parallel.

### Parameters

**thread\_count** – number of threads to use at maximum to build the configured number of time, defaults to *build/threads*

### Returns

list of base directories for the different builds

**build\_cmd**

Command to build this program block

**build\_dir**

Working directory in which the build command is run

**number**

Number of times to build this program

**revision**

Used version control system revision of the program

**vcs\_driver**

Used version control system driver

```
exception temci.build.builder.BuilderKeyboardInterrupt(error: BaseException, result: List[str])
```

Bases: KeyboardInterrupt

KeyboardInterrupt that wraps an error that occurred during the building of a program block

**error**

Wrapped error

**result**

Base directories of the succesfull builds

```
class temci.build.builder.BuilderQueueItem(id, number, tmp_build_dir, tmp_dir, build_cmd)
```

Bases: tuple

Create new instance of BuilderQueueItem(id, number, tmp\_build\_dir, tmp\_dir, build\_cmd)

**property build\_cmd**

Alias for field number 4

**property id**

Alias for field number 0

**property number**

Alias for field number 1

**property tmp\_build\_dir**

Alias for field number 2

**property tmp\_dir**

Alias for field number 3

```
class temci.build.builder.BuilderThread(id: int, submit_queue: Queue)
```

Bases: Thread

Thread that fetches configurations from a queue and builds the therein described program blocks.

Creates a new builder thread

**Parameters**

- **id** – id of the thread
- **submit\_queue** – used queue

**id**

Id of this thread

**run()**

Queue fetch loop, that builds the fetched program block configurations.

**stop**

Stop the queue fetch loop?

**submit\_queue**

Used queue

## Module contents

This module contains the build part of temci (usable from the command line with *temci build*).

It's separated into four parts with the following purposes:

- build\_processor.py: facade for the the builders
- builder.py: Build programs

## temci.misc package

### Submodules

#### temci.misc.game module

Benchmarks game inspired comparison of different implementations for a given language.

It doesn't really belong directly to the temci tool, but uses big parts of it. It's currently in a pre alpha state as it's a part of the evaluation for my bachelor thesis that I'm currently doing,

```
temci.misc.game.AV_GHC_VERSIONS = ['7.0.1', '7.2.1', '7.4.1', '7.6.1', '7.8.1', '7.10.1', '8.0.1']
```

These are (currently) the versions installable via the ppa on <https://launchpad.net/~hvr/+archive/ubuntu/ghc>  
Older versions can't be installed due to version conflicts and missing libraries

```
class temci.misc.game.BOTableColumn(title: str, format_str: str, property: Callable[[SingleProperty, List[float], List[SingleProperty], int], float], reduce: Callable[[List[float]], Any])
```

Bases: object

Column for BaseObject table\_html\_for\_vals\_per\_impl

```
class temci.misc.game.BaseObject(name: str, children: Optional[Union[Dict[str, BaseObject], InsertionTimeOrderedDict]] = None)
```

Bases: object

A base class for all other classes that provides helper methods.

```
boxplot_html(base_file_name: str, singles: List[SingleProperty], zoom_in: bool = False) → str
```

```
boxplot_html_for_data(name: str, base_file_name: str, data: Dict[str, List[float]], zoom_in: bool = False)
```

```
build(base_dir: str) → List[dict]
```

```
classmethod from_config_dict(*args) → BaseObject
```

```
get_geom_over_rel_means() → Dict[str, float]
```

```
get_geom_over_rel_stds() → Dict[str, float]
```

```
get_geom_std_over_rel_means() → Dict[str, float]
```

```
get_gsd_for_x_per_impl(property: Callable[[SingleProperty, List[float], List[SingleProperty], int], float]) → Dict[str, float]
```

Calculates the geometric standard deviation for the property for each implementation.

```
get_reduced_x_per_impl(property: Callable[[SingleProperty, List[float], List[SingleProperty], int], float], reduce: Callable[[List[float]], Any], x_per_impl_func: Optional[Callable[[Callable[[SingleProperty, List[float], List[SingleProperty], int], float]], Dict[str, List[float]]]] = None) → Dict[str, float]
```

Returns the reduced [property] for each implementation. To reduce the list of [property] it uses the passed reduce function. The returned implementations doesn't depend on one of the parameters.

```
get_x_per_impl(property: Callable[[SingleProperty, List[float], List[SingleProperty], int], float]) → Dict[str, List[float]]
```

Returns a list of [property] for each implementation.

#### Parameters

**property** – property function that gets a SingleProperty object and a list of all means and returns a float

```
table_html_for_vals_per_impl(columns: List[Union[BOTableColumn, Callable[[BOTableColumn]], base_file_name: str, x_per_impl_func: Optional[Callable[[Callable[[SingleProperty, List[float], List[SingleProperty], int], float]], Dict[str, List[float]]]] = None) → str
```

Returns the html for a table that has a row for each implementation and several columns (the first is the implementation column).

```
class temci.misc.game.Implementation(parent: ProgramWithInput, name: str, run_cmd: str, build_cmd: Optional[str] = None, run_data: Optional[List[Union[int, float]]] = None)
```

Bases: [BaseObject](#)

Represents an implementation of a program.

```
build(base_dir: str) → List[dict]
```

```
classmethod from_config_dict(parent: ProgramWithInput, config: dict) → Implementation
```

```
get_single_property() → SingleProperty
```

```
get_x_per_impl(property: Callable[[SingleProperty, List[float], List[SingleProperty], int], float])
```

Returns a list of [property] for each implementation.

#### Parameters

**property** – property function that gets a SingleProperty object and a list of all means and returns a float

```
mean() → float
```

```
class temci.misc.game.Input(prefix: Optional[str] = None, number: Optional[Union[int, float]] = None, appendix: Optional[str] = None)
```

Bases: [object](#)

Input with a variable numeric part.

```
classmethod from_config_dict(config: dict) → Input
```

```
classmethod list_from_numbers(*numbers: List[Union[int, float]]) → List[Input]
```

```
replace(search: str, replacement: str) → Input
```

Returns an input object in which the search string is replaced in the prefix and the appendix.

```
to_dict() → dict
```

```
class temci.misc.game.Language(name: str, categories: List[ProgramCategory])
    Bases: BaseObject

    apply_program_filter(filter: ~typing.Callable[[int, ~typing.List[~temci.misc.game.Program]], bool] =
        <function id_program_filter>)

    build(base_dir: str, multiprocess: bool = True) → List[dict]

    create_temci_run_file(base_build_dir: str, file: str)

    classmethod from_config_dict(config: dict) → Language

    get_box_plot_html(base_file_name: str) → str

    get_box_plot_per_input_per_impl_html(base_file_name: str, input_num: int) → str
        A box plot for each input that shows the mean scores (over all programs) for each implementation.

    get_full_html(base_dir: str, html_func: Optional[Callable[[str, int, bool], str]] = None) → str

    get_html(base_file_name: str, h_level: int, with_header: bool = True, multiprocess: bool = False) → str

    get_html2(base_file_name: str, h_level: int, with_header: bool = True, multiprocess: bool = False,
        show_entropy_distinction: bool = True)

    get_impl_mean_scores() → Dict[str, float]

    get_max_input_num() → int

    get_scores_per_impl() → Dict[str, List[float]]

    get_statistical_property_scores(func: ~typing.Callable[[~temci.report.stats.SingleProperty], float],
        reduce: ~typing.Callable[[~typing.List[float]],
        ~temci.utils.typecheck.Any] = <function gmean>) → Dict[str, float]

    get_statistical_property_scores_per_impl(func: Callable[[SingleProperty], float]) → Dict[str,
        List[float]]

    get_statistical_property_scores_per_input_per_impl(func: ~typ-
        ing.Callable[[~temci.report.stats.SingleProperty],
        float], input_num: int, reduce:
        ~typing.Callable[[~typing.List[float]],
        ~temci.utils.typecheck.Any] = <function
        gmean>) → Dict[str, List[float]]

    Assumptions:
        • Most programs have the same number of input (known as max input number)
        • The input number n takes roughly the same amount of time for every program category

    get_x_per_impl_and_input(property: Callable[[SingleProperty, List[float], List[SingleProperty], int],
        float], input_num: int) → Dict[str, List[float]]

    classmethod merge_different_versions_of_the_same(configs: List[dict], config_impl_apps:
        List[str], group_by_app: bool)

    process_result_file(file: str, property: str = 'task-clock')
```



```
set_difference_from_two_result_dicts(run_datas: Tuple[List[Dict[str, Any]]], app: str, property: str = 'task-clock')
```

First - Second for each measured value

```
set_merged_run_data_from_result_dict(run_datas: List[List[Dict[str, Any]]], impl_apps: List[str], property: str = 'task-clock')
```

```
set_run_data_from_result_dict(run_datas: List[Dict[str, Any]], property: str = 'task-clock')
```

```
store_html(base_dir: str, clear_dir: bool = True, html_func: Optional[Callable[[str, int, bool], str]] = None)
```

```
class temci.misc.game.Mode(value)
```

Bases: Enum

An enumeration.

```
geom_mean_rel_to_best = 1
```

calculate all mean scores as “mean / best mean” and use the geometric mean for summaries

```
mean_rel_to_first = 2
```

calculate all mean scores as “mean / mean of first” and use the arithmetic mean for summaries

```
mean_rel_to_one = 3
```

calculate all mean scores as “mean / 1” and use the arithmetic mean for summaries

```
class temci.misc.game.Program(parent: ProgramCategory, name: str, file: str, prog_inputs: Optional[List[ProgramWithInput]] = None, copied_files: Optional[List[str]] = None)
```

Bases: [BaseObject](#)

A program with several different inputs.

```
build(base_dir: str) → List[dict]
```

```
entropy
```

Entropy of the implementation

```
classmethod from_config_dict(parent: ProgramCategory, config: dict) → Implementation
```

```
get_box_plot_html(base_file_name: str) → str
```

```
get_box_plot_per_input_per_impl_html(base_file_name: str, input: str) → str
```

A box plot for each input that shows the execution times for each implementation.

```
get_html(base_file_name: str, h_level: int) → str
```

```
get_html2(base_file_name: str, h_level: int)
```

```
get_impl_mean_scores() → Dict[str, List[float]]
```

Geometric mean over the means relative to best per implementation (per input).

```
get_statistical_property_scores(func: Callable[[SingleProperty], float]) → Dict[str, List[float]]
```

```
get_statistical_property_scores_per_input_per_impl(func: Callable[[SingleProperty], float], input: str) → Dict[str, float]
```

```
class temci.misc.game.ProgramCategory(parent: Language, name: str, programs: List[Program])
```

Bases: [BaseObject](#)

Represents a specific abstract program that gives the specification for several implementations (aka “program”s).

```
apply_program_filter(filter: ~typing.Callable[[int, ~typing.List[~temci.misc.game.Program]], bool] =  
    <function id_program_filter>)
```

Filters the programs that make up self.children and self.programs. It stores the original set of programs else where.

#### Parameters

**filter** – the used filter, the id filter resets the original state

```
build(base_dir: str) → List[dict]
```

```
classmethod from_config_dict(parent: Language, config: dict) → ProgramCategory
```

```
get_box_plot_html(base_file_name: str) → str
```

```
get_box_plot_per_input_per_impl_html(base_file_name: str, input: str) → str
```

A box plot for each input that shows the mean scores (over all programs) for each implementation.

```
get_html(base_file_name: str, h_level: int) → str
```

```
get_html2(base_file_name: str, h_level: int)
```

```
get_impl_mean_scores() → Dict[str, float]
```

```
get_input_strs() → List[str]
```

```
get_scores_per_impl() → Dict[str, List[float]]
```

```
get_statistical_property_scores(func: ~typing.Callable[[~temci.report.stats.SingleProperty], float],  
    reduce: ~typing.Callable[[~typing.List[float]],  
    ~temci.utils.typecheck.Any] = <function gmean>) → Dict[str, float]
```

```
get_statistical_property_scores_per_impl(func:  
    ~typing.Callable[[~temci.report.stats.SingleProperty],  
    float], reduce: ~typing.Callable[[~typing.List[float]],  
    ~temci.utils.typecheck.Any] = <function gmean>) →  
    Dict[str, float]
```

```
get_statistical_property_scores_per_input_per_impl(func: Callable[[SingleProperty], float],  
    input: str) → Dict[str, List[float]]
```

```
get_x_per_impl_and_input(property: Callable[[SingleProperty, List[float], List[SingleProperty], int],  
    float], input: str) → Dict[str, List[float]]
```

```
temci.misc.game.ProgramFilterFunc
```

A function that get’s the current program index in the also passed list of all other programs and returns True if the program is okay and False otherwise.

alias of [Callable\[\[int, List\[Program\]\], bool\]](#)

```
class temci.misc.game.ProgramWithInput(parent: Program, input: Input, impls: List[Implementation], id:  
    int)
```

Bases: [BaseObject](#)

This represents the program with a specific input. It has several program implementations.

**build**(*base\_dir: str*) → List[dict]

**get\_box\_plot\_html**(*base\_file\_name: str*) → str

**get\_html**(*base\_file\_name: str, h\_level: int*) → str

**get\_html2**(*base\_file\_name: str, h\_level: int*) → str

**get\_means\_rel\_to\_best**() → Dict[str, float]

**get\_single**()

**get\_single\_properties**() → List[Tuple[str, *SingleProperty*]]

**get\_statistical\_properties\_for\_each**(*func: Callable[[SingleProperty], float]*) → Dict[str, float]

**get\_x\_per\_impl**(*property: Callable[[SingleProperty, List[float], List[SingleProperty], int], float]*) → Dict[str, List[float]]

Returns a list of [property] for each implementation.

#### Parameters

**property** – property function that gets a *SingleProperty* object and a list of all means and returns a float

**temci.misc.game.ReduceFunc**

Gets passed a list of values and returns a single value, e.g. stats.gmean

alias of Callable[[List[float]], *Any*]

**temci.misc.game.StatProperty**

Gets passed a *SingleProperty* object, the list of means (containing the object's mean), the list of all *SingleProperty* objects and the index of the first in it and returns a float.

alias of Callable[[*SingleProperty*, List[float], List[*SingleProperty*], int], float]

**temci.misc.game.StatisticalPropertyFunc**

Get's passed the *SingleProperty* object to process and min mean

alias of Callable[[*SingleProperty*], float]

**temci.misc.game.amean\_std**(*values: List[float]*) → float

Calculates the arithmetic mean.

**temci.misc.game.bench\_categories**(*ending: str, inputs: Dict[str, List[Input]]*) → List[dict]

**temci.misc.game.bench\_category**(*category: str, ending: str, inputs: List[Input], numbers: Optional[List[int]] = None*) → dict

**temci.misc.game.bench\_file**(*category: str, ending: str, number: int = 1*) → str

**temci.misc.game.bench\_program**(*category: str, ending: str, inputs: List[Input], number: int = 1*) → dict

**temci.misc.game.c\_config**(*inputs\_per\_category: Dict[str, List[Input]], optimisation: str = '-O2', clang\_version='3.7'*) → Dict[str, Union[str, dict]]

Generates a game config that compares gcc and clang.

**temci.misc.game.cparser\_config**(*inputs\_per\_category: Dict[str, List[Input]], optimisation: str = '-O2', clang\_version='3.7'*) → Dict[str, Union[str, dict]]

Generates a game config that compares gcc, clang and cparser.

`temci.misc.game.divide_inputs`(*inputs\_per\_category*: Dict[str, List[Input]], *divisor*: Union[int, float]) → Dict[str, List[Input]]

`temci.misc.game.empty_inputs`(*inputs\_per\_category*: Dict[str, List[Input]]) → Dict[str, List[Input]]

`temci.misc.game.file_entropy`(*file*: str) → int

Calculates the entropy of given file by taking the length of its gzip compressed content

`temci.misc.game.file_lines`(*file*: str) → int

Number of non empty lines in the file

`temci.misc.game.first`(*values*: List[float]) → float

`temci.misc.game.first_inputs`(*inputs\_per\_category*: Dict[str, List[Input]]) → Dict[str, List[Input]]

`temci.misc.game.haskel_config`(*inputs\_per\_category*: Dict[str, List[Input]], *optimisation*: str, *ghc\_versions*: Optional[List[str]] = None, *used\_c\_compilers*: Optional[List[str]] = None) → Dict[str, Union[str, dict]]

Generate a game config comparing all available ghc versions

#### Parameters

- **inputs\_per\_category** –
- **optimisation** – optimisation flags, e.g. ‘-Odph’ or ‘-O’
- **ghc\_versions** – compared ghc versions, if None, AV\_GHC\_VERSIONS is used

`temci.misc.game.id_program_filter`(*\_1*, *\_2*)

`temci.misc.game.last_inputs`(*inputs\_per\_category*: Dict[str, List[Input]]) → Dict[str, List[Input]]

`temci.misc.game.mean_rel_std`()

`temci.misc.game.mean_score_column`()

`temci.misc.game.mean_score_std_column`()

`temci.misc.game.prefix_inputs`(*prefix*: str, *inputs*: List[Input]) → List[Input]

`temci.misc.game.process`(*config*: ~typing.Dict[str, ~typing.Union[str, dict]], *name*: ~typing.Optional[str] = None, *build\_dir*: ~typing.Optional[str] = None, *build*: bool = True, *benchmark*: bool = True, *report*: bool = True, *temci\_runs*: int = 15, *temci\_options*: str = ‘--discarded\_blocks 1’, *temci\_stop\_start*: bool = True, *report\_dir*: ~typing.Optional[str] = None, *property*: str = ‘task-clock’, *report\_modes*: ~typing.List[~temci.misc.game.Mode] = [<Mode.mean\_rel\_to\_first: 2>, <Mode.geom\_mean\_rel\_to\_best: 1>])

Process a config dict. Simplifies the build, benchmarking and report generating.

#### Parameters

- **config** – processed config dict
- **name** – the name of the whole configuration (used to generate the file names), default “{config[‘language’]}”
- **build\_dir** – build dir that is used to build the programs, default is “/tmp/{name}”
- **build** – make a new build of all programs? (results in a “{name}.exec.yaml” file for temci)
- **benchmark** – benchmark the “{name}.exec.yaml” file (from a built)? (results in a “{name}.yaml” result file)

- **report** – generate a game report? (results in a report placed into the report\_dir)
- **temci\_runs** – number of benchmarking runs (if benchmark=True)
- **temci\_options** – used options for temci
- **temci\_stop\_start** – does temci use the StopStart plugin for decreasing the variance while benchmarking?
- **report\_dir** – the directory to place the report in, default is “{name}\_report”
- **property** – measured property for which the report is generated, default is “task-clock”

```
temci.misc.game.produce_ttest_comparison_table(datas: List[List[Dict[str, Union[Dict[str, List[float]], Any]]], impls: List[str], data_descrs: List[str],
                                              filename: str, property: str = 'task-clock', alpha: float = 0.05, tester_name: str = 't', ratio_format: str = '{:3.0%}')
```

```
temci.misc.game.property_filter_half(cur_index: int, all: List[Program], property_func: Callable[[Program], float], remove_upper_half: bool) → bool
```

Note: if the number of programs is uneven, then one program will belong to the upper and the lower half.

```
temci.misc.game.ref(name: str, value=None, _store={'fasta': ['250000', '2500000', '25000000']})
```

A simple YAML like reference utility. It to easily store a value under a given key and return it.

#### Parameters

- **name** – name of the reference
- **value** – new value of the reference (if value isn't None)
- **\_store** – dict to store everything in

#### Returns

the value of the reference

```
temci.misc.game.rel_mean_func(x, min)
```

```
temci.misc.game.rel_mean_property(single: SingleProperty, means: List[float], *args) → float
```

A property function that returns the relative mean (the mean of the single / minimum of means)

```
temci.misc.game.rel_std_dev_func(x: SingleProperty, min: float) → float
```

```
temci.misc.game.rel_std_dev_to_min_func(x: SingleProperty, min: float) → float
```

```
temci.misc.game.rel_std_property(single: SingleProperty, means: List[float], *args) → float
```

A property function that returns the relative standard deviation (relative to single's mean)

```
temci.misc.game.replace_run_with_build_cmd(config_dict: Dict[str, Union[str, dict]]) → Dict[str, Union[str, dict]]
```

```
temci.misc.game.rust_config(inputs_per_category: Dict[str, List[Input]], optimisation: int = 3) → Dict[str, Union[str, dict]]
```

Generates a game config that compares the different rust versions.

```
temci.misc.game.ttest_rel_to_first_property(single: SingleProperty, _, all_singles: List[SingleProperty], index: int) → float
```

```
temci.misc.game.ttest_summarize(values: List[float]) → float
```

```
temci.misc.game.ttest_to_first()
```

```
temci.misc.game.used_rel_mean_property(single: SingleProperty, means: List[float], *args) → float
```

```
temci.misc.game.used_std_property(single: SingleProperty, means: List[float], *args) → float
```

```
temci.misc.game.used_summarize_mean(values: List[float]) → float
```

```
temci.misc.game.used_summarize_mean_std(values: List[float]) → float
```

## Module contents

The stuff in this folder doesn't really belong to the temci tool but builds on top of it some cool applications, like a benchmarksgame inspired comparison of different implementations of several languages. The tools may depend on other code or packages than the temci tool itself.

## temci.run package

### Submodules

#### temci.run.cpuset module

```
temci.run.cpuset.BENCH_SET = 'temci.set'
```

Name of the base cpu set used by temci for benchmarking purposes

```
temci.run.cpuset.CONTROLLER_SUB_BENCH_SET = 'temci.set.controller'
```

Name of the cpu set used by the temci control process

```
temci.run.cpuset.CPUSET_DIR = '/cpuset'
```

Location that the cpu set pseudo file system is mounted at

```
class temci.run.cpuset.CPUSet(active: bool = True, base_core_number: Optional[int] = None, parallel: Optional[int] = None, sub_core_number: Optional[int] = None, temci_in_base_set: bool = True)
```

Bases: object

This class allows the usage of cpusets (see *man cpuset*) and therefore requires root privileges. It uses the program *cset* to modify the cpusets. This class needs root privileges to operate properly. Warns if not.

Initializes the cpu sets and determines the number of parallel programs (*parallel\_number* variable).

#### Parameters

- **active** – are cpu sets actually used?
- **base\_core\_number** – number of cpu cores for the base (remaining part of the) system
- **parallel** – 0: benchmark sequential, > 0: benchmark parallel with n instances, -1: determine n automatically
- **sub\_core\_number** – number of cpu cores per parallel running program
- **temci\_in\_base\_set** – place temci in the same cpu set as the rest of the system?

#### Raises

- **ValueError** – if the passed parameters don't work together on the current platform

- **EnvironmentError** – if the environment can't be setup properly (e.g. no root privileges)

**active**

Are cpu sets actually used?

**av\_cores**

Number of available cpu cores

**base\_core\_number**

Number of cpu cores for the base (remaining part of the) system

**get\_sub\_set**(*set\_id: int*) → str

Gets the name of the benchmarking cpu set with the given id / number (starting at zero).

**move\_process\_to\_set**(*pid: int, set\_id: int*)

Moves the process with the passed id to the parallel sub cpuset with the passed id.

**Parameters**

- **pid** – passed process id
- **set\_id** – passed parallel sub cpuset id

**parallel**

0: benchmark sequential, > 0: benchmark parallel with n instances, -1: determine n automatically

**parallel\_number**

Number of used parallel instances, zero if the benchmarking is done sequentially

**sub\_core\_number**

Number of cpu cores per parallel running program

**teardown()**

Tears the created cpusets down and makes the system usable again.

**temci\_in\_base\_set**

Place temci in the same cpu set as the rest of the system?

```
temci.run.cpuset.NEW_ROOT_SET = 'bench.root'
```

Name of the new root cpu set that contains most of the processes of the original root set

```
temci.run.cpuset.SUB_BENCH_SET = 'temci.set.{}'
```

Format of cpu sub set names for benchmarking

**temci.run.run\_driver module**

This modules contains the base run driver, needed helper classes and registries.

```
class temci.run.run_driver.AbstractRunDriver(misc_settings: Optional[dict] = None)
```

Bases: *AbstractRegistry*

A run driver that does the actual benchmarking and supports plugins to modify the benchmarking environment.

The constructor also calls the setup methods on all registered plugins. It calls the setup() method.

Creates an instance. Also calls the setup methods on all registered plugins. It calls the setup() method.

**Parameters**

**misc\_settings** – further settings

**benchmark**(*block*: [RunProgramBlock](#), *runs*: *int*, *cpuset*: *Optional[CPUSet] = None*, *set\_id*: *int = 0*, *timeout*: *float = - 1*) → [BenchmarkingResultBlock](#)

Benchmark the passed program block “runs” times and return the benchmarking results.

#### Parameters

- **block** – run program block to benchmark
- **runs** – number of benchmarking runs
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarked block should be executed in
- **timeout** – timeout or -1 if no timeout is given

#### Returns

object that contains a dictionary of properties with associated raw run data

**block\_type\_scheme** = `Dict({}, False, keys=Any, values=Any, default = {})`

Type scheme for the program block configuration

**default** = []

Name(s) of the class(es) used by default. Type depends on the *use\_list* property.

**classmethod** `get_full_block_typescheme()` → [Type](#)

**get\_property\_descriptions()** → `Dict[str, str]`

Returns a dictionary that maps some properties to their short descriptions.

**get\_used\_plugins()** → `List[str]`

**misc\_settings**

Further settings

**plugin\_synonym** = ('run driver plugin', 'run driver plugins')

Singular and plural version of the word that is used in the documentation for the registered entities

**registry** = {}

Registered classes (indexed by their name)

**runs\_benchmarks** = `True`

**settings\_key\_path** = 'run/plugins'

Used settings key path

**setup()**

Call the setup() method on all used plugins for this driver.

**classmethod** `store_example_config(file: str, comment_out_defaults: bool = False)`

**store\_files** = `True`

**teardown()**

Call the teardown() method on all used plugins for this driver.

**use\_key** = 'active'

Used key that sets which registered class is currently used

**use\_list** = `True`

Allow more than one class to used at a specific moment in time



**used\_plugins**

Used and active plugins

**exception** temci.run.run\_driver.BenchmarkingError

Bases: RuntimeError

Thrown when the benchmarking of a program block fails.

**exception** temci.run.run\_driver.BenchmarkingProgramError(recorded\_error: RecordedProgramError)

Bases: *BenchmarkingError*

Thrown when the benchmarked program fails

**class** temci.run.run\_driver.BenchmarkingResultBlock(data: Dict[str, List[Union[int, float]]] = None, error: BaseException = None, recorded\_error: RecordedError = None)

Bases: object

Result of the benchmarking of one block. It includes the error object if an error occurred.

Creates an instance.

**Parameters**

- **data** – measured data per measured property
- **error** – exception object if something went wrong during benchmarking

**Returns**

**add\_run\_data**(data: Dict[str, Union[int, float, List[Union[int, float]]]])

Add data.

**Parameters**

**data** – data to be added (measured data per property)

**data**

Measured data per measured property

**error**

Exception object if something went wrong during benchmarking

**properties**() → List[str]

Get a list of the measured properties

**class** temci.run.run\_driver.CPUSpecExecRunner(block: RunProgramBlock)

Bases: *ExecRunner*

A runner that uses a tool that runs the SPEC CPU benchmarks and parses the resulting files.

To use this runner with name {name} either set the `runner` property of a run configuration or the setting under the key `run/exec_misc/runner` to its name.

The runner is configured by modifying the `spec.py` property of a run configuration. This configuration has the following structure:

```
# File patterns (the newest file will be used)
files:      ListOrTuple(Str())
            default: [result/CINT2000.*.raw, result/CFP2000.*.raw]

# Randomize the assembly during compiling?
randomize:  Bool()
```

Creates an instance.

**Parameters**

**block** – run program block to measure

**Raises**

**KeyboardInterrupt** – if the runner can't be used (e.g. if the used tool isn't installed or compiled)

```
misc_options = # File patterns (the newest file will be used) files:
ListOrTuple(Str()) default: [result/CINT2000.*.raw, result/CFP2000.*.raw] #
Randomize the assembly during compiling? randomize: Bool()
```

Type scheme of the options for this type of runner

```
name = 'spec.py'
```

Name of the runner

```
parse_result_impl(exec_res: ExecResult, res: Optional[BenchmarkingResultBlock] = None) →
    BenchmarkingResultBlock
```

Parse the output of a program and turn it into benchmarking results.

**Parameters**

- **exec\_res** – program output
- **res** – benchmarking result to which the extracted results should be added or None if they should be added

to an empty one :return: the modified benchmarking result block

```
setup_block(block: RunProgramBlock, cpuset: Optional[CPUSet] = None, set_id: int = 0)
```

Configure the passed copy of a run program block (e.g. the run command).

**Parameters**

- **block** – modified copy of a block
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarking takes place in

```
class temci.run.run_driver.ExecRunDriver(misc_settings: Optional[dict] = None)
```

Bases: [AbstractRunDriver](#)

Implements a simple run driver that just executes one of the passed run\_cmds in each benchmarking run. It measures the time using the perf stat tool (runner=perf\_stat).

The constructor calls the setup method.

Configuration format:

```
# Argument passed to all benchmarked commands by replacing $ARGUMENT with this_
↳value in the command
argument: ''

# Parse the program output as a YAML dictionary of that gives for a specific_
↳property a measurement.
# Not all runners support it.
parse_output: false

# Order in which the plugins are used, plugins that do not appear in this list are_
↳used before all
```

(continues on next page)

(continued from previous page)

```

# others
plugin_order: [drop_fs_caches, sync, sleep, preheat, flush_cpu_caches]

# Enable other plugins by default: none = (enable none by default); all = cpu_
↳governor,disable_swap,s
# ync,stop_start,other_nice,nice,disable_aslr,disable_ht,cpuset,disable_turbo_boost_
↳(enable all, might
# freeze your system); usable =
# cpu_governor,disable_swap,sync,nice,disable_aslr,disable_ht,cpuset,disable_turbo_
↳boost (like 'all'
# but doesn't affect other processes)
preset: none

# Pick a random command if more than one run command is passed.
random_cmd: true

# If not " overrides the runner setting for each program block
runner: ''

```

This run driver can be configured under the settings key `run/exec_misc`.

To use this run driver set the currently used run driver (at key `run/driver`) to its name (`exec`).

The default run driver is `exec`.

Block configuration format for the run configuration:

```

# Optional build config to integrate the build step into the run step
build_config:      Either(Dict(, keys=Any, values=Any, default = {})|non_
↳existent)

# Optional attributes that describe the block
attributes:
    description:      Optional(Str())

    # Tags of this block
    tags:              ListOrTuple(Str())

run_config:
    # Command to benchmark, adds to run_cmd
    cmd:                Str()

    # Command to append before the commands to benchmark
    cmd_prefix:         List(Str())

    # Execution directories for each command
    cwd:                Either(List(Str())|Str())
                        default: .

    # Disable the address space layout randomization
    disable_aslr:       Bool()

    # Override all other max runspecifications if > -1
    max_runs:           Int(constraint=<function>)

```

(continues on next page)

(continued from previous page)

```

        default: -1

# Override all other min runspecifications if > -1
min_runs:      Int(constraint=<function>)
                default: -1

# Parse the program output as a YAML dictionary of that gives for a specific_
↪property a
# measurement. Not all runners support it.
parse_output:      Bool()

# Used revision (or revision number). -1 is the current revision, checks out the_
↪revision
revision:      Either(Int(constraint=<function>)|Str())
                default: -1

# Commands to benchmark
run_cmd:      Either(List(Str())|Str())

# Used runner
runner:      ExactEither()
                default: time

# Override min run and max runspecifications if > -1
runs:      Int(constraint=<function>)
                default: -1

# Measured properties for rusage that are stored in the benchmarking result
rusage_properties:      ValidRusagePropertyList()

# Environment variables
env:      Dict(, keys=Str(), values=Any, default = {})

# Configuration for the output and return code validator
validator:
    # Program error output without ignoring line breaks and spaces at the_
↪beginning and the end
    expected_err_output:      Optional(Str())

    # Strings that should be present in the program error output
    expected_err_output_contains:      Either(List(Str())|Str())

    # Program output without ignoring line breaks and spaces at the beginning_
↪and the end
    expected_output:      Optional(Str())

    # Strings that should be present in the program output
    expected_output_contains:      Either(List(Str())|Str())

    # Allowed return code(s)
    expected_return_code:      Either(List(Int())|Int())

```

(continues on next page)

(continued from previous page)

```
# Strings that shouldn't be present in the program output
unexpected_err_output_contains:      Either(List(Str())|Str())

# Strings that shouldn't be present in the program output
unexpected_output_contains:         Either(List(Str())|Str())
```

Creates an instance. Also calls the setup methods on all registered plugins. It calls the setup() method.

#### Parameters

**misc\_settings** – further settings

**class ExecResult**(time, stderr, stdout, rusage)

Bases: tuple

A simple named tuple named ExecResult with to properties: time, stderr and stdout

#### property rusage

Alias for field number 3

#### property stderr

Alias for field number 1

#### property stdout

Alias for field number 2

#### property time

Alias for field number 0

**benchmark**(block: [RunProgramBlock](#), runs: int, cpuset: Optional[CPUSet] = None, set\_id: int = 0, timeout: float = - 1) → [BenchmarkingResultBlock](#)

Benchmark the passed program block “runs” times and return the benchmarking results.

#### Parameters

- **block** – run program block to benchmark
- **runs** – number of benchmarking runs
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarked block should be executed in
- **timeout** – timeout or -1 if no timeout is given

#### Returns

object that contains a dictionary of properties with associated raw run data

```

block_type_scheme = # Command to benchmark, adds to run_cmd cmd: Str() # Command to
append before the commands to benchmark cmd_prefix: List(Str()) # Execution
directories for each command cwd: Either(List(Str())|Str()) default: . # Disable
the address space layout randomization disable_aslr: Bool() # Override all other
max runspecifications if > -1 max_runs: Int(constraint=<function>) default: -1 #
Override all other min runspecifications if > -1 min_runs:
Int(constraint=<function>) default: -1 # Parse the program output as a YAML
dictionary of that gives for a specific property a measurement. # Not all runners
support it. parse_output: Bool() # Used revision (or revision number).-1 is the
current revision, checks out the revision revision:
Either(Int(constraint=<function>)|Str()) default: -1 # Commands to benchmark
run_cmd: Either(List(Str())|Str()) # Used runner runner:
ExactEither('perf_stat'|'rusage'|'spec'|'spec.py'|'time'|'output') default: time #
Override min run and max runspecifications if > -1 runs: Int(constraint=<function>)
default: -1 # Measured properties for rusage that are stored in the benchmarking
result rusage_properties: ValidRusagePropertyList() # Environment variables env:
Dict(keys=Str(), values=Any, default = {}) output: Dict({}, False, keys=Any,
values=Any, default = {}) perf_stat: # Measured properties. The number of
properties that can be measured at once is limited. properties: List(Str())
default: [wall-clock, cycles, cpu-clock, task-clock, instructions, branch-misses,
cache-references] # If runner=perf_stat make measurements of the program repeated n
times. Therefore scale the # number of times a program is benchmarked. repeat:
Int(constraint=<function>) default: 1 rusage: # Measured properties that are
stored in the benchmarking result properties: ValidRusagePropertyList() default:
[idrss, inblock, isrss, ixrss, majflt, maxrss, minflt, msgrcv, msgsnd, nivcs,
nsignals, nswap, nvcs, ouble, stime, utime] spec: # Base property path that all
other paths are relative to. base_path: Str() # Code that is executed for each
matched path. The code should evaluate to the actual measured # value for the
path.it can use the function get(sub_path: str = '') and the modules pytimeparse, #
numpy, math, random, datetime and time. code: Str() default: get() # SPEC result
file file: Str() # Regexp matching the base property path for each measured
property path_regexp: Str() default: .* spec.py: # File patterns (the newest file
will be used) files: ListOrTuple(Str()) default: [result/CINT2000.*.raw,
result/CFP2000.*.raw] # Randomize the assembly during compiling? randomize: Bool()
time: # Measured properties that are included in the benchmarking results
properties: ValidTimePropertyList() default: [utime, stime, etime, avg_mem_usage,
max_res_set, avg_res_set] # Configuration for the output and return code validator
validator: # Program error output without ignoring line breaks and spaces at the
beginning and the end expected_err_output: Optional(Str()) # Strings that should be
present in the program error output expected_err_output_contains:
Either(List(Str())|Str()) # Program output without ignoring line breaks and spaces
at the beginning and the end expected_output: Optional(Str()) # Strings that should
be present in the program output expected_output_contains:
Either(List(Str())|Str()) # Allowed return code(s) expected_return_code:
Either(List(Int())|Int()) # Strings that shouldn't be present in the program output
unexpected_err_output_contains: Either(List(Str())|Str()) # Strings that shouldn't
be present in the program output unexpected_output_contains:
Either(List(Str())|Str())

```

Type scheme for the program block configuration

```
default = []
```

Name(s) of the class(es) used by default. Type depends on the *use\_list* property.

```
get_property_descriptions() → Dict[str, str]
```

Returns a dictionary that maps some properties to their short descriptions.

**classmethod** `get_runner(block: RunProgramBlock) → ExecRunner`

Create the suitable runner for the passed run program block.

**Parameters**

**block** – passed run program block

**get\_used\_plugins()** → List[str]

Get the list of name of the used plugins (use\_list=True) or the names of the used plugin (use\_list=False).

**classmethod** `register_runner()` → Callable[[type], type]

Decorator to register a runner (has to be sub class of ExecRunner).

**registry = {}**

Registered classes (indexed by their name)

```
runners = {'output': <class 'temci.run.run_driver.OutputExecRunner'>, 'perf_stat':
<class 'temci.run.run_driver.PerfStatExecRunner'>, 'rusage': <class
'temci.run.run_driver.RusageExecRunner'>, 'spec': <class
'temci.run.run_driver.SpecExecRunner'>, 'spec.py': <class
'temci.run.run_driver.CPUSpecExecRunner'>, 'time': <class
'temci.run.run_driver.TimeExecRunner'>}
```

Dictionary mapping a runner name to a runner class

**settings\_key\_path = 'run/exec\_plugins'**

Used settings key path

**teardown()**

Call the teardown() method on all used plugins for this driver.

**use\_key = 'exec\_active'**

Used key that sets which registered class is currently used

**use\_list = True**

Allow more than one class to used at a specific moment in time

**class** `temci.run.run_driver.ExecRunner(block: RunProgramBlock)`

Bases: object

Base class for runners for the ExecRunDriver. A runner deals with creating the commands that actually measure a program and parse their outputs.

Creates an instance.

**Parameters**

**block** – run program block to measure

**Raises**

**KeyboardInterrupt** – if the runner can't be used (e.g. if the used tool isn't installed or compiled)

**get\_property\_descriptions()** → Dict[str, str]

Returns a dictionary that maps some properties to their short descriptions.

**misc**

Options for this runner

```
misc_options = Dict({}, False, keys=Any, values=Any, default = {})
```

Type scheme of the options for this type of runner

```
name = None
```

Name of the runner

```
parse_result(exec_res: ExecResult, res: Optional[BenchmarkingResultBlock] = None, parse_output: bool = False) → BenchmarkingResultBlock
```

Parse the output of a program and turn it into benchmarking results.

#### Parameters

- **exec\_res** – program output
- **res** – benchmarking result to which the extracted results should be added or None if they should be added

to an empty one :param parse\_output: parse standard out to get additional properties :return: the modified benchmarking result block

```
parse_result_impl(exec_res: ExecResult, res: Optional[BenchmarkingResultBlock] = None) → BenchmarkingResultBlock
```

Parse the output of a program and turn it into benchmarking results.

#### Parameters

- **exec\_res** – program output
- **res** – benchmarking result to which the extracted results should be added or None if they should be added

to an empty one :return: the modified benchmarking result block

```
setup_block(block: RunProgramBlock, cpuset: Optional[CPUSet] = None, set_id: int = 0)
```

Configure the passed copy of a run program block (e.g. the run command).

#### Parameters

- **block** – modified copy of a block
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarking takes place in

```
supports_parsing_out = False
```

Is the captured output on standard out useful for parsing

```
class temci.run.run_driver.ExecValidator(config: dict)
```

Bases: object

Output validator.

Configuration:

```
# Program error output without ignoring line breaks and spaces at the beginning and
↳ the end
expected_err_output: Optional(Str())

# Strings that should be present in the program error output
expected_err_output_contains: Either(List(Str())|Str())
```

(continues on next page)



(continued from previous page)

```

# Program output without ignoring line breaks and spaces at the beginning and the
↪end
expected_output:          Optional(Str())

# Strings that should be present in the program output
expected_output_contains: Either(List(Str())|Str())

# Allowed return code(s)
expected_return_code:     Either(List(Int())|Int())

# Strings that shouldn't be present in the program output
unexpected_err_output_contains: Either(List(Str())|Str())

# Strings that shouldn't be present in the program output
unexpected_output_contains: Either(List(Str())|Str())

```

Creates an instance.

#### Parameters

**config** – validator configuration

#### config

Validator configuration

```

config_type_scheme = # Program error output without ignoring line breaks and spaces
at the beginning and the end expected_err_output: Optional(Str()) # Strings that
should be present in the program error output expected_err_output_contains:
Either(List(Str())|Str()) # Program output without ignoring line breaks and spaces
at the beginning and the end expected_output: Optional(Str()) # Strings that should
be present in the program output expected_output_contains:
Either(List(Str())|Str()) # Allowed return code(s) expected_return_code:
Either(List(Int())|Int()) # Strings that shouldn't be present in the program output
unexpected_err_output_contains: Either(List(Str())|Str()) # Strings that shouldn't
be present in the program output unexpected_output_contains:
Either(List(Str())|Str())

```

Configuration type scheme

**validate**(cmd: str, out: str, err: str, return\_code: int)

Validate the passed program output, error output and return code.

#### Parameters

- **cmd** – program command for better error messages
- **out** – passed program output
- **err** – passed program error output
- **return\_code** – passed program return code

#### Raises

**BenchmarkingError** – if the check failed

temci.run.run\_driver.**Number**

Numeric value

alias of Union[int, float]

**class** temci.run.run\_driver.**OutputExecRunner**(*block*: [RunProgramBlock](#))

Bases: [ExecRunner](#)

Parses the output of the called command as YAML dictionary (or list of dictionaries) populate the benchmark results (string key and int or float value).

For the simplest case, a program just outputs something like *time: 1000.0*.

To use this runner with name {name} either set the `runner` property of a run configuration or the setting under the key `run/exec_misc/runner` to its name.

Creates an instance.

**Parameters**

**block** – run program block to measure

**Raises**

**KeyboardInterrupt** – if the runner can't be used (e.g. if the used tool isn't installed or compiled)

**get\_property\_descriptions**() → Dict[str, str]

Returns a dictionary that maps some properties to their short descriptions.

**misc\_options** = Dict({}, False, keys=Any, values=Any, default = {})

Type scheme of the options for this type of runner

**name** = 'output'

Name of the runner

**parse\_result\_impl**(*exec\_res*: [ExecResult](#), *res*: Optional[[BenchmarkingResultBlock](#)] = None) → [BenchmarkingResultBlock](#)

Parse the output of a program and turn it into benchmarking results.

**Parameters**

- **exec\_res** – program output
- **res** – benchmarking result to which the extracted results should be added or None if they should be added

to an empty one :return: the modified benchmarking result block

**setup\_block**(*block*: [RunProgramBlock](#), *cpuset*: Optional[[CPUSet](#)] = None, *set\_id*: int = 0)

Configure the passed copy of a run program block (e.g. the run command).

**Parameters**

- **block** – modified copy of a block
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarking takes place in

**class** temci.run.run\_driver.**PerfStatExecRunner**(*block*: [RunProgramBlock](#))

Bases: [ExecRunner](#)

Runner that uses `perf stat` for measurements.

To use this runner with name {name} either set the `runner` property of a run configuration or the setting under the key `run/exec_misc/runner` to its name.

This runner supports the `parse_output` option.

The runner is configured by modifying the `perf_stat` property of a run configuration. This configuration has the following structure:

```
# Measured properties. The number of properties that can be measured at once is
↳ limited.
properties:          List(Str())
                      default: [wall-clock, cycles, cpu-clock, task-clock, instructions,
↳ branch-misses, cache-references]

# If runner=perf_stat make measurements of the program repeated n times. Therefore
↳ scale the number of
# times a program is benchmarked.
repeat:             Int(constraint=<function>)
                      default: 1
```

Creates an instance.

#### Parameters

**block** – run program block to measure

#### Raises

**KeyboardInterrupt** – if the runner can't be used (e.g. if the used tool isn't installed or compiled)

```
misc_options = # Measured properties. The number of properties that can be measured
at once is limited. properties: List(Str()) default: [wall-clock, cycles,
cpu-clock, task-clock, instructions, branch-misses, cache-references] # If
runner=perf_stat make measurements of the program repeated n times. Therefore scale
the number of # times a program is benchmarked. repeat: Int(constraint=<function>)
default: 1
```

Type scheme of the options for this type of runner

```
name = 'perf_stat'
```

Name of the runner

```
parse_result_impl(exec_res: ExecResult, res: Optional[BenchmarkingResultBlock] = None) →
BenchmarkingResultBlock
```

Parse the output of a program and turn it into benchmarking results.

#### Parameters

- **exec\_res** – program output
- **res** – benchmarking result to which the extracted results should be added or None if they should be added

to an empty one :return: the modified benchmarking result block

```
setup_block(block: RunProgramBlock, cpuset: Optional[CPUSet] = None, set_id: int = 0)
```

Configure the passed copy of a run program block (e.g. the run command).

#### Parameters

- **block** – modified copy of a block
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarking takes place in

```
supports_parsing_out = True
```

Is the captured output on standard out useful for parsing

**class** temci.run.run\_driver.RunDriverRegistry

Bases: *AbstractRegistry*

The registry for run drivers.

The used run driver can be configured by editing the settings key *run/driver*. Possible run drivers are 'exec' and 'shell'

**default** = 'exec'

Name(s) of the class(es) used by default. Type depends on the *use\_list* property.

**plugin\_synonym** = ('run driver', 'run drivers')

Singular and plural version of the word that is used in the documentation for the registered entities

**classmethod** **register**(name: str, klass: type, misc\_type: Type, deprecated: bool = False)

Registers a new class. The constructor of the class gets as first argument the misc settings.

#### Parameters

- **name** – common name of the registered class
- **klass** – actual class
- **misc\_type** – type scheme of the {name}\_misc settings
- **misc\_default** – default value of the {name}\_misc settings
- **deprecated** – is the registered class deprecated and should not be used?

**registry** = {'exec': <class 'temci.run.run\_driver.ExecRunDriver'>, 'shell': <class 'temci.run.run\_driver.ShellRunDriver'>}

Registered classes (indexed by their name)

**settings\_key\_path** = 'run'

Used settings key path

**use\_key** = 'driver'

Used key that sets which registered class is currently used

**use\_list** = False

Allow more than one class to used at a specific moment in time

**class** temci.run.run\_driver.RunProgramBlock(id: int, data: Dict[str, Any], attributes: Dict[str, str], run\_driver\_class: Optional[type] = None)

Bases: object

An object that contains every needed information of a program block.

Creates an instance.

#### Parameters

- **data** – run driver configuration for this run program block
- **attributes** – attributes of this run program block
- **run\_driver\_class** – used type of run driver with this instance

#### attributes

Describing attributes of this run program block

**copy**() → *RunProgramBlock*

Copy this run program block. Deep copies the data and uses the same type scheme and attributes.

**data**

Run driver configuration

**description()** → str

**classmethod from\_dict**(*id: int, data: Dict, run\_driver: Optional[type] = None*)

Structure of data:

```
{
  "attributes": {"attr1": ..., ...},
  "run_config": {"prop1": ..., ...},
  "build_config": {"prop1": ..., ...}
}
```

**Parameters**

- **id** – id of the block (only used to track them later)
- **data** – used data
- **run\_driver** – used RunDriver subclass

**Returns**

new RunProgramBlock

**id**

Id of this run program block

**is\_enqueued**

Is this program block enqueued in a run worker pool queue?

**run\_driver\_class**

Used type of run driver

**to\_dict()** → Dict

Serializes this instance into a data structure that is accepted by the `from_dict` method.

**type\_scheme**

Configuration type scheme of the used run driver

**class** temci.run.run\_driver.**RusageExecRunner**(*block: RunProgramBlock*)

Bases: [ExecRunner](#)

Runner that uses the `getrusage(2)` function to obtain resource measurements.

To use this runner with name {name} either set the `runner` property of a run configuration or the setting under the key `run/exec_misc/runner` to its name.

The runner is configured by modifying the `rusage` property of a run configuration. This configuration has the following structure:

```
# Measured properties that are stored in the benchmarking result
properties:          ValidRusagePropertyList()
                      default: [idrss, inblock, isrss, ixrss, majflt, maxrss, minflt, msgrcv, ↵
↵msgsnd, nivcsw, nsignals,
                      nswap, nvcsw, oublock, stime, utime]
```

Creates an instance.

**Parameters**

**block** – run program block to measure

**Raises**

**KeyboardInterrupt** – if the runner can't be used (e.g. if the used tool isn't installed or compiled)

**get\_property\_descriptions()** → Dict[str, str]

Returns a dictionary that maps some properties to their short descriptions.

**misc\_options** = # Measured properties that are stored in the benchmarking result  
**properties:** ValidRusagePropertyList() default: [idrss, inblock, isrss, ixrss, majflt, maxrss, minflt, msgrcv, msgsnd, nivcs, nsignals, nswap, nvcs, oublock, stime, utime]

Type scheme of the options for this type of runner

**name** = 'rusage'

Name of the runner

**parse\_result\_impl**(*exec\_res*: ExecResult, *res*: Optional[BenchmarkingResultBlock] = None) → BenchmarkingResultBlock

Parse the output of a program and turn it into benchmarking results.

**Parameters**

- **exec\_res** – program output
- **res** – benchmarking result to which the extracted results should be added or None if they should be added

to an empty one :return: the modified benchmarking result block

**setup\_block**(*block*: RunProgramBlock, *cpuset*: Optional[CPUSet] = None, *set\_id*: int = 0)

Configure the passed copy of a run program block (e.g. the run command).

**Parameters**

- **block** – modified copy of a block
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarking takes place in

**class** temci.run.run\_driver.ShellRunDriver(*misc\_settings*: Optional[dict] = None)

Bases: ExecRunDriver

Implements a run driver that runs the benched command a single time with redirected in- and output. It can be used to run own benchmarking commands inside a sane benchmarking environment

The constructor calls the setup method.

Configuration format:

```
# Order in which the plugins are used, plugins that do not appear in this list are_
↳used before all
# others
plugin_order: [drop_fs_caches, sync, sleep, preheat, flush_cpu_caches]

# Enable other plugins by default: none = (enable none by default); all = cpu_
↳governor,disable_swap,s
# ync,stop_start,other_nice,nice,disable_aslr,disable_ht,cpuset,disable_turbo_boost_
↳(enable all, might
```

(continues on next page)

(continued from previous page)

```
# freeze your system); usable =
# cpu_governor,disable_swap,sync,nice,disable_aslr,disable_ht,cpuset,disable_turbo_
→boost (like 'all'
# but doesn't affect other processes)
preset: none
```

This run driver can be configured under the settings key *run/shell\_misc*.

To use this run driver set the currently used run driver (at key *run/driver*) to its name (*shell*). Another usable run driver is *exec*. The default run driver is *exec*.

Block configuration format for the run configuration:

```
# Optional build config to integrate the build step into the run step
build_config:          Either(Dict(, keys=Any, values=Any, default = {})|non_
→existent)

# Optional attributes that describe the block
attributes:
    description:          Optional(Str())

    # Tags of this block
    tags:                  ListOrTuple(Str())

run_config:
    # Execution directory
    cwd:                   Either(List(Str())|Str())
                           default: .

    # Command to run
    run_cmd:               Str()
                           default: sh

    # Environment variables
    env:                   Dict(, keys=Str(), values=Any, default = {})
```

Creates an instance. Also calls the setup methods on all registered plugins. It calls the *setup()* method.

#### Parameters

**misc\_settings** – further settings

**benchmark**(*block*: *RunProgramBlock*, *runs*: *int*, *cpuset*: *Optional[CPUSet] = None*, *set\_id*: *int = 0*, *timeout*: *float = -1*) → *BenchmarkingResultBlock*

Benchmark the passed program block “runs” times and return the benchmarking results.

#### Parameters

- **block** – run program block to benchmark
- **runs** – number of benchmarking runs
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarked block should be executed in
- **timeout** – timeout or -1 if no timeout is given

**Returns**

object that contains a dictionary of properties with associated raw run data

```
block_type_scheme = # Execution directory cwd: Either(List(Str())|Str()) default:
. # Command to run run_cmd: Str() default: sh # Environment variables env:
Dict(, keys=Str(), values=Any, default = {})
```

Type scheme for the program block configuration

```
runs_benchmarks = False
```

```
store_files = False
```

```
teardown()
```

Call the teardown() method on all used plugins for this driver.

```
class temci.run.run_driver.SpecExecRunner(block: RunProgramBlock)
```

Bases: *ExecRunner*

Runner for SPEC like single benchmarking suites. It works with resulting property files, in which the properties are colon separated from their values.

To use this runner with name {name} either set the runner property of a run configuration or the setting under the key run/exec\_misc/runner to its name.

The runner is configured by modifying the spec property of a run configuration. This configuration has the following structure:

```
# Base property path that all other paths are relative to.
base_path:          Str()

# Code that is executed for each matched path. The code should evaluate to the
↳ actual measured value
# for the path.it can use the function get(sub_path: str = "") and the modules
↳ pytimeparse, numpy,
# math, random, datetime and time.
code:               Str()
                  default: get()

# SPEC result file
file:              Str()

# Regexp matching the base property path for each measured property
path_regexp:       Str()
                  default: .*
```

Creates an instance.

**Parameters**

**block** – run program block to measure

**Raises**

**KeyboardInterrupt** – if the runner can't be used (e.g. if the used tool isn't installed or compiled)



```
misc_options = # Base property path that all other paths are relative to. base_path:
Str() # Code that is executed for each matched path. The code should evaluate to the
actual measured value # for the path.it can use the function get(sub_path: str =
'') and the modules datetime and time. code:
Str() default: get() # SPEC result file file: Str() # Regexp matching the base
property path for each measured property path_regexp: Str() default: .*
```

Type scheme of the options for this type of runner

```
name = 'spec'
```

Name of the runner

```
parse_result_impl(exec_res: ExecResult, res: Optional[BenchmarkingResultBlock] = None) →
    BenchmarkingResultBlock
```

Parse the output of a program and turn it into benchmarking results.

#### Parameters

- **exec\_res** – program output
- **res** – benchmarking result to which the extracted results should be added or `None` if they should be added

to an empty one :return: the modified benchmarking result block

```
setup_block(block: RunProgramBlock, cpuset: Optional[CPUSet] = None, set_id: int = 0)
```

Configure the passed copy of a run program block (e.g. the run command).

#### Parameters

- **block** – modified copy of a block
- **cpuset** – used `CPUSet` instance
- **set\_id** – id of the cpu set the benchmarking takes place in

```
class temci.run.run_driver.TimeExecRunner(block: RunProgramBlock)
```

Bases: `ExecRunner`

Uses the GNU `time` tool and is mostly equivalent to the `rusage` runner but more user friendly.

To use this runner with name {name} either set the `runner` property of a run configuration or the setting under the key `run/exec_misc/runner` to its name.

This runner supports the `parse_output` option.

The runner is configured by modifying the `time` property of a run configuration. This configuration has the following structure:

```
# Measured properties that are included in the benchmarking results
properties: ValidTimePropertyList()
            default: [utime, stime, etime, avg_mem_usage, max_res_set, avg_res_set]
```

Creates an instance.

#### Parameters

**block** – run program block to measure

#### Raises

**KeyboardInterrupt** – if the runner can't be used (e.g. if the used tool isn't installed or compiled)

**get\_property\_descriptions()** → Dict[str, str]

Returns a dictionary that maps some properties to their short descriptions.

**misc\_options** = # Measured properties that are included in the benchmarking results  
**properties:** ValidTimePropertyList() default: [utime, stime, etime, avg\_mem\_usage, max\_res\_set, avg\_res\_set]

Type scheme of the options for this type of runner

**name** = 'time'

Name of the runner

**parse\_result\_impl**(exec\_res: ExecResult, res: Optional[BenchmarkingResultBlock] = None) → BenchmarkingResultBlock

Parse the output of a program and turn it into benchmarking results.

#### Parameters

- **exec\_res** – program output
- **res** – benchmarking result to which the extracted results should be added or None if they should be added

to an empty one :return: the modified benchmarking result block

**setup\_block**(block: RunProgramBlock, cpuset: Optional[CPUSet] = None, set\_id: int = 0)

Configure the passed copy of a run program block (e.g. the run command).

#### Parameters

- **block** – modified copy of a block
- **cpuset** – used CPUSet instance
- **set\_id** – id of the cpu set the benchmarking takes place in

**supports\_parsing\_out** = True

Is the captured output on standard out useful for parsing

**exception** temci.run.run\_driver.TimeoutException(cmd: str, timeout: float, out: str, err: str, ret\_code: int)

Bases: BenchmarkingProgramError

Thrown whenever a benchmarked program timeouts

**class** temci.run.run\_driver.ValidPerfStatPropertyList

Bases: Type

Checks for the value to be a valid perf stat measurement property list or the perf tool to be missing.

Creates an instance.

#### Parameters

**completion\_hints** – completion hints for supported shells for this type instance

**class** temci.run.run\_driver.ValidPropertyList(av\_properties: Iterable[str])

Bases: Type

Checks for the value to be a valid property list that contains only elements from a given list.

Creates an instance.

#### Parameters

**av\_properties** – allowed list elements

**av**

Allowed list elements

**class** temci.run.run\_driver.ValidRusagePropertyList

Bases: [ValidPropertyList](#)

Checks for the value to be a valid rusage runner measurement property list.

Creates an instance.

#### Parameters

**av\_properties** – allowed list elements

**class** temci.run.run\_driver.ValidTimePropertyList

Bases: [ValidPropertyList](#)

Checks for the value to be a valid time runner measurement property list.

Creates an instance.

#### Parameters

**av\_properties** – allowed list elements

temci.run.run\_driver.clean\_output(output: str) → str

Remove everything after the header

temci.run.run\_driver.filter\_runs(blocks: List[Union[RunProgramBlock, RunData]], included: List[str])  
→ List[RunProgramBlock]

Filter run blocks (all: include all), identified by their description or tag or their number in the file (starting at zero) and run datas (only identified by their description and tag). The include query can also consist of regular expressions

#### Parameters

- **blocks** – blocks or run datas to filter
- **included** – include query

#### Returns

filtered list

temci.run.run\_driver.get\_av\_perf\_stat\_properties() → List[str]

Returns the list of properties that are measurable with the used perf stat tool.

temci.run.run\_driver.get\_av\_rusage\_properties() → Dict[str, str]

Returns the available properties for the RusageExecRunner mapped to their descriptions.

temci.run.run\_driver.get\_av\_time\_properties() → Dict[str, str]

Returns the available properties for the TimeExecRunner mapped to their descriptions.

temci.run.run\_driver.get\_av\_time\_properties\_with\_format\_specifiers() → Dict[str, Tuple[str, str]]

Returns the available properties for the TimeExecRunner mapped to their descriptions and time format specifiers.

temci.run.run\_driver.header() → str

A header to use for measurement formatting

temci.run.run\_driver.is\_perf\_available() → bool

Is the perf tool available?

temci.run.run\_driver.log\_program\_error(recorded\_error: RecordedInternalError)

`temci.run.run_driver.time_file(_tmp=[]) → str`

Returns the command used to execute the (GNU) `time` tool (not the built in shell tool).

## temci.run.run\_driver\_plugin module

This module consists of run driver plugin implementations.

**class** `temci.run.run_driver_plugin.AbstractRunDriverPlugin(misc_settings)`

Bases: `object`

A plugin for a run driver. It allows additional modifications. The object is instantiated before the benchmarking starts and used for the whole benchmarking runs.

Creates an instance.

### Parameters

**`misc_settings`** – configuration of this plugin

**`needs_root_privileges`** = `False`

Does this plugin work only with root privileges?

**`setup()`**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**`setup_block(block: RunProgramBlock, runs: int = 1)`**

Called before each run program block is run “runs” time.

### Parameters

- **`block`** – run program block to modify
- **`runs`** – number of times the program block is run (and measured) at once.

**`setup_block_run(block: RunProgramBlock)`**

Called before each run program block is run.

### Parameters

**`block`** – run program block to modify

**`teardown()`**

Called after the whole benchmarking is finished.

**`teardown_block(block: RunProgramBlock)`**

Called after each run program block is run.

### Parameters

**`block`** – run program block

**class** `temci.run.run_driver_plugin.CPUGovernor(misc_settings)`

Bases: `AbstractRunDriverPlugin`

Allows the setting of the scaling governor of all cpu cores, to ensure that all use the same.

Configuration format:

```
# New scaling governor for all cpus
governor: performance
```

This run driver plugin can be configured under the settings key `run/exec_plugins/cpu_governor_misc`.

To use this run driver plugin add its name (`cpu_governor`) to the list at settings key `run/exec_plugins/exec_active` or set `run/exec_plugins/cpu_governor_active` to true. Other usable run driver plugins are `nice`, `env_randomize`, `preheat`, `other_nice`, `stop_start`, `sync`, `sleep`, `drop_fs_caches`, `disable_swap`, `disable_cpu_caches` and `flush_cpu_caches`.

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

#### setup()

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

#### teardown()

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.CPUSet(*misc\_settings*)

Bases: [AbstractRunDriverPlugin](#)

Enable cpusets, simply sets run/cpuset/active to true

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

#### setup()

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**class** temci.run.run\_driver\_plugin.DisableASLR(*misc\_settings*)

Bases: [AbstractRunDriverPlugin](#)

Disable address space randomization

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

#### setup()

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

#### teardown()

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.DisableAmdTurbo(*misc\_settings*)

Bases: [DisableTurboBoost](#)

Disable amd turbo boost

Creates an instance.

**Parameters**

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**class** temci.run.run\_driver\_plugin.**DisableCPUCaches**(*misc\_settings*)

Bases: [\*AbstractRunDriverPlugin\*](#)

Disable the L1 and L2 caches on x86 and x86-64 architectures. Uses a small custom kernel module (be sure to compile it via ‘temci setup –build\_kernel\_modules’).

**Warning**

slows program down significantly and has probably other weird consequences

**Warning**

this is untested

**Warning**

a linux-forum user declared: Disabling cpu caches gives you a pentium I like processor!!!

Creates an instance.

**Parameters**

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**teardown()**

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.**DisableHyperThreading**(*misc\_settings*)

Bases: [\*AbstractRunDriverPlugin\*](#)

Disable hyper-threading

Creates an instance.

**Parameters**

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**teardown()**

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.**DisableIntelTurbo**(*misc\_settings*)

Bases: [\*DisableTurboBoost\*](#)

Disable intel turbo mode

Creates an instance.

**Parameters****misc\_settings** – configuration of this plugin**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**class** temci.run.run\_driver\_plugin.**DisableSwap**(misc\_settings)Bases: [AbstractRunDriverPlugin](#)

Disables swapping on the system before the benchmarking and enables it after.

Creates an instance.

**Parameters****misc\_settings** – configuration of this plugin**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**teardown()**

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.**DisableTurboBoost**(misc\_settings)Bases: [AbstractRunDriverPlugin](#)

Disable amd and intel turbo boost

Creates an instance.

**Parameters****misc\_settings** – configuration of this plugin

```
CPU_PATHS = {'amd': ('/sys/devices/system/cpu/cpufreq/boost', <class 'int'>),
'intel': ('/sys/devices/system/cpu/intel_pstate/no_turbo', <function
DisableTurboBoost.<lambda>>)}

```

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**teardown()**

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.**DiscardedRuns**(misc\_settings)Bases: [AbstractRunDriverPlugin](#)

Sets run/discarded\_runs

Configuration format:

```
# Number of discarded runs
runs: 1

```

This run driver plugin can be configured under the settings key `run/exec_plugins/discarded_runs_misc`.

To use this run driver plugin add its name (`discarded_runs`) to the list at settings key `run/exec_plugins/exec_active` or set `run/exec_plugins/discarded_runs_active` to true. Other usable run driver plugins are `nice`, `env_randomize`, `preheat`, `other_nice`, `stop_start`, `sync`, `sleep`, `drop_fs_caches`, `disable_swap`, `disable_cpu_caches`, `flush_cpu_caches`, `cpu_governor`, `disable_aslr`, `disable_ht`, `disable_turbo_boost`, `disable_intel_turbo`, `disable_amd_boost` and `cpuset`.

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

#### setup()

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**class** temci.run.run\_driver\_plugin.DropFSCaches(*misc\_settings*)

Bases: `AbstractRunDriverPlugin`

Drop page cache, directoy entries and inodes before every benchmarking run.

Configuration format:

```
# Free dentries and inodes
free_dentries_inodes: true

# Free the page cache
free_pagecache: true
```

This run driver plugin can be configured under the settings key `run/exec_plugins/drop_fs_caches_misc`.

To use this run driver plugin add its name (`drop_fs_caches`) to the list at settings key `run/exec_plugins/exec_active` or set `run/exec_plugins/drop_fs_caches_active` to true. Other usable run driver plugins are `nice`, `env_randomize`, `preheat`, `other_nice`, `stop_start`, `sync` and `sleep`.

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

**setup\_block**(*block*: `RunProgramBlock`, *runs*: `int = 1`)

Called before each run program block is run “runs” time.

#### Parameters

- **block** – run program block to modify
- **runs** – number of times the program block is run (and measured) at once.

**class** temci.run.run\_driver\_plugin.EnvRandomizePlugin(*misc\_settings*)

Bases: `AbstractRunDriverPlugin`

Adds random environment variables.

Configuration format:



```
# Maximum length of each random key
key_max: 4096

# Maximum number of added random environment variables
max: 4

# Minimum number of added random environment variables
min: 4

# Maximum length of each random value
var_max: 4096
```

This run driver plugin can be configured under the settings key `run/exec_plugins/env_randomize_misc`.

To use this run driver plugin add its name (`env_randomize`) to the list at settings key `run/exec_plugins/exec_active` or set `run/exec_plugins/env_randomize_active` to true. Another usable run driver plugin is `nice`.

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**setup\_block\_run**(*block*: `RunProgramBlock`, *runs*: `int = 1`)

Called before each run program block is run.

#### Parameters

**block** – run program block to modify

**class** `temci.run.run_driver_plugin.FlushCPUCaches`(*misc\_settings*)

Bases: `AbstractRunDriverPlugin`

Flushes the CPU caches on a x86 CPU using a small kernel module, see `WBINVD`

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges** = `True`

Does this plugin work only with root privileges?

**setup\_block\_run**(*block*: `RunProgramBlock`)

Called before each run program block is run.

#### Parameters

**block** – run program block to modify

**class** `temci.run.run_driver_plugin.NicePlugin`(*misc\_settings*)

Bases: `AbstractRunDriverPlugin`

Allows the setting of the nice and ionice values of the benchmarking process.

Configuration format:

```
# Specify the name or number of the scheduling class to use;0 for none, 1 for
↪realtime, 2 for best-
# effort, 3 for idle.
io_nice: 1
```

(continues on next page)

(continued from previous page)

```
# Niceness values range from -20 (most favorable to the process) to 19 (least
↪favorable to the
# process).
nice: -15
```

This run driver plugin can be configured under the settings key `run/exec_plugins/nice_misc`.

To use this run driver plugin add its name (*nice*) to the list at settings key `run/exec_plugins/exec_active` or set `run/exec_plugins/nice_active` to true.

Creates an instance.

**Parameters**

**misc\_settings** – configuration of this plugin

**needs\_root\_privileges = True**

Does this plugin work only with root privileges?

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**teardown()**

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.OtherNicePlugin(*misc\_settings*)

Bases: *AbstractRunDriverPlugin*

Allows the setting of the nice value of most other processes (as far as possible).

Configuration format:

```
# Processes with lower nice values are ignored.
min_nice: -10

# Niceness values for other processes.
nice: 19
```

This run driver plugin can be configured under the settings key `run/exec_plugins/other_nice_misc`.

To use this run driver plugin add its name (*other\_nice*) to the list at settings key `run/exec_plugins/exec_active` or set `run/exec_plugins/other_nice_active` to true. Other usable run driver plugins are *nice*, *env\_randomize* and *preheat*.

Creates an instance.

**Parameters**

**misc\_settings** – configuration of this plugin

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**teardown()**

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.PreheatPlugin(*misc\_settings*)

Bases: *AbstractRunDriverPlugin*

Preheats the system with a cpu bound task (calculating the inverse of a big random matrix with numpy).

Configuration format:

```
# Number of seconds to preheat the system with an cpu bound task
time: 10

# When to preheat
when: [before_each_run]
```

This run driver plugin can be configured under the settings key `run/exec_plugins/preheat_misc`.

To use this run driver plugin add its name (*preheat*) to the list at settings key `run/exec_plugins/exec_active` or set `run/exec_plugins/preheat_active` to true. Other usable run driver plugins are *nice* and *env\_randomize*.

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

#### setup()

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**setup\_block**(*block*: [RunProgramBlock](#), *runs*: int = 1)

Called before each run program block is run “runs” time.

#### Parameters

- **block** – run program block to modify
- **runs** – number of times the program block is run (and measured) at once.

**class** temci.run.run\_driver\_plugin.SleepPlugin(*misc\_settings*)

Bases: [AbstractRunDriverPlugin](#)

Sleep a given amount of time before the benchmarking begins.

See Gernot Heisers Systems Benchmarking Crimes: Make sure that the system is really quiescent when starting an experiment, leave enough time to ensure all previous data is flushed out.

Configuration format:

```
# Seconds to sleep
seconds: 10
```

This run driver plugin can be configured under the settings key `run/exec_plugins/sleep_misc`.

To use this run driver plugin add its name (*sleep*) to the list at settings key `run/exec_plugins/exec_active` or set `run/exec_plugins/sleep_active` to true. Other usable run driver plugins are *nice*, *env\_randomize*, *preheat*, *other\_nice*, *stop\_start* and *sync*.

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**setup\_block**(*block*: [RunProgramBlock](#), *runs*: int = 1)

Called before each run program block is run “runs” time.

#### Parameters

- **block** – run program block to modify
- **runs** – number of times the program block is run (and measured) at once.

**class** temci.run.run\_driver\_plugin.StopStartPlugin(misc\_settings)

Bases: *AbstractRunDriverPlugin*

Stop almost all other processes (as far as possible).

Configuration format:

```
# Each process which name (lower cased) starts with one of the prefixes is not_
↳ ignored. Overrides the
# decision based on the min_id.
comm_prefixes: [ssh, xorg, bluetoothd]

# Each process which name (lower cased) starts with one of the prefixes is ignored.
↳ It overrides the
# decisions based on comm_prefixes and min_id.
comm_prefixes_ignored: [dbus, kworker]

# Just output the to be stopped processes but don't actually stop them?
dry_run: false

# Processes with lower id are ignored.
min_id: 1500

# Processes with lower nice values are ignored.
min_nice: -10

# Suffixes of processes names which are stopped.
subtree_suffixes: [dm, apache]
```

This run driver plugin can be configured under the settings key *run/exec\_plugins/stop\_start\_misc*.

To use this run driver plugin add its name (*stop\_start*) to the list at settings key *run/exec\_plugins/exec\_active* or set *run/exec\_plugins/stop\_start\_active* to true. Other usable run driver plugins are *nice*, *env\_randomize*, *preheat* and *other\_nice*.

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**parse\_processes()**

**setup()**

Called before the whole benchmarking starts (e.g. to set the “nice” value of the benchmarking process).

**teardown()**

Called after the whole benchmarking is finished.

**class** temci.run.run\_driver\_plugin.SyncPlugin(misc\_settings)

Bases: *AbstractRunDriverPlugin*

Calls sync before each program execution.

Creates an instance.

#### Parameters

**misc\_settings** – configuration of this plugin

**setup\_block\_run**(*block*: [RunProgramBlock](#), *runs*: *int* = 1)

Called before each run program block is run.

#### Parameters

**block** – run program block to modify

### temci.run.run\_processor module

**class** temci.run.run\_processor.**RunProcessor**(*runs*: *Optional*[*List*[*dict*]] = *None*, *append*: *Optional*[*bool*] = *None*, *show\_report*: *Optional*[*bool*] = *None*)

Bases: object

This class handles the coordination of the whole benchmarking process. It is configured by setting the settings of the stats and run domain.

Important note: the constructor also setups the cpu sets and plugins that can alter the system, e.g. confine most processes on only one core. Be sure to call the `teardown()` or the `benchmark()` method to make the system usable again.

Creates an instance and setup everything.

#### Parameters

- **runs** – list of dictionaries that represent run program blocks if `None` `Settings()["run/in"]` is used
- **append** – append to the old benchmarks if there are any in the result file?
- **show\_report** – show a short report after finishing the benchmarking?

#### append

Append to the old benchmarks if there are any in the result file?

#### benchmark()

Benchmark and teardown.

#### block\_run\_count

Number of benchmarked blocks

#### build()

Build before benchmarking, essentially calls *temci build* where necessary and modifies the run configs

#### discarded\_runs

First n runs that are discarded

#### end\_time

Unix time stamp of the point in time that the benchmarking can at most reach

#### erroneous\_run\_blocks

List of all failing run blocks (id and results till failing)

#### fixed\_runs

Do a fixed number of benchmarking runs?

#### max\_runs

Maximum number of benchmarking runs

**maximum\_of\_max\_runs**() → int

**maximum\_of\_min\_runs()** → int

**min\_runs**

Minimum number of benchmarking runs

**pool**

Used run worker pool that abstracts the benchmarking

**print\_report()** → str

**recorded\_error()** → bool

**run\_block\_size**

Number of benchmarking runs that are done together

**run\_blocks**

Run program blocks for each dictionary in `runs``

**runs**

List of dictionaries that represent run program blocks

**show\_report**

Show a short report after finishing the benchmarking?

**shuffle**

Randomize the order in which the program blocks are benchmarked.

**start\_time**

Unix time stamp of the start of the benchmarking

**stats\_helper**

Used stats helper to help with measurements

**store()**

Store the result file

**store\_and\_teardown()**

Teardown everything, store the result file, print a short report and send an email if configured to do so.

**store\_erroneous()**

Store the failing program blocks in a file ending with `.erroneous.yaml`.

**store\_often**

Store the result file after each set of blocks is benchmarked

**teardown()**

Teardown everything (make the system useable again)

## temci.run.run\_worker\_pool module

This module consists of the abstract run worker pool class and several implementations.

```
class temci.run.run_worker_pool.AbstractRunWorkerPool(run_driver_name: Optional[str] = None,  
                                                    end_time: float = - 1)
```

Bases: object

An abstract run worker pool that just deals with the hyper threading setting.

Create an instance.

**Parameters**

**run\_driver\_name** – name of the used run driver, if None the one configured in the settings is used

**cpuset**

Used cpu set instance

**classmethod disable\_hyper\_threading()** → List[int]

**classmethod enable\_hyper\_threading**(*disabled\_cores: List[int]*)

**classmethod get\_hyper\_threading\_cores()** → List[int]

Adapted from <http://unix.stackexchange.com/a/223322>

**has\_time\_left()** → bool

**next\_block\_timeout()** → float

**parallel\_number**

Number of instances in which the benchmarks takes place in parallel

**result\_queue**

Queue of benchmarking results. The queue items are tuples consisting of the benchmarked block, the benchmarking result and the blocks id.

**results**(*expected\_num: int*) → Iterator[Tuple[RunProgramBlock, BenchmarkingResultBlock, int]]

A generator for all available benchmarking results. The items of this generator are tuples consisting of the benchmarked block, the benchmarking result and the blocks id.

**Parameters**

**expected\_num** – expected number of results

**run\_driver**

Used run driver instance

**submit**(*block: RunProgramBlock, id: int, runs: int*)

Submits the passed block for “runs” times benchmarking. It also sets the blocks `is_enqueued` property to True.

**Parameters**

- **block** – passed run program block
- **id** – id of the passed block
- **runs** – number of individual benchmarking runs

**submit\_queue**

Queue for submitted but not benchmarked run program blocks

**teardown()**

Tears down the inherited run driver. This should be called if all benchmarking with this pool is finished.

**time\_left()** → float

Does not work properly if `self.end_time == -1`

**class** temci.run.run\_worker\_pool.**BenchmarkingThread**(*id: int, pool: ParallelRunWorkerPool, driver: AbstractRunDriver, cpuset: CPUSet*)

Bases: `Thread`

A thread that allows parallel benchmarking.

Creates an instance.

**Parameters**

- **id** – id of this thread
- **pool** – parent run worked pool
- **driver** – use run driver instance
- **cpuset** – used CPUSet instance

**cpuset**

Used CPUSet instance

**driver**

Used run driver instance

**id**

Id of this thread

**pool**

Parent run worker pool

**run()**

Start the run loop. It fetches run program blocks from the pool's submit queue, benchmarks them and stores the results in the pool's result queue. It stops if `stop` is true.

**stop**

Stop the run loop?

```
class temci.run.run_worker_pool.ParallelRunWorkerPool(run_driver_name: Optional[str] = None,
                                                       end_time: float = - 1)
```

Bases: `AbstractRunWorkerPool`

This run worker pool implements the parallel benchmarking of program blocks. It uses a server-client-model to benchmark on different cpu cores.

Create an instance.

**Parameters**

**run\_driver\_name** – name of the used run driver, if None the one configured in the settings is used

**results**(*expected\_num: int*) → `Iterator[Tuple[RunProgramBlock, BenchmarkingResultBlock, int]]`

A generator for all available benchmarking results. The items of this generator are tuples consisting of the benchmarked block, the benchmarking result and the blocks id.

**Parameters**

**expected\_num** – expected number of results

**submit**(*block: RunProgramBlock, id: int, runs: int*)

Submits the passed block for “runs” times benchmarking. It also sets the blocks `is_enqueued` property to True.

**Parameters**

- **block** – passed run program block



- **id** – id of the passed block
- **runs** – number of individual benchmarking runs

**teardown()**

Tears down the inherited run driver. This should be called if all benchmarking with this pool is finished.

**threads**

Running benchmarking threads

`temci.run.run_worker_pool.ResultGenerator`

Return type of the run worker pool `results` method

alias of `Iterator[Tuple[RunProgramBlock, BenchmarkingResultBlock, int]]`

**class** `temci.run.run_worker_pool.RunWorkerPool`(*run\_driver\_name: Optional[str] = None, end\_time: float = - 1*)

Bases: `AbstractRunWorkerPool`

This run worker pool implements the sequential benchmarking of program blocks.

Create an instance.

**Parameters**

**run\_driver\_name** – name of the used run driver, if None the one configured in the settings is used

**results**(*expected\_num: int*) → `Iterator[Tuple[RunProgramBlock, BenchmarkingResultBlock, int]]`

A generator for all available benchmarking results. The items of this generator are tuples consisting of the benchmarked block, the benchmarking result and the blocks id.

**Parameters**

**expected\_num** – expected number of results

**submit**(*block: RunProgramBlock, id: int, runs: int*)

Submits the passed block for “runs” times benchmarking. It also sets the blocks `is_enqueued` property to True.

**Parameters**

- **block** – passed run program block
- **id** – id of the passed block
- **runs** – number of individual benchmarking runs

**teardown()**

Tears down the inherited run driver. This should be called if all benchmarking with this pool is finished.

## Module contents

This module contains code to make the actual benchmarks.

## temci.scripts package

### Submodules

#### temci.scripts.cli module

**class** temci.scripts.cli.**ErrorCode**(*value*)

Bases: Enum

An enumeration.

**NO\_ERROR** = 0

**PROGRAM\_ERROR** = 1

**TEMCI\_ERROR** = 255

temci.scripts.cli.**benchmark\_and\_exit**(*runs: Optional[List[dict]] = None*)

Benchmark and exit with an exit code if an error occurred: 0 if everything went okay, 1 if at least one benchmarked program failed, 255 if temci itself failed

temci.scripts.cli.**cli\_with\_error\_catching**()

Process the command line arguments and catch (some) errors.

temci.scripts.cli.**cli\_with\_verb\_arg\_handling**()

Handles ` - ` properly and calls the cli

temci.scripts.cli.**create\_run\_driver\_function**(*driver: str, options: CmdOptionList*)

temci.scripts.cli.**temci\_\_build**(*build\_file: str, \*\*kwargs*)

**Language**

python

**Command**

temci build

**Description**

Build programs and output modified run config file

**Argument**

build configuration YAML file

**Options:**

--settings:

**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

--config:

**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

**--tmp\_dir:****Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

**--log\_level:****Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

**--sudo|--no-sudo:****Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

**--in:****Description**

Input file with the program blocks to build

**Argument type**

Str()

**Default**

'build.yaml'

**--out:****Description**

Resulting run config file

**Argument type**

Str()

**Default**

'run\_config.yaml'

**--threads:****Description**

Number of simultaneous builds for a specific program block, only makes sense when build\_config/number &gt; 1, and if the build commands create a different binary every time they are run

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

`temci.scripts.cli.temci__clean(**kwargs)`**Language**

python

**Command**

temci clean

**Description**

Clean up the temporary files

**Options:**`--settings:`**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

`--config:`**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

`--tmp_dir:`**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

`--log_level:`**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

`--sudo | --no-sudo:`

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

`temci.scripts.cli.temci__completion()`

Creates completion files for several shells.

`temci.scripts.cli.temci__completion__bash()`

**Language**

python

**Command**

temci completion bash

**Description**

Creates a new tab completion file for zsh and returns its file name

**Options:**

`--settings:`

**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

`--config:`

**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

`--tmp_dir:`

**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

`--log_level:`

**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

--sudo|--no-sudo:

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

temci.scripts.cli.temci\_\_completion\_\_zsh()

**Language**

python

**Command**

temci completion zsh

**Description**

Creates a new tab completion file for zsh and returns its file name

**Options:**

--settings:

**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

--config:

**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

--tmp\_dir:

**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci '

--log\_level:

**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

--sudo|--no-sudo:

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

```
temci.scripts.cli.temci__exec(run_file, **kwargs)
```

**Language**

python

**Command**

temci exec

**Description**

Implements a simple run driver that just executes one of the passed run\_cmds

**Argument**

configuration YAML file

**Options:**

--settings:

**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

--config:

**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

--tmp\_dir:

**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci '

--log\_level:

**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

--sudo|--no-sudo:

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

--discarded\_runs:

**Description**

First n runs that are discarded

**Argument type**

Int(constraint=<function>)

**Default**

1

--min\_runs:

**Description**

Minimum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

20

--max\_runs:

**Description**

Maximum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

100

--runs:

**Description**

if != -1 sets max and min runs to its value

**Argument type**

Int(constraint=<function>)

**Default**

-1

--max\_time:

**Description**

Maximum time the whole benchmarking should take, -1 == no timeout, supports normal time span expressions



**Argument type**  
ValidTimespan()

**Default**  
'-1'

--max\_block\_time:

**Description**  
Maximum time one run block should take, -1 == no timeout, supports normal time span expressions

**Argument type**  
ValidTimespan()

**Default**  
'-1'

--run\_block\_size:

**Description**  
Number of benchmarking runs that are done together

**Argument type**  
Int(constraint=<function>)

**Default**  
1

--in:

**Description**  
Input file with the program blocks to benchmark

**Argument type**  
Str()

**Default**  
'input.exec.yaml'

--out:

**Description**  
Output file for the benchmarking results

**Argument type**  
Str()

**Default**  
'run\_output.yaml'

--store\_often|--no-store\_often:

**Description**  
Store the result file after each set of blocks is benchmarked

**Default**  
False

--included\_blocks:

**Description**  
List of included run blocks (all: include all), or their tag attribute or their number in the file (starting with 0), can be regular expressions

**Argument type**  
ListOrTuple(Str())

**Default**  
['all']

--disable\_hyper\_threading|--no-disable\_hyper\_threading:

**Description**  
Disable the hyper threaded cores. Good for cpu bound programs.

**Default**  
False

--show\_report|--no-show\_report:

**Description**  
Print console report if log\_level=info

**Default**  
True

--append|--no-append:

**Description**  
Append to the output file instead of overwriting by adding new run data blocks

**Default**  
False

--shuffle|--no-shuffle:

**Description**  
Randomize the order in which the program blocks are benchmarked.

**Default**  
True

--send\_mail:

**Description**  
If not empty, recipient of a mail after the benchmarking finished.

**Argument type**  
Str()

**Default**  
' '

--discard\_all\_data\_for\_block\_on\_error|--no-discard\_all\_data\_for\_block\_on\_error:

**Description**  
Discard all run data for the failing program on error

**Default**  
False

--record\_errors\_in\_file|--no-record\_errors\_in\_file:

**Description**  
Record the caught errors in the run\_output file

**Default**  
True

--no\_build|--no-no\_build:

**Description**

Do not build if build configs are present, only works if the working directory of the blocks does not change

**Default**

False

--only\_build|--no-only\_build:

**Description**

Only build

**Default**

False

--abort\_after\_build\_error|--no-abort\_after\_build\_error:

**Description**

Abort after the first failing build

**Default**

True

--watch|--no-watch:

**Description**

Show the report continuously

**Default**

False

--watch\_every:

**Description**

Update the screen nth run (less updates are better for benchmarks)

**Argument type**

Int(constraint=<function>)

**Default**

1

--driver:

**Description**

Possible run drivers are 'exec' and 'shell'

**Argument type**

ExactEither('exec'|'shell')

**Default**

'exec'

--stats\_properties:

**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

`--stats_uncertainty_range:`

**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(<class 'float'>):<function>, T(<class 'float'>):<function>)

**Default**

[0.05, 0.15]

`--stats_tester:`

**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

`--report_number_parentheses|--no-report_number_parentheses:`

**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

`--report_number_min_decimal_places:`

**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=<function>)

**Default**

3

`--report_number_max_decimal_places:`

**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=<function>)

**Default**

5

`--report_number_scientific_notation|--no-report_number_scientific_notation:`

**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

`--report_number_scientific_notation_si_prefixes|--no-report_number_scientific_notation_si_prefixes:`

**Description**

Use si prefixes instead of 'e...'

**Default**

True

`--report_number_omit_insignificant_decimal_places|--no-report_number_omit_insignificant_decimal_places:`**Description**

Omit insignificant decimal places

**Default**

False

`--report_number_force_min_decimal_places|--no-report_number_force_min_decimal_places:`**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

`--report_number_percentages|--no-report_number_percentages:`**Description**

Show as percentages

**Default**

False

`--report_number_sigmas:`**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

2

`--report_number_parentheses_mode:`**Description**

Mode for showing the parentheses: either d (Digits are considered significant if they don't change if the number itself changes  $\pm \$sigmas * \text{std dev}$ ) or o (digits are considered significant if they are bigger than  $\$sigmas * \text{std dev}$ )

**Argument type**

ExactEither('d'|'o')

**Default**

'o'

`--nice|--no-nice:`**Description**

Enable: Allows the setting of the nice and ionice values of the benchmarking process.

**Default**

None

`--nice_nice:`**Description**

Niceness values range from -20 (most favorable to the process) to 19 (least favorable to the process).

**Argument type**

Int(range=range(-20, 20))

**Default**

-15

--nice\_io\_nice:

**Description**

Specify the name or number of the scheduling class to use;0 for none, 1 for real-time, 2 for best-effort, 3 for idle.

**Argument type**

Int(range=range(0, 4))

**Default**

1

--env\_randomize|--no-env\_randomize:

**Description**

Enable: Adds random environment variables.

**Default**

None

--env\_randomize\_min:

**Description**

Minimum number of added random environment variables

**Argument type**

Int(constraint=<function>)

**Default**

4

--env\_randomize\_max:

**Description**

Maximum number of added random environment variables

**Argument type**

Int(constraint=<function>)

**Default**

4

--env\_randomize\_var\_max:

**Description**

Maximum length of each random value

**Argument type**

Int(constraint=<function>)

**Default**

4096

--env\_randomize\_key\_max:

**Description**

Maximum length of each random key

**Argument type**

Int(constraint=<function>)

**Default**

4096

--preheat|--no-preheat:

**Description**

Enable: Preheats the system with a cpu bound task

**Default**

None

--preheat\_time:

**Description**

Number of seconds to preheat the system with an cpu bound task

**Argument type**

Int(constraint=<function>)

**Default**

10

--preheat\_when:

**Description**

When to preheat

**Argument type**

ListOrTuple(ExactEither('before\_each\_run','at\_setup'))

**Default**

['before\_each\_run']

--other\_nice|--no-other\_nice:

**Description**

Enable: Allows the setting of the nice value of most other processes (as far as possible).

**Default**

None

--other\_nice\_nice:

**Description**

Niceness values for other processes.

**Argument type**

Int(range=range(-20, 20))

**Default**

19

--other\_nice\_min\_nice:

**Description**

Processes with lower nice values are ignored.

**Argument type**

Int(range=range(-15, 20))

**Default**

-10

--stop\_start|--no-stop\_start:

**Description**

Enable: Stop almost all other processes (as far as possible).

**Default**

None

--stop\_start\_min\_nice:

**Description**

Processes with lower nice values are ignored.

**Argument type**

Int(range=range(-15, 20))

**Default**

-10

--stop\_start\_min\_id:

**Description**

Processes with lower id are ignored.

**Argument type**

Int(constraint=<function>)

**Default**

1500

--stop\_start\_comm\_prefixes:

**Description**

Each process which name (lower cased) starts with one of the prefixes is not ignored. Overrides the decision based on the min\_id.

**Argument type**

ListOrTuple(Str())

**Default**

['ssh', 'xorg', 'bluetoothd']

--stop\_start\_comm\_prefixes\_ignored:

**Description**

Each process which name (lower cased) starts with one of the prefixes is ignored. It overrides the decisions based on comm\_prefixes and min\_id.

**Argument type**

ListOrTuple(Str())

**Default**

['dbus', 'kworker']

--stop\_start\_subtree\_suffixes:

**Description**

Suffixes of processes names which are stopped.

**Argument type**

ListOrTuple(Str())



**Default**

['dm', 'apache']

`--stop_start_dry_run|--no-stop_start_dry_run:`**Description**

Just output the to be stopped processes but don't actually stop them?

**Default**

False

`--sync|--no-sync:`**Description**

Enable: Calls sync before each program execution.

**Default**

None

`--sleep|--no-sleep:`**Description**

Enable: Sleep a given amount of time before the benchmarking begins.

**Default**

None

`--sleep_seconds:`**Description**

Seconds to sleep

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

10

`--drop_fs_caches|--no-drop_fs_caches:`**Description**

Enable: Drop page cache, directoy entries and inodes before every benchmarking run.

**Default**

None

`--drop_fs_caches_free_pagecache|--no-drop_fs_caches_free_pagecache:`**Description**

Free the page cache

**Default**

True

`--drop_fs_caches_free_dentries_inodes|--no-drop_fs_caches_free_dentries_inodes:`**Description**

Free dentries and inodes

**Default**

True

`--disable_swap|--no-disable_swap:`

**Description**

Enable: Disables swapping on the system before the benchmarking and enables it after.

**Default**

None

--disable\_cpu\_caches|--no-disable\_cpu\_caches:

**Description**

Enable: Disable the L1 and L2 caches on x86 and x86-64 architectures.

**Default**

None

--flush\_cpu\_caches|--no-flush\_cpu\_caches:

**Description**

Enable: Flushes the CPU caches on a x86 CPU using a small kernel module,

**Default**

None

--cpu\_governor|--no-cpu\_governor:

**Description**

Enable: Allows the setting of the scaling governor of all cpu cores, to ensure that all use the same.

**Default**

None

--cpu\_governor\_governor:

**Description**

New scaling governor for all cpus

**Argument type**

Str()

**Default**

'performance'

--disable\_aslr|--no-disable\_aslr:

**Description**

Enable: Disable address space randomization

**Default**

None

--disable\_ht|--no-disable\_ht:

**Description**

Enable: Disable hyper-threading

**Default**

None

--disable\_turbo\_boost|--no-disable\_turbo\_boost:

**Description**

Enable: Disable amd and intel turbo boost

**Default**

None

`--disable_intel_turbo|--no-disable_intel_turbo:`**Description**

Enable: Disable intel turbo mode

**Default**

None

`--disable_amd_boost|--no-disable_amd_boost:`**Description**

Enable: Disable amd turbo boost

**Default**

None

`--cpuset|--no-cpuset:`**Description**

Enable: Enable cpusets, simply sets run/cpuset/active to true

**Default**

None

`--discarded_runs|--no-discarded_runs:`**Description**

Enable: Sets run/discarded\_runs

**Default**

None

`--discarded_runs_runs:`**Description**

Number of discarded runs

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

`--runner:`**Description**

If not "" overrides the runner setting for each program block

**Argument type**

ExactEither('','perf\_stat','rusage','spec','spec.py','time','output')

**Default**

''

`--random_cmd|--no-random_cmd:`**Description**

Pick a random command if more than one run command is passed.

**Default**

True

`--preset:`

**Description**

Enable other plugins by default: none = (enable none by default); all = cpu\_governor,disable\_swap,sync,stop\_start,other\_nice,nice,disable\_aslr,disable\_ht,cpuset,disable\_turbo\_boost (enable all, might freeze your system); usable = cpu\_governor,disable\_swap,sync,nice,disable\_aslr,disable\_ht,cpuset,disable\_turbo\_boost (like 'all' but doesn't affect other processes)

**Argument type**

ExactEither('none'|'all'|'usable')

**Default**

'none'

--parse\_output|--no-parse\_output:

**Description**

Parse the program output as a YAML dictionary of that gives for a specific property a measurement. Not all runners support it.

**Default**

False

--plugin\_order:

**Description**

Order in which the plugins are used, plugins that do not appear in this list are used before all others

**Argument type**

ListOrTuple(Str())

**Default**

['drop\_fs\_caches', 'sync', 'sleep', 'preheat', 'flush\_cpu\_caches']

--argument:

**Description**

Argument passed to all benchmarked commands by replacing \$ARGUMENT with this value in the command

**Argument type**

Str()

**Default**

''

--cpuset\_active|--no-cpuset\_active:

**Description**

Use cpuset functionality?

**Default**

False

--cpuset\_base\_core\_number:

**Description**

Number of cpu cores for the base (remaining part of the) system

**Argument type**

Int(range=range(0, 2))

**Default**

1

`--cpuset_parallel:`**Description**

0: benchmark sequential, > 0: benchmark parallel with n instances, -1: determine n automatically

**Argument type**

Int(constraint=<function>)

**Default**

0

`--cpuset_sub_core_number:`**Description**

Number of cpu cores per parallel running program.

**Argument type**

Int(range=range(0, 2))

**Default**

1

`--cpuset_temci_in_base_set|--no-cpuset_temci_in_base_set:`**Description**

place temci in the same cpu set as the rest of the system?

**Default**

True

`--discarded_runs:`**Description**

First n runs that are discarded

**Argument type**

Int(constraint=<function>)

**Default**

1

`--min_runs:`**Description**

Minimum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

20

`--max_runs:`**Description**

Maximum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

100

**--runs:****Description**

if != -1 sets max and min runs to its value

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

-1

**--max\_time:****Description**

Maximum time the whole benchmarking should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

**--max\_block\_time:****Description**

Maximum time one run block should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

**--run\_block\_size:****Description**

Number of benchmarking runs that are done together

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

**--in:****Description**

Input file with the program blocks to benchmark

**Argument type**

Str()

**Default**

'input.exec.yaml'

**--out:****Description**

Output file for the benchmarking results

**Argument type**

Str()

**Default**

'run\_output.yaml'

`--store_often|--no-store_often:`**Description**

Store the result file after each set of blocks is benchmarked

**Default**

False

`--included_blocks:`**Description**

List of included run blocks (all: include all), or their tag attribute or their number in the file (starting with 0), can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

`--disable_hyper_threading|--no-disable_hyper_threading:`**Description**

Disable the hyper threaded cores. Good for cpu bound programs.

**Default**

False

`--show_report|--no-show_report:`**Description**

Print console report if log\_level=info

**Default**

True

`--append|--no-append:`**Description**

Append to the output file instead of overwriting by adding new run data blocks

**Default**

False

`--shuffle|--no-shuffle:`**Description**

Randomize the order in which the program blocks are benchmarked.

**Default**

True

`--send_mail:`**Description**

If not empty, recipient of a mail after the benchmarking finished.

**Argument type**

Str()

**Default**

''

`--discard_all_data_for_block_on_error|--no-discard_all_data_for_block_on_error:`**Description**

Discard all run data for the failing program on error

**Default**

False

`--record_errors_in_file|--no-record_errors_in_file:`**Description**

Record the caught errors in the run\_output file

**Default**

True

`--no_build|--no-no_build:`**Description**

Do not build if build configs are present, only works if the working directory of the blocks does not change

**Default**

False

`--only_build|--no-only_build:`**Description**

Only build

**Default**

False

`--abort_after_build_error|--no-abort_after_build_error:`**Description**

Abort after the first failing build

**Default**

True

`--watch|--no-watch:`**Description**

Show the report continuously

**Default**

False

`--watch_every:`**Description**

Update the screen nth run (less updates are better for benchmarks)

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

`--driver:`



**Description**

Possible run drivers are 'exec' and 'shell'

**Argument type**

ExactEither('exec'|'shell')

**Default**

'exec'

--stats\_properties:

**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

--stats\_uncertainty\_range:

**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(<class 'float'>):<function>, T(<class 'float'>):<function>)

**Default**

[0.05, 0.15]

--stats\_tester:

**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

--report\_number\_parentheses|--no-report\_number\_parentheses:

**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

--report\_number\_min\_decimal\_places:

**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=<function>)

**Default**

3

--report\_number\_max\_decimal\_places:

**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=<function>)

**Default**

5

`--report_number_scientific_notation|--no-report_number_scientific_notation:`

**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

`--report_number_scientific_notation_si_prefixes|--no-report_number_scientific_notation_si_pref:`

**Description**

Use si prefixes instead of 'e...'

**Default**

True

`--report_number_omit_insignificant_decimal_places|--no-report_number_omit_insignificant_decima:`

**Description**

Omit insignificant decimal places

**Default**

False

`--report_number_force_min_decimal_places|--no-report_number_force_min_decimal_places:`

**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

`--report_number_percentages|--no-report_number_percentages:`

**Description**

Show as percentages

**Default**

False

`--report_number_sigmas:`

**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=<function>)

**Default**

2

`--report_number_parentheses_mode:`

**Description**

Mode for showing the parentheses: either d (Digits are considered significant if

they don't change if the number itself changes  $\pm \text{\$sigmas} * \text{std dev}$ ) or o (digits are considered significant if they are bigger than  $\text{\$sigmas} * \text{std dev}$ )

**Argument type**

ExactEither('d'|'o')

**Default**

'o'

`temci.scripts.cli.temci__format(number, abs_deviation, **kwargs)`

**Language**

python

**Command**

temci format

**Description**

Format a number

**Options:**

--settings:

**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

--config:

**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

--tmp\_dir:

**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

--log\_level:

**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

--sudo|--no-sudo:

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

--parentheses|--no-parentheses:

**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

--min\_decimal\_places:

**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=<function>)

**Default**

3

--max\_decimal\_places:

**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=<function>)

**Default**

5

--scientific\_notation|--no-scientific\_notation:

**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

--scientific\_notation\_si\_prefixes|--no-scientific\_notation\_si\_prefixes:

**Description**

Use si prefixes instead of 'e...'

**Default**

True

--omit\_insignificant\_decimal\_places|--no-omit\_insignificant\_decimal\_places:

**Description**

Omit insignificant decimal places

**Default**

False

--force\_min\_decimal\_places|--no-force\_min\_decimal\_places:

**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

--percentages|--no-percentages:

**Description**

Show as percentages

**Default**

False

--sigmas:

**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=<function>)

**Default**

2

--parentheses\_mode:

**Description**

Mode for showing the parentheses: either d (Digits are considered significant if they don't change if the number itself changes  $\pm \$sigmas * \text{std dev}$ ) or o (digits are considered significant if they are bigger than  $\$sigmas * \text{std dev}$ )

**Argument type**

ExactEither('d'|'o')

**Default**

'o'

temci.scripts.cli.temci\_\_init()

Helper commands to initialize files (like settings)

temci.scripts.cli.temci\_\_init\_\_build\_config(*file*, *\*\*kwargs*)

**Language**

python

**Command**

temci init build\_config

**Description**

Creates a sample build config file build\_config.yaml (or the file name passed) in the current directory

**Options:**

--settings:

**Description**

Additional settings file

**Argument type**

Str()

**Default****' '****--config:****Description**

Alias for settings

**Argument type**

Str()

**Default****' '****--tmp\_dir:****Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

**--log\_level:****Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

**--sudo|--no-sudo:****Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

`temci.scripts.cli.temci__init__run_config(file, **kwargs)`**Language**

python

**Command**`temci init run_config`**Description**Creates a sample exec run config file `run_config.yaml` (or the file name passed) in the current directory**Options:****--settings:****Description**

Additional settings file

**Argument type**

Str()

**Default**

''

`--config:`**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

`--tmp_dir:`**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

`--log_level:`**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

`--sudo|--no-sudo:`**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

`temci.scripts.cli.temci__init__settings(file, **kwargs)`**Language**

python

**Command**

temci init settings

**Description**

Create a new settings file temci.yaml (or the file name passed) in the current directory

**Options:**`--settings:`**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

`--config:`**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

`--tmp_dir:`**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

`--log_level:`**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

`--sudo|--no-sudo:`**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

`temci.scripts.cli.temci__report(report_file: List[str], **kwargs)`**Language**

python

**Command**

temci report

**Description**

Generate a report from benchmarking result

**Options:**`--settings:`**Description**

Additional settings file



**Argument type**

Str()

**Default**

''

`--config:`**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

`--tmp_dir:`**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

`--log_level:`**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

`--sudo|--no-sudo:`**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

`--in:`**Description**

Files that contain the benchmarking results

**Argument type**

Either(Str())|ListOrTuple(Str())

**Default**

'run\_output.yaml'

`--excluded_properties:`**Description**

Properties that aren't shown in the report.

**Argument type**

ListOrTuple(Str())

**Default**

['\_\_ov-time']

`--exclude_invalid|--no-exclude_invalid:`**Description**

Exclude all data sets that contain only NaNs.

**Default**

True

`--long_properties|--no-long_properties:`**Description**

Replace the property names in reports with longer more descriptive versions?

**Default**

False

`--xkcd_like_plots|--no-xkcd_like_plots:`**Description**

Produce xkcd like plots (requires the humor sans font to be installed)

**Default**

False

`--included_blocks:`**Description**

List of included run blocks (all: include all), identified by their description or tag attribute, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

`--reporter:`**Description**

Possible reporter are 'console', 'html2', 'csv', 'codespeed', 'codespeed2' and 'velcom'

**Argument type**

ExactEither('console'|'html2'|'csv'|'codespeed'|'codespeed2'|'velcom')

**Default**

'console'

`--properties:`**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

--uncertainty\_range:

**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(<class 'float'>):<function>, T(<class 'float'>):<function>)

**Default**

[0.05, 0.15]

--tester:

**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

--console\_out:

**Description**

Output file name or - (stdout)

**Argument type**

FileName(allow\_std=True)

**Default**

'\_'

--console\_with\_tester\_results|--no-console\_with\_tester\_results:

**Description**

Print statistical tests for every property for every two programs

**Default**

True

--console\_mode:

**Description**

'auto': report clusters (runs with the same description) and singles (clusters with a single entry, combined) separately, 'single': report all clusters together as one, 'cluster': report all clusters separately, 'both': append the output of 'cluster' to the output of 'single'

**Argument type**

ExactEither('both'|'cluster'|'single'|'auto')

**Default**

'auto'

--console\_report\_errors|--no-console\_report\_errors:

**Description**

Report on the failing blocks

**Default**

True

--console\_baseline:

**Description**

Matches the baseline block

**Argument type**

Str()

**Default**

''

--console\_baseline\_position:

**Description**

Position of the baseline comparison: each: after each block, after: after each cluster, both: after each and after cluster, instead: instead of the non baselined

**Argument type**

ExactEither('each'|'after'|'both'|'instead')

**Default**

'each'

--html2\_out:

**Description**

Output directory

**Argument type**

Str()

**Default**

'report'

--html2\_html\_filename:

**Description**

Name of the HTML file

**Argument type**

Str()

**Default**

'report.html'

--html2\_fig\_width\_small:

**Description**

Width of all small plotted figures

**Argument type**

T(<class 'float'>)

**Default**

15.0

--html2\_fig\_width\_big:

**Description**

Width of all big plotted figures

**Argument type**

T(<class 'float'>)

**Default**

25.0

--html2\_boxplot\_height:

**Description**

Height per run block for the big comparison box plots

**Argument type**

T(<class 'float'>)

**Default**

2.0

--html2\_alpha:

**Description**

Alpha value for confidence intervals

**Argument type**

T(<class 'float'>)

**Default**

0.05

--html2\_gen\_tex|--no-html2\_gen\_tex:

**Description**

Generate simple latex versions of the plotted figures?

**Default**

True

--html2\_gen\_pdf|--no-html2\_gen\_pdf:

**Description**

Generate pdf versions of the plotted figures?

**Default**

False

--html2\_gen\_xls|--no-html2\_gen\_xls:

**Description**

Generate excel files for all tables

**Default**

False

--html2\_show\_zoomed\_out|--no-html2\_show\_zoomed\_out:

**Description**

Show zoomed out (x min = 0) figures in the extended summaries?

**Default**

True

--html2\_percent\_format:

**Description**

Format string used to format floats as percentages

**Argument type**

Str()

**Default**

'{:5.2%}'

--html2\_float\_format:

**Description**

Format string used to format floats

**Argument type**

Str()

**Default**

'{:5.2e}'

--html2\_min\_in\_comparison\_tables|--no-html2\_min\_in\_comparison\_tables:

**Description**

Show the minimum related values in the big comparison table

**Default**

False

--html2\_mean\_in\_comparison\_tables|--no-html2\_mean\_in\_comparison\_tables:

**Description**

Show the mean related values in the big comparison table

**Default**

True

--html2\_force\_override|--no-html2\_force\_override:

**Description**

Override the contents of the output directory if it already exists?

**Default**

False

--html2\_hide\_stat\_warnings|--no-html2\_hide\_stat\_warnings:

**Description**

Hide warnings and errors related to statistical properties

**Default**

False

--html2\_local|--no-html2\_local:

**Description**

Use local versions of all third party resources

**Default**

False

--csv\_out:

**Description**

Output file name or standard out (-)

**Argument type**

FileName(allow\_std=True)

**Default**

'\_'

--csv\_columns:

**Description**

List of valid column specs, format is a comma separated list of 'PROPERTY[mod]' or 'ATTRIBUTE' mod is one of: mean, stddev, property, min,

max and stddev per mean, optionally a formatting option can be given via PROPERTY[mod|OPT1OPT2...], where the OPTs are one of the following: % (format as percentage), p (wrap insignificant digits in parentheses (+- 2 std dev)), s (use scientific notation, configured in report/number) and o (wrap digits in the order of magnitude of 2 std devs in parentheses). PROPERTY can be either the description or the short version of the property. Configure the number formatting further via the number settings in the settings file

#### Argument type

ListOrTuple(Str()):List of valid column specs, format is a comma separated list of 'PROPERTY[mod]' or 'ATTRIBUTE' mod is one of: mean, stddev, property, min, max and stddev per mean, optionally a formatting option can be given via aPROPERTY[mod|OPT1OPT2...], where the OPTs are one of the following: % (format as percentage), p (wrap insignificant digits in parentheses (+- 2 std dev)), s (use scientific notation, configured in report/number) and o (wrap digits in the order of magnitude of 2 std devs in parentheses). PROPERTY can be either the description or the short version of the property. Configure the number formatting further via the number settings in the settings file

#### Default

['description']

--codespeed\_project:

#### Description

Project name reported to codespeed.

#### Argument type

Str()

#### Default

''

--codespeed\_executable:

#### Description

Executable name reported to codespeed. Defaults to the project name.

#### Argument type

Str()

#### Default

''

--codespeed\_environment:

#### Description

Environment name reported to codespeed. Defaults to current host name.

#### Argument type

Str()

#### Default

''

--codespeed\_branch:

#### Description

Branch name reported to codespeed. Defaults to current branch or else 'master'.

#### Argument type

Str()

**Default**

''

`--codespeed_commit_id:`**Description**

Commit ID reported to codespeed. Defaults to current commit.

**Argument type**`Str()`**Default**

''

`temci.scripts.cli.temci__setup(build_kernel_modules: bool)`**Language**`python`**Command**`temci setup`**Description**

Compile all needed binaries in the temci scripts folder

**Options:**`--build_kernel_modules|--no-build_kernel_modules:`**Description**

Build the rusage helper program and the kernel modules used for disabling the CPU caches (with the `--build_kernel_modules` option)

**Default**`False``temci.scripts.cli.temci__shell(run_file, **kwargs)`**Language**`python`**Command**`temci shell`**Description**

Implements a run driver that runs the benched command a single time with redirected in- and output.

**Argument**

configuration YAML file

**Options:**`--settings:`**Description**

Additional settings file

**Argument type**`Str()`**Default**

''

`--config:`



**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

--tmp\_dir:

**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

--log\_level:

**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

--sudo|--no-sudo:

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

--discarded\_runs:

**Description**

First n runs that are discarded

**Argument type**

Int(constraint=<function>)

**Default**

1

--min\_runs:

**Description**

Minimum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

20

--max\_runs:

**Description**

Maximum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

100

--runs:

**Description**

if != -1 sets max and min runs to its value

**Argument type**

Int(constraint=<function>)

**Default**

-1

--max\_time:

**Description**

Maximum time the whole benchmarking should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

--max\_block\_time:

**Description**

Maximum time one run block should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

--run\_block\_size:

**Description**

Number of benchmarking runs that are done together

**Argument type**

Int(constraint=<function>)

**Default**

1

--in:

**Description**

Input file with the program blocks to benchmark

**Argument type**

Str()

**Default**

'input.exec.yaml'

--out:

**Description**

Output file for the benchmarking results

**Argument type**

Str()

**Default**

'run\_output.yaml'

--store\_often|--no-store\_often:

**Description**

Store the result file after each set of blocks is benchmarked

**Default**

False

--included\_blocks:

**Description**

List of included run blocks (all: include all), or their tag attribute or their number in the file (starting with 0), can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

--disable\_hyper\_threading|--no-disable\_hyper\_threading:

**Description**

Disable the hyper threaded cores. Good for cpu bound programs.

**Default**

False

--show\_report|--no-show\_report:

**Description**

Print console report if log\_level=info

**Default**

True

--append|--no-append:

**Description**

Append to the output file instead of overwriting by adding new run data blocks

**Default**

False

--shuffle|--no-shuffle:

**Description**

Randomize the order in which the program blocks are benchmarked.

**Default**

True

--send\_mail:

**Description**

If not empty, recipient of a mail after the benchmarking finished.

**Argument type**

Str()

**Default**

''

--discard\_all\_data\_for\_block\_on\_error|--no-discard\_all\_data\_for\_block\_on\_error:

**Description**

Discard all run data for the failing program on error

**Default**

False

--record\_errors\_in\_file|--no-record\_errors\_in\_file:

**Description**

Record the caught errors in the run\_output file

**Default**

True

--no\_build|--no-no\_build:

**Description**

Do not build if build configs are present, only works if the working directory of the blocks does not change

**Default**

False

--only\_build|--no-only\_build:

**Description**

Only build

**Default**

False

--abort\_after\_build\_error|--no-abort\_after\_build\_error:

**Description**

Abort after the first failing build

**Default**

True

--watch|--no-watch:

**Description**

Show the report continuously

**Default**

False

--watch\_every:

**Description**

Update the screen nth run (less updates are better for benchmarks)

**Argument type**

Int(constraint=<function>)

**Default**

1

`--driver:`**Description**

Possible run drivers are 'exec' and 'shell'

**Argument type**

ExactEither('exec'|'shell')

**Default**

'exec'

`--stats_properties:`**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

`--stats_uncertainty_range:`**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(&lt;class 'float'&gt;):&lt;function&gt;, T(&lt;class 'float'&gt;):&lt;function&gt;)

**Default**

[0.05, 0.15]

`--stats_tester:`**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

`--report_number_parentheses|--no-report_number_parentheses:`**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

`--report_number_min_decimal_places:`**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

3

`--report_number_max_decimal_places:`**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

5

`--report_number_scientific_notation|--no-report_number_scientific_notation:`**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

`--report_number_scientific_notation_si_prefixes|--no-report_number_scientific_notation_si_prefixes:`**Description**

Use si prefixes instead of 'e...'

**Default**

True

`--report_number_omit_insignificant_decimal_places|--no-report_number_omit_insignificant_decimal_places:`**Description**

Omit insignificant decimal places

**Default**

False

`--report_number_force_min_decimal_places|--no-report_number_force_min_decimal_places:`**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

`--report_number_percentages|--no-report_number_percentages:`**Description**

Show as percentages

**Default**

False

`--report_number_sigmas:`**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

2

`--report_number_parentheses_mode:`

**Description**

Mode for showing the parentheses: either d (Digits are considered significant if they don't change if the number itself changes  $\pm \text{\$sigmas} * \text{std dev}$ ) or o (digits are considered significant if they are bigger than  $\text{\$sigmas} * \text{std dev}$ )

**Argument type**

ExactEither('d'|'o')

**Default**

'o'

--nice|--no-nice:

**Description**

Enable: Allows the setting of the nice and ionice values of the benchmarking process.

**Default**

None

--nice\_nice:

**Description**

Niceness values range from -20 (most favorable to the process) to 19 (least favorable to the process).

**Argument type**

Int(range=range(-20, 20))

**Default**

-15

--nice\_io\_nice:

**Description**

Specify the name or number of the scheduling class to use; 0 for none, 1 for real-time, 2 for best-effort, 3 for idle.

**Argument type**

Int(range=range(0, 4))

**Default**

1

--env\_randomize|--no-env\_randomize:

**Description**

Enable: Adds random environment variables.

**Default**

None

--env\_randomize\_min:

**Description**

Minimum number of added random environment variables

**Argument type**

Int(constraint=<function>)

**Default**

4

--env\_randomize\_max:

**Description**

Maximum number of added random environment variables

**Argument type**

Int(constraint=<function>)

**Default**

4

--env\_randomize\_var\_max:

**Description**

Maximum length of each random value

**Argument type**

Int(constraint=<function>)

**Default**

4096

--env\_randomize\_key\_max:

**Description**

Maximum length of each random key

**Argument type**

Int(constraint=<function>)

**Default**

4096

--preheat|--no-preheat:

**Description**

Enable: Preheats the system with a cpu bound task

**Default**

None

--preheat\_time:

**Description**

Number of seconds to preheat the system with an cpu bound task

**Argument type**

Int(constraint=<function>)

**Default**

10

--preheat\_when:

**Description**

When to preheat

**Argument type**

ListOrTuple(ExactEither('before\_each\_run'|'at\_setup'))

**Default**

['before\_each\_run']

--other\_nice|--no-other\_nice:



**Description**

Enable: Allows the setting of the nice value of most other processes (as far as possible).

**Default**

None

--other\_nice\_nice:

**Description**

Niceness values for other processes.

**Argument type**

Int(range=range(-20, 20))

**Default**

19

--other\_nice\_min\_nice:

**Description**

Processes with lower nice values are ignored.

**Argument type**

Int(range=range(-15, 20))

**Default**

-10

--stop\_start|--no-stop\_start:

**Description**

Enable: Stop almost all other processes (as far as possible).

**Default**

None

--stop\_start\_min\_nice:

**Description**

Processes with lower nice values are ignored.

**Argument type**

Int(range=range(-15, 20))

**Default**

-10

--stop\_start\_min\_id:

**Description**

Processes with lower id are ignored.

**Argument type**

Int(constraint=<function>)

**Default**

1500

--stop\_start\_comm\_prefixes:

**Description**

Each process which name (lower cased) starts with one of the prefixes is not ignored. Overrides the decision based on the min\_id.

**Argument type**

ListOrTuple(Str())

**Default**

['ssh', 'xorg', 'bluetoothd']

`--stop_start_comm_prefixes_ignored:`**Description**

Each process which name (lower cased) starts with one of the prefixes is ignored.  
It overrides the decisions based on `comm_prefixes` and `min_id`.

**Argument type**

ListOrTuple(Str())

**Default**

['dbus', 'kworker']

`--stop_start_subtree_suffixes:`**Description**

Suffixes of processes names which are stopped.

**Argument type**

ListOrTuple(Str())

**Default**

['dm', 'apache']

`--stop_start_dry_run|--no-stop_start_dry_run:`**Description**

Just output the to be stopped processes but don't actually stop them?

**Default**

False

`--sync|--no-sync:`**Description**

Enable: Calls sync before each program execution.

**Default**

None

`--sleep|--no-sleep:`**Description**

Enable: Sleep a given amount of time before the benchmarking begins.

**Default**

None

`--sleep_seconds:`**Description**

Seconds to sleep

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

10

`--drop_fs_caches|--no-drop_fs_caches:`

**Description**

Enable: Drop page cache, directoy entries and inodes before every benchmarking run.

**Default**

None

--drop\_fs\_caches\_free\_pagecache|--no-drop\_fs\_caches\_free\_pagecache:

**Description**

Free the page cache

**Default**

True

--drop\_fs\_caches\_free\_dentries\_inodes|--no-drop\_fs\_caches\_free\_dentries\_inodes:

**Description**

Free dentries and inodes

**Default**

True

--disable\_swap|--no-disable\_swap:

**Description**

Enable: Disables swapping on the system before the benchmarking and enables it after.

**Default**

None

--disable\_cpu\_caches|--no-disable\_cpu\_caches:

**Description**

Enable: Disable the L1 and L2 caches on x86 and x86-64 architectures.

**Default**

None

--flush\_cpu\_caches|--no-flush\_cpu\_caches:

**Description**

Enable: Flushes the CPU caches on a x86 CPU using a small kernel module,

**Default**

None

--cpu\_governor|--no-cpu\_governor:

**Description**

Enable: Allows the setting of the scaling governor of all cpu cores, to ensure that all use the same.

**Default**

None

--cpu\_governor\_governor:

**Description**

New scaling governor for all cpus

**Argument type**

Str()

**Default**

'performance'

--disable\_aslr|--no-disable\_aslr:

**Description**

Enable: Disable address space randomization

**Default**

None

--disable\_ht|--no-disable\_ht:

**Description**

Enable: Disable hyper-threading

**Default**

None

--disable\_turbo\_boost|--no-disable\_turbo\_boost:

**Description**

Enable: Disable amd and intel turbo boost

**Default**

None

--disable\_intel\_turbo|--no-disable\_intel\_turbo:

**Description**

Enable: Disable intel turbo mode

**Default**

None

--disable\_amd\_boost|--no-disable\_amd\_boost:

**Description**

Enable: Disable amd turbo boost

**Default**

None

--cpuset|--no-cpuset:

**Description**

Enable: Enable cpusets, simply sets run/cpuset/active to true

**Default**

None

--discarded\_runs|--no-discarded\_runs:

**Description**

Enable: Sets run/discarded\_runs

**Default**

None

--discarded\_runs\_runs:

**Description**

Number of discarded runs

**Argument type**

Int(constraint=<function>)

**Default**

1

`--preset:`**Description**

Enable other plugins by default: none = (enable none by default); all = cpu\_governor,disable\_swap,sync,stop\_start,other\_nice,nice,disable\_aslr,disable\_ht,cpuset,disable\_turbo\_boost (enable all, might freeze your system); usable = cpu\_governor,disable\_swap,sync,nice,disable\_aslr,disable\_ht,cpuset,disable\_turbo\_boost (like 'all' but doesn't affect other processes)

**Argument type**

ExactEither('none'|'all'|'usable')

**Default**

'none'

`--plugin_order:`**Description**

Order in which the plugins are used, plugins that do not appear in this list are used before all others

**Argument type**

ListOrTuple(Str())

**Default**

['drop\_fs\_caches', 'sync', 'sleep', 'preheat',  
'flush\_cpu\_caches']

`--cpuset_active|--no-cpuset_active:`**Description**

Use cpuset functionality?

**Default**

False

`--cpuset_base_core_number:`**Description**

Number of cpu cores for the base (remaining part of the) system

**Argument type**

Int(range=range(0, 2))

**Default**

1

`--cpuset_parallel:`**Description**

0: benchmark sequential, > 0: benchmark parallel with n instances, -1: determine n automatically

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

0

`--cpuset_sub_core_number:`

**Description**

Number of cpu cores per parallel running program.

**Argument type**

Int(range=range(0, 2))

**Default**

1

--cpuset\_temci\_in\_base\_set|--no-cpuset\_temci\_in\_base\_set:

**Description**

place temci in the same cpu set as the rest of the system?

**Default**

True

--discarded\_runs:

**Description**

First n runs that are discarded

**Argument type**

Int(constraint=<function>)

**Default**

1

--min\_runs:

**Description**

Minimum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

20

--max\_runs:

**Description**

Maximum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

100

--runs:

**Description**

if != -1 sets max and min runs to its value

**Argument type**

Int(constraint=<function>)

**Default**

-1

--max\_time:

**Description**

Maximum time the whole benchmarking should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

--max\_block\_time:

**Description**

Maximum time one run block should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

--run\_block\_size:

**Description**

Number of benchmarking runs that are done together

**Argument type**

Int(constraint=<function>)

**Default**

1

--in:

**Description**

Input file with the program blocks to benchmark

**Argument type**

Str()

**Default**

'input.exec.yaml'

--out:

**Description**

Output file for the benchmarking results

**Argument type**

Str()

**Default**

'run\_output.yaml'

--store\_often|--no-store\_often:

**Description**

Store the result file after each set of blocks is benchmarked

**Default**

False

--included\_blocks:

**Description**

List of included run blocks (all: include all), or their tag attribute or their number in the file (starting with 0), can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

--disable\_hyper\_threading|--no-disable\_hyper\_threading:

**Description**

Disable the hyper threaded cores. Good for cpu bound programs.

**Default**

False

--show\_report|--no-show\_report:

**Description**

Print console report if log\_level=info

**Default**

True

--append|--no-append:

**Description**

Append to the output file instead of overwriting by adding new run data blocks

**Default**

False

--shuffle|--no-shuffle:

**Description**

Randomize the order in which the program blocks are benchmarked.

**Default**

True

--send\_mail:

**Description**

If not empty, recipient of a mail after the benchmarking finished.

**Argument type**

Str()

**Default**

''

--discard\_all\_data\_for\_block\_on\_error|--no-discard\_all\_data\_for\_block\_on\_error:

**Description**

Discard all run data for the failing program on error

**Default**

False

--record\_errors\_in\_file|--no-record\_errors\_in\_file:

**Description**

Record the caught errors in the run\_output file



**Default**

True

`--no_build|--no-no_build:`**Description**

Do not build if build configs are present, only works if the working directory of the blocks does not change

**Default**

False

`--only_build|--no-only_build:`**Description**

Only build

**Default**

False

`--abort_after_build_error|--no-abort_after_build_error:`**Description**

Abort after the first failing build

**Default**

True

`--watch|--no-watch:`**Description**

Show the report continuously

**Default**

False

`--watch_every:`**Description**

Update the screen nth run (less updates are better for benchmarks)

**Argument type**

Int(constraint=<function>)

**Default**

1

`--driver:`**Description**

Possible run drivers are 'exec' and 'shell'

**Argument type**

ExactEither('exec','shell')

**Default**

'exec'

`--stats_properties:`**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

`--stats_uncertainty_range:`**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(&lt;class 'float'&gt;):&lt;function&gt;, T(&lt;class 'float'&gt;):&lt;function&gt;)

**Default**

[0.05, 0.15]

`--stats_tester:`**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

`--report_number_parentheses|--no-report_number_parentheses:`**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

`--report_number_min_decimal_places:`**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

3

`--report_number_max_decimal_places:`**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

5

`--report_number_scientific_notation|--no-report_number_scientific_notation:`**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

--report\_number\_scientific\_notation\_si\_prefixes|--no-report\_number\_scientific\_notation\_si\_pref

**Description**

Use si prefixes instead of 'e...'

**Default**

True

--report\_number\_omit\_insignificant\_decimal\_places|--no-report\_number\_omit\_insignificant\_decima

**Description**

Omit insignificant decimal places

**Default**

False

--report\_number\_force\_min\_decimal\_places|--no-report\_number\_force\_min\_decimal\_places:

**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

--report\_number\_percentages|--no-report\_number\_percentages:

**Description**

Show as percentages

**Default**

False

--report\_number\_sigmas:

**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=<function>)

**Default**

2

--report\_number\_parentheses\_mode:

**Description**

Mode for showing the parentheses: either d (Digits are considered significant if they don't change if the number itself changes  $\pm \text{\$sigmas} * \text{std dev}$ ) or o (digits are considered significant if they are bigger than  $\text{\$sigmas} * \text{std dev}$ )

**Argument type**

ExactEither('d'|'o')

**Default**

'o'

temci.scripts.cli.temci\_\_short()

Utility commands to ease working directly on the command line

temci.scripts.cli.temci\_\_short\_\_exec(commands: list, with\_description: Optional[list] = None, without\_description: Optional[list] = None, \*\*kwargs)

**Language**

python

**Command**

temci short exec

**Description**

Execute commands directly with the exec run driver

**Options:**

--settings:

**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

--config:

**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

--tmp\_dir:

**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

--log\_level:

**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

--sudo|--no-sudo:

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

--with\_description|-d:

**Description**

DESCRIPTION COMMAND: Benchmark the command and set its description

attribute. Appends '\$ARGUMENT' to the command if the string isn't present. Use the '-argument' option to set the value that this string is replaced with.

**Argument type**

ListOrTuple(Tuple(Str(), Str()))

**Default**

None

--without\_description|-wd:

**Description**

COMMAND: Benchmark the command and use itself as its description. Appends '\$ARGUMENT' to the command if the string isn't present. Use the '-argument' option to set the value that this string is replaced with.

**Argument type**

ListOrTuple(Str())

**Default**

None

--nice|--no-nice:

**Description**

Enable: Allows the setting of the nice and ionice values of the benchmarking process.

**Default**

None

--nice\_nice:

**Description**

Niceness values range from -20 (most favorable to the process) to 19 (least favorable to the process).

**Argument type**

Int(range=range(-20, 20))

**Default**

-15

--nice\_io\_nice:

**Description**

Specify the name or number of the scheduling class to use; 0 for none, 1 for real-time, 2 for best-effort, 3 for idle.

**Argument type**

Int(range=range(0, 4))

**Default**

1

--env\_randomize|--no-env\_randomize:

**Description**

Enable: Adds random environment variables.

**Default**

None

--env\_randomize\_min:

**Description**

Minimum number of added random environment variables

**Argument type**

Int(constraint=<function>)

**Default**

4

--env\_randomize\_max:

**Description**

Maximum number of added random environment variables

**Argument type**

Int(constraint=<function>)

**Default**

4

--env\_randomize\_var\_max:

**Description**

Maximum length of each random value

**Argument type**

Int(constraint=<function>)

**Default**

4096

--env\_randomize\_key\_max:

**Description**

Maximum length of each random key

**Argument type**

Int(constraint=<function>)

**Default**

4096

--preheat|--no-preheat:

**Description**

Enable: Preheats the system with a cpu bound task

**Default**

None

--preheat\_time:

**Description**

Number of seconds to preheat the system with an cpu bound task

**Argument type**

Int(constraint=<function>)

**Default**

10

--preheat\_when:

**Description**

When to preheat

**Argument type**

ListOrTuple(ExactEither('before\_each\_run'|'at\_setup'))

**Default**

['before\_each\_run']

--other\_nice|--no-other\_nice:

**Description**

Enable: Allows the setting of the nice value of most other processes (as far as possible).

**Default**

None

--other\_nice\_nice:

**Description**

Niceness values for other processes.

**Argument type**

Int(range=range(-20, 20))

**Default**

19

--other\_nice\_min\_nice:

**Description**

Processes with lower nice values are ignored.

**Argument type**

Int(range=range(-15, 20))

**Default**

-10

--stop\_start|--no-stop\_start:

**Description**

Enable: Stop almost all other processes (as far as possible).

**Default**

None

--stop\_start\_min\_nice:

**Description**

Processes with lower nice values are ignored.

**Argument type**

Int(range=range(-15, 20))

**Default**

-10

--stop\_start\_min\_id:

**Description**

Processes with lower id are ignored.

**Argument type**

Int(constraint=<function>)

**Default**

1500

`--stop_start_comm_prefixes:`**Description**

Each process which name (lower cased) starts with one of the prefixes is not ignored. Overrides the decision based on the `min_id`.

**Argument type**

ListOrTuple(Str())

**Default**

['ssh', 'xorg', 'bluetoothd']

`--stop_start_comm_prefixes_ignored:`**Description**

Each process which name (lower cased) starts with one of the prefixes is ignored. It overrides the decisions based on `comm_prefixes` and `min_id`.

**Argument type**

ListOrTuple(Str())

**Default**

['dbus', 'kworker']

`--stop_start_subtree_suffixes:`**Description**

Suffixes of processes names which are stopped.

**Argument type**

ListOrTuple(Str())

**Default**

['dm', 'apache']

`--stop_start_dry_run|--no-stop_start_dry_run:`**Description**

Just output the to be stopped processes but don't actually stop them?

**Default**

False

`--sync|--no-sync:`**Description**

Enable: Calls sync before each program execution.

**Default**

None

`--sleep|--no-sleep:`**Description**

Enable: Sleep a given amount of time before the benchmarking begins.

**Default**

None

`--sleep_seconds:`



**Description**

Seconds to sleep

**Argument type**

Int(constraint=<function>)

**Default**

10

--drop\_fs\_caches|--no-drop\_fs\_caches:

**Description**

Enable: Drop page cache, directoy entries and inodes before every benchmarking run.

**Default**

None

--drop\_fs\_caches\_free\_pagecache|--no-drop\_fs\_caches\_free\_pagecache:

**Description**

Free the page cache

**Default**

True

--drop\_fs\_caches\_free\_dentries\_inodes|--no-drop\_fs\_caches\_free\_dentries\_inodes:

**Description**

Free dentries and inodes

**Default**

True

--disable\_swap|--no-disable\_swap:

**Description**

Enable: Disables swapping on the system before the benchmarking and enables it after.

**Default**

None

--disable\_cpu\_caches|--no-disable\_cpu\_caches:

**Description**

Enable: Disable the L1 and L2 caches on x86 and x86-64 architectures.

**Default**

None

--flush\_cpu\_caches|--no-flush\_cpu\_caches:

**Description**

Enable: Flushes the CPU caches on a x86 CPU using a small kernel module,

**Default**

None

--cpu\_governor|--no-cpu\_governor:

**Description**

Enable: Allows the setting of the scaling governor of all cpu cores, to ensure that all use the same.

**Default**

None

`--cpu_governor_governor:`**Description**

New scaling governor for all cpus

**Argument type**

Str()

**Default**

'performance'

`--disable_aslr|--no-disable_aslr:`**Description**

Enable: Disable address space randomization

**Default**

None

`--disable_ht|--no-disable_ht:`**Description**

Enable: Disable hyper-threading

**Default**

None

`--disable_turbo_boost|--no-disable_turbo_boost:`**Description**

Enable: Disable amd and intel turbo boost

**Default**

None

`--disable_intel_turbo|--no-disable_intel_turbo:`**Description**

Enable: Disable intel turbo mode

**Default**

None

`--disable_amd_boost|--no-disable_amd_boost:`**Description**

Enable: Disable amd turbo boost

**Default**

None

`--cpuset|--no-cpuset:`**Description**

Enable: Enable cpusets, simply sets run/cpuset/active to true

**Default**

None

`--discarded_runs|--no-discarded_runs:`**Description**

Enable: Sets run/discarded\_runs

**Default**

None

`--discarded_runs_runs:`**Description**

Number of discarded runs

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

`--runner:`**Description**

If not "" overrides the runner setting for each program block

**Argument type**

ExactEither(''|'perf\_stat'|'rusage'|'spec'|'spec.py'|'time'|'output')

**Default**

''

`--random_cmd | --no-random_cmd:`**Description**

Pick a random command if more than one run command is passed.

**Default**

True

`--preset:`**Description**

Enable other plugins by default: none = (enable none by default); all =  
 cpu\_governor,disable\_swap,sync,stop\_start,other\_nice,nice,disable\_aslr,disable\_ht,cpuset,disable\_turbo\_boost  
 (enable all, might freeze your system); usable =  
 cpu\_governor,disable\_swap,sync,nice,disable\_aslr,disable\_ht,cpuset,disable\_turbo\_boost  
 (like 'all' but doesn't affect other processes)

**Argument type**

ExactEither('none'|'all'|'usable')

**Default**

'none'

`--parse_output | --no-parse_output:`**Description**

Parse the program output as a YAML dictionary of that gives for a specific prop-  
 erty a measurement. Not all runners support it.

**Default**

False

`--plugin_order:`**Description**

Order in which the plugins are used, plugins that do not appear in this list are  
 used before all others

**Argument type**

ListOrTuple(Str())

**Default**['drop\_fs\_caches', 'sync', 'sleep', 'preheat',  
'flush\_cpu\_caches']`--argument:`**Description**Argument passed to all benchmarked commands by replacing \$ARGUMENT  
with this value in the command**Argument type**

Str()

**Default**

''

`--cpuset_active|--no-cpuset_active:`**Description**

Use cpuset functionality?

**Default**

False

`--cpuset_base_core_number:`**Description**

Number of cpu cores for the base (remaining part of the) system

**Argument type**

Int(range=range(0, 2))

**Default**

1

`--cpuset_parallel:`**Description**0: benchmark sequential, > 0: benchmark parallel with n instances, -1: determine  
n automatically**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

0

`--cpuset_sub_core_number:`**Description**

Number of cpu cores per parallel running program.

**Argument type**

Int(range=range(0, 2))

**Default**

1

`--cpuset_temci_in_base_set|--no-cpuset_temci_in_base_set:`**Description**

place temci in the same cpu set as the rest of the system?

**Default**  
True

--discarded\_runs:

**Description**  
First n runs that are discarded

**Argument type**  
Int(constraint=<function>)

**Default**  
1

--min\_runs:

**Description**  
Minimum number of benchmarking runs

**Argument type**  
Int(constraint=<function>)

**Default**  
20

--max\_runs:

**Description**  
Maximum number of benchmarking runs

**Argument type**  
Int(constraint=<function>)

**Default**  
100

--runs:

**Description**  
if != -1 sets max and min runs to its value

**Argument type**  
Int(constraint=<function>)

**Default**  
-1

--max\_time:

**Description**  
Maximum time the whole benchmarking should take, -1 == no timeout, supports normal time span expressions

**Argument type**  
ValidTimespan()

**Default**  
'-1'

--max\_block\_time:

**Description**  
Maximum time one run block should take, -1 == no timeout, supports normal time span expressions

**Argument type**  
ValidTimespan()

**Default**  
'-1'

--run\_block\_size:

**Description**  
Number of benchmarking runs that are done together

**Argument type**  
Int(constraint=<function>)

**Default**  
1

--in:

**Description**  
Input file with the program blocks to benchmark

**Argument type**  
Str()

**Default**  
'input.exec.yaml'

--out:

**Description**  
Output file for the benchmarking results

**Argument type**  
Str()

**Default**  
'run\_output.yaml'

--store\_often|--no-store\_often:

**Description**  
Store the result file after each set of blocks is benchmarked

**Default**  
False

--included\_blocks:

**Description**  
List of included run blocks (all: include all), or their tag attribute or their number in the file (starting with 0), can be regular expressions

**Argument type**  
ListOrTuple(Str())

**Default**  
['all']

--disable\_hyper\_threading|--no-disable\_hyper\_threading:

**Description**  
Disable the hyper threaded cores. Good for cpu bound programs.

**Default**

False

`--show_report|--no-show_report:`**Description**

Print console report if log\_level=info

**Default**

True

`--append|--no-append:`**Description**

Append to the output file instead of overwriting by adding new run data blocks

**Default**

False

`--shuffle|--no-shuffle:`**Description**

Randomize the order in which the program blocks are benchmarked.

**Default**

True

`--send_mail:`**Description**

If not empty, recipient of a mail after the benchmarking finished.

**Argument type**

Str()

**Default**

''

`--discard_all_data_for_block_on_error|--no-discard_all_data_for_block_on_error:`**Description**

Discard all run data for the failing program on error

**Default**

False

`--record_errors_in_file|--no-record_errors_in_file:`**Description**

Record the caught errors in the run\_output file

**Default**

True

`--no_build|--no-no_build:`**Description**

Do not build if build configs are present, only works if the working directory of the blocks does not change

**Default**

False

`--only_build|--no-only_build:`

**Description**

Only build

**Default**

False

--abort\_after\_build\_error|--no-abort\_after\_build\_error:

**Description**

Abort after the first failing build

**Default**

True

--watch|--no-watch:

**Description**

Show the report continuously

**Default**

False

--watch\_every:

**Description**

Update the screen nth run (less updates are better for benchmarks)

**Argument type**

Int(constraint=<function>)

**Default**

1

--driver:

**Description**

Possible run drivers are 'exec' and 'shell'

**Argument type**

ExactEither('exec'|'shell')

**Default**

'exec'

--stats\_properties:

**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

--stats\_uncertainty\_range:

**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(<class 'float'>):<function>, T(<class 'float'>):<function>)



**Default**

[0.05, 0.15]

`--stats_tester:`**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

`--report_number_parentheses|--no-report_number_parentheses:`**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

`--report_number_min_decimal_places:`**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

3

`--report_number_max_decimal_places:`**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

5

`--report_number_scientific_notation|--no-report_number_scientific_notation:`**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

`--report_number_scientific_notation_si_prefixes|--no-report_number_scientific_notation_si_prefixes:`**Description**

Use si prefixes instead of 'e...'

**Default**

True

`--report_number_omit_insignificant_decimal_places|--no-report_number_omit_insignificant_decimal_places:`**Description**

Omit insignificant decimal places

**Default**

False

`--report_number_force_min_decimal_places|--no-report_number_force_min_decimal_places:`**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

`--report_number_percentages|--no-report_number_percentages:`**Description**

Show as percentages

**Default**

False

`--report_number_sigmas:`**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

2

`--report_number_parentheses_mode:`**Description**Mode for showing the parentheses: either d (Digits are considered significant if they don't change if the number itself changes  $\pm \text{\$sigmas} * \text{std dev}$ ) or o (digits are considered significant if they are bigger than  $\text{\$sigmas} * \text{std dev}$ )**Argument type**

ExactEither('d'|'o')

**Default**

'o'

`--discarded_runs:`**Description**

First n runs that are discarded

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

`--min_runs:`**Description**

Minimum number of benchmarking runs

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

20

--max\_runs:

**Description**

Maximum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

100

--runs:

**Description**

if != -1 sets max and min runs to its value

**Argument type**

Int(constraint=<function>)

**Default**

-1

--max\_time:

**Description**

Maximum time the whole benchmarking should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

--max\_block\_time:

**Description**

Maximum time one run block should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

--run\_block\_size:

**Description**

Number of benchmarking runs that are done together

**Argument type**

Int(constraint=<function>)

**Default**

1

--in:

**Description**

Input file with the program blocks to benchmark

**Argument type**

Str()

**Default**  
'input.exec.yaml'

--out:

**Description**  
Output file for the benchmarking results

**Argument type**  
Str()

**Default**  
'run\_output.yaml'

--store\_often|--no-store\_often:

**Description**  
Store the result file after each set of blocks is benchmarked

**Default**  
False

--included\_blocks:

**Description**  
List of included run blocks (all: include all), or their tag attribute or their number in the file (starting with 0), can be regular expressions

**Argument type**  
ListOrTuple(Str())

**Default**  
['all']

--disable\_hyper\_threading|--no-disable\_hyper\_threading:

**Description**  
Disable the hyper threaded cores. Good for cpu bound programs.

**Default**  
False

--show\_report|--no-show\_report:

**Description**  
Print console report if log\_level=info

**Default**  
True

--append|--no-append:

**Description**  
Append to the output file instead of overwriting by adding new run data blocks

**Default**  
False

--shuffle|--no-shuffle:

**Description**  
Randomize the order in which the program blocks are benchmarked.

**Default**  
True

--send\_mail:

**Description**

If not empty, recipient of a mail after the benchmarking finished.

**Argument type**

Str()

**Default**

''

--discard\_all\_data\_for\_block\_on\_error|--no-discard\_all\_data\_for\_block\_on\_error:

**Description**

Discard all run data for the failing program on error

**Default**

False

--record\_errors\_in\_file|--no-record\_errors\_in\_file:

**Description**

Record the caught errors in the run\_output file

**Default**

True

--no\_build|--no-no\_build:

**Description**

Do not build if build configs are present, only works if the working directory of the blocks does not change

**Default**

False

--only\_build|--no-only\_build:

**Description**

Only build

**Default**

False

--abort\_after\_build\_error|--no-abort\_after\_build\_error:

**Description**

Abort after the first failing build

**Default**

True

--watch|--no-watch:

**Description**

Show the report continuously

**Default**

False

--watch\_every:

**Description**

Update the screen nth run (less updates are better for benchmarks)

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

`--driver:`**Description**

Possible run drivers are 'exec' and 'shell'

**Argument type**

ExactEither('exec'|'shell')

**Default**

'exec'

`--stats_properties:`**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

`--stats_uncertainty_range:`**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(&lt;class 'float'&gt;):&lt;function&gt;, T(&lt;class 'float'&gt;):&lt;function&gt;)

**Default**

[0.05, 0.15]

`--stats_tester:`**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

`--report_number_parentheses|--no-report_number_parentheses:`**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

`--report_number_min_decimal_places:`**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

3

`--report_number_max_decimal_places:`**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

5

`--report_number_scientific_notation|--no-report_number_scientific_notation:`**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

`--report_number_scientific_notation_si_prefixes|--no-report_number_scientific_notation_si_prefixes:`**Description**

Use si prefixes instead of 'e...'

**Default**

True

`--report_number_omit_insignificant_decimal_places|--no-report_number_omit_insignificant_decimal_places:`**Description**

Omit insignificant decimal places

**Default**

False

`--report_number_force_min_decimal_places|--no-report_number_force_min_decimal_places:`**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

`--report_number_percentages|--no-report_number_percentages:`**Description**

Show as percentages

**Default**

False

`--report_number_sigmas:`**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

2

`--report_number_parentheses_mode:`**Description**

Mode for showing the parentheses: either d (Digits are considered significant if they don't change if the number itself changes  $\pm \text{\$sigmas} * \text{std dev}$ ) or o (digits are considered significant if they are bigger than  $\text{\$sigmas} * \text{std dev}$ )

**Argument type**`ExactEither('d'|'o')`**Default**`'o'``temci.scripts.cli.temci__short__shell(command: str, **kwargs)`**Language**`python`**Command**`temci short shell`**Description**

Execute a command in a shell with benchmarking setup

**Options:**`--settings:`**Description**

Additional settings file

**Argument type**`Str()`**Default**`''``--config:`**Description**

Alias for settings

**Argument type**`Str()`**Default**`''``--tmp_dir:`**Description**

Used temporary directory

**Argument type**`Str()`**Default**`'/tmp/temci'``--log_level:`



**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

--sudo|--no-sudo:

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

--nice|--no-nice:

**Description**

Enable: Allows the setting of the nice and ionice values of the benchmarking process.

**Default**

None

--nice\_nice:

**Description**

Niceness values range from -20 (most favorable to the process) to 19 (least favorable to the process).

**Argument type**

Int(range=range(-20, 20))

**Default**

-15

--nice\_io\_nice:

**Description**

Specify the name or number of the scheduling class to use;0 for none, 1 for real-time, 2 for best-effort, 3 for idle.

**Argument type**

Int(range=range(0, 4))

**Default**

1

--env\_randomize|--no-env\_randomize:

**Description**

Enable: Adds random environment variables.

**Default**

None

--env\_randomize\_min:

**Description**

Minimum number of added random environment variables

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

4

`--env_randomize_max:`**Description**

Maximum number of added random environment variables

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

4

`--env_randomize_var_max:`**Description**

Maximum length of each random value

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

4096

`--env_randomize_key_max:`**Description**

Maximum length of each random key

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

4096

`--preheat | --no-preheat:`**Description**

Enable: Preheats the system with a cpu bound task

**Default**

None

`--preheat_time:`**Description**

Number of seconds to preheat the system with an cpu bound task

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

10

`--preheat_when:`**Description**

When to preheat

**Argument type**

ListOrTuple(ExactEither('before\_each\_run'|'at\_setup'))

**Default**

['before\_each\_run']

`--other_nice|--no-other_nice:`**Description**

Enable: Allows the setting of the nice value of most other processes (as far as possible).

**Default**

None

`--other_nice_nice:`**Description**

Niceness values for other processes.

**Argument type**

Int(range=range(-20, 20))

**Default**

19

`--other_nice_min_nice:`**Description**

Processes with lower nice values are ignored.

**Argument type**

Int(range=range(-15, 20))

**Default**

-10

`--stop_start|--no-stop_start:`**Description**

Enable: Stop almost all other processes (as far as possible).

**Default**

None

`--stop_start_min_nice:`**Description**

Processes with lower nice values are ignored.

**Argument type**

Int(range=range(-15, 20))

**Default**

-10

`--stop_start_min_id:`**Description**

Processes with lower id are ignored.

**Argument type**

Int(constraint=<function>)

**Default**

1500

`--stop_start_comm_prefixes:`

**Description**

Each process which name (lower cased) starts with one of the prefixes is not ignored. Overrides the decision based on the min\_id.

**Argument type**

ListOrTuple(Str())

**Default**

['ssh', 'xorg', 'bluetoothd']

--stop\_start\_comm\_prefixes\_ignored:

**Description**

Each process which name (lower cased) starts with one of the prefixes is ignored. It overrides the decisions based on comm\_prefixes and min\_id.

**Argument type**

ListOrTuple(Str())

**Default**

['dbus', 'kworker']

--stop\_start\_subtree\_suffixes:

**Description**

Suffixes of processes names which are stopped.

**Argument type**

ListOrTuple(Str())

**Default**

['dm', 'apache']

--stop\_start\_dry\_run|--no-stop\_start\_dry\_run:

**Description**

Just output the to be stopped processes but don't actually stop them?

**Default**

False

--sync|--no-sync:

**Description**

Enable: Calls sync before each program execution.

**Default**

None

--sleep|--no-sleep:

**Description**

Enable: Sleep a given amount of time before the benchmarking begins.

**Default**

None

--sleep\_seconds:

**Description**

Seconds to sleep

**Argument type**

Int(constraint=<function>)

**Default**

10

`--drop_fs_caches|--no-drop_fs_caches:`**Description**

Enable: Drop page cache, directoy entries and inodes before every benchmarking run.

**Default**

None

`--drop_fs_caches_free_pagecache|--no-drop_fs_caches_free_pagecache:`**Description**

Free the page cache

**Default**

True

`--drop_fs_caches_free_dentries_inodes|--no-drop_fs_caches_free_dentries_inodes:`**Description**

Free dentries and inodes

**Default**

True

`--disable_swap|--no-disable_swap:`**Description**

Enable: Disables swapping on the system before the benchmarking and enables it after.

**Default**

None

`--disable_cpu_caches|--no-disable_cpu_caches:`**Description**

Enable: Disable the L1 and L2 caches on x86 and x86-64 architectures.

**Default**

None

`--flush_cpu_caches|--no-flush_cpu_caches:`**Description**

Enable: Flushes the CPU caches on a x86 CPU using a small kernel module,

**Default**

None

`--cpu_governor|--no-cpu_governor:`**Description**

Enable: Allows the setting of the scaling governor of all cpu cores, to ensure that all use the same.

**Default**

None

`--cpu_governor_governor:`

**Description**

New scaling governor for all cpus

**Argument type**

Str()

**Default**

'performance'

--disable\_aslr|--no-disable\_aslr:

**Description**

Enable: Disable address space randomization

**Default**

None

--disable\_ht|--no-disable\_ht:

**Description**

Enable: Disable hyper-threading

**Default**

None

--disable\_turbo\_boost|--no-disable\_turbo\_boost:

**Description**

Enable: Disable amd and intel turbo boost

**Default**

None

--disable\_intel\_turbo|--no-disable\_intel\_turbo:

**Description**

Enable: Disable intel turbo mode

**Default**

None

--disable\_amd\_boost|--no-disable\_amd\_boost:

**Description**

Enable: Disable amd turbo boost

**Default**

None

--cpuset|--no-cpuset:

**Description**

Enable: Enable cpusets, simply sets run/cpuset/active to true

**Default**

None

--discarded\_runs|--no-discarded\_runs:

**Description**

Enable: Sets run/discarded\_runs

**Default**

None

--discarded\_runs\_runs:

**Description**

Number of discarded runs

**Argument type**

Int(constraint=<function>)

**Default**

1

--runner:

**Description**

If not "" overrides the runner setting for each program block

**Argument type**

ExactEither('', 'perf\_stat', 'rusage', 'spec', 'spec.py', 'time', 'output')

**Default**

''

--random\_cmd | --no-random\_cmd:

**Description**

Pick a random command if more than one run command is passed.

**Default**

True

--preset:

**Description**

Enable other plugins by default: none = (enable none by default); all = cpu\_governor,disable\_swap,sync,stop\_start,other\_nice,nice,disable\_aslr,disable\_ht,cpuset,disable\_turbo\_boost (enable all, might freeze your system); usable = cpu\_governor,disable\_swap,sync,nice,disable\_aslr,disable\_ht,cpuset,disable\_turbo\_boost (like 'all' but doesn't affect other processes)

**Argument type**

ExactEither('none','all','usable')

**Default**

'none'

--parse\_output | --no-parse\_output:

**Description**

Parse the program output as a YAML dictionary of that gives for a specific property a measurement. Not all runners support it.

**Default**

False

--plugin\_order:

**Description**

Order in which the plugins are used, plugins that do not appear in this list are used before all others

**Argument type**

ListOrTuple(Str())

**Default**

['drop\_fs\_caches', 'sync', 'sleep', 'preheat', 'flush\_cpu\_caches']

--argument:

**Description**

Argument passed to all benchmarked commands by replacing \$ARGUMENT with this value in the command

**Argument type**

Str()

**Default**

''

--cpuset\_active|--no-cpuset\_active:

**Description**

Use cpuset functionality?

**Default**

False

--cpuset\_base\_core\_number:

**Description**

Number of cpu cores for the base (remaining part of the) system

**Argument type**

Int(range=range(0, 2))

**Default**

1

--cpuset\_parallel:

**Description**

0: benchmark sequential, > 0: benchmark parallel with n instances, -1: determine n automatically

**Argument type**

Int(constraint=<function>)

**Default**

0

--cpuset\_sub\_core\_number:

**Description**

Number of cpu cores per parallel running program.

**Argument type**

Int(range=range(0, 2))

**Default**

1

--cpuset\_temci\_in\_base\_set|--no-cpuset\_temci\_in\_base\_set:

**Description**

place temci in the same cpu set as the rest of the system?

**Default**

True

--discarded\_runs:



**Description**

First n runs that are discarded

**Argument type**

Int(constraint=<function>)

**Default**

1

--min\_runs:

**Description**

Minimum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

20

--max\_runs:

**Description**

Maximum number of benchmarking runs

**Argument type**

Int(constraint=<function>)

**Default**

100

--runs:

**Description**

if != -1 sets max and min runs to its value

**Argument type**

Int(constraint=<function>)

**Default**

-1

--max\_time:

**Description**

Maximum time the whole benchmarking should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

--max\_block\_time:

**Description**

Maximum time one run block should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

`--run_block_size:`

**Description**

Number of benchmarking runs that are done together

**Argument type**

Int(constraint=<function>)

**Default**

1

`--in:`

**Description**

Input file with the program blocks to benchmark

**Argument type**

Str()

**Default**

'input.exec.yaml'

`--out:`

**Description**

Output file for the benchmarking results

**Argument type**

Str()

**Default**

'run\_output.yaml'

`--store_often|--no-store_often:`

**Description**

Store the result file after each set of blocks is benchmarked

**Default**

False

`--included_blocks:`

**Description**

List of included run blocks (all: include all), or their tag attribute or their number in the file (starting with 0), can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

`--disable_hyper_threading|--no-disable_hyper_threading:`

**Description**

Disable the hyper threaded cores. Good for cpu bound programs.

**Default**

False

`--show_report|--no-show_report:`

**Description**

Print console report if log\_level=info

**Default**

True

`--append|--no-append:`**Description**

Append to the output file instead of overwriting by adding new run data blocks

**Default**

False

`--shuffle|--no-shuffle:`**Description**

Randomize the order in which the program blocks are benchmarked.

**Default**

True

`--send_mail:`**Description**

If not empty, recipient of a mail after the benchmarking finished.

**Argument type**

Str()

**Default**

''

`--discard_all_data_for_block_on_error|--no-discard_all_data_for_block_on_error:`**Description**

Discard all run data for the failing program on error

**Default**

False

`--record_errors_in_file|--no-record_errors_in_file:`**Description**

Record the caught errors in the run\_output file

**Default**

True

`--no_build|--no-no_build:`**Description**

Do not build if build configs are present, only works if the working directory of the blocks does not change

**Default**

False

`--only_build|--no-only_build:`**Description**

Only build

**Default**

False

`--abort_after_build_error|--no-abort_after_build_error:`

**Description**

Abort after the first failing build

**Default**

True

--watch|--no-watch:

**Description**

Show the report continuously

**Default**

False

--watch\_every:

**Description**

Update the screen nth run (less updates are better for benchmarks)

**Argument type**

Int(constraint=<function>)

**Default**

1

--driver:

**Description**

Possible run drivers are 'exec' and 'shell'

**Argument type**

ExactEither('exec'|'shell')

**Default**

'exec'

--stats\_properties:

**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

--stats\_uncertainty\_range:

**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(<class 'float'>):<function>, T(<class 'float'>):<function>)

**Default**

[0.05, 0.15]

--stats\_tester:

**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

`--report_number_parentheses|--no-report_number_parentheses:`**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

`--report_number_min_decimal_places:`**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

3

`--report_number_max_decimal_places:`**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

5

`--report_number_scientific_notation|--no-report_number_scientific_notation:`**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

`--report_number_scientific_notation_si_prefixes|--no-report_number_scientific_notation_si_prefixes:`**Description**

Use si prefixes instead of 'e...'

**Default**

True

`--report_number_omit_insignificant_decimal_places|--no-report_number_omit_insignificant_decimal_places:`**Description**

Omit insignificant decimal places

**Default**

False

`--report_number_force_min_decimal_places|--no-report_number_force_min_decimal_places:`**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

`--report_number_percentages|--no-report_number_percentages:`**Description**

Show as percentages

**Default**

False

`--report_number_sigmas:`**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

2

`--report_number_parentheses_mode:`**Description**

Mode for showing the parentheses: either d (Digits are considered significant if they don't change if the number itself changes  $\pm \text{\$sigmas} * \text{std dev}$ ) or o (digits are considered significant if they are bigger than  $\text{\$sigmas} * \text{std dev}$ )

**Argument type**

ExactEither('d'|'o')

**Default**

'o'

`--discarded_runs:`**Description**

First n runs that are discarded

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

`--min_runs:`**Description**

Minimum number of benchmarking runs

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

20

`--max_runs:`**Description**

Maximum number of benchmarking runs

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

100

`--runs:`**Description**

if != -1 sets max and min runs to its value

**Argument type**

Int(constraint=<function>)

**Default**

-1

`--max_time:`**Description**

Maximum time the whole benchmarking should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

`--max_block_time:`**Description**

Maximum time one run block should take, -1 == no timeout, supports normal time span expressions

**Argument type**

ValidTimespan()

**Default**

'-1'

`--run_block_size:`**Description**

Number of benchmarking runs that are done together

**Argument type**

Int(constraint=<function>)

**Default**

1

`--in:`**Description**

Input file with the program blocks to benchmark

**Argument type**

Str()

**Default**

'input.exec.yaml'

`--out:`**Description**

Output file for the benchmarking results

**Argument type**

Str()

**Default**

'run\_output.yaml'

`--store_often|--no-store_often:`**Description**

Store the result file after each set of blocks is benchmarked

**Default**

False

`--included_blocks:`**Description**

List of included run blocks (all: include all), or their tag attribute or their number in the file (starting with 0), can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

`--disable_hyper_threading|--no-disable_hyper_threading:`**Description**

Disable the hyper threaded cores. Good for cpu bound programs.

**Default**

False

`--show_report|--no-show_report:`**Description**

Print console report if log\_level=info

**Default**

True

`--append|--no-append:`**Description**

Append to the output file instead of overwriting by adding new run data blocks

**Default**

False

`--shuffle|--no-shuffle:`**Description**

Randomize the order in which the program blocks are benchmarked.

**Default**

True

`--send_mail:`**Description**

If not empty, recipient of a mail after the benchmarking finished.

**Argument type**

Str()



**Default**

''

`--discard_all_data_for_block_on_error|--no-discard_all_data_for_block_on_error:`**Description**

Discard all run data for the failing program on error

**Default**

False

`--record_errors_in_file|--no-record_errors_in_file:`**Description**

Record the caught errors in the run\_output file

**Default**

True

`--no_build|--no-no_build:`**Description**

Do not build if build configs are present, only works if the working directory of the blocks does not change

**Default**

False

`--only_build|--no-only_build:`**Description**

Only build

**Default**

False

`--abort_after_build_error|--no-abort_after_build_error:`**Description**

Abort after the first failing build

**Default**

True

`--watch|--no-watch:`**Description**

Show the report continuously

**Default**

False

`--watch_every:`**Description**

Update the screen nth run (less updates are better for benchmarks)

**Argument type**

Int(constraint=&lt;function&gt;)

**Default**

1

`--driver:`

**Description**

Possible run drivers are 'exec' and 'shell'

**Argument type**

ExactEither('exec'|'shell')

**Default**

'exec'

--stats\_properties:

**Description**

Properties to use for reporting and null hypothesis tests, can be regular expressions

**Argument type**

ListOrTuple(Str())

**Default**

['all']

--stats\_uncertainty\_range:

**Description**

Range of p values that allow no conclusion.

**Argument type**

Tuple(T(<class 'float'>):<function>, T(<class 'float'>):<function>)

**Default**

[0.05, 0.15]

--stats\_tester:

**Description**

Possible testers are 't', 'ks' and 'anderson'

**Argument type**

ExactEither('t'|'ks'|'anderson')

**Default**

't'

--report\_number\_parentheses|--no-report\_number\_parentheses:

**Description**

Show parentheses around non significant digits? (If a std dev is given)

**Default**

True

--report\_number\_min\_decimal\_places:

**Description**

The minimum number of shown decimal places if decimal places are shown

**Argument type**

Int(constraint=<function>)

**Default**

3

--report\_number\_max\_decimal\_places:

**Description**

The maximum number of decimal places

**Argument type**

Int(constraint=<function>)

**Default**

5

--report\_number\_scientific\_notation|--no-report\_number\_scientific\_notation:

**Description**

Use the exponential notation, i.e. '10e3' for 1000

**Default**

True

--report\_number\_scientific\_notation\_si\_prefixes|--no-report\_number\_scientific\_notation\_si\_pref:

**Description**

Use si prefixes instead of 'e...'

**Default**

True

--report\_number\_omit\_insignificant\_decimal\_places|--no-report\_number\_omit\_insignificant\_decima:

**Description**

Omit insignificant decimal places

**Default**

False

--report\_number\_force\_min\_decimal\_places|--no-report\_number\_force\_min\_decimal\_places:

**Description**

Don't omit the minimum number of decimal places if insignificant?

**Default**

True

--report\_number\_percentages|--no-report\_number\_percentages:

**Description**

Show as percentages

**Default**

False

--report\_number\_sigmas:

**Description**

Number of standard deviation used for the digit significance evaluation

**Argument type**

Int(constraint=<function>)

**Default**

2

--report\_number\_parentheses\_mode:

**Description**

Mode for showing the parentheses: either d (Digits are considered significant if

they don't change if the number itself changes  $\pm \sigma$  or  $\sigma$  (digits are considered significant if they are bigger than  $\sigma$ )

**Argument type**

ExactEither('d'|'o')

**Default**

'o'

temci.scripts.cli.temci\_\_version(\*\*kwargs)

**Language**

python

**Command**

temci version

**Description**

Print the current version (0.8.5)

**Options:**

--settings:

**Description**

Additional settings file

**Argument type**

Str()

**Default**

''

--config:

**Description**

Alias for settings

**Argument type**

Str()

**Default**

''

--tmp\_dir:

**Description**

Used temporary directory

**Argument type**

Str()

**Default**

'/tmp/temci'

--log\_level:

**Description**

Logging level

**Argument type**

ExactEither('debug'|'info'|'warn'|'error'|'quiet')

**Default**

'info'

--sudo | --no-sudo:

**Description**

Acquire sudo privileges and run benchmark programs with non-sudo user. Only supported on the command line.

**Default**

False

## temci.scripts.temci\_completion module

Just a more performant version of *temci completion* that rebuilds the completion files only if the temci version changed. The advantage over using *temci completion* directly is, that it's normally significantly faster.

Usage:

```

```
temci_completion [zsh|bash]
```

``` This returns the location of the completion file.

`temci.scripts.temci_completion.cli()`

Process the command line arguments and call `temci completion` if needed.

`temci.scripts.temci_completion.completion_dir() → str`

Get the name of the completion directory

`temci.scripts.temci_completion.completion_file_name(shell: str) → str`

Get the completion file name for the passed shell and the current temci version

`temci.scripts.temci_completion.create_completion_dir() → str`

Create the directory for the completion files if it doesn't already exist.

`temci.scripts.temci_completion.print_help()`

## temci.scripts.version module

Contains the current version of temci.

`temci.scripts.version.version = '0.8.5'`

The current version of temci

## Module contents

This directory contains the command line interface and tab completion code and also the several wrapper scripts and the projects C++ code in sub directories.

## temci.setup package

### Submodules

#### temci.setup.setup module

This module helps to build the C and C++ code in the scripts directory.

**exception** temci.setup.setup.**ExecError**(cmd: str, out: str, err: str)

Bases: Exception

Error raised if a command failed.

**cmd**

Failed command

**err**

Error output of the command

**out**

Output of the command

temci.setup.setup.**exec**(dir: str, cmd: str)

Run the passed command in the passed directory

#### Parameters

- **dir** – passed directory
- **cmd** – passed command

#### Raises

**ExecError** – if the executed program has a > 0 error code

temci.setup.setup.**make\_scripts**(build\_kernel\_modules: bool = False)

Builds the C and C++ code inside the scripts directory.

#### Parameters

**build\_kernel\_modules** – build the kernel modules for disabling the CPU caches too

temci.setup.setup.**script\_relative**(file: str) → str

Returns the absolute version of the passed file name. :param file: passed file name relative to the scripts directory

### Module contents

## temci.report package

### Submodules

#### temci.report.report module

```
class temci.report.report.AbstractReporter(misc_settings=None, stats_helper:
Optional[RunDataStatsHelper] = None,
excluded_properties: Optional[List[str]] = None)
```

Bases: `object`

Produces a meaningful report out of measured data.

Creates an instance.

#### Parameters

- **misc\_settings** – configuration
- **stats\_helper** – used stats helper instance
- **excluded\_properties** – measured properties that are excluded from the reports

#### **misc**

Configuration

#### **report()**

Create a report and output or store it as configured.

#### **stats**

This object is used to simplify the work with the data and the statistics

#### **stats\_helper**

Used stats helper

#### **to\_long\_prop\_dict**

Maps a property name to a long property name

```
class temci.report.report.CSVReporter(misc_settings=None, stats_helper: Optional[RunDataStatsHelper]
                                     = None, excluded_properties: Optional[List[str]] = None)
```

Bases: `AbstractReporter`

Simple reporter that outputs just a configurable csv table with rows for each run block

Configuration format:

```
# List of valid column specs, format is a comma separated list of 'PROPERTY\[mod\]'
↳or 'ATTRIBUTE' mod
# is one of: mean, stddev, property, min, max and stddev per mean, optionally a
↳formatting option can
# be given viaPROPERTY\[mod|OPT1OPT2...\], where the OPTs are one of the following:
↳% (format as
# percentage), p (wrap insignificant digits in parentheses (+- 2 std dev)), s (use
↳scientific
# notation, configured in report/number) and o (wrap digits in the order of
↳magnitude of 2 std devs in
# parentheses). PROPERTY can be either the description or the short version of the
↳property. Configure
# the number formatting further via the number settings in the settings file
columns: [description]

# Output file name or standard out (-)
out: '-'
```

This reporter can be configured under the settings key `report/csv_misc`.

To use this reporter set the currently used reporter (at key `report/reporter`) to its name (`csv`). Other usable reporter are `console` and `html2`. The default reporter is `console`.

Creates an instance.

#### Parameters

- **misc\_settings** – configuration
- **stats\_helper** – used stats helper instance
- **excluded\_properties** – measured properties that are excluded from the reports

**report()** → Optional[str]

Create an report and output it as configured.

#### Returns

the report string if `to_string == True`

```
class temci.report.report.Codespeed2Reporter(misc_settings=None, stats_helper:
Optional[RunDataStatsHelper] = None,
excluded_properties: Optional[List[str]] = None)
```

Bases: [AbstractReporter](#)

Reporter that outputs JSON as specified by the [codespeed runner spec](#).

Creates an instance.

#### Parameters

- **misc\_settings** – configuration
- **stats\_helper** – used stats helper instance
- **excluded\_properties** – measured properties that are excluded from the reports

**report()**

Create a report and output it as configured.

```
class temci.report.report.CodespeedReporter(misc_settings=None, stats_helper:
Optional[RunDataStatsHelper] = None,
excluded_properties: Optional[List[str]] = None)
```

Bases: [AbstractReporter](#)

Reporter that outputs JSON as expected by [codespeed](#). Branch name and commit ID are taken from the current directory. Use it like this:

```
temci report --reporter codespeed ... | curl --data-urlencode json@- http://
↪localhost:8000/result/add/json/
```

Configuration format:

```
# Branch name reported to codespeed. Defaults to current branch or else 'master'.
branch: ''

# Commit ID reported to codespeed. Defaults to current commit.
commit_id: ''

# Environment name reported to codespeed. Defaults to current host name.
environment: ''

# Executable name reported to codespeed. Defaults to the project name.
executable: ''
```

(continues on next page)



(continued from previous page)

```
# Project name reported to codespeed.
project: ''
```

This reporter can be configured under the settings key *report/codespeed\_misc*.

To use this reporter set the currently used reporter (at key *report/reporter*) to its name (*codespeed*). Other usable reporter are *console*, *html2* and *csv*. The default reporter is *console*.

Creates an instance.

#### Parameters

- **misc\_settings** – configuration
- **stats\_helper** – used stats helper instance
- **excluded\_properties** – measured properties that are excluded from the reports

#### report()

Create a report and output it as configured.

```
class temci.report.report.ConsoleReporter(misc_settings=None, stats_helper:
Optional[RunDataStatsHelper] = None, excluded_properties:
Optional[List[str]] = None)
```

Bases: *AbstractReporter*

Simple reporter that outputs just text.

Configuration format:

```
# Matches the baseline block
baseline: ''

# Position of the baseline comparison: each: after each block, after: after each
↳ cluster, both: after
# each and after cluster, instead: instead of the non baselined
baseline_position: each

# 'auto': report clusters (runs with the same description) and singles (clusters with
↳ a single entry,
# combined) separately, 'single': report all clusters together as one, 'cluster':
↳ report all clusters
# separately, 'both': append the output of 'cluster' to the output of 'single'
mode: auto

# Output file name or `` (stdout)
out: '-'

# Report on the failing blocks
report_errors: true

# Print statistical tests for every property for every two programs
with_tester_results: true
```

This reporter can be configured under the settings key *report/console\_misc*.

To use this reporter set the currently used reporter (at key *report/reporter*) to its name (*console*).

The default reporter is *console*.

Creates an instance.

#### Parameters

- **misc\_settings** – configuration
- **stats\_helper** – used stats helper instance
- **excluded\_properties** – measured properties that are excluded from the reports

**report**(*with\_tester\_results: bool = True, to\_string: bool = False*) → Optional[str]

Create an report and output it as configured.

#### Parameters

- **with\_tester\_results** – include the hypothesis tester results
- **to\_string** – return the report as a string and don't output it?

#### Returns

the report string if `to_string == True`

```
class temci.report.report.HTMLReporter(misc_settings=None, stats_helper:
    Optional[RunDataStatsHelper] = None, excluded_properties:
    Optional[List[str]] = None)
```

Bases: [\*AbstractReporter\*](#)

Deprecated reporter that just lives as a hull. It might be useful to revive it as a basic reporter without JavaScript.

Creates an instance.

#### Parameters

- **misc\_settings** – configuration
- **stats\_helper** – used stats helper instance
- **excluded\_properties** – measured properties that are excluded from the reports

**report()**

Create a report and output or store it as configured.

```
class temci.report.report.HTMLReporter2(misc_settings=None, stats_helper:
    Optional[RunDataStatsHelper] = None, excluded_properties:
    Optional[List[str]] = None)
```

Bases: [\*AbstractReporter\*](#)

Reporter that produces a HTML based report with lot's of graphics. A rewrite of the original HTMLReporter

Configuration format:

```
# Alpha value for confidence intervals
alpha: 0.05

# Height per run block for the big comparison box plots
boxplot_height: 2.0

# Width of all big plotted figures
fig_width_big: 25.0

# Width of all small plotted figures
```

(continues on next page)

(continued from previous page)

```

fig_width_small: 15.0

# Format string used to format floats
float_format: '{:5.2e}'

# Override the contents of the output directory if it already exists?
force_override: false

# Generate pdf versions of the plotted figures?
gen_pdf: false

# Generate simple latex versions of the plotted figures?
gen_tex: true

# Generate excel files for all tables
gen_xls: false

# Hide warnings and errors related to statistical properties
hide_stat_warnings: false

# Name of the HTML file
html_filename: report.html

# Use local versions of all third party resources
local: false

# Show the mean related values in the big comparison table
mean_in_comparison_tables: true

# Show the minimum related values in the big comparison table
min_in_comparison_tables: false

# Output directory
out: report

# Format string used to format floats as percentages
percent_format: '{:5.2%}'

# Show zoomed out (x min = 0) figures in the extended summaries?
show_zoomed_out: true

```

This reporter can be configured under the settings key *report/html2\_misc*.

To use this reporter set the currently used reporter (at key *report/reporter*) to its name (*html2*). Another usable reporter is *console*. The default reporter is *console*.

Creates an instance.

#### Parameters

- **misc\_settings** – configuration
- **stats\_helper** – used stats helper instance
- **excluded\_properties** – measured properties that are excluded from the reports

`get_random_filename()` → str

`classmethod html_escape_property(property: str) → str`

`report()`

Create a report and output or store it as configured.

**class** temci.report.report.ReporterRegistry

Bases: [\*AbstractRegistry\*](#)

Registry for reporters.

The used reporter can be configured by editing the settings key *report/reporter*. Possible reporter are ‘console’, ‘html2’, ‘csv’, ‘codespeed’, ‘codespeed2’ and ‘velcom’

**default** = 'console'

Name(s) of the class(es) used by default. Type depends on the *use\_list* property.

**plugin\_synonym** = ('reporter', 'reporter')

Singular and plural version of the word that is used in the documentation for the registered entities

```
registry = {'codespeed': <class 'temci.report.report.CodespeedReporter'>,
'codespeed2': <class 'temci.report.report.Codespeed2Reporter'>, 'console': <class
'temci.report.report.ConsoleReporter'>, 'csv': <class
'temci.report.report.CSVReporter'>, 'html2': <class
'temci.report.report.HTMLReporter2'>, 'velcom': <class
'temci.report.report.VelcomReporter'>}
```

Registered classes (indexed by their name)

**settings\_key\_path** = 'report'

Used settings key path

**use\_key** = 'reporter'

Used key that sets which registered class is currently used

**use\_list** = False

Allow more than one class to used at a specific moment in time

**class** temci.report.report.VelcomReporter(*misc\_settings=None, stats\_helper:*  
*Optional[RunDataStatsHelper] = None, excluded\_properties:*  
*Optional[List[str]] = None)*

Bases: [\*AbstractReporter\*](#)

Reporter that outputs JSON as specified by the *velcom runner spec*.

Creates an instance.

#### Parameters

- **misc\_settings** – configuration
- **stats\_helper** – used stats helper instance
- **excluded\_properties** – measured properties that are excluded from the reports

`report()`

Create a report and output it as configured.

`temci.report.report.html_escape_property(property: str) → str`

Escape the name of a measured property.

**Parameters**

**property** – name of a measured property

**Returns**

escaped property name

## temci.report.report\_processor module

`class temci.report.report_processor.ReportProcessor(stats_helper: Optional[RunDataStatsHelper] = None)`

Bases: object

Simplifies the work with reporters.

Creates an instance.

**Parameters**

**stats\_helper** – used data wrapper or None if an empty one should be used

**report()**

Create a report with the used reporter

**reporter**

Used reporter

## temci.report.rundata module

Contains the RunData object for benchmarking data of specific program block and the RunDataStatsHelper that provides helper methods for working with these objects.

`class temci.report.rundata.ExcludedInvalidData`

Bases: object

Info object that contains information about the excluded invalid data.

**excluded\_properties\_per\_run\_data**

Run data descriptions mapped to the excluded properties per run data. Only includes not fully excluded run datas.

**excluded\_run\_datas**

Descriptions of the fully excluded run datas

`temci.report.rundata.Number`

Numeric value

alias of Union[int, float]

`class temci.report.rundata.RecordedError(message: str)`

Bases: object

`class temci.report.rundata.RecordedInternalError(message: str)`

Bases: [RecordedError](#)

**classmethod for\_exception(ex: BaseException) → [RecordedInternalError](#)**

```
class temci.report.rundata.RecordedProgramError(message: str, out: str, err: str, ret_code: int)
```

Bases: *RecordedError*

```
class temci.report.rundata.RunData(data: Optional[Dict[str, List[Union[int, float]]]] = None, attributes:
    Optional[Dict[str, str]] = None, recorded_error:
    Optional[RecordedError] = None, external: bool = False)
```

Bases: object

A set of benchmarking data for a specific program block.

An entry in the run output list

```
error:          Either(Dict({'message': Str(), 'return_code': Int(), 'output': Str(),
↪ 'error_output': Str()}), False, keys=Any, values=Any)|non existent)

internal_error:      Either(Dict({'message': Str()}), False, keys=Any, ↪
↪ values=Any)|non existent)

# Optional attributes that describe the block
attributes:
    description:      Optional(Str())

    # Tags of this block
    tags:             ListOrTuple(Str())

data:           Dict(, keys=Str(), values=List(Either(Int()|T(<class 'float'>))), ↪
↪ default = {}))
# with constrain: Either 'error' or 'internal_error' can be present
```

An entry in the run output list that specifies the long names for the properties

```
property_descriptions:      Either(non existent|Dict(, keys=Str(), values=Str()), ↪
↪ default = {}))
```

Initializes a new run data object.

#### Parameters

- **data** – optional dictionary mapping each property to a list of actual values
- **attributes** – dictionary of optional attributes that describe its program block
- **recorded\_error** – either program error or internal error
- **external** – does the data come from a prior benchmarking?

```
add_data_block(data_block: Dict[str, List[Union[int, float]]])
```

Adds a block of data.

#### Parameters

**data\_block** – maps each of the run datas properties to list of actual values (from each benchmarking run).

#### attributes

Dictionary of optional attributes that describe its program block

```
benchmarks() → int
```

Returns the maximum number of measured values for the associated program block over all properties.

```
block_type_scheme = Dict('data': Dict(, keys=Str(),
values=List(Either(Int()|T(<class 'float'>))), default = {}), 'attributes':
Dict('tags': ListOrTuple(Str()), 'description': Optional(Str()), keys=Str(),
values=Any, default = {'tags': [], 'description': ''}), 'error':
Either(Dict({'message': Str(), 'return_code': Int(), 'output': Str(),
'error_output': Str()}), False, keys=Any, values=Any)|non existent),
'internal_error': Either(Dict({'message': Str()}), False, keys=Any, values=Any)|non
existent), keys=Any, values=Any):Either 'error' or 'internal_error' can be present
```

```
clone(data: Optional[Dict[str, List[Union[int, float]]]] = None, attributes: Optional[Dict[str, str]] = None,
recorded_error: Optional[RecordedError] = None, external: Optional[bool] = None) → RunData
```

Clone this instance and replaces thereby some instance properties.

#### Parameters

- **data** – optional dictionary mapping each property to a list of actual values
- **attributes** – dictionary of optional attributes that describe its program block
- **external** – does the data come from a prior benchmarking?

#### Returns

new instance

#### data

Raw benchmarking data, mapping properties to their corresponding values

**description()** → str

Description of this instance based on the attributes

```
env_info_scheme = env_info: Either(non existent|List(Tuple(Str(), List(Tuple(Str(),
Str())))))
```

**exclude\_invalid()** → Tuple[Optional[RunData], List[str]]

Exclude properties that only have NaNs as measurements.

#### Returns

(new run data instance or None if all properties are excluded or the current if nothing changed,  
excluded properties)

**exclude\_properties(properties: List[str])** → RunData

Creates a new run data instance without the passed properties.

#### Parameters

**properties** – excluded properties

#### Returns

new run data instance

#### external

Does the data come from a prior benchmarking?

**get\_single\_properties()** → Dict[str, SingleProperty]

Returns single property stat objects per property that support statistical analysis (e.g. mean, standard deviation, ...)

#### Returns

stat object per property

**has\_error()** → bool

**include\_properties**(*properties: List[str]*) → *RunData*

Creates a new run data instance with only the passed properties.

**Parameters**

**properties** – included properties, can be regular expressions

**Returns**

new run data instance

**long\_properties**(*long\_versions: Dict[str, str]*) → *RunData*

Replace the short properties names with their long version from the passed dictionary.

**Parameters**

**long\_versions** – long versions of some properties

**Returns**

new run data instance (or current instance if nothing changed)

**min\_values**() → int

Returns the minimum number of measured values for the associated program block over all properties.

**properties**

List of measured properties. They might not all be measured the same number of times.

**property\_descriptions\_scheme = property\_descriptions: Either(non existent|Dict(, keys=Str(), values=Str(), default = {}))**

**to\_dict**() → Dict[str, Union[Dict[str, str], Dict[str, List[Union[int, float]]]]]

Returns a dictionary that represents this run data object.

```
class temci.report.rundata.RunDataStatsHelper(runs: List[RunData], tester: Optional[Tester] = None,  
                                             external_count: int = 0, property_descriptions:  
                                             Optional[Dict[str, str]] = None, erroneous_runs:  
                                             Optional[List[RunData]] = None, included_blocks:  
                                             Optional[str] = None, env_info: Optional[List[Tuple[str,  
                                             List[Tuple[str, str]]]] = None)
```

Bases: object

This class helps to simplify the work with a set of run data observations.

Don't use the constructor use `init_from_dicts` if possible.

**Parameters**

- **runs** – list of run data objects
- **tester** – used tester or tester that is set in the settings
- **external\_count** – Number of external program blocks (blocks for which the data was obtained in a

different benchmarking session) :param `property_descriptions`: mapping of some properties to their descriptions or longer versions :param `erroneous_runs`: runs that resulted in errors :param `included_blocks`: include query :param `env_info`: formatted environment info

**add\_data\_block**(*program\_id: int, data\_block: Dict[str, List[Union[int, float]]]*)

Add block of data for the program block with the given id.

**Parameters**

- **program\_id** – id of the program.



- **data\_block** – list of data from several benchmarking runs of the program block

#### Raises

**ValueError** – if the program block with the given id doesn't exist

**add\_error**(*program\_id: id, error: RecordedError*)

Set the error for a program

**add\_property\_descriptions**(*property\_descriptions: Dict[str, str]*)

Adds the given property descriptions.

#### Parameters

**property\_descriptions** – mapping of some properties to their descriptions or longer versions

**clone**(*runs: Optional[List[RunData]] = None, tester: Optional[Tester] = None, external\_count: Optional[int] = None, property\_descriptions: Optional[Dict[str, str]] = None*) → *RunDataStatsHelper*

Clones this instance and replaces the given instance properties.

#### Parameters

- **runs** – list of run data objects
- **tester** – used tester or tester that is set in the settings
- **external\_count** – Number of external program blocks (blocks for which the data was obtained in a

different benchmarking session) :param property\_descriptions: mapping of some properties to their descriptions or longer versions :return: cloned instance

**discard\_run\_data**(*id: int*)

Disable that run data object with the given id.

**estimate\_time**(*run\_bin\_size: int, min\_runs: int, max\_runs: int*) → float

Roughly estimates the time needed to finish benchmarking all program blocks. It doesn't take any parallelism into account. Therefore divide the number by the used parallel processes.

#### Warning

Doesn't work well.

#### Parameters

- **run\_bin\_size** – times a program block is benchmarked in a single block of time
- **min\_runs** – minimum number of allowed runs
- **max\_runs** – maximum number of allowed runs

#### Returns

estimated time in seconds or float("inf") if no proper estimation could be made

**estimate\_time\_for\_next\_round**(*run\_bin\_size: int, all: bool*) → float

Roughly estimates the time needed for the next benchmarking round.

#### Parameters

- **run\_bin\_size** – times a program block is benchmarked in a single block of time and the size of a round
- **all** – expect all program block to be benchmarked

**Returns**

estimated time in seconds

**exclude\_invalid()** → Tuple[RunDataStatsHelper, ExcludedInvalidData]

Exclude all properties of run datas that only have zeros or NaNs as measurements.

**Returns**

(new instance, info about the excluded data)

**exclude\_properties(properties: List[str])** → RunDataStatsHelper

Create a new instance without the passed properties.

**Parameters**

**properties** – excluded properties

**Returns**

new instance

**external\_count**

Number of external program blocks (blocks for which the data was obtained in a different benchmarking session)

**get\_description\_clusters()** → Dict[str, List[RunData]]

Set of runs per description, call RunDataStatsHelper.make\_descriptions\_distinct first

**Returns**

set of runs per description

**get\_description\_clusters\_and\_single()** → Tuple[List[RunData], Dict[str, List[RunData]]]

Set of runs per description, call RunDataStatsHelper.make\_descriptions\_distinct first

**Returns**

set of runs per description

**get\_evaluation(with\_equal: bool, with\_unequal: bool, with\_uncertain: bool, blocks:**

*Optional[List[RunData]] = None*) → dict

Structure of the returned list items:

```
- data: # set of two run data objects
properties: # information for each property that is equal, ...
-prop:
  - equal: True/False
    uncertain: True/False
    p_val: probability of the null hypothesis
    speed_up: speed up from the first to the second
    description: description of the property
```

**Parameters**

- **with\_equal** – with tuple with at least one “equal” property
- **with\_unequal** – ... unequal property
- **with\_uncertain** – include also uncertain properties
- **blocks** – blocks to compare

**Returns**

list of tuples for which at least one property matches the criteria

**get\_program\_ids\_to\_bench()** → List[int]

Returns the ids (the first gets id 0, ...) of the program block / run data object that should be benchmarked again.

**has\_error**(*program\_id: int*) → bool

Is there an error recorded for the program with the given id?

**include\_properties**(*properties: List[str]*) → *RunDataStatsHelper*

Create a new instance with only the passed properties

#### Parameters

**properties** – included properties, can be regular expressions

#### Returns

new instance

**classmethod init\_from\_dicts**(*runs: Optional[List[Union[Dict[str, str], Dict[str, List[Union[int, float]]]]]] = None, external: bool = False, included\_blocks: Optional[str] = None*) → *RunDataStatsHelper*

Expected structure of the stats settings and the runs parameter:

```
"stats": {
    "tester": ...,
    "properties": ["prop1", ...],
    # or
    "properties": ["prop1", ...],
    "uncertainty_range": (0.1, 0.3)
}

"runs": [
    {"attributes": {"attr1": ..., ..., ["description": ...], ["tags": ...]},
    "data": {"__ov-time": [...], ...},
    "error": {"return_code": ..., "output": "...", "error_output": "..."},
    "internal_error": {"message": "..."} (either "error" or "internal_error"
    ↪ might be present)
    ["property_descriptions": {"__ov-time": "Overall time", ...}]
    ["env_info": ... ]
    },
    ...
]
```

#### Parameters

- **runs** – list of dictionaries representing the benchmarking runs for each program block
- **external** – are the passed runs not from this benchmarking session but from another?
- **included\_blocks** – include query

#### Raises

**ValueError** – if runs parameter has an incorrect structure

#### Returns

created stats helper

**classmethod init\_from\_file**(*filename: str*) → *RunDataStatsHelper*

Load the runs from the passed file. This file has to encode the runs using YAML in the format stated in *RunDataStatsHelper.init\_from\_dicts()*

**Parameters**

**filename** – name of the file

**Raises**

**ValueError** – if the file has an incorrect structure

**Returns**

created stats helper

**is\_equal**(*p\_val: float*) → bool

Is the passed value above the uncertainty range for null hypothesis probabilities?

**is\_uncertain**(*p\_val: float*) → bool

Does the passed probability of the null hypothesis for two samples lie in the uncertainty range? :param p\_val: passed probability of the null hypothesis

**is\_unequal**(*p\_val: float*) → bool

Is the passed value above the uncertainty range for null hypothesis probabilities?

**long\_properties**(*property\_format: str = '[{}]'*) → Tuple[RunDataStatsHelper, Dict[str, str]]

Replace the short properties names with their descriptions if possible.

**Parameters**

**property\_format** – format string that gets a property description and produces a longer property name

**Returns**

new instance, a dict that returns a long property name for a given short property name

**make\_descriptions\_distinct**()

Append numbers to descriptions if needed to make them unique

**properties**() → List[str]

Returns a sorted list of all properties that exist in all run data blocks.

**runs**

Data of serveral runs from several measured program blocks

**serialize**() → List[Union[Dict[str, str], Dict[str, List[Union[int, float]]], List[Tuple[str, List[Tuple[str, str]]]]]]

Serialize this instance into a data structure that is accepted by the `init_from_dicts` method.

**tester**

Used statistical tester

**update\_env\_info**()

Obtain the environment information for the current system and store it

**valid\_runs**() → List[RunData]

Number of valid (with measured data) runs

`temci.report.rundata.get_for_tag(per_tag_settings_key: str, main_key: str, tag: Optional[str])`

`temci.report.rundata.get_for_tags(per_tag_settings_key: str, main_key: str, tags: List[str], combinator: Callable[[Any, Any], Any])`

**temci.report.stats module**

Statistical helper classes for tested pairs and single blocks.

**class** temci.report.stats.**BaseStatObject**

Bases: object

Class that gives helper methods for the extending stat object classes.

**description()** → str

Returns the description of this stat object.

Should be implemented by sub classes of BaseStatObject.

**eq\_except\_property**(other) → bool

Is this stat object equal to the passed stat object (without regarding the actual property)?

Implemented by sub classes of BaseStatObject.

**errors()** → List[StatMessage]

Returns a list of all (merged) statistical errors for this stat object.

**get\_data\_frame**(\*\*kwargs)

Get the data frame that is associated with this stat object.

Implemented by sub classes of BaseStatObject. :return pd.DataFrame instance

**get\_stat\_messages()** → List[StatMessage]

Returns a list of all (merged) statistical warnings and errors for this stat object.

**has\_errors()** → bool

Does this stat object has any statistical errors?

**has\_warnings()** → bool

Does this stat object has any statistical errors?

**histogram**(fig\_width: Union[float, int], fig\_height: Optional[Union[int, float]] = None, x\_ticks: Optional[List[Union[int, float]]] = None, y\_ticks: Optional[List[Union[int, float]]] = None, show\_legend: Optional[bool] = None, type: Optional[str] = None, align: str = 'mid', x\_label: Optional[str] = None, y\_label: Optional[str] = None, zoom\_in: bool = True, other\_objs: Optional[List[BaseStatObject]] = None, other\_obj\_names: Optional[List[str]] = None, own\_name: Optional[str] = None, \*\*kwargs)

Plots a histogram as the current figure. Don't forget to close it via fig.close()

**Parameters**

- **fig\_width** – width of the figure in cm
- **fig\_height** – height of the figure in cm
- **x\_ticks** – None: use default ticks, list: use the given ticks
- **y\_ticks** – None: use default ticks, list: use the given ticks
- **show\_legend** – show a legend in the plot? If None only show one if there are more than one sub histograms
- **type** – histogram type (either 'bar', 'barstacked', 'step', 'stepfilled' or None for auto)
- **align** – controls where each bar centered ('left', 'mid' or 'right')
- **x\_label** – if not None, shows the given x label

- **y\_label** – if not None: shows the given y label
- **zoom\_in** – does the x axis start at the minimum x value?
- **kwargs** – optional arguments passed to the `get_data_frame` method
- **other\_objs** – additional objects to plot on the same histogram (only `SingleProperty` objects allowed)
- **other\_obj\_names** – names of the additional objects
- **own\_name** – used with `other_objs` option

**img\_filename\_ending** = `'.svg'`

File ending for image file (and therefore their format)

**is\_single\_valued()** → bool

Does the data consist only of one unique value?

**reset\_plt()**

Reset the current matplotlib plot style.

**store\_figure**(*filename: str, fig\_width: float, fig\_height: Optional[float] = None, pdf: bool = True, tex: bool = True, tex\_standalone: bool = True, img: bool = True, zoom\_in: bool = False*) → Dict[str, str]

**warnings()** → List[*StatMessage*]

Returns a list of all (merged) statistical warnings for this stat object.

**class** `temci.report.stats.EffectToSmallError`(*parent: BaseStatObject, properties: Union[List[str], str], values: Union[List[Union[int, float]], float, int]*)

Bases: *EffectToSmallWarning*

Error message regarding an only insignificant mean difference regarding the standard deviation.

Create a new instance.

#### Parameters

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

**border\_value** = 2

**type** = 10

Type of this *StatMessage*, warning or error

**class** `temci.report.stats.EffectToSmallWarning`(*parent: BaseStatObject, properties: Union[List[str], str], values: Union[List[Union[int, float]], float, int]*)

Bases: *StatWarning*

Warning regarding a not really significant mean difference regarding the standard deviation.

Create a new instance.

#### Parameters

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

```
border_value = 1
```

```
classmethod check_value(value) → bool
```

Is this type of message inappropriate for the passed value of the related statistical property?

```
hint = 'Try to reduce the standard deviation if you think that the measured
difference is significant: If you use the exec run driver, consider using a preset.
Also consider increasing the number of measured runs.'
```

Text that should give a hint about what to do to prevent the possible mistake

```
message = 'The mean difference per standard deviation of {props} is less than
{b_val}.'
```

Format string for the message, {b\_val} is replaced by the border value and {props} by the list of properties and program block names that have this problem.

```
value_format = '{:5.3f}'
```

Format string used to format the border and the actual value

```
class temci.report.stats.NotEnoughObservationsError(parent: BaseStatObject, properties:
                                                    Union[List[str], str], values:
                                                    Union[List[Union[int, float]], float, int])
```

Bases: [NotEnoughObservationsWarning](#)

Error message regarding too few observations or measurements.

Create a new instance.

#### Parameters

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

```
border_value = 15
```

```
type = 10
```

Type of this StatMessage, warning or error

```
class temci.report.stats.NotEnoughObservationsWarning(parent: BaseStatObject, properties:
                                                    Union[List[str], str], values:
                                                    Union[List[Union[int, float]], float, int])
```

Bases: [StatWarning](#)

Warning regarding too few observations or measurements.

Create a new instance.

#### Parameters

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

```
border_value = 30
```

```
classmethod check_value(value) → bool
```

Is this type of message inappropriate for the passed value of the related statistical property?

```
hint = 'Increase the number of measured runs.'
```

Text that should give a hint about what to do to prevent the possible mistake

```
message = 'The number of observations of {props} is less than {b_val}.'
```

Format string for the message, {b\_val} is replaced by the border value and {props} by the list of properties and program block names that have this problem.

```
value_format = '{}'
```

Format string used to format the border and the actual value

```
temci.report.stats.Number
```

Numeric value type

alias of Union[float, int]

```
class temci.report.stats.SignificanceTooLowError(parent: BaseStatObject, properties: Union[List[str],
                                                                                          str], values: Union[List[Union[int, float]], float,
                                                                                          int])
```

Bases: [SignificanceTooLowWarning](#)

Error message regarding an only insignificant difference regarding a statistical test. The probability of the null hypothesis is larger than the larger end of the configured uncertainty range.

Create a new instance.

#### Parameters

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

```
classmethod check_value(value) → bool
```

Is this type of message inappropriate for the passed value of the related statistical property?

```
type = 10
```

Type of this StatMessage, warning or error

```
class temci.report.stats.SignificanceTooLowWarning(parent: BaseStatObject, properties:
                                                                                          Union[List[str], str], values:
                                                                                          Union[List[Union[int, float]], float, int])
```

Bases: [StatWarning](#)

Warning regarding an only insignificant difference regarding a statistical test. The probability of the null hypothesis lies in the configured uncertainty range.

Create a new instance.

#### Parameters

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

```
classmethod check_value(value) → bool
```

Is this type of message inappropriate for the passed value of the related statistical property?

```
hint = 'Increase the number of benchmarking runs.'
```

Text that should give a hint about what to do to prevent the possible mistake



```
message = 'The used statistical significance test showed that the significance of
the\n difference with {props} is too low.'
```

Format string for the message, {b\_val} is replaced by the border value and {props} by the list of properties and program block names that have this problem.

```
class temci.report.stats.Single(data: Union[RunData, Single])
```

Bases: *BaseStatObject*

A statistical wrapper around a single run data object.

Create an instance.

#### Parameters

**data** – run data wrapped by this instance or another instance of which the run data is used

#### attributes

Attributes for this instance

**description()** → str

Returns the description of this stat object.

Should be implemented by sub classes of BaseStatObject.

**eq\_except\_property(other)** → bool

Is this stat object equal to the passed stat object (without regarding the actual property)?

Implemented by sub classes of BaseStatObject.

**get\_data\_frame()**

Get the data frame that is associated with this stat object.

Implemented by sub classes of BaseStatObject. :return pd.DataFrame instance

#### properties

SingleProperty objects for each property

#### rundata

Run data wrapped by this instance

```
class temci.report.stats.SingleProperty(parent: Single, data: Union[RunData, SingleProperty],
                                         property: str)
```

Bases: *BaseStatObject*

A statistical wrapper around a single run data block for a specific measured property.

Creates an instance.

#### Parameters

- **parent** – parent single object that contains this instance
- **data** – measured data for all properties or another single property instance of which the data is used
- **property** – actually measured property

#### array

NumPy array version of the measured data for the specific property. Using this in NumPy contexts might speed up the calculations.

#### data

Measured data for the specific property

**description()** → str

Returns the description of this stat object.

Should be implemented by sub classes of BaseStatObject.

**eq\_except\_property**(*other*) → bool

Is this stat object equal to the passed stat object (without regarding the actual property)?

Implemented by sub classes of BaseStatObject.

**get\_data\_frame()**

Get the data frame that is associated with this stat object.

Implemented by sub classes of BaseStatObject. :return pd.DataFrame instance

**iqr()** → float

Calculates the interquartile range.

**is\_single\_valued()** → bool

Does the data consist only of one unique value?

**max()** → float

Maximum value of the measurements

**mean()** → float

Mean value of the measurements

**mean\_ci**(*alpha: float*) → Tuple[float, float]

Calculates the confidence interval in which the population mean lies with the given probability. Assumes normal distribution.

Adopted from <http://stackoverflow.com/a/15034143>

**Parameters**

**alpha** – given probability

**Returns**

lower, upper bound

**median()** → float

Median of the measurements

**min()** → float

Minimum value of the measurements

**normality()** → float

Calculates the probability of the data being normal distributed.

**Warning**

return NaN if the number of observations is less than 8

**observations()** → int

Number of measurements or observations

**outliers**(*whis: float = 1.5*) → List[float]

Returns the values that don't lie in the in the range fenced by the whiskers.

**parent**

Parent single object that contains this instance

**percentile**(*q: int*) → float

Calculates the *q* th percentile. *q* must be between 0 and 100 inclusive.

**property**

Actually measured property

**quartiles**() → Tuple[float, float, float]

Calculates the 3 quartiles (1, 2 and 3)

**rundata**

RunData object that contains the measurements for all properties

**sem**() → float

Standard error of the mean (standard deviation / sqrt(observations))

**skewedness**() → float

Calculates the skewedness of the data.

**Warning**

return NaN if the number of observations is less than 8

**std**() → float

Standard deviation of the measurements

**std\_dev**() → float

Standard deviation of the measurements

**std\_dev\_ci**(*alpha: float*) → Tuple[float, float]

Calculates the confidence interval in which the standard deviation lies with the given probability. Assumes normal distribution.

Adopted from [http://www.stat.purdue.edu/~tlzhang/stat511/chapter7\\_4.pdf](http://www.stat.purdue.edu/~tlzhang/stat511/chapter7_4.pdf)

**Parameters**

**alpha** – given probability

**Returns**

lower, upper bound

**std\_dev\_per\_mean**() → float

Standard deviation per mean of the measurements (also known as variation coefficient)

**std\_devs**() → Tuple[float, float]

Calculates the standard deviation of elements <= mean and of the elements > mean.

**Returns**

(lower, upper)

**std\_error\_mean**() → float

Standard error of the mean (standard deviation / sqrt(observations))

**variance**() → float

Variance of the measurements

**whiskers**(*whis: float = 1.5*) → Tuple[float, float]

Calculates the upper and the lower whisker for a boxplot. I.e. the minimum and the maximum value of the data set lie in the range (Q1 - whis \* IQR, Q3 + whis \* IQR). IQR being the interquartil distance, Q1 the lower and Q2 the upper quartile.

Adapted from <http://stackoverflow.com/a/20096945>

**class** temci.report.stats.**SinglesProperty**(singles: List[Union[Single, SingleProperty]], property: str)

Bases: [BaseStatObject](#)

Creates an instance.

#### Parameters

- **singles** – compared single property objects or single objects that are turned into one
- **property** – regarded measured property

**boxplot**(fig\_width: Union[float, int], fig\_height: Optional[Union[int, float]] = None, zoom\_in: bool = False)

Creates a (horizontal) box plot comparing all single object for a given property.

#### Parameters

- **fig\_width** – width of the figure in cm
- **fig\_height** – height of the figure in cm, if None it is calculated from the figure width using the aesthetic ratio

**get\_data\_frame**(\*\*kwargs)

Get the data frame that is associated with this stat object.

Implemented by sub classes of BaseStatObject. :return pd.DataFrame instance

**max**() → float

Calculates the maximum value of all compared single property objects.

**property**

Regarded measured property

**singles**

Compared single property objects

**class** temci.report.stats.**StatError**(parent: [BaseStatObject](#), properties: Union[List[str], str], values: Union[List[Union[int, float]], float, int])

Bases: [StatWarning](#), [StatMessage](#)

A message that signals a possible warning that is probably severe

Create a new instance.

#### Parameters

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

**type** = 10

Type of this StatMessage, warning or error

**class** temci.report.stats.**StatMessage**(parent: [BaseStatObject](#), properties: Union[List[str], str], values: Union[List[Union[int, float]], float, int])

Bases: object

A message that warns of possible mistakes and gives hints. It's related to a statistical property.

Create a new instance.

#### Parameters

- **parent** – parent stat object

- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

**border\_value** = 0

**classmethod check\_value**(*value: Union[float, int]*) → bool

Is this type of message inappropriate for the passed value of the related statistical property?

**static combine**(\**messages: Tuple[Optional[StatMessage]]*) → List[StatMessage]

Combines all message of the same type and with the same parent in the passed list. Ignores None entries.

#### Parameters

**messages** – passed list of messages

#### Returns

new reduced list

**classmethod create\_if\_valid**(*parent: BaseStatObject, property: str, value: Union[float, int], \*\*kwargs*) → Optional[StatMessage]

Returns a message object if this type of message is appropriate for the passed value of the related statistical property or None otherwise.

#### Parameters

- **parent** – parent stat object
- **property** – property of the stat object for which this message is might by relevant and the value is given
- **value** – a value of the related statistical property
- **kwargs** – additional arguments for the constructor of this message type class

**generate\_msg\_text**(*show\_parent: bool*) → str

Generates the text of this message object.

#### Parameters

**show\_parent** – Is the parent shown in after the properties? E.g. “blub of bla parent: ...”

#### Returns

message text

**hint** = ''

Text that should give a hint about what to do to prevent the possible mistake

**message** = '{props}: {b\_val}'

Format string for the message, {b\_val} is replaced by the border value and {props} by the list of properties and program block names that have this problem.

**classmethod overridden**(*value: Union[int, float]*) → bool

**parent**

Parent stat object

**properties**

Properties of the stat object for which this message is relevant

**type** = None

Type of this StatMessage, warning or error

```
value_format = '{:5.3f}'
```

Format string used to format the border and the actual value

**values**

a value of the related statistical property for each relevant property

```
class temci.report.stats.StatMessageType(value)
```

Bases: Enum

Types of StatMessages.

**ERROR = 10**

Error type that signals a possible error or mistake, but is called ‘severe warning’

**WARNING = 5**

Warning type that signals a possible but not severe mistake

```
class temci.report.stats.StatMessageValueFormat
```

Bases: object

Format strings for ints, floats and percentage values used in the StatMessages.

**FLOAT = '{:5.3f}'**

Format string for float values that aren’t shown as percentages

**INT = '{}'**

Format string for int values

**PERCENT = '{:5.3%}'**

Format string for float values that are shown as percentages

```
class temci.report.stats.StatWarning(parent: BaseStatObject, properties: Union[List[str], str], values: Union[List[Union[int, float]], float, int])
```

Bases: [StatMessage](#)

A message that signals a possible warning that isn’t severe

Create a new instance.

**Parameters**

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

```
classmethod overridden(value: Union[int, float]) → bool
```

**type = 5**

Type of this StatMessage, warning or error

```
class temci.report.stats.StdDeviationToHighError(parent: BaseStatObject, properties: Union[List[str], str], values: Union[List[Union[int, float]], float, int])
```

Bases: [StdDeviationToHighWarning](#)

Error message regarding a far too high standard deviation.

Create a new instance.

**Parameters**

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

**border\_value** = 0.05

**type** = 10

Type of this StatMessage, warning or error

```
class temci.report.stats.StdDeviationToHighWarning(parent: BaseStatObject, properties:
                                                    Union[List[str], str], values:
                                                    Union[List[Union[int, float]], float, int])
```

Bases: [StatWarning](#)

A warning about a too high standard deviation.

Create a new instance.

#### Parameters

- **parent** – parent stat object
- **properties** – properties of the stat object for which this message is relevant
- **values** – a value of the related statistical property for each relevant property

**border\_value** = 0.01

**classmethod check\_value**(value) → bool

Is this type of message inappropriate for the passed value of the related statistical property?

**hint** = 'With the exec run driver you can probably use a preset. Also consider to increase the number of measured runs.'

Text that should give a hint about what to do to prevent the possible mistake

**message** = 'The standard deviation per mean of {props} is too high. It should be {b\_val}.'

Format string for the message, {b\_val} is replaced by the border value and {props} by the list of properties and program block names that have this problem.

**value\_format** = '{:5.3%}'

Format string used to format the border and the actual value

```
class temci.report.stats.TestedPair(first: Union[RunData, Single], second: Union[RunData, Single],
                                     tester: Optional[Tester] = None)
```

Bases: [BaseStatObject](#)

A statistical wrapper around two run data objects that are compared via a tester.

Creates an instance.

#### Parameters

- **first** – first of the two compared run data or single objects
- **second** – second of the two compared run data or single objects
- **tester** – used statistical tester, if None the default tester is used (configured in the settings)

**description()** → str

Returns the description of this stat object.

Should be implemented by sub classes of BaseStatObject.

**eq\_except\_property**(*other*) → bool

Is this stat object equal to the passed stat object (without regarding the actual property)?

Implemented by sub classes of BaseStatObject.

**first**

First of the two compared single objects

**first\_rel\_to\_second()** → float

Calculates the geometric mean of the first means relative to the second means. Ignores NaNs in calculating the geometric mean.

See [http://www.cse.unsw.edu.au/~cs9242/15/papers/Fleming\\_Wallace\\_86.pdf](http://www.cse.unsw.edu.au/~cs9242/15/papers/Fleming_Wallace_86.pdf)

**first\_rel\_to\_second\_std()** → float

Calculates the geometric standard deviation for the first\_rel\_to\_second method.

Ignores NaNs in calculating the geometric std.

**properties**

TestedPairProperty objects for each shared property of the inherited Single objects

**second**

Second of the two compared single objects

**swap()** → *TestedPair*

Creates a new pair with the elements swapped.

**Returns**

new pair object

**tester**

Used statistical tester for the comparisons

**class** temci.report.stats.**TestedPairProperty**(*parent*: *TestedPair*, *first*: *Single*, *second*: *Single*, *property*: *str*, *tester*: *Optional[Tester]* = *None*)

Bases: *BaseStatObject*

Statistical helper for comparing a pair of run data blocks for a specific measured property.

Creates an instance.

**Parameters**

- **parent** – parent tested pair object
- **first** – first of the two compared single objects
- **second** – second of the two compared single objects
- **property** – regarded property
- **tester** – used statistical tester, if None the default tester is used (configured in the settings)

**Returns**



**description()** → str

Returns the description of this stat object.

Should be implemented by sub classes of BaseStatObject.

**eq\_except\_property(*other*)** → bool

Is this stat object equal to the passed stat object (without regarding the actual property)?

Implemented by sub classes of BaseStatObject.

**equal\_prob()** → float

Probability of the null hypothesis being not not correct (tertiary logic).

**Returns**

p value between 0 and 1

**first**

First of the two compared single property objects

**first\_rel\_to\_second()** → float

Calculates the mean of the first relative to the mean of the second ( $\text{mean}(\text{first}) / \text{mean}(\text{second})$ ).

Returns 1 if both means are equal, regardless of their values.

**get\_data\_frame(*show\_property=True*)**

Get the data frame that is associated with this stat object.

Implemented by sub classes of BaseStatObject. :return pd.DataFrame instance

**is\_equal()** → Optional[bool]

Checks the null hypothesis.

**Returns**

True or False if the p val isn't in the uncertainty range of the tester, None else

**is\_single\_valued()** → bool

Does the data consist only of one unique value?

**max\_rel\_std\_dev()** → float

**max\_std\_dev()** → float

Calculates the maximum of the standard deviations of the first and the second.

**mean\_diff()** → float

Calculates the difference of the means between the first and the second single property object ( $\text{mean}(\text{first}) - \text{mean}(\text{second})$ ).

**mean\_diff\_ci(*alpha: float*)** → Tuple[float, float]

Calculates the confidence interval in which the mean difference lies with the given probability. Assumes normal distribution.

Adopted from <http://www.kean.edu/~fosborne/bstat/06b2means.html>

**Parameters**

**alpha** – given probability

**Returns**

lower, upper bound

**mean\_diff\_per\_dev()** → float

Calculates the mean difference per standard deviation (maximum of the first's and the second's deviation).

**mean\_diff\_per\_mean()** → float

Calculates the mean difference relative to the mean of the first ((mean(first) - mean(second)) / mean(first)).

**mean\_std\_dev()** → float

Calculates the mean of the standard deviations of the first and the second.

**min\_observations()** → int

Returns the minimum number of observations of the first and the second

**parent**

Parent tested pair object

**property**

Regarded specific property

**second**

Second of the two compared single property objects

**swap()** → *TestedPairProperty*

Swap the first and the second single object.

**Returns**

new instance

**tester**

Used statistical tester for the comparisons

**class** temci.report.stats.**TestedPairsAndSingles**(*singles: List[Union[RunData, Single]]*, *pairs: Optional[List[TestedPair]] = None*)

Bases: *BaseStatObject*

A wrapper around a list of tested pairs and singles.

Creates an instance.

**Parameters**

- **singles** – compared single objects or run data objects that are turned into single object
- **pairs** – compared pairs of single objects, if None they created out of the passed single objects

**get\_pair**(*first\_id: int*, *second\_id: int*) → *TestedPair*

Get the tested pair consisting of the two single objects with the passed ids. The id of a single objects is its index (starting at zero) in the internal single list.

**Parameters**

- **first\_id** – id of the first single object
- **second\_id** – id of the second single object

**Returns**

created tested pair object comparing the two single objects

**get\_stat\_messages()** → List[*StatMessage*]

Combines the messages for all inherited TestedPair and Single objects,

**Returns**

simplified list of all messages

**number\_of\_singles()** → int  
 Number of compared single objects

**property pairs:** `List[TestedPair]`

**properties()** → List[str]  
 Returns the properties that are shared among all single run data objects.

**singles**  
 Compared single objects

**singles\_properties**  
 Singles property object for every measured property

### temci.report.testers module

Contains the tester base class and several simple implementations that simplify the work with statistical hypothesis tests.

**class** temci.report.testers.**AndersonTester**(\*args, \*\*kwargs)

Bases: `Tester`

Tester that uses the Anderson statistic on 2 samples.

Creates a new instance. :param misc\_settings: Additional settings :param uncertainty\_range: (start, end) probability tuple that gives range in which the tester doesn't give

a definitive result on the nullhypothesis check

**name** = 'anderson'

Name of the implemented statistical test

**scipy\_stat\_method** = 'anderson\_ksamp'

Used method of the scipy.stats module if the \_test\_impl isn't reimplemented

**class** temci.report.testers.**KSTester**(\*args, \*\*kwargs)

Bases: `Tester`

Tester that uses the Kolmogorov-Smirnov statistic on 2 samples.

Creates a new instance. :param misc\_settings: Additional settings :param uncertainty\_range: (start, end) probability tuple that gives range in which the tester doesn't give

a definitive result on the nullhypothesis check

**name** = 'kolmogorov smirnov'

Name of the implemented statistical test

**scipy\_stat\_method** = 'ks\_2samp'

Used method of the scipy.stats module if the \_test\_impl isn't reimplemented

**class** temci.report.testers.**TTester**(\*args, \*\*kwargs)

Bases: `Tester`

Tester that uses the student's t test.

Creates a new instance. :param misc\_settings: Additional settings :param uncertainty\_range: (start, end) probability tuple that gives range in which the tester doesn't give

a definitive result on the nullhypothesis check

**name** = 't'

Name of the implemented statistical test

**scipy\_stat\_method** = 'ttest\_ind'

Used method of the scipy.stats module if the `_test_impl` isn't reimplemented

**class** temci.report.testers.**Tester**(\*args, \*\*kwargs)

Bases: object

A tester tests the probability of the null hypothesis of two same length list of observations.

This is a base class that shouldn't be instantiated.

Creates a new instance. :param misc\_settings: Additional settings :param uncertainty\_range: (start, end) probability tuple that gives range in which the tester doesn't give

a definitive result on the null hypothesis check

**estimate\_needed\_runs**(data1: list, data2: list, run\_bin\_size: int, min\_runs: int, max\_runs: int) → int

Calculate a approximation of the needed length of both observations that is needed for the p value to lie outside the uncertainty range.

It uses the simple observation that the graph of the p value plotted against the size of the sets has a exponential, logarithmic or root shape.

#### Warning

Doesn't work well.

#### Parameters

- **data1** – list of observations
- **data2** – list of observations
- **run\_bin\_size** – granularity of the observation (> 0)
- **min\_runs** – minimum number of allowed runs
- **max\_runs** – maximum number of allowed runs

#### Returns

approximation of needed runs or float("inf")

**is\_equal**(data1: List[Union[int, float]], data2: List[Union[int, float]]) → bool

Are the two samples not significantly unequal regarding the probability of the null hypothesis?

**is\_uncertain**(data1: List[Union[int, float]], data2: List[Union[int, float]]) → bool

Does the probability of the null hypothesis for two samples lie in the uncertainty range?

**is\_unequal**(data1: List[Union[int, float]], data2: List[Union[int, float]]) → bool

Are the two samples significantly unequal regarding the probability of the null hypothesis?

**misc\_settings**

Additional settings

**name** = ''

Name of the implemented statistical test

**scipy\_stat\_method** = None

Used method of the scipy.stats module if the `_test_impl` isn't reimplemented

**test**(*data1*: List[Union[int, float]], *data2*: List[Union[int, float]]) → float

Calculates the probability of the null hypotheses for two samples.

**uncertainty\_range**

(**start**, **end**) probability tuple that gives range in which the tester doesn't give

a definitive result on the nullhypothesis check

**class** temci.report.testers.TesterRegistry

Bases: [AbstractRegistry](#)

The used tester can be configured by editing the settings key *stats/tester*. Possible testers are 't', 'ks' and 'anderson'

**default** = 't'

Name(s) of the class(es) used by default. Type depends on the *use\_list* property.

**plugin\_synonym** = ('tester', 'testers')

Singular and plural version of the word that is used in the documentation for the registered entities

**registry** = {'anderson': <class 'temci.report.testers.AndersonTester'>, 'ks': <class 'temci.report.testers.KSTester'>, 't': <class 'temci.report.testers.TTester'>}

Registered classes (indexed by their name)

**settings\_key\_path** = 'stats'

Used settings key path

**use\_key** = 'tester'

Used key that sets which registered class is currently used

**use\_list** = False

Allow more than one class to used at a specific moment in time

## Module contents

This module is about generating meaningful reports an working with the resulting measurements of serveral benchmarks.

## temci.utils package

### Submodules

### temci.utils.click\_helper module

This module simplifies the creation of click options from settings and type schemes.

**class** temci.utils.click\_helper.CmdOption(*option\_name*: str, *settings\_key*: Optional[str] = None, *type\_scheme*: Optional[Type] = None, *short*: Optional[str] = None, *completion\_hints*: Optional[Dict[str, Any]] = None, *is\_flag*: Optional[bool] = None)

Bases: object

Represents a command line option.

Initializes a option either based on a setting (via settings key) or on a type scheme. If this is backed by a settings key, the setting is automatically set. If `is_flag` is `None`, it is set `True` if `type_scheme` is an instance of `Bool()` or `BoolOrNone()`

**Parameters**

- **option\_name** – name of the option
- **settings\_key** – settings key of the option
- **type\_scheme** – type scheme with default value
- **short** – short version of the option (ignored if `is_flag=True`)
- **completion\_hints** – additional completion hints (dict with keys for each shell)
- **is\_flag** – is the option a “-ABC/-no-ABC” flag like option?

**callback**

Callback that sets the setting

**completion\_hints**

Additional completion hints (dict with keys for each shell)

**default**

Default value of this option

**description**

Description of this option

**classmethod from\_non\_plugin\_settings**(*settings\_domain: str, exclude: Optional[List[Str]] = None, name\_prefix: Optional[str] = None*) → *CmdOptionList*

Creates a list of `CmdOption` object from all sub settings (in the settings domain). It excludes all sub settings that are either in the exclude list or end with “\_active” or “\_misc” (used for plugin settings). Also every setting that is of type `Dict` is ignored.

**Parameters**

- **settings\_domain** – settings domain to look into (or “” for the root domain)
- **exclude** – list of sub keys to exclude

**Returns**

list of `CmdOptions`

**classmethod from\_registry**(*registry: type, name\_prefix: Optional[str] = None*) → *CmdOptionList*

Creates a list of `CmdOption` objects from an registry. It creates an activation flag (-OPT/-no-OPT) for each registered plugin and creates for each plugin preference an option with name `OPT_PREF`. Deeper nesting is intentionally not supported.

**Parameters**

- **registry** – used registry
- **name\_prefix** – prefix of each option name (usable to avoid ambiguity problems)

**Returns**

list of `CmdOptions`

**has\_completion\_hints**

Does this option has completion hints?

**has\_default**

Does this option has a default value?

**has\_description**

Does this option has a description?

**has\_short**

Does this option has a short version?

**is\_flag**

Is this option flag like?

**option\_name**

Name of this option

**settings\_key**

Settings key of this option

**short**

Short version of the option (ignored if is\_flag=True)

**type\_scheme**

Type scheme with default value

**class** temci.utils.click\_helper.**CmdOptionList**(\*options: *Tuple[Union[CmdOption, CmdOptionList]]*)

Bases: object

A simple list for CmdOptions that supports list flattening.

Create an instance.

**Parameters**

**options** – options that this list consists of

**append**(options: *Union[CmdOption, CmdOptionList]*) → *CmdOptionList*

Appends the passed CmdOptionList or CmdOption and flattens the resulting list.

**Parameters**

**options** – CmdOptionList or CmdOption

**Returns**

self

**get\_sphinx\_doc**() → str

Returns the documentation string for the enclosed options

**options**

Options that build up this list

**set\_short**(option\_name: str, new\_short: str) → *CmdOptionList*

Sets the short option name of the included option with the passed name.

**Parameters**

- **option\_name** – passed option name
- **new\_short** – new short option name

**Returns**

self

**Raises**

IndexError if the option with the passed name doesn't exist

```
temci.utils.click_helper.cmd_option(option: Union[CmdOption, CmdOptionList], name_prefix:
    Optional[str] = None, validate: Optional[bool] = None) →
    Callable[[Callable], Callable]
```

Wrapper around click.option that works with CmdOption objects. If option is a list of CmdOptions then the type\_scheme\_option decorators are chained. Also supports nested lists in the same manner.

**Parameters**

- **option** – CmdOption or (possibly nested) list of CmdOptions
- **name\_prefix** – prefix of all options
- **validate** – validate setting or validate only for outer most if None

**Returns**

click.option(...) like decorator

```
temci.utils.click_helper.document_func(description: str, *options: Tuple[Union[CmdOptionList,
    CmdOption]], argument: Optional[str] = None,
    only_if_sphinx_doc: bool = True) → Callable[[Callable],
    Callable]
```

Function decorator that appends an command documentation to the functions \_\_doc\_\_ attribute.

**Parameters**

- **description** – description of the command
- **options** – options to generate a documentation for
- **argument** – optional argument description of the command
- **only\_if\_sphinx\_doc** – only generate the documentation if the file is executed for documentation generation

```
temci.utils.click_helper.type_scheme_option(option_name: str, type_scheme: Type, is_flag: bool =
    False, callback=typing.Callable[[click.core.Context, str,
    typing.Any], typing.Any], short: Optional[str] = None,
    with_default: bool = True, default=None, validate_settings:
    bool = False) → Callable[[Callable], Callable]
```

Is essentially a wrapper around click.option that works with type schemes.

**Parameters**

- **option\_name** – name of the option
- **type\_scheme** – type scheme to use
- **is\_flag** – is this option a “-ABC/-no-ABC” like flag
- **callback** – callback that is called with the parameter and the argument and has to returns its argument
- **short** – short name of the option (ignored if flag=True)
- **with\_default** – set a default value for the option if possible?
- **default** – default value (if with\_default is true), default: default value of the type scheme
- **validate\_settings** – call Settings().validate() in the callback



`temci.utils.click_helper.validate`(*type\_scheme*: [Type](#)) → Callable[[Context, str, Any], Any]

Creates a valid click option validator function that can be passed to click via the callback parameter. The validator function expects the type of the value to be the raw type of the type scheme.

**Parameters**

**type\_scheme** – type scheme the validator validates against

**Returns**

the validator function

## temci.utils.config\_utils module

Types shared between different file config definitions

## temci.utils.library\_init module

Initialize temci for use as a library

## temci.utils.mail module

Utilities to send mails.

`temci.utils.mail.hostname`() → str

Returns the hostname of the current machine

`temci.utils.mail.send_mail`(*recipient*: str, *subject*: str, *content*: str, *attached\_files*: Optional[List[str]] = None)

Sends a mail to the recipient with the passed subject, content and attached files.

**Parameters**

- **recipient** – recipient of the mail, i.e. a mail address
- **subject** – subject of the mail
- **content** – content of the mail
- **attached\_files** – optional list of names of files that are attached to the mail

## temci.utils.number module

```
class temci.utils.number.FNumber(number: Union[int, float], rel_deviation: Optional[Union[int, float]] =
    None, abs_deviation: Optional[Union[int, float]] = None, is_percent:
    Optional[bool] = None, scientific_notation: Optional[bool] = None,
    parentheses_mode: Optional[Union[str, ParenthesesMode]] = None,
    parentheses: Optional[bool] = None)
```

Bases: object

A formattable number wrapper.

Configuration format, is in the settings under report/number

```

# Don't omit the minimum number of decimal places if insignificant?
force_min_decimal_places:          Bool()
                                default: true

# The maximum number of decimal places
max_decimal_places:             Int(constraint=<function>)
                                default: 5

# The minimum number of shown decimal places if decimal places are shown
min_decimal_places:             Int(constraint=<function>)
                                default: 3

# Omit insignificant decimal places
omit_insignificant_decimal_places: Bool()

# Show parentheses around non significant digits? (If a std dev is given)
parentheses:                   Bool()
                                default: true

# Mode for showing the parentheses: either d (Digits are considered significant if
↳ they don't change
# if the number itself changes += $sigmas * std dev) or o (digits are considered
↳ significant if they
# are bigger than $sigmas * std dev)
parentheses_mode:              ExactEither('d'|'o')
                                default: o

# Show as percentages
percentages:                   Bool()

# Use the exponential notation, i.e. '10e3' for 1000
scientific_notation:           Bool()
                                default: true

# Use si prefixes instead of 'e...'
scientific_notation_si_prefixes: Bool()
                                default: true

# Number of standard deviation used for the digit significance evaluation
sigmas:                        Int(constraint=<function>)
                                default: 2

```

**deviation**

Relative deviation

**format()** → str**classmethod init\_settings**(new\_settings: Dict[str, Union[int, bool]])

```

settings = {'force_min_decimal_places': True, 'max_decimal_places': 5,
'min_decimal_places': 3, 'omit_insignificant_decimal_places': False,
'parentheses': True, 'parentheses_mode': 'o', 'percentages': False,
'scientific_notation': True, 'scientific_notation_si_prefixes': True, 'sigmas':
2}

```

```

settings_format = # Don't omit the minimum number of decimal places if
insignificant? force_min_decimal_places: Bool() default: true # The maximum number
of decimal places max_decimal_places: Int(constraint=<function>) default: 5 # The
minimum number of shown decimal places if decimal places are shown
min_decimal_places: Int(constraint=<function>) default: 3 # Omit insignificant
decimal places omit_insignificant_decimal_places: Bool() # Show parentheses around
non significant digits? (If a std dev is given) parentheses: Bool() default: true
# Mode for showing the parentheses: either d (Digits are considered significant if
they don't change # if the number itself changes += $sigmas * std dev) or o (digits
are considered significant if they # are bigger than $sigmas * std dev)
parentheses_mode: ExactEither('d' | 'o') default: o # Show as percentages
percentages: Bool() # Use the exponential notation, i.e. '10e3' for 1000
scientific_notation: Bool() default: true # Use si prefixes instead of 'e...'
scientific_notation_si_prefixes: Bool() default: true # Number of standard
deviation used for the digit significance evaluation sigmas:
Int(constraint=<function>) default: 2

```

temci.utils.number.Number

Numeric type

alias of Union[int, float]

class temci.utils.number.ParenthesesMode(value)

Bases: Enum

An enumeration.

DIGIT\_CHANGE = 'd'

ORDER\_OF\_MAGNITUDE = 'o'

classmethod map(key: Union[str, ParenthesesMode]) → ParenthesesMode

temci.utils.number.fnumber(number: Union[int, float], rel\_deviation: Optional[Union[int, float]] = None, abs\_deviation: Optional[Union[int, float]] = None, is\_percent: bool = False) → str

temci.utils.number.format\_number(number: Union[int, float], deviation: float = 0.0, parentheses: bool = True, explicit\_deviation: bool = False, is\_deviation\_absolute: bool = True, min\_decimal\_places: int = 3, max\_decimal\_places: Optional[int] = None, omit\_insignificant\_decimal\_places: bool = True, scientific\_notation: bool = False, scientific\_notation\_steps: int = 3, scientific\_notation\_decimal\_places: Optional[int] = None, scientific\_notation\_si\_prefixes: bool = True, force\_min\_decimal\_places: bool = True, relative\_to\_deviation: bool = False, sigmas: int = 2, parentheses\_mode: ParenthesesMode = ParenthesesMode.ORDER\_OF\_MAGNITUDE) → str

Format the passed number

```
>>> format_number(1.0, 0.5)
'1.(000)'
```

```
>>> format_number(1.56, 0.005)
'1.56(0)'
```

```
>>> format_number(1560, scientific_notation=True)
'1.560k'
```

```
>>> format_number(1560, scientific_notation_si_prefixes=False, scientific_
↪ notation=True)
'1.560e3'
```

```
>>> format_number(float("inf"))
'inf'
```

### Parameters

- **number** – formatted number
- **deviation** – standard deviation associated with the number
- **parentheses** – show parentheses around non significant digits?
- **explicit\_deviation** – show the absolute deviation, e.g. “100±456.4”
- **is\_deviation\_absolute** – is the given deviation absolute?
- **min\_decimal\_places** – the minimum number of shown decimal places if decimal places are shown
- **max\_decimal\_places** – the maximum number of decimal places
- **omit\_insignificant\_decimal\_places** – omit insignificant decimal places
- **scientific\_notation** – use the exponential notation, i.e. “10e3” for 1000
- **scientific\_notation\_steps** – steps in which the exponential part is incremented
- **scientific\_notation\_decimal\_places** – number of decimal places that are shown in the scientific notation
- **scientific\_notation\_si\_prefixes** – use si prefixes instead of “e...”
- **force\_min\_decimal\_places** – don’t omit the minimum number of decimal places if insignificant?
- **relative\_to\_deviation** – format the number relative to its deviation, i.e. “10 sigma”
- **sigmas** – number of standard deviations for significance
- **parentheses\_mode** – mode for selecting the significant digits

### Returns

the number formatted as a string

```
temci.utils.number.format_number_sn(number: Union[int, float], scientific_notation_steps: int = 3,
                                     deviation: Optional[float] = None, decimal_places: Optional[int] =
                                     None, si_prefixes: bool = True, **kwargs)
```

## temci.utils.plugin module

Utilities for loading plugins. Plugins are python files (ending *.py*) that are loaded prior to building the cli. These files may e.g. add runners, plugins, ...

Plugins are loaded from the application directory (*~/temci*) and from the paths given in the environment variable *TEMCI\_PLUGIN\_PATH* which contains a colon separated list of paths.

`temci.utils.plugin.load_plugins()`

Load the plugins from the plugin folders :return:

`temci.utils.plugin.plugin_paths()` → List[str]

Returns the paths that plugins are located in (might return folders and files)

## temci.utils.registry module

**class** `temci.utils.registry.AbstractRegistry`

Bases: `object`

An abstract registry. To create an own registry set the *settings\_key\_path* (type str), the *use\_key* (type str), the *use\_list* (type bool) and the default attribute (type (use\_list ? list of strings : str).

Important: Be sure to have a “register = { }” line in your extending class.

**default = None**

Name(s) of the class(es) used by default. Type depends on the *use\_list* property.

**classmethod** `get_class(name: str)` → type

**classmethod** `get_for_name(name: str, *args, **kwargs)` → Any

Creates a plugin with the given name.

**Parameters**

**name** – name of the registered class

**Returns**

object of the registered class

**Raises**

ValueError if there isn't such a class

**classmethod** `get_tester()` → *Tester*

Returns the tester that is configured in the settings

**Returns**

tester instance

**classmethod** `get_used()` → Union[str, List[str]]

Get the list of name of the used plugins (use\_list=True) or the names of the used plugin (use\_list=False).

**plugin\_synonym = ('plugin', 'plugins')**

Singular and plural version of the word that is used in the documentation for the registered entities

**classmethod** `register(name: str, klass: type, misc_type: Type, activate_by_default: Optional[bool] = None, deprecated: bool = False)`

Registers a new class. The constructor of the class gets as first argument the misc settings.

**Parameters**

- **name** – common name of the registered class
- **klass** – actual class
- **misc\_type** – type scheme of the {name}\_misc settings
- **misc\_default** – default value of the {name}\_misc settings
- **deprecated** – is the registered class deprecated and should not be used?

**registry = {}**

Registered classes (indexed by their name)

**settings\_key\_path = ''**

Used settings key path

**use\_key = None**

Used key that sets which registered class is currently used

**use\_list = False**

Allow more than one class to used at a specific moment in time

`temci.utils.registry.register(registry: type, name: str, misc_type: Type, deprecated: bool = False)`

Class decorator that calls the register method for the decorated method.

#### Parameters

- **registry** – the registry class to register the class in
- **name** – common name of the registered class
- **misc\_type** – type scheme of the {name}\_misc settings (each dict key must have a default value)
- **deprecated** – is the registered class deprecated and should not be used?

## temci.utils.settings module

**class** `temci.utils.settings.Settings(*args, **kwargs)`

Bases: object

Manages the Settings. The settings keys and sub keys are combined by a slash, e.g. “report/in”.

The current settings are:

The whole configuration file has the following structure:

```
# Alias for settings
config:          Str()

# Logging level
log_level:       ExactEither('debug' | 'info' | 'warn' | 'error' | 'quiet')
                  default: info

# Additional settings file
settings:        Str()

# Acquire sudo privileges and run benchmark programs with non-sudo user. Only
↳ supported on the command
```

(continues on next page)

(continued from previous page)

```

# line.
sudo:          Bool()

# Used temporary directory
tmp_dir:      Str()
                default: /tmp/temci

build:
  # Input file with the program blocks to build
  in:          Str()
                default: build.yaml

  # Resulting run config file
  out:         Str()
                default: run_config.yaml

  # Number of simultaneous builds for a specific program block, only makes sense
  ↳when
  # build_config/number > 1, and if the build commands create a different binary
  ↳every time they are
  # run
  threads:     Int(constraint=<function>)
                default: 1

# Environment variables for the benchmarked programs, includes the user used for
↳generated files
env:
  PATH:        Str()

  USER:        Str()

report:
  # Exclude all data sets that contain only NaNs.
  exclude_invalid: BoolOrNone()
                    default: true

  # Properties that aren't shown in the report.
  excluded_properties: ListOrTuple(Str())
                        default: [__ov-time]

  # Files that contain the benchmarking results
  in:           Either(Str()|ListOrTuple(Str()))
                    default: run_output.yaml

  # List of included run blocks (all: include all), identified by their
  ↳description or tag
  # attribute, can be regular expressions
  included_blocks:   ListOrTuple(Str())
                        default: [all]

  # Replace the property names in reports with longer more descriptive versions?
  long_properties:   BoolOrNone()

```

(continues on next page)

(continued from previous page)

```

# Possible reporter are 'console', 'html2', 'csv', 'codespeed', 'codespeed2' and
↳ 'velcom'
reporter:          ExactEither('console'|'html2'|'csv'|'codespeed'|'codespeed2'|
↳ 'velcom')
                    default: console

# Produce xkcd like plots (requires the humor sans font to be installed)
xkcd_like_plots:    BoolOrNone()

# Reporter that outputs JSON as specified by
# the `codespeed runner spec <https://git.scc.kit.edu/IPDSnelting/codespeed-
↳ runner>`.
codespeed2_misc:    Dict({}, False, keys=Any, values=Any, default = {})

# Reporter that outputs JSON as expected by `codespeed <https://github.com/
↳ tobami/codespeed>`.
# Branch name and commit ID are taken from the current directory.
# Use it like this:
# .. code:: sh
# temci report --reporter codespeed ... | curl --data-urlencode json@-
# http://localhost:8000/result/add/json/
codespeed_misc:
# Branch name reported to codespeed. Defaults to current branch or else
↳ 'master'.
branch:            Str()

# Commit ID reported to codespeed. Defaults to current commit.
commit_id:         Str()

# Environment name reported to codespeed. Defaults to current host name.
environment:       Str()

# Executable name reported to codespeed. Defaults to the project name.
executable:         Str()

# Project name reported to codespeed.
project:           Str()

# Simple reporter that outputs just text.
console_misc:
# Matches the baseline block
baseline:          Str()

# Position of the baseline comparison: each: after each block, after: after
↳ each cluster, both:
# after each and after cluster, instead: instead of the non baselined
baseline_position:    ExactEither('each'|'after'|'both'|'instead')
                        default: each

# 'auto': report clusters (runs with the same description) and singles
↳ (clusters with a single

```

(continues on next page)



(continued from previous page)

```

    # entry, combined) separately, 'single': report all clusters together as one,
    ↪ 'cluster': report
    # all clusters separately, 'both': append the output of 'cluster' to the
    ↪ output of 'single'
    mode:          ExactEither('both'|'cluster'|'single'|'auto')
                  default: auto

    # Output file name or '-' (stdout)
    out:           FileName(allow_std=True)
                  default: '-'

    # Report on the failing blocks
    report_errors: Bool()
                  default: true

    # Print statistical tests for every property for every two programs
    with_tester_results: Bool()
                       default: true

    # Simple reporter that outputs just a configurable csv table with rows for
    ↪ each run block
    csv_misc:
        # List of valid column specs, format is a comma separated list of 'PROPERTY\
    ↪ [mod\]' or 'ATTRIBUTE'
        # mod is one of: mean, stddev, property, min, max and stddev per mean,
    ↪ optionally a formatting
        # option can be given via PROPERTY\[mod|OPT1OPT2...\], where the OPTs are
    ↪ one of the following: %
        # (format as percentage), p (wrap insignificant digits in parentheses (+- 2
    ↪ std dev)), s (use
        # scientific notation, configured in report/number) and o (wrap digits in
    ↪ the order of magnitude
        # of 2 std devs in parentheses). PROPERTY can be either the description or
    ↪ the short version of
        # the property. Configure the number formatting further via the number
    ↪ settings in the settings
        # file
    columns:       ListOrTuple(Str())
                  default: [description]
        # with constrain: List of valid column specs, format is a comma
    ↪ separated list of 'PROPERTY\[mod\]' or 'ATTRIBUTE' mod is one of: mean, stddev,
    ↪ property, min, max and stddev per mean, optionally a formatting option can be
    ↪ given via PROPERTY\[mod|OPT1OPT2...\], where the OPTs are one of the following: %
    ↪ (format as percentage), p (wrap insignificant digits in parentheses (+- 2 std
    ↪ dev)), s (use scientific notation, configured in report/number) and o (wrap
    ↪ digits in the order of magnitude of 2 std devs in parentheses). PROPERTY can be
    ↪ either the description or the short version of the property. Configure the number
    ↪ formatting further via the number settings in the settings file

    # Output file name or standard out (-)
    out:           FileName(allow_std=True)
                  default: '-'

```

(continues on next page)

(continued from previous page)

```

# Reporter that produces a HTML based report with lot's of graphics.
# A rewrite of the original HTMLReporter
html2_misc:
# Alpha value for confidence intervals
alpha:          T(<class 'float'>)
                default: 0.05

# Height per run block for the big comparison box plots
boxplot_height: T(<class 'float'>)
                default: 2.0

# Width of all big plotted figures
fig_width_big:  T(<class 'float'>)
                default: 25.0

# Width of all small plotted figures
fig_width_small: T(<class 'float'>)
                default: 15.0

# Format string used to format floats
float_format:   Str()
                default: '{:5.2e}'

# Override the contents of the output directory if it already exists?
force_override: Bool()

# Generate pdf versions of the plotted figures?
gen_pdf:        Bool()

# Generate simple latex versions of the plotted figures?
gen_tex:        Bool()
                default: true

# Generate excel files for all tables
gen_xls:        Bool()

# Hide warnings and errors related to statistical properties
hide_stat_warnings: Bool()

# Name of the HTML file
html_filename:   Str()
                default: report.html

# Use local versions of all third party resources
local:          Bool()

# Show the mean related values in the big comparison table
mean_in_comparison_tables: Bool()
                default: true

# Show the minimum related values in the big comparison table

```

(continues on next page)

(continued from previous page)

```

min_in_comparison_tables:      Bool()

# Output directory
out:      Str()
        default: report

# Format string used to format floats as percentages
percent_format:      Str()
        default: '{:5.2%}'

# Show zoomed out (x min = 0) figures in the extended summaries?
show_zoomed_out:      Bool()
        default: true

number:
# Don't omit the minimum number of decimal places if insignificant?
force_min_decimal_places:      Bool()
        default: true

# The maximum number of decimal places
max_decimal_places:      Int(constraint=<function>)
        default: 5

# The minimum number of shown decimal places if decimal places are shown
min_decimal_places:      Int(constraint=<function>)
        default: 3

# Omit insignificant decimal places
omit_insignificant_decimal_places:      Bool()

# Show parentheses around non significant digits? (If a std dev is given)
parentheses:      Bool()
        default: true

# Mode for showing the parentheses: either d (Digits are considered
↳ significant if they don't
# change if the number itself changes += $sigmas * std dev) or o (digits
↳ are considered
# significant if they are bigger than $sigmas * std dev)
parentheses_mode:      ExactEither('d'|'o')
        default: o

# Show as percentages
percentages:      Bool()

# Use the exponential notation, i.e. '10e3' for 1000
scientific_notation:      Bool()
        default: true

# Use si prefixes instead of 'e...'
scientific_notation_si_prefixes:      Bool()
        default: true

```

(continues on next page)

(continued from previous page)

```

    # Number of standard deviation used for the digit significance evaluation
    sigmas:          Int(constraint=<function>)
                    default: 2

    # Reporter that outputs JSON as specified by
    # the `velcom runner spec` <https://github.com/IPDSnelting/velcom/wiki/
    ↪ Benchmark-Repo-
    # Specification>`.
    velcom_misc:      Dict({}, False, keys=Any, values=Any, default = {})

run:
    # Abort after the first failing build
    abort_after_build_error:      Bool()
                                default: true

    # Append to the output file instead of overwriting by adding new run data blocks
    append:                      Bool()

    # Disable the hyper threaded cores. Good for cpu bound programs.
    disable_hyper_threading:      Bool()

    # Discard all run data for the failing program on error
    discard_all_data_for_block_on_error:      Bool()

    # First n runs that are discarded
    discarded_runs:              Int(constraint=<function>)
                                default: 1

    # Possible run drivers are 'exec' and 'shell'
    driver:                     ExactEither('exec'|'shell')
                                default: exec

    # Input file with the program blocks to benchmark
    in:                         Str()
                                default: input.exec.yaml

    # List of included run blocks (all: include all), or their tag attribute or
    ↪ their number in the
    # file (starting with 0), can be regular expressions
    included_blocks:            ListOrTuple(Str())
                                default: [all]

    # Maximum time one run block should take, -1 == no timeout, supports normal
    ↪ time span expressions
    max_block_time:             ValidTimespan()
                                default: '-1'

    # Maximum number of benchmarking runs
    max_runs:                   Int(constraint=<function>)
                                default: 100

```

(continues on next page)

(continued from previous page)

```

# Maximum time the whole benchmarking should take, -1 == no timeout, supports_
↪normal time span
# expressions
max_time:          ValidTimespan()
                    default: '-1'

# Minimum number of benchmarking runs
min_runs:         Int(constraint=<function>)
                    default: 20

# Do not build if build configs are present, only works if the working_
↪directory of the blocks
# does not change
no_build:         Bool()

# Only build
only_build:      Bool()

# Output file for the benchmarking results
out:              Str()
                    default: run_output.yaml

# Record the caught errors in the run_output file
record_errors_in_file: Bool()
                    default: true

# Number of benchmarking runs that are done together
run_block_size:   Int(constraint=<function>)
                    default: 1

# if != -1 sets max and min runs to its value
runs:             Int(constraint=<function>)
                    default: -1

# If not empty, recipient of a mail after the benchmarking finished.
send_mail:        Str()

# Print console report if log_level=info
show_report:      Bool()
                    default: true

# Randomize the order in which the program blocks are benchmarked.
shuffle:          Bool()
                    default: true

# Store the result file after each set of blocks is benchmarked
store_often:      Bool()

# Show the report continuously
watch:            Bool()

# Update the screen nth run (less updates are better for benchmarks)

```

(continues on next page)

(continued from previous page)

```

watch_every:          Int(constraint=<function>)
                        default: 1

cpuset:
  # Use cpuset functionality?
  active:             Bool()

  # Number of cpu cores for the base (remaining part of the) system
  base_core_number:   Int(range=range(0, 2))
                        default: 1

  # 0: benchmark sequential, > 0: benchmark parallel with n instances, -1:
  ↪ determine n automatically
  parallel:          Int(constraint=<function>)

  # Number of cpu cores per parallel running program.
  sub_core_number:    Int(range=range(0, 2))
                        default: 1

  # place temci in the same cpu set as the rest of the system?
  temci_in_base_set:  Bool()
                        default: true

  # Implements a simple run driver that just executes one of the passed run_cmds
  # in each benchmarking run.
  # It measures the time using the perf stat tool (runner=perf_stat).
  # The constructor calls the ``setup`` method.
exec_misc:
  # Argument passed to all benchmarked commands by replacing $ARGUMENT with
  ↪ this value in the
  # command
  argument:          Str()

  # Parse the program output as a YAML dictionary of that gives for a
  ↪ specific property a
  # measurement. Not all runners support it.
  parse_output:      Bool()

  # Order in which the plugins are used, plugins that do not appear in this
  ↪ list are used before all
  # others
  plugin_order:      ListOrTuple(Str())
                        default: [drop_fs_caches, sync, sleep, preheat, flush_cpu_
  ↪ caches]

  # Enable other plugins by default: none = (enable none by default); all =
  ↪ cpu_governor,disable_sw
  # ap,sync,stop_start,other_nice,nice,disable_aslr,disable_ht,cpuset,disable_
  ↪ turbo_boost (enable
  # all, might freeze your system); usable =
  # cpu_governor,disable_swap,sync,nice,disable_aslr,disable_ht,cpuset,
  ↪ disable_turbo_boost (like

```

(continues on next page)

(continued from previous page)

```

# 'all' but doesn't affect other processes)
preset:          ExactEither('none'|'all'|'usable')
                  default: none

# Pick a random command if more than one run command is passed.
random_cmd:      Bool()
                  default: true

# If not " overrides the runner setting for each program block
runner:          ExactEither(''|'perf_stat'|'rusage'|'spec'|'spec.py'|'time'|
↳ 'output')

exec_plugins:
# Enable:  Allows the setting of the scaling governor of all cpu cores, to
↳ ensure that all use
# the same.
cpu_governor_active:      BoolOrNone()

# Enable:  Enable cpusets, simply sets run/cpuset/active to true
cpuset_active:           BoolOrNone()

# Enable:  Disable amd turbo boost
disable_amd_boost_active: BoolOrNone()

# Enable:  Disable address space randomization
disable_aslr_active:      BoolOrNone()

# Enable:  Disable the L1 and L2 caches on x86 and x86-64 architectures.
# Uses a small custom kernel module (be sure to compile it via 'temci setup
# --build_kernel_modules').
# :warning: slows program down significantly and has probably other weird
↳ consequences
# :warning: this is untested
# :warning: a linux-forum user declared: Disabling cpu caches gives you a
↳ pentium I like
# processor!!!
disable_cpu_caches_active: BoolOrNone()

# Enable:  Disable hyper-threading
disable_ht_active:         BoolOrNone()

# Enable:  Disable intel turbo mode
disable_intel_turbo_active: BoolOrNone()

# Enable:  Disables swapping on the system before the benchmarking and
↳ enables it after.
disable_swap_active:       BoolOrNone()

# Enable:  Disable amd and intel turbo boost
disable_turbo_boost_active: BoolOrNone()

# Enable:  Sets run/discarded_runs

```

(continues on next page)

(continued from previous page)

```

discarded_runs_active:      BoolOrNone()

# Enable: Drop page cache, directoy entries and inodes before every
↳ benchmarking run.
drop_fs_caches_active:      BoolOrNone()

# Enable: Adds random environment variables.
env_randomize_active:       BoolOrNone()

# Possible run driver plugins are 'nice', 'env_randomize', 'preheat', 'other_nice'
↳ ', 'stop_start',
# 'sync', 'sleep', 'drop_fs_caches', 'disable_swap', 'disable_cpu_caches', 'flush_
↳ cpu_caches',
# 'cpu_governor', 'disable_aslr', 'disable_ht', 'disable_turbo_boost', 'disable_
↳ intel_turbo',
# 'disable_amd_boost', 'cpuset' and 'discarded_runs'
exec_active:                StrList(allowed=['nice', 'env_randomize', 'preheat',
↳ 'other_nice', 'stop_start', 'sync', 'sleep', 'drop_fs_caches', 'disable_swap',
↳ 'disable_cpu_caches', 'flush_cpu_caches', 'cpu_governor', 'disable_aslr',
↳ 'disable_ht', 'disable_turbo_boost', 'disable_intel_turbo', 'disable_amd_boost',
↳ 'cpuset', 'discarded_runs'])

# Enable: Flushes the CPU caches on a x86 CPU using a small kernel module,
# see `WBINVD` <https://www.felixcloutier.com/x86/wbinvd>`
flush_cpu_caches_active:     BoolOrNone()

# Enable: Allows the setting of the nice and ionice values of the
↳ benchmarking process.
nice_active:                 BoolOrNone()

# Enable: Allows the setting of the nice value of most other processes
↳ (as far as possible).
other_nice_active:           BoolOrNone()

# Enable: Preheats the system with a cpu bound task
# (calculating the inverse of a big random matrix with numpy).
preheat_active:              BoolOrNone()

# Enable: Sleep a given amount of time before the benchmarking begins.
# See Gernot Heisers Systems Benchmarking Crimes:
# Make sure that the system is really quiescent when starting an
↳ experiment,
# leave enough time to ensure all previous data is flushed out.
sleep_active:                BoolOrNone()

# Enable: Stop almost all other processes (as far as possible).
stop_start_active:           BoolOrNone()

# Enable: Calls ``sync`` before each program execution.
sync_active:                 BoolOrNone()

# Allows the setting of the scaling governor of all cpu cores, to ensure
↳ that all use the same.

```

(continues on next page)



(continued from previous page)

```

cpu_governor_misc:
    # New scaling governor for all cpus
    governor:          Str()
                        default: performance

    # Enable cpusets, simply sets run/cpuset/active to true
    cpuset_misc:       Dict({}, False, keys=Any, values=Any, default = {})

    # Disable amd turbo boost
    disable_amd_boost_misc: Dict({}, False, keys=Any, values=Any,
↪ default = {})

    # Disable address space randomization
    disable_aslr_misc:   Dict({}, False, keys=Any, values=Any, default =
↪ {})

    # Disable the L1 and L2 caches on x86 and x86-64 architectures.
    # Uses a small custom kernel module (be sure to compile it via 'temci setup
    # --build_kernel_modules').
    # :warning: slows program down significantly and has probably other weird
↪ consequences
    # :warning: this is untested
    # :warning: a linux-forum user declared: Disabling cpu caches gives you a
↪ pentium I like
    # processor!!!
    disable_cpu_caches_misc: Dict({}, False, keys=Any, values=Any,
↪ default = {})

    # Disable hyper-threading
    disable_ht_misc:     Dict({}, False, keys=Any, values=Any, default = {})

    # Disable intel turbo mode
    disable_intel_turbo_misc: Dict({}, False, keys=Any, values=Any,
↪ default = {})

    # Disables swapping on the system before the benchmarking and enables it
↪ after.
    disable_swap_misc:   Dict({}, False, keys=Any, values=Any, default =
↪ {})

    # Disable amd and intel turbo boost
    disable_turbo_boost_misc: Dict({}, False, keys=Any, values=Any,
↪ default = {})

    # Sets run/discarded_runs
    discarded_runs_misc:
        # Number of discarded runs
        runs:           Int(constraint=<function>)
                        default: 1

    # Drop page cache, directoy entries and inodes before every benchmarking
↪ run.

```

(continues on next page)

(continued from previous page)

```

drop_fs_caches_misc:
    # Free dentries and inodes
    free_dentries_inodes:      Bool()
                                default: true

    # Free the page cache
    free_pagecache:           Bool()
                                default: true

# Adds random environment variables.
env_randomize_misc:
    # Maximum length of each random key
    key_max:                  Int(constraint=<function>)
                                default: 4096

    # Maximum number of added random environment variables
    max:                      Int(constraint=<function>)
                                default: 4

    # Minimum number of added random environment variables
    min:                      Int(constraint=<function>)
                                default: 4

    # Maximum length of each random value
    var_max:                  Int(constraint=<function>)
                                default: 4096

# Flushes the CPU caches on a x86 CPU using a small kernel module,
# see `WBINVD` <https://www.felixcloutier.com/x86/wbinvd>`
flush_cpu_caches_misc:      Dict({}, False, keys=Any, values=Any,
↪ default = {})

# Allows the setting of the nice and ionice values of the benchmarking_
↪ process.
nice_misc:
    # Specify the name or number of the scheduling class to use; 0 for none,
↪ 1 for realtime, 2 for
    # best-effort, 3 for idle.
    io_nice:                  Int(range=range(0, 4))
                                default: 1

    # Niceness values range from -20 (most favorable to the process) to 19_
↪ (least favorable to the
    # process).
    nice:                    Int(range=range(-20, 20))
                                default: -15

# Allows the setting of the nice value of most other processes (as far as_
↪ possible).
other_nice_misc:
    # Processes with lower nice values are ignored.
    min_nice:                 Int(range=range(-15, 20))

```

(continues on next page)

(continued from previous page)

```

        default: -10

    # Niceness values for other processes.
    nice:      Int(range=range(-20, 20))
               default: 19

    # Preheats the system with a cpu bound task
    # (calculating the inverse of a big random matrix with numpy).
    preheat_misc:
        # Number of seconds to preheat the system with an cpu bound task
        time:      Int(constraint=<function>)
                   default: 10

    # When to preheat
    when:      ListOrTuple(ExactEither('before_each_run' | 'at_setup'))
               default: [before_each_run]

    # Sleep a given amount of time before the benchmarking begins.
    # See Gernot Heisers Systems Benchmarking Crimes:
    # Make sure that the system is really quiescent when starting an
    ↪ experiment,
    # leave enough time to ensure all previous data is flushed out.
    sleep_misc:
        # Seconds to sleep
        seconds:   Int(constraint=<function>)
                   default: 10

    # Stop almost all other processes (as far as possible).
    stop_start_misc:
        # Each process which name (lower cased) starts with one of the prefixes
    ↪ is not ignored. Overrides
        # the decision based on the min_id.
        comm_prefixes: ListOrTuple(Str())
                       default: [ssh, xorg, bluetoothd]

        # Each process which name (lower cased) starts with one of the prefixes
    ↪ is ignored. It overrides
        # the decisions based on comm_prefixes and min_id.
        comm_prefixes_ignored: ListOrTuple(Str())
                               default: [dbus, kworker]

    # Just output the to be stopped processes but don't actually stop them?
    dry_run:      Bool()

    # Processes with lower id are ignored.
    min_id:      Int(constraint=<function>)
                 default: 1500

    # Processes with lower nice values are ignored.
    min_nice:    Int(range=range(-15, 20))
                 default: -10

```

(continues on next page)

(continued from previous page)

```

# Suffixes of processes names which are stopped.
subtree_suffixes:      ListOrTuple(Str())
                        default: [dm, apache]

# Calls ``sync`` before each program execution.
sync_misc:            Dict({}, False, keys=Any, values=Any, default = {})

# Maximum runs per tag (block attribute 'tag'), min('max_runs', 'per_tag') is used
max_runs_per_tag:      Dict(, keys=Str(), values=Int(constraint=<function>),
↳ default = {})

# Minimum runs per tag (block attribute 'tag'), max('min_runs', 'per_tag') is used
min_runs_per_tag:      Dict(, keys=Str(), values=Int(constraint=<function>),
↳ default = {})

# Runs per tag (block attribute 'tag'), max('runs', 'per_tag') is used
runs_per_tag:          Dict(, keys=Str(), values=Int(constraint=<function>),
↳ default = {})

# Implements a run driver that runs the benched command a single time with
↳ redirected in- and
# output.
# It can be used to run own benchmarking commands inside a sane benchmarking
↳ environment
# The constructor calls the ``setup`` method.
shell_misc:
# Order in which the plugins are used, plugins that do not appear in this
↳ list are used before all
# others
plugin_order:          ListOrTuple(Str())
                        default: [drop_fs_caches, sync, sleep, preheat, flush_cpu_
↳ caches]

# Enable other plugins by default: none = (enable none by default); all =
↳ cpu_governor,disable_sw
# ap,sync,stop_start,other_nice,nice,disable_aslr,disable_ht,cpuset,disable_
↳ turbo_boost (enable
# all, might freeze your system); usable =
# cpu_governor,disable_swap,sync,nice,disable_aslr,disable_ht,cpuset,
↳ disable_turbo_boost (like
# 'all' but doesn't affect other processes)
preset:                ExactEither('none'|'all'|'usable')
                        default: none

stats:
# Properties to use for reporting and null hypothesis tests, can be regular
↳ expressions
properties:            ListOrTuple(Str())
                        default: [all]

# Possible testers are 't', 'ks' and 'anderson'
tester:                ExactEither('t'|'ks'|'anderson')

```

(continues on next page)

(continued from previous page)

```

        default: t

    # Range of p values that allow no conclusion.
    uncertainty_range:      Tuple(T(<class 'float'>):<function>, T(<class 'float
→ '>):<function>)
        default: [0.05, 0.15]

    # Tester that uses the Anderson statistic on 2 samples.
    anderson_misc:         Dict({}, False, keys=Any, values=Any, default = {})

    # Tester that uses the Kolmogorov-Smirnov statistic on 2 samples.
    ks_misc:               Dict({}, False, keys=Any, values=Any, default = {})

    # Tester that uses the student's t test.
    t_misc:                Dict({}, False, keys=Any, values=Any, default = {})

```

Initializes a Settings singleton object and thereby loads the Settings files. It loads the settings files from the app folder (config.yaml) and the current working directory (temci.yaml) if they exist.

#### Raises

SettingsError if some of the settings aren't in the format described via the type\_scheme class property

#### apply\_override\_actions()

Applies actions like overriding max\_runs with runs

#### config\_file\_name = 'temci.yaml'

Default name of the configuration files

#### default(value: Optional[Any], key: str)

Returns the passed value if isn't None else the settings value under the passed key.

#### Parameters

- **value** – passed value
- **key** – passed settings key

#### get(key: Union[str, List[str]]) → Any

Get the setting with the given key.

#### Parameters

**key** – name of the setting

#### Returns

value of the setting

#### Raises

SettingsError if the setting doesn't exist

#### get\_type\_scheme(key: Union[str, List[str]]) → Type

Returns the type scheme of the given key.

#### Parameters

**key** – given key

#### Returns

type scheme

**Raises**

SettingsError if the setting with the given key doesn't exist

**has\_key**(key: str) → bool

Does the passed key exist?

**has\_log\_level**(level: str) → bool

Is the current log level the passed level?

**Parameters**

**level** – passed level (in ["error", "warn", "info", "debug"])

**is\_obsolete**(key: Union[str, List[str]]) → bool

Is the setting with the passed key obsolete?

**Parameters**

**key** – key or key path

**Returns**

obsolete setting?

**load\_file**(file: str)

Loads the configuration from the configuration YAML file.

**Parameters**

**file** – path to the file

**Raises**

SettingsError if the settings file is incorrect or doesn't exist

**load\_files**()

Loads the configuration files from the current and the config directory

**load\_from\_config\_dir**()

Load the config file from the application directory (e.g. in the users home folder) if it exists.

**load\_from\_current\_dir**()

Load the configuration from the configuration file in the current working directory if it exists.

**load\_from\_dict**(config\_dict: Dict[str, Any])

Load the configuration from the passed dictionary.

**Parameters**

**config\_dict** – passed configuration dictionary

**load\_from\_dir**(dir: str)

Load the configuration from the configuration file inside the passed directory.

**Parameters**

**dir** – path of the directory

**modify\_setting**(key: str, type\_scheme: Type)

Modifies the setting with the given key and adds it if it doesn't exist.

**Parameters**

- **key** – key of the setting
- **type\_scheme** – Type of the setting
- **default\_value** – default value of the setting

**Raises**

SettingsError if the settings domain (the key without the last element) doesn't exist

**Raises**

TypeError if the default value doesn't adhere the type scheme

**modify\_type\_scheme**(*key: str, modifier: Callable[[Type], Type]*)

Modifies the type scheme of the given key via a modifier function.

**Parameters**

- **key** – given key
- **modifier** – gets the type scheme and returns its modified version

**Raises**

SettingsError if the setting with the given key doesn't exist

**obsoleteness\_reason**(*key: Union[str, List[str]]*) → Optional[Obsolete]

Returns the obsolete type object for obsolete settings

**Parameters**

**key** – key or path

**Returns**

object that contains information on the obsoleteness or None

**prefs**

The set configurations

**reset()**

Resets the current settings to the defaults.

**set**(*key: str, value, validate: bool = True, setup: bool = True*)

Sets the setting key to the passed new value

**Parameters**

- **key** – settings key
- **value** – new value
- **validate** – validate after the setting operation
- **setup** – call the setup function

**Raises**

SettingsError if the setting isn't valid

**store\_into\_file**(*file\_name: str, comment\_out\_defaults: bool = False*)

Stores the current settings into a yaml file with comments.

**Parameters**

- **file\_name** – name of the resulting file
- **comment\_out\_defaults** – comment out the default values

```

type_scheme = # Alias for settings config: Str() # Logging level log_level:
ExactEither('debug'|'info'|'warn'|'error'|'quiet') default: info # Additional
settings file settings: Str() # Acquire sudo privileges and run benchmark programs
with non-sudo user. Only supported on the command # line. sudo: Bool() # Used
temporary directory tmp_dir: Str() default: /tmp/temci build: # Input file with
the program blocks to build in: Str() default: build.yaml # Resulting run config
file out: Str() default: run_config.yaml # Number of simultaneous builds for a
specific program block, only makes sense when # build_config/number > 1, and if the
build commands create a different binary every time they are # run threads:
Int(constraint=<function>) default: 1 # Environment variables for the benchmarked
programs, includes the user used for generated files env: PATH: Str() USER: Str()
report: # Exclude all data sets that contain only NaNs. exclude_invalid:
BoolOrNone() default: true # Properties that aren't shown in the report.
excluded_properties: ListOrTuple(Str()) default: [__ov-time] # Files that contain
the benchmarking results in: Either(Str()|ListOrTuple(Str())) default:
run_output.yaml # List of included run blocks (all: include all), identified by
their description or tag # attribute, can be regular expressions included_blocks:
ListOrTuple(Str()) default: [all] # Replace the property names in reports with
longer more descriptive versions? long_properties: BoolOrNone() # Possible reporter
are 'console', 'html2', 'csv', 'codespeed', 'codespeed2' and 'velcom' reporter:
ExactEither('console'|'html2'|'csv'|'codespeed'|'codespeed2'|'velcom') default:
console # Produce xkcd like plots (requires the humor sans font to be installed)
xkcd_like_plots: BoolOrNone() # Reporter that outputs JSON as specified by # the
`codespeed runner spec <https://git.scc.kit.edu/IPDSnelting/codespeed-runner>`_.
codespeed2_misc: Dict({}, False, keys=Any, values=Any, default = {}) # Reporter
that outputs JSON as expected by `codespeed <https://github.com/tobami/codespeed>`_.
# Branch name and commit ID are taken from the current directory. # Use it like
this: # .. code:: sh # temci report --reporter codespeed ... | curl
--data-urlencode json@- # http://localhost:8000/result/add/json/ codespeed_misc: #
Branch name reported to codespeed. Defaults to current branch or else 'master'.
branch: Str() # Commit ID reported to codespeed. Defaults to current commit.
commit_id: Str() # Environment name reported to codespeed. Defaults to current host
name. environment: Str() # Executable name reported to codespeed. Defaults to the
project name. executable: Str() # Project name reported to codespeed. project:
Str() # Simple reporter that outputs just text. console_misc: # Matches the
baseline block baseline: Str() # Position of the baseline comparison: each: after
each block, after: after each cluster, both: # after each and after cluster,
instead: instead of the non baselined baseline_position:
ExactEither('each'|'after'|'both'|'instead') default: each # 'auto': report
clusters (runs with the same description) and singles (clusters with a single #
entry, combined) separately, 'single': report all clusters together as one,
'cluster': report # all clusters separately, 'both': append the output of
'cluster' to the output of 'single' mode:
ExactEither('both'|'cluster'|'single'|'auto') default: auto # Output file name or
`-` (stdout) out: FileName(allow_std=True) default: '-' # Report on the failing
blocks report_errors: Bool() default: true # Print statistical tests for every
property for every two programs with_tester_results: Bool() default: true # Simple
reporter that outputs just a configurable csv table with rows for each run block
csv_misc: # List of valid column specs, format is a comma separated list of
'PROPERTY\[mod\]' or 'ATTRIBUTE' # mod is one of: mean, stddev, property, min, max
and stddev per mean, optionally a formatting # option can be given
viaPROPERTY\[mod|OPT1OPT2...\], where the OPTs are one of the following: % #
(format as percentage), p (wrap insignificant digits in parentheses (+- 2 std dev)),
s (use # scientific notation, configured in report/number) and o (wrap digits in the
order of magnitude # of 2 std devs in parentheses). PROPERTY can be either the
description or the short version of # the property. Configure the number formatting
further via the number settings in the settings file.
default: [description] # with constrain: List of valid column specs, format is a
comma separated list of 'PROPERTY\[mod\]' or 'ATTRIBUTE' mod is one of: mean,
stddev, property, min, max and stddev per mean, optionally a formatting option can

```



Type scheme of the settings

**validate()**

Validate this settings object

**Raises**

SettingsError if the setting isn't valid

**validate\_key\_path**(*path: List[str]*) → bool

Validates a path into in to the settings trees, :param path: list of sub keys :return: Is this key path valid?

**exception** temci.utils.settings.SettingsError

Bases: ValueError

Error raised if something with the settings goes wrong

temci.utils.settings.ValidCPUCoreNumber() → *Int*

Creates a Type instance that matches all valid CPU core numbers.

### temci.utils.sudo\_utils module

temci.utils.sudo\_utils.bench\_as\_different\_user() → bool

temci.utils.sudo\_utils.chown(*path: Union[str, TextIOWrapper, BufferedRandom, BufferedRWPair, BufferedWriter, IOBase]*)

temci.utils.sudo\_utils.get\_bench\_uid\_and\_gid() → Tuple[int, int]

temci.utils.sudo\_utils.get\_bench\_user() → str

temci.utils.sudo\_utils.get\_env\_setting() → Dict[str, str]

### temci.utils.typecheck module

Implements basic type checking for complex types.

Why? Because it's nice to be able to type check complex structures that come directly from the user (e.g. from YAML config files).

The Type instance are usable with the standard isinstance function:

```
isinstance(4, Either(Float(), Int()))
```

Type instances also support the "&" (produces All(one, two)) and "|" (produces Either(one, two)) operators. The above sample code can therefore be written as:

```
isinstance(4, Float() | Int())
```

The native type wrappers also support custom constraints. With help of the fn module one can write:

```
t = Float(_ > 0) | Int(_ > 10)
isinstance(var, t)
```

"t" is a Type that matches only floats greater than 0 and ints greater than 10.

For more examples look into the test\_typecheck.py file.

**class** temci.utils.typecheck.All(\*types: Tuple[Type])

Bases: [Type](#)

Checks for the value to be of all of several types.

Creates an All instance.

**Parameters**

**types** – list of types (or SpecialType subclasses)

**Raises**

ConstraintError if some of the constraints aren't (typechecker) Types

**types**

Expected types

**class** temci.utils.typecheck.Any(completion\_hints: Optional[Dict[str, Any]] = None)

Bases: [Type](#)

Checks for the value to be of any type.

Creates an instance.

**Parameters**

**completion\_hints** – completion hints for supported shells for this type instance

**class** temci.utils.typecheck.Bool

Bases: [Type](#), ParamType

Like Bool but with a third value none that declares that the value no boolean value. It has None as its default value (by default).

Creates an instance.

**Parameters**

**completion\_hints** – completion hints for supported shells for this type instance

**completion\_hints**

Completion hints for supported shells for this type instance

**name:** str = 'bool'

click.ParamType name, that makes this class usable as a click type

**class** temci.utils.typecheck.BoolOrNone

Bases: [Type](#), ParamType

Like Bool but with a third value none that declares that the value no boolean value. It has None as its default value (by default).

Creates an instance.

**Parameters**

**completion\_hints** – completion hints for supported shells for this type instance

**completion\_hints**

Completion hints for supported shells for this type instance

**convert**(value, param, ctx: Context) → Optional[bool]

Convert method that makes this class usable as a click type.

**default**

The default value of this instance

**name:** `str = 'bool_or_none'`

click.ParamType name, that makes this class usable as a click type

**class** `temci.utils.typecheck.CompletionHint(**hints)`

Bases: `object`

A completion hint annotation for a type. Usage example:

```
Int() // Completion(zsh="_files")
```

**hints**

Completion hints for every supported shell

**class** `temci.utils.typecheck.Constraint(constraint: Callable[[Any], bool], constrained_type: Type = Any, description: Optional[str] = None)`

Bases: `Type`

Checks the passed value by an user defined constraint.

Creates an Constraint instance.

**Parameters**

- **constraint** – function that returns True if the user defined constraint is satisfied
- **constrained\_type** – Type that the constraint is applied on
- **description** – short description of the constraint (e.g. “>0”)

**Raises**

ConstraintError if constrained\_type isn’t a (typechecker) Types

**constrained\_type**

Type that the constraint is applied on

**constraint**

Function that returns True if the user defined constraint is satisfied

**description**

Short description of the constraint (e.g. “>0”)

**string\_representation**(*indents: int = 0, indentation: int = 4, str\_list: bool = False, defaults=None*) → Union[str, List[str]]

Produce a YAML string that contains the default value (if possible), the description of this type and more and its possible sub types.

**Parameters**

- **indents** – number of indents in front of each produced line
- **indentation** – indentation width in number of white spaces
- **str\_list** – return a list of lines instead of a combined string?
- **defaults** – default value that should be used instead of the default value of this instance

**class** `temci.utils.typecheck.Default(default)`

Bases: `object`

A default value annotation for a Type. Usage example:

```
Int() // Default(3)
```

Especially useful to declare the default value for a key of an dictionary. Allows to use Dict(...).get\_default() -> dict.

**default**

Default value of the annotated type

```
class temci.utils.typecheck.Description(description: str)
```

Bases: object

A description of a Type, that annotates it. Usage example:

```
Int() // Description("Description of Int()")
```

**description**

Description string

```
class temci.utils.typecheck.Dict(data: Optional[Dict[Any, Type]] = None, unknown_keys: bool = False,
                                key_type: Type = Any, value_type: Type = Any)
```

Bases: *Type*

Checks for the value to be a dictionary with expected keys and values satisfy given type constraints.

Creates a new instance.

**Parameters**

- **data** – dictionary with the expected keys and the expected types of the associated values
- **unknown\_keys** – accept unknown keys in value
- **key\_type** – expected Type of all dictionary keys
- **value\_type** – expected Type of all dictionary values

**Raises**

ConstraintError if one of the given types isn't a (typechecker) Types

**get\_default()** → dict

Returns the default value of this type :raises: ValueError if the default value isn't set

```
get_default_yaml(indents: int = 0, indentation: int = 4, str_list: bool = False, defaults=None,
                 comment_out_defaults: bool = False) → Union[str, List[str]]
```

Produce a YAML like string that contains the default value and the description of this type and its possible sub types.

**Parameters**

- **indents** – number of indents in front of each produced line
- **indentation** – indentation width in number of white spaces
- **str\_list** – return a list of lines instead of a combined string?
- **defaults** – default value that should be used instead of the default value of this instance
- **comment\_out\_defaults** – comment out default values

**get\_description(key: str)** → str

Returns the description for the passed key or None if there isn't one.

**Parameters****key** – passed key**has\_default()** → bool

Does this type instance have an default value?

**is\_obsolete**(key: str) → bool

Is the type that belongs to the key Obsolete?

**Parameters****key** – dict key**Returns**

obsolete?

**key\_type**

Expected Type of all dictionary keys

**obsoleteness\_reason**(key: str) → Optional[Obsolete]

Return the obsoleteness reason (the Obsolete type object) if the key is obsolete

**Parameters****key** – dict key**Returns**

Obsolete object or none

**string\_representation**(indents: int = 0, indentation: int = 4, str\_list: bool = False, defaults=None) → Union[str, List[str]]

Produce a YAML string that contains the default value (if possible), the description of this type and more and its possible sub types.

**Parameters**

- **indents** – number of indents in front of each produced line
- **indentation** – indentation width in number of white spaces
- **str\_list** – return a list of lines instead of a combined string?
- **defaults** – default value that should be used instead of the default value of this instance

**unknown\_keys**

Fail if value contains unknown keys

**value\_type**

Expected Type of all dictionary values

**class** temci.utils.typecheck.DirName(constraint: Optional[Callable[[Any], bool]] = None)Bases: [Str](#)

A valid directory name. If the directory doesn't exist, at least the parent directory must exist.

Creates an instance.

**Parameters****constraint** – function that returns True if the user defined constraint is satisfied**completion\_hints**

Completion hints for supported shells for this type instance

**constraint**

Function that returns True if the user defined constraint is satisfied

`temci.utils.typecheck.E(exp_value) → Exact`

Alias for Exact.

**class** `temci.utils.typecheck.Either(*types: tuple)`

Bases: *Type*

Checks for the value to be of one of several types.

Creates an Either instance.

**Parameters**

**types** – list of types (or SpecialType subclasses)

**Raises**

ConstraintError if some of the constraints aren't (typechecker) Types

**types**

Possible types

**class** `temci.utils.typecheck.Exact(exp_value)`

Bases: *Type*

Checks for value equivalence.

Creates an Exact object.

**Parameters**

**exp\_value** – value to check for

**exp\_value**

Expected value

**class** `temci.utils.typecheck.ExactEither(*exp_values: tuple)`

Bases: *Type*

Checks for the value to be of one of several exact values.

Creates an ExactEither instance.

**Parameters**

**exp\_values** – list of types (or SpecialType subclasses)

**Raises**

ConstraintError if some of the constraints aren't (typechecker) Types

**exp\_values**

Expected values

**class** `temci.utils.typecheck.FileName(constraint: Optional[Callable[[Any], bool]] = None, allow_std: bool = False, allow_non_existent: bool = True)`

Bases: *Str*

A valid file name. If the file doesn't exist, at least the parent directory must exist and the file must be creatable.

Creates an instance.

**Parameters**

- **constraint** – function that returns True if the user defined constraint is satisfied
- **allow\_std** – allow '-' as standard out or in

- **allow\_non\_existent** – allow files that don't exist

**allow\_non\_existent**

Allow files that don't exist

**allow\_std**

Allow '-' as standard out or in

**completion\_hints**

Completion hints for supported shells for this type instance

**constraint**

Function that returns True if the user defined constraint is satisfied

`temci.utils.typecheck.FileNameOrStdOut()` → *FileName*

A valid file name or "-" for standard out.

`temci.utils.typecheck.Float(constraint: Optional[Callable[[Any], bool]] = None)` → `Union[T, Constraint]`

Alias for `Constraint(constraint, T(float))` or `T(float)`

**Parameters**

**constraint** – function that returns True if the user defined constraint is satisfied

**class** `temci.utils.typecheck.Info(value_name: Optional[str] = None, value=None, _app_str: Optional[str] = None)`

Bases: `object`

Information object that is used to produce meaningful type check error messages.

Creates a new info object.

**Parameters**

- **value\_name** – name of the value that is type checked
- **value** – value that is type checked

`add_to_name(app_str: str)` → *Info*

Creates a new info object based on this one with the given appendix to its value representation. It's used to give information about what part of the main value is currently examined.

**Parameters**

**app\_str** – app string appended to the own app string to create the app string for the new info object

**Returns**

new info object

`errorMsg(constraint: Type, value, msg: Optional[str] = None)` → *InfoMsg*

Creates an info message object with the passed expected type and the optional message.

**Parameters**

- **constraint** – passed expected type
- **value** – value that violates the constraint
- **msg** – additional message, it should give more information about why the constraint isn't met

**errmsg\_cond**(*cond: bool, constraint: Type, value, msg: Optional[str] = None*) → *InfoMsg*

Creates an info message object with the passed expected type and the optional message.

**Parameters**

- **cond** – if this is false *InfoMsg(True)* is returned.
- **constraint** – passed expected type
- **value** – value that violates th constraint
- **msg** – additional message, it should give more information about why the constraint isn't met

**errmsg\_key\_non\_existent**(*constraint: Type, key: str*) → *InfoMsg*

Creates an info message object with the passed expected type that contains the message that currently examined part of the value is unexpected.

**Parameters**

**constraint** – passed expected type

**errmsg\_unexpected**(*key: str*) → *InfoMsg*

Creates an info message object with the passed expected type that contains the message that currently examined part of the value has an unexpected key.

**Parameters**

**key** – the unexpected key

**get\_value**() → *Any*

Get the main value of this object. :raises: *ValueError* if the main value isn't set

**has\_value**

Is the value property of this info object set to a meaningful value?

**set\_value**(*value*)

Set the main value of this object

**value**

Main value that is type checked

**value\_name**

Name of the value that is typechecked

**wrap**(*result: bool*) → *InfoMsg*

Wrap the passed bool into a *InfoMsg* object.

**class** `temci.utils.typecheck.Int`(*constraint: Optional[Callable[[Any], bool]] = None, range: Optional[range] = None, description: Optional[str] = None*)

Bases: *Type*

Checks for the value to be of type int and to adhere to some constraints.

Creates an instance.

**Parameters**

- **constraint** – function that returns True if the user defined constraint is satisfied
- **range** – range (or list) that the value has to be part of
- **description** – description of the constraints



**completion\_hints**

Completion hints for supported shells for this type instance

**constraint**

Function that returns True if the user defined constraint is satisfied

**description**

Description of the constraints

**range**

Range (or list) that the value has to be part of

**class** temci.utils.typecheck.**List**(*elem\_type*: [Type](#) = Any)

Bases: [Type](#)

Checks for the value to be a list with elements of a given type.

Creates a new instance.

**Parameters**

**elem\_type** – type of the list elements

**Raises**

ConstraintError if elem\_type isn't a (typechecker) Types

**elem\_type**

Expected type of the list elements

**get\_default**() → Any

Returns the default value of this type :raises: ValueError if the default value isn't set

**get\_default\_yaml**(*indents*: int = 0, *indentation*: int = 4, *str\_list*: bool = False, *defaults*=None, *comment\_out\_defaults*: bool = False) → Union[str, List[str]]

Produce a YAML like string that contains the default value and the description of this type and its possible sub types.

**Parameters**

- **indents** – number of indents in front of each produced line
- **indentation** – indentation width in number of white spaces
- **str\_list** – return a list of lines instead of a combined string?
- **defaults** – default value that should be used instead of the default value of this instance
- **comment\_out\_defaults** – comment out default values

**class** temci.utils.typecheck.**ListOrTuple**(*elem\_type*: [Type](#) = Any)

Bases: [Type](#)

Checks for the value to be a list or tuple with elements of a given type.

Creates an instance.

**Parameters**

**elem\_type** – type of the list or tuple elements

**Raises**

ConstraintError if elem\_type isn't a (typechecker) Types

**elem\_type**

Expected type of the list or tuple elements

`temci.utils.typecheck.NaturalNumber`(*constraint: Optional[Callable[[Any], bool]] = None*) → *Int*

Matches all natural numbers (ints >= 0) that satisfy the optional user defined constrained.

**Parameters**

**constraint** – function that returns True if the user defined constraint is satisfied

**class** `temci.utils.typecheck.NonErrorConstraint`(*constraint: Callable[[Any], Any], error\_cls: type, constrained\_type: Type = Any, description: Optional[str] = None*)

Bases: *Type*

Checks the passed value by an user defined constraint that fails if it raises an error.

Creates a new instance

**Parameters**

- **constraint** – function that doesn't raise an error if the user defined constraint is satisfied
- **error\_cls** – class of the errors the constraint method raises
- **constrained\_type** – Type that the constraint is applied on
- **description** – short description of the constraint (e.g. ">0")

**Raises**

ConstraintError if constrained\_type isn't a (typechecker) Types

**constrained\_type**

Type that the constraint is applied on

**constraint**

Function that returns True if the user defined constraint is satisfied

**description**

Short description of the constraint (e.g. ">0")

**error\_cls**

Class of the errors the constraint method raises

**class** `temci.utils.typecheck.NonExistent`(*completion\_hints: Optional[Dict[str, Any]] = None*)

Bases: *Type*

Checks a key of a dictionary for existence if its associated value has this type.

Creates an instance.

**Parameters**

**completion\_hints** – completion hints for supported shells for this type instance

**class** `temci.utils.typecheck.Optional`(*other\_type: Type*)

Bases: *Either*

Checks the value and checks that its either of native type None or of another Type constraint. Alias for Either(Exact(None), other\_type)

Creates an Optional instance.

**Parameters**

**other\_type** – type to make optional

**Raises**

ConstraintError if other\_type isn't a (typechecker) Types

`temci.utils.typecheck.PositiveInt`(*constraint: Optional[Callable[[Any], bool]] = None*) → *Int*

Matches all positive integers that satisfy the optional user defined constrained.

#### Parameters

**constraint** – function that returns True if the user defined constraint is satisfied

**class** `temci.utils.typecheck.Str`(*constraint: Optional[Callable[[Any], bool]] = None*)

Bases: *Type*

Checks for the value to be a string an optionally meet some constraints.

Creates an instance.

#### Parameters

**constraint** – function that returns True if the user defined constraint is satisfied

#### constraint

Function that returns True if the user defined constraint is satisfied

**class** `temci.utils.typecheck.StrList`

Bases: *Type*, *ParamType*

A comma separated string list which contains elements from a fixed set of allowed values.

Creates an instance.

#### Parameters

**completion\_hints** – completion hints for supported shells for this type instance

#### allowed\_values

Possible values that can appear in the string list, if None all values are allowed.

**convert**(*value, param, ctx: Context*) → *List[str]*

Convert method that makes this class usable as a click type.

**get\_default\_yaml**(*indents: int = 0, indentation: int = 4, str\_list: bool = False, defaults=None, comment\_out\_defaults: bool = False*) → *str*

Produce a YAML like string that contains the default value and the description of this type and its possible sub types.

#### Parameters

- **indents** – number of indents in front of each produced line
- **indentation** – indentation width in number of white spaces
- **str\_list** – return a list of lines instead of a combined string?
- **defaults** – default value that should be used instead of the default value of this instance
- **comment\_out\_defaults** – comment out default values

**name:** `str = 'comma_sep_str_list'`

click.ParamType name, that makes this class usable as a click type

**class** `temci.utils.typecheck.T`(*native\_type: type*)

Bases: *Type*

Wrapper around a native type.

Creates an instance.

#### Parameters

**native\_type** – wrapped native type

**native\_type**

Native type that is wrapped

**class** temci.utils.typecheck.**Tuple**(\*elem\_types: Tuple[Type])

Bases: [Type](#)

Checks for the value to be a tuple (or a list) with elements of the given types.

Creates a new instance.

**Parameters**

**elem\_types** – types of each tuple element

**Raises**

ConstraintError if elem\_type isn't a (typechecker) Types

**elem\_types**

Expected type of each tuple element

**class** temci.utils.typecheck.**Type**(completion\_hints: Optional[Dict[str, Any]] = None)

Bases: object

A simple type checker type class.

Creates an instance.

**Parameters**

**completion\_hints** – completion hints for supported shells for this type instance

**check**(value) → bool

Checks whether or not the passed value has the type specified by this instance.

**Parameters**

- **value** – passed value
- **info** – info object for creating error messages

**completion\_hints**

Completion hints for supported shells for this type instance

**default**

Default value of this type instance

**description**

Description of this type instance

**dont\_typecheck\_default**() → [Type](#)

Disable type checking the default value. :return: self

**get\_default**() → Any

Returns the default value of this type :raises: ValueError if the default value isn't set

**get\_default\_yaml**(indents: int = 0, indentation: int = 4, str\_list: bool = False, defaults=None, comment\_out\_defaults: bool = False) → Union[str, List[str]]

Produce a YAML like string that contains the default value and the description of this type and its possible sub types.

**Parameters**

- **indents** – number of indents in front of each produced line
- **indentation** – indentation width in number of white spaces

- **str\_list** – return a list of lines instead of a combined string?
- **defaults** – default value that should be used instead of the default value of this instance
- **comment\_out\_defaults** – comment out default values

**has\_default()** → bool

Does this type instance have a default value?

**string\_representation**(*indents: int = 0, indentation: int = 4, str\_list: bool = False, defaults=None*) → Union[str, List[str]]

Produce a YAML string that contains the default value (if possible), the description of this type and more and its possible sub types.

#### Parameters

- **indents** – number of indents in front of each produced line
- **indentation** – indentation width in number of white spaces
- **str\_list** – return a list of lines instead of a combined string?
- **defaults** – default value that should be used instead of the default value of this instance

**typecheck\_default**

Type check the default value

**class** temci.utils.typecheck.ValidTimeSpan

Bases: [Type](#), ParamType

A string that is parseable as timespan by pytimeparse. E.g. “32m” or “2h 32m”.

Creates an instance.

#### Parameters

**completion\_hints** – completion hints for supported shells for this type instance

**convert**(*value, param, ctx: Context*) → int

Convert method that makes this class usable as a click type.

**name:** **str** = 'valid\_timespan'

click.ParamType name, that makes this class usable as a click type

**class** temci.utils.typecheck.ValidYamlFileName(*allow\_non\_existent: bool = False*)

Bases: [Str](#)

A valid file name that refers to a valid YAML file.

Create an instance.

#### Parameters

**allow\_non\_existent** – allow files that don’t exist

**allow\_non\_existent**

Allow files that don’t exist

**completion\_hints**

Completion hints for supported shells for this type instance

temci.utils.typecheck.YAML\_FILE\_COMPLETION\_HINT = "\_files -g '\*\\\*.yaml'"

YAML file name completion hint for ZSH

`temci.utils.typecheck.typecheck(value, type: Union[Type, type], value_name: Optional[str] = None)`

Like `verbose_isinstance` but raises an error if the value hasn't the expected type.

#### Parameters

- **value** – passed value
- **type** – expected type of the value
- **value\_name** – optional description of the value

#### Raises

`TypeError`

`temci.utils.typecheck.typecheck_locals(locals: Optional[Dict[str, Any]] = None, **variables: Dict[str, Union[Type, type]])`

Like `typecheck` but checks several variables for their associated expected type. The advantage against `typecheck` is that it sets the value descriptions properly. Example usage:

```
def func(a: str, b: int):
    typecheck_locals(locals(), a=Str(), b=Int())
```

#### Parameters

- **locals** – directory to get the variable values from
- **variables** – variable names with their associated expected types

#### Raises

`TypeError`

`temci.utils.typecheck.verbose_isinstance(value, type: Union[Type, type], value_name: Optional[str] = None) → InfoMsg`

Verbose version of `isinstance` that returns a `InfoMsg` object.

#### Parameters

- **value** – value to check
- **type** – type or `Type` to check for
- **value\_name** – name of the passed value (improves the error message)

## temci.utils.util module

Utility functions and classes that don't depend on the rest of the temci code base.

**class** `temci.utils.util.InsertionTimeOrderedDict`

Bases: `object`

A dictionary which's elements are sorted by their insertion time.

**classmethod** `from_list(items: Optional[list], key_func: Callable[[Any], Any]) → InsertionTimeOrderedDict`

Creates an ordered dict out of a list of elements.

#### Parameters

- **items** – list of elements
- **key\_func** – function that returns a key for each passed list element

**Returns**

created ordered dict with the elements in the same order as in the passed list

**items()** → List[Tuple[Any, Any]]

**keys()** → List

Returns all keys of this dictionary. They are sorted by their insertion time.

**values()** → List

Returns all values of this dictionary. They are sorted by their insertion time.

**class** temci.utils.util.Singleton

Bases: type

Singleton meta class. @see <http://stackoverflow.com/a/6798042>

temci.utils.util.allow\_all\_imports = True

Allow all imports (should the can\_import method return true for every module)?

temci.utils.util.can\_import(*module: str*) → bool

Can a module (like scipy or numpy) be imported without a severe and avoidable performance penalty? The rational behind this is that some parts of temci don't need scipy or numpy.

**Parameters**

**module** – name of the module

temci.utils.util.document(\*\**kwargs: Dict[str, str]*)

Document

**Parameters**

**kwargs** – class attribute, documentation prefix

temci.utils.util.does\_command\_succeed(*cmd: str*) → bool

Does the passed command succeed (when executed by /bin/sh)?

temci.utils.util.does\_program\_exist(*program: str*) → bool

Does the passed program exist?

temci.utils.util.geom\_std(*values: List[float]*) → float

Calculates the geometric standard deviation for the passed values. Source: [https://en.wikipedia.org/wiki/Geometric\\_standard\\_deviation](https://en.wikipedia.org/wiki/Geometric_standard_deviation)

temci.utils.util.get\_cache\_line\_size(*cache\_level: Optional[int] = None*) → Optional[int]

Returns the cache line size of the cache on the given level. Level 0 and 1 are actually on the same level.

**Parameters**

**cache\_level** – if None the highest level cache is used

**Returns**

cache line size or none if the cache on the given level doesn't exist

temci.utils.util.get\_distribution\_name() → str

Returns the name of the current linux distribution (requires *lsb\_release* to be installed)

temci.utils.util.get\_distribution\_release() → str

Returns the used release of the current linux distribution (requires *lsb\_release* to be installed)

temci.utils.util.get\_doc\_for\_type\_scheme(*type\_scheme: Type*) → str

Return a class documentation string for the given type scheme. Use the default\_yaml method.

`temci.utils.util.get_memory_page_size()` → int

Returns the size of a main memory page

`temci.utils.util.handler = <RainbowLoggingHandler <stderr> (NOTSET)>`

Colored logging handler that is used for the root logger

`temci.utils.util.has_pdflatex()` → bool

Is pdflatex installed?

`temci.utils.util.has_root_privileges()` → bool

Has the current user root privileges?

`temci.utils.util.in_standalone_mode = False`

In rudimentary standalone mode (executed via run.py)

`temci.utils.util.join_strs(strs: List[str], last_word: str = 'and')` → str

Joins the passed strings together with “,” except for the last to strings that separated by the passed word.

#### Parameters

- **strs** – strings to join
- **last\_word** – passed word that is used between the two last strings

`temci.utils.util.on_apple_os()` → bool

Is the current operating system an apple OS X?

`temci.utils.util.parse_timespan(time: str)` → float

Parse a time span expression, see <https://pypi.org/project/pytimeparse/>

Supports -1 to express an infinite time span

#### Parameters

**time** – time span expression, mixture of different time units is possible

#### Returns

time span in seconds

**class** `temci.utils.util.proc_wait_with_rusage`

Bases: object

Each Popen object gets a field rusage

`temci.utils.util.recursive_exec_for_leafs(data: dict, func, _path_prep=[])`

Executes the function for every leaf key (a key without any sub keys) of the data dict tree.

#### Parameters

- **data** – dict tree
- **func** – function that gets passed the leaf key, the key path and the actual value

`temci.utils.util.rusage_header()` → str

`temci.utils.util.sphinx_doc()` → bool

Is the code only loaded to document it with sphinx?

`temci.utils.util.warn_for_pdflatex_non_existence_once(_warned=[False])`

Log a warning if the pdflatex isn't available, but only if this function is called the first time



**temci.utils.vcs module**

**class** temci.utils.vcs.**FileDriver**(*dir*: str = '.', *branch*: Optional[str] = None)

Bases: [VCSDriver](#)

The default driver, that works with plain old files and directories without any vcs. Therefore its also usable with every directory. It has only one revision: Number -1 or 'HEAD', the current directory content.

This class is also a simple example implementation of a VCSDriver.

Initializes the VCS driver for a given base directory. It also sets the current branch if it's defined in the Settings

**Parameters**

- **dir** – base directory
- **branch** – used branch

**copy\_revision**(*id\_or\_num*: Union[int, str], *sub\_dir*: str, *dest\_dirs*: List[str])

Copy the sub directory of the current vcs base directory into all of the destination directories.

**Parameters**

- **id\_or\_num** – id or number of the revision (-1 and 'HEAD' represent the uncommitted changes)
- **sub\_dir** – sub directory of the current vcs base directory relative to it
- **dest\_dirs** – list of destination directories in which the content of the sub dir is placed or dest dir string

**Raises**

VCSError if something goes wrong while copying the directories

**get\_branch**() → Optional[str]

Gets the current branch.

**Returns**

current branch name

**Raises**

VCSError if something goes terribly wrong

**get\_info\_for\_revision**(*id\_or\_num*: Union[int, str]) → dict

Get an info dict for the given commit (-1 and 'HEAD' represent the uncommitted changes). Structure of the info dict:

```
"commit_id"; ...,
"commit_message": ...,
"commit_number": ...,
"is_uncommitted": True/False,
"is_from_other_branch": True/False,
"branch": ... # branch name or empty string if this commit belongs to no branch
```

**Parameters**

**id\_or\_num** – id or number of the commit

**Returns**

info dict

**Raises**

VCSError if the number or id isn't valid

**has\_uncommitted()** → bool

Check for uncommitted changes in the repository.

**classmethod is\_suited\_for\_dir**(*dir: str = '.'*)

Checks whether or not this vcs driver can work with the passed base directory.

**Parameters**

**dir** – passed base directory path

**number\_of\_revisions()** → int

Number of committed revisions in the current branch (if branches are supported). :return: number of revisions

**set\_branch**(*new\_branch: str*)

Sets the current branch and throws an error if the branch doesn't exist.

**Parameters**

**new\_branch** – new branch to set

**Raises**

VCSError if new\_branch doesn't exist

**validate\_revision**(*id\_or\_num: Union[int, str]*) → bool

Validate the existence of the referenced revision.

**Parameters**

**id\_or\_num** – id or number of the referenced revision

**Returns**

does it exists?

**class** temci.utils.vcs.**GitDriver**(*dir: str = '.', branch: Optional[str] = None*)

Bases: [\*VCSDriver\*](#)

The driver for git repositories.

Initializes the VCS driver for a given base directory. It also sets the current branch if it's defined in the Settings

**Parameters**

- **dir** – base directory
- **branch** – used branch

**copy\_revision**(*id\_or\_num: Union[int, str], sub\_dir: str, dest\_dirs: List[str]*)

Copy the sub directory of the current vcs base directory into all of the destination directories.

**Parameters**

- **id\_or\_num** – id or number of the revision (-1 and 'HEAD' represent the uncommitted changes)
- **sub\_dir** – sub directory of the current vcs base directory relative to it
- **dest\_dirs** – list of destination directories in which the content of the sub dir is placed or dest dir string

**Raises**

VCSError if something goes wrong while copying the directories

**get\_branch()** → Optional[str]

Gets the current branch.

**Returns**

current branch name

**Raises**

VCSError if something goes terribly wrong

**get\_info\_for\_revision**(*id\_or\_num*: Union[int, str]) → dict

Get an info dict for the given commit (-1 and 'HEAD' represent the uncommitted changes). Structure of the info dict:

```
"commit_id": ...,
"commit_message": ...,
"commit_number": ...,
"is_uncommitted": True/False,
"is_from_other_branch": True/False,
"branch": ... # branch name or empty string if this commit belongs to no branch
```

**Parameters**

**id\_or\_num** – id or number of the commit

**Returns**

info dict

**Raises**

VCSError if the number or id isn't valid

**get\_valid\_branches()** → Optional[List[str]]

Gets the valid branches for the associated repository or None if the vcs doesn't support branches.

**has\_uncommitted()** → bool

Check for uncommitted changes in the repository.

**classmethod is\_suited\_for\_dir**(*dir*: str = '.') → bool

Checks whether or not this vcs driver can work with the passed base directory.

**Parameters**

**dir** – passed base directory path

**number\_of\_revisions()** → str

Number of committed revisions in the current branch (if branches are supported). :return: number of revisions

**set\_branch**(*new\_branch*: str)

Sets the current branch and throws an error if the branch doesn't exist.

**Parameters**

**new\_branch** – new branch to set

**Raises**

VCSError if new\_branch doesn't exist

**validate\_revision**(*id\_or\_num*: Union[int, str]) → bool

Validate the existence of the referenced revision.

**Parameters**

**id\_or\_num** – id or number of the referenced revision

**Returns**

does it exists?

**exception** `temci.utils.vcs.NoSuchRevision`

Bases: `VCSError`

Thrown if a specific revision doesn't exist

**exception** `temci.utils.vcs.NoSuchVCSError`

Bases: `VCSError`

Thrown if there isn't a vcs with the specific name

**class** `temci.utils.vcs.VCSDriver`(*dir: str = '.', branch: Optional[str] = None*)

Bases: `object`

Abstract version control system driver class used to support different vcss.

Initializes the VCS driver for a given base directory. It also sets the current branch if it's defined in the Settings

**Parameters**

- **dir** – base directory
- **branch** – used branch

**branch**

Used branch

**copy\_revision**(*id\_or\_num: Union[int, str], sub\_dir: str, dest\_dirs: List[str]*)

Copy the sub directory of the current vcs base directory into all of the destination directories.

**Parameters**

- **id\_or\_num** – id or number of the revision (-1 and 'HEAD' represent the uncommitted changes)
- **sub\_dir** – sub directory of the current vcs base directory relative to it
- **dest\_dirs** – list of destination directories in which the content of the sub dir is placed or dest dir string

**Raises**

`VCSError` if something goes wrong while copying the directories

**dir**

Base directory

**get\_branch**() → `Optional[str]`

Gets the current branch.

**Returns**

current branch name

**Raises**

`VCSError` if something goes terribly wrong

**get\_info\_for\_all\_revisions**(*max: int = -1*) → `List[Dict[str, Any]]`

Get an info dict for all revisions. A single info dict has the following structure:

```
"commit_id"; ...,
"commit_message": ...,
"commit_number": ...,
"is_uncommitted": True/False,
"is_from_other_branch": True/False,
"branch": ... # branch name or empty string if this commit belongs to no branch
```

**Parameters**

**max** – if max isn't -1 it gives the maximum number of revision infos returned

**Returns**

list of info dicts

**get\_info\_for\_revision**(*id\_or\_num: Union[int, str]*) → dict

Get an info dict for the given commit (-1 and 'HEAD' represent the uncommitted changes). Structure of the info dict:

```
"commit_id"; ...,
"commit_message": ...,
"commit_number": ...,
"is_uncommitted": True/False,
"is_from_other_branch": True/False,
"branch": ... # branch name or empty string if this commit belongs to no branch
```

**Parameters**

**id\_or\_num** – id or number of the commit

**Returns**

info dict

**Raises**

VCSError if the number or id isn't valid

**classmethod get\_suited\_vcs**(*mode='auto', dir='.', branch: Optional[str] = None*) → *VCSDriver*

Chose the best suited vcs driver for the passed base directory and the passed mode. If mode is "auto" the best suited vcs driver is chosen. If mode is "git" or "file", the GitDriver or the FileDriver is chosen. If the chosen driver isn't applicable than a VCSError is raised.

**Parameters**

- **mode** – passed mode
- **dir** – base directory
- **branch** – used branch

**Returns**

vcs driver for the base directory

**Raises**

VCSError if the selected driver isn't applicable

**get\_valid\_branches**() → Optional[List[str]]

Gets the valid branches for the associated repository or None if the vcs doesn't support branches.

**has\_uncommitted**() → bool

Check for uncommitted changes in the repository.

```
id_type = Either(Str()|Int())
```

```
classmethod is_suited_for_dir(dir: str = '.') → bool
```

Checks whether or not this vcs driver can work with the passed base directory.

**Parameters**

**dir** – passed base directory path

```
number_of_revisions() → int
```

Number of committed revisions in the current branch (if branches are supported). :return: number of revisions

```
set_branch(new_branch: str)
```

Sets the current branch and throws an error if the branch doesn't exist.

**Parameters**

**new\_branch** – new branch to set

**Raises**

VCSError if new\_branch doesn't exist

```
validate_revision(id_or_num: Union[int, str]) → bool
```

Validate the existence of the referenced revision.

**Parameters**

**id\_or\_num** – id or number of the referenced revision

**Returns**

does it exists?

```
exception temci.utils.vcs.VCSError
```

Bases: OSError

Base error for the errors that occur during vcs handling

## Module contents

Package with utility modules.

### 7.13.2 Module contents

## 7.14 Resources

This is a collection of additional resources that are related to temci.

### Bachelor thesis

The development of temci started as part of this bachelor thesis at the Karlsruhe Institute of Technology. It's written in German.

A talk in german on the topic of benchmarking in german:

- [modindex](#)
- [search](#)

## PYTHON MODULE INDEX

### t

- temci, 290
- temci.build, 66
- temci.build.build\_processor, 62
- temci.build.builder, 64
- temci.misc, 74
- temci.misc.game, 66
- temci.report, 241
- temci.report.report, 210
- temci.report.report\_processor, 217
- temci.report.rundata, 217
- temci.report.stats, 225
- temci.report.testers, 239
- temci.run, 109
- temci.run.cpuset, 74
- temci.run.run\_driver, 75
- temci.run.run\_driver\_plugin, 96
- temci.run.run\_processor, 105
- temci.run.run\_worker\_pool, 106
- temci.scripts, 209
- temci.scripts.cli, 110
- temci.scripts.temci\_completion, 209
- temci.scripts.version, 209
- temci.setup, 210
- temci.setup.setup, 210
- temci.utils, 290
- temci.utils.click\_helper, 241
- temci.utils.config\_utils, 245
- temci.utils.library\_init, 245
- temci.utils.mail, 245
- temci.utils.number, 245
- temci.utils.plugin, 249
- temci.utils.registry, 249
- temci.utils.settings, 250
- temci.utils.sudo\_utils, 269
- temci.utils.typecheck, 269
- temci.utils.util, 282
- temci.utils.vcs, 285





## A

AbstractRegistry (class in *temci.utils.registry*), 249  
 AbstractReporter (class in *temci.report.report*), 210  
 AbstractRunDriver (class in *temci.run.run\_driver*), 75  
 AbstractRunDriverPlugin (class in *temci.run.run\_driver\_plugin*), 96  
 AbstractRunWorkerPool (class in *temci.run.run\_worker\_pool*), 106  
 active (*temci.run.cpuset.CPUSet* attribute), 75  
 add\_data\_block() (*temci.report.rundata.RunData* method), 218  
 add\_data\_block() (*temci.report.rundata.RunDataStatsHelper* method), 220  
 add\_error() (*temci.report.rundata.RunDataStatsHelper* method), 221  
 add\_property\_descriptions() (*temci.report.rundata.RunDataStatsHelper* method), 221  
 add\_run\_data() (*temci.run.run\_driver.BenchmarkingResultBlock* method), 77  
 add\_to\_name() (*temci.utils.typecheck.Info* method), 275  
 All (class in *temci.utils.typecheck*), 269  
 allow\_all\_imports (in module *temci.utils.util*), 283  
 allow\_non\_existent (*temci.utils.typecheck.FileName* attribute), 275  
 allow\_non\_existent (*temci.utils.typecheck.ValidYamlFile* attribute), 281  
 allow\_std (*temci.utils.typecheck.FileName* attribute), 275  
 allowed\_values (*temci.utils.typecheck.StrList* attribute), 279  
 amean\_std() (in module *temci.misc.game*), 71  
 AndersonTester (class in *temci.report.testers*), 239  
 Any (class in *temci.utils.typecheck*), 270  
 append (*temci.run.run\_processor.RunProcessor* attribute), 105  
 append() (*temci.utils.click\_helper.CmdOptionList* method), 243  
 apply\_override\_actions() (*temci.utils.settings.Settings* method), 265  
 apply\_program\_filter() (*temci.misc.game.Language* method), 68

apply\_program\_filter() (*temci.misc.game.ProgramCategory* method), 70  
 array (*temci.report.stats.SingleProperty* attribute), 229  
 attributes (*temci.report.rundata.RunData* attribute), 218  
 attributes (*temci.report.stats.Single* attribute), 229  
 attributes (*temci.run.run\_driver.RunProgramBlock* attribute), 88  
 av (*temci.run.run\_driver.ValidPropertyList* attribute), 94  
 av\_cores (*temci.run.cpuset.CPUSet* attribute), 75  
 AV\_GHC\_VERSIONS (in module *temci.misc.game*), 66

## B

base\_core\_number (*temci.run.cpuset.CPUSet* attribute), 75  
 BaseObject (class in *temci.misc.game*), 66  
 BaseStatObject (class in *temci.report.stats*), 225  
 bench\_as\_different\_user() (in module *temci.utils.sudo\_utils*), 269  
 bench\_categories() (in module *temci.misc.game*), 71  
 bench\_category() (in module *temci.misc.game*), 71  
 bench\_file() (in module *temci.misc.game*), 71  
 bench\_program() (in module *temci.misc.game*), 71  
 BENCH\_SET (in module *temci.run.cpuset*), 74  
 benchmark() (*temci.run.run\_driver.AbstractRunDriver* method), 75  
 benchmark() (*temci.run.run\_driver.ExecRunDriver* method), 81  
 benchmark() (*temci.run.run\_driver.ShellRunDriver* method), 91  
 benchmark() (*temci.run.run\_processor.RunProcessor* method), 105  
 benchmark\_and\_exit() (in module *temci.scripts.cli*), 110  
 BenchmarkingError, 77  
 BenchmarkingProgramError, 77  
 BenchmarkingResultBlock (class in *temci.run.run\_driver*), 77  
 BenchmarkingThread (class in *temci.run.run\_worker\_pool*), 107

- [benchmarks\(\)](#) (*temci.report.rundata.RunData* method), 218  
[block\\_run\\_count](#) (*temci.run.run\_processor.RunProcessor* attribute), 105  
[block\\_scheme](#) (*temci.build.build\_processor.BuildProcessor* attribute), 63  
[block\\_type\\_scheme](#) (*temci.report.rundata.RunData* attribute), 218  
[block\\_type\\_scheme](#) (*temci.run.run\_driver.AbstractRunDriver* attribute), 76  
[block\\_type\\_scheme](#) (*temci.run.run\_driver.ExecRunDriver* attribute), 81  
[block\\_type\\_scheme](#) (*temci.run.run\_driver.ShellRunDriver* attribute), 92  
[Bool](#) (class in *temci.utils.typecheck*), 270  
[BoolOrNone](#) (class in *temci.utils.typecheck*), 270  
[border\\_value](#) (*temci.report.stats.EffectToSmallError* attribute), 226  
[border\\_value](#) (*temci.report.stats.EffectToSmallWarning* attribute), 227  
[border\\_value](#) (*temci.report.stats.NotEnoughObservationsError* attribute), 227  
[border\\_value](#) (*temci.report.stats.NotEnoughObservationsWarning* attribute), 227  
[border\\_value](#) (*temci.report.stats.StatMessage* attribute), 233  
[border\\_value](#) (*temci.report.stats.StdDeviationToHighError* attribute), 235  
[border\\_value](#) (*temci.report.stats.StdDeviationToHighWarning* attribute), 235  
[BOTableColumn](#) (class in *temci.misc.game*), 66  
[boxplot\(\)](#) (*temci.report.stats.SinglesProperty* method), 232  
[boxplot\\_html\(\)](#) (*temci.misc.game.BaseObject* method), 66  
[boxplot\\_html\\_for\\_data\(\)](#) (*temci.misc.game.BaseObject* method), 66  
[branch](#) (*temci.utils.vcs.VCSDriver* attribute), 288  
[build\(\)](#) (*temci.build.build\_processor.BuildProcessor* method), 63  
[build\(\)](#) (*temci.build.builder.Builder* method), 64  
[build\(\)](#) (*temci.misc.game.BaseObject* method), 66  
[build\(\)](#) (*temci.misc.game.Implementation* method), 67  
[build\(\)](#) (*temci.misc.game.Language* method), 68  
[build\(\)](#) (*temci.misc.game.Program* method), 69  
[build\(\)](#) (*temci.misc.game.ProgramCategory* method), 70  
[build\(\)](#) (*temci.misc.game.ProgramWithInput* method), 70  
[build\(\)](#) (*temci.run.run\_processor.RunProcessor* method), 105  
[build\\_cmd](#) (*temci.build.builder.Builder* attribute), 64  
[build\\_cmd](#) (*temci.build.builder.BuilderQueueItem* property), 65  
[build\\_dir](#) (*temci.build.builder.Builder* attribute), 64  
[Builder](#) (class in *temci.build.builder*), 64  
[BuilderKeyboardInterrupt](#), 64  
[BuilderQueueItem](#) (class in *temci.build.builder*), 65  
[BuildError](#), 64  
[BuilderThread](#) (class in *temci.build.builder*), 65  
[BuildProcessor](#) (class in *temci.build.build\_processor*), 62  
**C**  
[c\\_config\(\)](#) (in module *temci.misc.game*), 71  
[callback](#) (*temci.utils.click\_helper.CmdOption* attribute), 242  
[can\\_import\(\)](#) (in module *temci.utils.util*), 283  
[check\(\)](#) (*temci.utils.typecheck.Type* method), 280  
[check\\_value\(\)](#) (*temci.report.stats.EffectToSmallWarning* class method), 227  
[check\\_value\(\)](#) (*temci.report.stats.NotEnoughObservationsWarning* class method), 227  
[check\\_value\(\)](#) (*temci.report.stats.SignificanceTooLowError* class method), 228  
[check\\_value\(\)](#) (*temci.report.stats.SignificanceTooLowWarning* class method), 228  
[check\\_value\(\)](#) (*temci.report.stats.StatMessage* class method), 233  
[check\\_value\(\)](#) (*temci.report.stats.StdDeviationToHighWarning* class method), 235  
[chown\(\)](#) (in module *temci.utils.sudo\_utils*), 269  
[clean\\_output\(\)](#) (in module *temci.run.run\_driver*), 95  
[cli\(\)](#) (in module *temci.scripts.temci\_completion*), 209  
[cli\\_with\\_error\\_catching\(\)](#) (in module *temci.scripts.cli*), 110  
[cli\\_with\\_verb\\_arg\\_handling\(\)](#) (in module *temci.scripts.cli*), 110  
[clone\(\)](#) (*temci.report.rundata.RunData* method), 219  
[clone\(\)](#) (*temci.report.rundata.RunDataStatsHelper* method), 221  
[cmd](#) (*temci.setup.setup.ExecError* attribute), 210  
[cmd\\_option\(\)](#) (in module *temci.utils.click\_helper*), 244  
[CmdOption](#) (class in *temci.utils.click\_helper*), 241  
[CmdOptionList](#) (class in *temci.utils.click\_helper*), 243  
[Codespeed2Reporter](#) (class in *temci.report.report*), 212  
[CodespeedReporter](#) (class in *temci.report.report*), 212  
[combine\(\)](#) (*temci.report.stats.StatMessage* static method), 233  
[completion\\_dir\(\)](#) (in module *temci.scripts.temci\_completion*), 209  
[completion\\_file\\_name\(\)](#) (in module *temci.scripts.temci\_completion*), 209  
[completion\\_hints](#) (*temci.utils.click\_helper.CmdOption* attribute), 242  
[completion\\_hints](#) (*temci.utils.typecheck.Bool* attribute), 270

completion\_hints (temci.utils.typecheck.BoolOrNone attribute), 270  
 completion\_hints (temci.utils.typecheck.DirName attribute), 273  
 completion\_hints (temci.utils.typecheck.FileName attribute), 275  
 completion\_hints (temci.utils.typecheck.Int attribute), 276  
 completion\_hints (temci.utils.typecheck.Type attribute), 280  
 completion\_hints (temci.utils.typecheck.ValidYamlFileName attribute), 281  
 CompletionHint (class in temci.utils.typecheck), 271  
 config (temci.run.run\_driver.ExecValidator attribute), 85  
 config\_file\_name (temci.utils.settings.Settings attribute), 265  
 config\_type\_scheme (temci.run.run\_driver.ExecValidator attribute), 85  
 ConsoleReporter (class in temci.report.report), 213  
 constrained\_type (temci.utils.typecheck.Constraint attribute), 271  
 constrained\_type (temci.utils.typecheck.NonErrorConstraint attribute), 278  
 Constraint (class in temci.utils.typecheck), 271  
 constraint (temci.utils.typecheck.Constraint attribute), 271  
 constraint (temci.utils.typecheck.DirName attribute), 273  
 constraint (temci.utils.typecheck.FileName attribute), 275  
 constraint (temci.utils.typecheck.Int attribute), 277  
 constraint (temci.utils.typecheck.NonErrorConstraint attribute), 278  
 constraint (temci.utils.typecheck.Str attribute), 279  
 CONTROLLER\_SUB\_BENCH\_SET (in module temci.run.cpuset), 74  
 convert() (temci.utils.typecheck.BoolOrNone method), 270  
 convert() (temci.utils.typecheck.StrList method), 279  
 convert() (temci.utils.typecheck.ValidTimeSpan method), 281  
 copy() (temci.run.run\_driver.RunProgramBlock method), 88  
 copy\_revision() (temci.utils.vcs.FileDriver method), 285  
 copy\_revision() (temci.utils.vcs.GitDriver method), 286  
 copy\_revision() (temci.utils.vcs.VCSDriver method), 288  
 cparser\_config() (in module temci.misc.game), 71  
 CPU\_PATHS (temci.run.run\_driver\_plugin.DisableTurboBoost attribute), 99  
 CPUGovernor (class in temci.run.run\_driver\_plugin), 96  
 CPUSet (class in temci.run.cpuset), 74  
 CPUSet (class in temci.run.run\_driver\_plugin), 97  
 cpuset (temci.run.run\_worker\_pool.AbstractRunWorkerPool attribute), 107  
 cpuset (temci.run.run\_worker\_pool.BenchmarkingThread attribute), 108  
 CPUSSET\_DIR (in module temci.run.cpuset), 74  
 CPUSpecExecRunner (class in temci.run.run\_driver), 77  
 create\_completion\_dir() (in module temci.scripts.temci\_completion), 209  
 create\_if\_valid() (temci.report.stats.StatMessage class method), 233  
 create\_run\_driver\_function() (in module temci.scripts.cli), 110  
 create\_temci\_run\_file() (temci.misc.game.Language method), 68  
 CSVReporter (class in temci.report.report), 211

## D

data (temci.report.rundata.RunData attribute), 219  
 data (temci.report.stats.SingleProperty attribute), 229  
 data (temci.run.run\_driver.BenchmarkingResultBlock attribute), 77  
 data (temci.run.run\_driver.RunProgramBlock attribute), 88  
 Default (class in temci.utils.typecheck), 271  
 default (temci.report.report.ReporterRegistry attribute), 216  
 default (temci.report.testers.TesterRegistry attribute), 241  
 default (temci.run.run\_driver.AbstractRunDriver attribute), 76  
 default (temci.run.run\_driver.ExecRunDriver attribute), 82  
 default (temci.run.run\_driver.RunDriverRegistry attribute), 88  
 default (temci.utils.click\_helper.CmdOption attribute), 242  
 default (temci.utils.registry.AbstractRegistry attribute), 249  
 default (temci.utils.typecheck.BoolOrNone attribute), 270  
 default (temci.utils.typecheck.Default attribute), 272  
 default (temci.utils.typecheck.Type attribute), 280  
 default() (temci.utils.settings.Settings method), 265  
 Description (class in temci.utils.typecheck), 272  
 description (temci.utils.click\_helper.CmdOption attribute), 242  
 description (temci.utils.typecheck.Constraint attribute), 271  
 description (temci.utils.typecheck.Description attribute), 272  
 description (temci.utils.typecheck.Int attribute), 277

- description (*temci.utils.typecheck.NonErrorConstraint attribute*), 278
- description (*temci.utils.typecheck.Type attribute*), 280
- description() (*temci.report.rundata.RunData method*), 219
- description() (*temci.report.stats.BaseStatObject method*), 225
- description() (*temci.report.stats.Single method*), 229
- description() (*temci.report.stats.SingleProperty method*), 229
- description() (*temci.report.stats.TestedPair method*), 235
- description() (*temci.report.stats.TestedPairProperty method*), 236
- description() (*temci.run.run\_driver.RunProgramBlock method*), 89
- deviation (*temci.utils.number.FNumber attribute*), 246
- Dict (*class in temci.utils.typecheck*), 272
- DIGIT\_CHANGE (*temci.utils.number.ParenthesesMode attribute*), 247
- dir (*temci.utils.vcs.VCSDriver attribute*), 288
- DirName (*class in temci.utils.typecheck*), 273
- disable\_hyper\_threading() (*temci.run.run\_worker\_pool.AbstractRunWorkerPool class method*), 107
- DisableAmdTurbo (*class in temci.run.run\_driver\_plugin*), 97
- DisableASLR (*class in temci.run.run\_driver\_plugin*), 97
- DisableCPUCaches (*class in temci.run.run\_driver\_plugin*), 98
- DisableHyperThreading (*class in temci.run.run\_driver\_plugin*), 98
- DisableIntelTurbo (*class in temci.run.run\_driver\_plugin*), 98
- DisableSwap (*class in temci.run.run\_driver\_plugin*), 99
- DisableTurboBoost (*class in temci.run.run\_driver\_plugin*), 99
- discard\_run\_data() (*temci.report.rundata.RunDataStatsHelper method*), 221
- discarded\_runs (*temci.run.run\_processor.RunProcessor attribute*), 105
- DiscardedRuns (*class in temci.run.run\_driver\_plugin*), 99
- divide\_inputs() (*in module temci.misc.game*), 71
- document() (*in module temci.utils.util*), 283
- document\_func() (*in module temci.utils.click\_helper*), 244
- does\_command\_succeed() (*in module temci.utils.util*), 283
- does\_program\_exist() (*in module temci.utils.util*), 283
- dont\_typecheck\_default() (*temci.utils.typecheck.Type method*), 280
- driver (*temci.run.run\_worker\_pool.BenchmarkingThread attribute*), 108
- DropFSCaches (*class in temci.run.run\_driver\_plugin*), 100
- ## E
- E() (*in module temci.utils.typecheck*), 274
- EffectToSmallError (*class in temci.report.stats*), 226
- EffectToSmallWarning (*class in temci.report.stats*), 226
- Either (*class in temci.utils.typecheck*), 274
- elem\_type (*temci.utils.typecheck.List attribute*), 277
- elem\_type (*temci.utils.typecheck.ListOrTuple attribute*), 277
- elem\_types (*temci.utils.typecheck.Tuple attribute*), 280
- empty\_inputs() (*in module temci.misc.game*), 72
- enable\_hyper\_threading() (*temci.run.run\_worker\_pool.AbstractRunWorkerPool class method*), 107
- end\_time (*temci.run.run\_processor.RunProcessor attribute*), 105
- entropy (*temci.misc.game.Program attribute*), 69
- env\_info\_scheme (*temci.report.rundata.RunData attribute*), 219
- EnvRandomizePlugin (*class in temci.run.run\_driver\_plugin*), 100
- eq\_except\_property() (*temci.report.stats.BaseStatObject method*), 225
- eq\_except\_property() (*temci.report.stats.Single method*), 229
- eq\_except\_property() (*temci.report.stats.SingleProperty method*), 230
- eq\_except\_property() (*temci.report.stats.TestedPair method*), 236
- eq\_except\_property() (*temci.report.stats.TestedPairProperty method*), 237
- equal\_prob() (*temci.report.stats.TestedPairProperty method*), 237
- err (*temci.setup.setup.ExecError attribute*), 210
- erroneous\_run\_blocks (*temci.run.run\_processor.RunProcessor attribute*), 105
- error (*temci.build.builder.BuilderKeyboardInterrupt attribute*), 64
- ERROR (*temci.report.stats.StatMessageType attribute*), 234
- error (*temci.run.run\_driver.BenchmarkingResultBlock attribute*), 77
- error\_cls (*temci.utils.typecheck.NonErrorConstraint attribute*), 278
- ErrorCode (*class in temci.scripts.cli*), 110
- errmsg() (*temci.utils.typecheck.Info method*), 275



- [errmsg\\_cond\(\)](#) (*temci.utils.typecheck.Info* method), 275  
[errmsg\\_key\\_non\\_existent\(\)](#) (*temci.utils.typecheck.Info* method), 276  
[errmsg\\_unexpected\(\)](#) (*temci.utils.typecheck.Info* method), 276  
[errors\(\)](#) (*temci.report.stats.BaseStatObject* method), 225  
[estimate\\_needed\\_runs\(\)](#) (*temci.report.testers.Tester* method), 240  
[estimate\\_time\(\)](#) (*temci.report.rundata.RunDataStatsHelper* method), 221  
[estimate\\_time\\_for\\_next\\_round\(\)](#) (*temci.report.rundata.RunDataStatsHelper* method), 221  
[Exact](#) (class in *temci.utils.typecheck*), 274  
[ExactEither](#) (class in *temci.utils.typecheck*), 274  
[exclude\\_invalid\(\)](#) (*temci.report.rundata.RunData* method), 219  
[exclude\\_invalid\(\)](#) (*temci.report.rundata.RunDataStatsHelper* method), 222  
[exclude\\_properties\(\)](#) (*temci.report.rundata.RunData* method), 219  
[exclude\\_properties\(\)](#) (*temci.report.rundata.RunDataStatsHelper* method), 222  
[excluded\\_properties\\_per\\_run\\_data](#) (*temci.report.rundata.ExcludedInvalidData* attribute), 217  
[excluded\\_run\\_datas](#) (*temci.report.rundata.ExcludedInvalidData* attribute), 217  
[ExcludedInvalidData](#) (class in *temci.report.rundata*), 217  
[exec\(\)](#) (in module *temci.setup.setup*), 210  
[ExecError](#), 210  
[ExecRunDriver](#) (class in *temci.run.run\_driver*), 78  
[ExecRunDriver.ExecResult](#) (class in *temci.run.run\_driver*), 81  
[ExecRunner](#) (class in *temci.run.run\_driver*), 83  
[ExecValidator](#) (class in *temci.run.run\_driver*), 84  
[exp\\_value](#) (*temci.utils.typecheck.Exact* attribute), 274  
[exp\\_values](#) (*temci.utils.typecheck.ExactEither* attribute), 274  
[external](#) (*temci.report.rundata.RunData* attribute), 219  
[external\\_count](#) (*temci.report.rundata.RunDataStatsHelper* attribute), 222
- F**
- [file\\_entropy\(\)](#) (in module *temci.misc.game*), 72  
[file\\_lines\(\)](#) (in module *temci.misc.game*), 72  
[FileDriver](#) (class in *temci.utils.vcs*), 285  
[FileName](#) (class in *temci.utils.typecheck*), 274  
[FileNameOrStdOut\(\)](#) (in module *temci.utils.typecheck*), 275  
[filter\\_runs\(\)](#) (in module *temci.run.run\_driver*), 95  
[first](#) (*temci.report.stats.TestedPair* attribute), 236  
[first](#) (*temci.report.stats.TestedPairProperty* attribute), 237  
[first\(\)](#) (in module *temci.misc.game*), 72  
[first\\_inputs\(\)](#) (in module *temci.misc.game*), 72  
[first\\_rel\\_to\\_second\(\)](#) (*temci.report.stats.TestedPair* method), 236  
[first\\_rel\\_to\\_second\(\)](#) (*temci.report.stats.TestedPairProperty* method), 237  
[first\\_rel\\_to\\_second\\_std\(\)](#) (*temci.report.stats.TestedPair* method), 236  
[fixed\\_runs](#) (*temci.run.run\_processor.RunProcessor* attribute), 105  
[FLOAT](#) (*temci.report.stats.StatMessageValueFormat* attribute), 234  
[Float](#) (in module *temci.utils.typecheck*), 275  
[FlushCPUCaches](#) (class in *temci.run.run\_driver\_plugin*), 101  
[FNumber](#) (class in *temci.utils.number*), 245  
[fnumber\(\)](#) (in module *temci.utils.number*), 247  
[for\\_exception\(\)](#) (*temci.report.rundata.RecordedInternalError* class method), 217  
[format\(\)](#) (*temci.utils.number.FNumber* method), 246  
[format\\_number\(\)](#) (in module *temci.utils.number*), 247  
[format\\_number\\_sn\(\)](#) (in module *temci.utils.number*), 248  
[from\\_config\\_dict\(\)](#) (*temci.misc.game.BaseObject* class method), 66  
[from\\_config\\_dict\(\)](#) (*temci.misc.game.Implementation* class method), 67  
[from\\_config\\_dict\(\)](#) (*temci.misc.game.Input* class method), 67  
[from\\_config\\_dict\(\)](#) (*temci.misc.game.Language* class method), 68  
[from\\_config\\_dict\(\)](#) (*temci.misc.game.Program* class method), 69  
[from\\_config\\_dict\(\)](#) (*temci.misc.game.ProgramCategory* class method), 70  
[from\\_dict\(\)](#) (*temci.run.run\_driver.RunProgramBlock* class method), 89  
[from\\_list\(\)](#) (*temci.utils.util.InsertionTimeOrderedDict* class method), 282  
[from\\_non\\_plugin\\_settings\(\)](#) (*temci.utils.click\_helper.CmdOption* class method), 242  
[from\\_registry\(\)](#) (*temci.utils.click\_helper.CmdOption* class method), 242
- G**
- [generate\\_msg\\_text\(\)](#) (*temci.report.stats.StatMessage*

- method*), 233
- `geom_mean_rel_to_best` (*temci.misc.game.Mode attribute*), 69
- `geom_std()` (*in module temci.utils.util*), 283
- `get()` (*temci.utils.settings.Settings method*), 265
- `get_av_perf_stat_properties()` (*in module temci.run.run\_driver*), 95
- `get_av_rusage_properties()` (*in module temci.run.run\_driver*), 95
- `get_av_time_properties()` (*in module temci.run.run\_driver*), 95
- `get_av_time_properties_with_format_specifiers` (*in module temci.run.run\_driver*), 95
- `get_bench_uid_and_gid()` (*in module temci.utils.sudo\_utils*), 269
- `get_bench_user()` (*in module temci.utils.sudo\_utils*), 269
- `get_box_plot_html()` (*temci.misc.game.Language method*), 68
- `get_box_plot_html()` (*temci.misc.game.Program method*), 69
- `get_box_plot_html()` (*temci.misc.game.ProgramCategory method*), 70
- `get_box_plot_html()` (*temci.misc.game.ProgramWithInput method*), 71
- `get_box_plot_per_input_per_impl_html()` (*temci.misc.game.Language method*), 68
- `get_box_plot_per_input_per_impl_html()` (*temci.misc.game.Program method*), 69
- `get_box_plot_per_input_per_impl_html()` (*temci.misc.game.ProgramCategory method*), 70
- `get_branch()` (*temci.utils.vcs.FileDriver method*), 285
- `get_branch()` (*temci.utils.vcs.GitDriver method*), 286
- `get_branch()` (*temci.utils.vcs.VCSDriver method*), 288
- `get_cache_line_size()` (*in module temci.utils.util*), 283
- `get_class()` (*temci.utils.registry.AbstractRegistry class method*), 249
- `get_data_frame()` (*temci.report.stats.BaseStatObject method*), 225
- `get_data_frame()` (*temci.report.stats.Single method*), 229
- `get_data_frame()` (*temci.report.stats.SingleProperty method*), 230
- `get_data_frame()` (*temci.report.stats.SinglesProperty method*), 232
- `get_data_frame()` (*temci.report.stats.TestedPairProperty method*), 237
- `get_default()` (*temci.utils.typecheck.Dict method*), 272
- `get_default()` (*temci.utils.typecheck.List method*), 277
- `get_default()` (*temci.utils.typecheck.Type method*), 280
- `get_default_yaml()` (*temci.utils.typecheck.Dict method*), 272
- `get_default_yaml()` (*temci.utils.typecheck.List method*), 277
- `get_default_yaml()` (*temci.utils.typecheck.StrList method*), 279
- `get_default_yaml()` (*temci.utils.typecheck.Type method*), 280
- `get_description()` (*temci.utils.typecheck.Dict method*), 272
- `get_description_clusters()` (*temci.report.rundata.RunDataStatsHelper method*), 222
- `get_description_clusters_and_single()` (*temci.report.rundata.RunDataStatsHelper method*), 222
- `get_distribution_name()` (*in module temci.utils.util*), 283
- `get_distribution_release()` (*in module temci.utils.util*), 283
- `get_doc_for_type_scheme()` (*in module temci.utils.util*), 283
- `get_env_setting()` (*in module temci.utils.sudo\_utils*), 269
- `get_evaluation()` (*temci.report.rundata.RunDataStatsHelper method*), 222
- `get_for_name()` (*temci.utils.registry.AbstractRegistry class method*), 249
- `get_for_tag()` (*in module temci.report.rundata*), 224
- `get_for_tags()` (*in module temci.report.rundata*), 224
- `get_full_block_typescheme()` (*temci.run.run\_driver.AbstractRunDriver class method*), 76
- `get_full_html()` (*temci.misc.game.Language method*), 68
- `get_geom_over_rel_means()` (*temci.misc.game.BaseObject method*), 66
- `get_geom_over_rel_stds()` (*temci.misc.game.BaseObject method*), 66
- `get_geom_std_over_rel_means()` (*temci.misc.game.BaseObject method*), 66
- `get_gsd_for_x_per_impl()` (*temci.misc.game.BaseObject method*), 66
- `get_html()` (*temci.misc.game.Language method*), 68
- `get_html()` (*temci.misc.game.Program method*), 69
- `get_html()` (*temci.misc.game.ProgramCategory method*), 70
- `get_html()` (*temci.misc.game.ProgramWithInput method*), 71
- `get_html2()` (*temci.misc.game.Language method*), 68
- `get_html2()` (*temci.misc.game.Program method*), 69
- `get_html2()` (*temci.misc.game.ProgramCategory method*), 70

[get\\_html2\(\)](#) (*temci.misc.game.ProgramWithInput method*), 71  
[get\\_hyper\\_threading\\_cores\(\)](#) (*temci.run.run\_worker\_pool.AbstractRunWorkerPool class method*), 107  
[get\\_impl\\_mean\\_scores\(\)](#) (*temci.misc.game.Language method*), 68  
[get\\_impl\\_mean\\_scores\(\)](#) (*temci.misc.game.Program method*), 69  
[get\\_impl\\_mean\\_scores\(\)](#) (*temci.misc.game.ProgramCategory method*), 70  
[get\\_info\\_for\\_all\\_revisions\(\)](#) (*temci.utils.vcs.VCSDriver method*), 288  
[get\\_info\\_for\\_revision\(\)](#) (*temci.utils.vcs.FileDriver method*), 285  
[get\\_info\\_for\\_revision\(\)](#) (*temci.utils.vcs.GitDriver method*), 287  
[get\\_info\\_for\\_revision\(\)](#) (*temci.utils.vcs.VCSDriver method*), 289  
[get\\_input\\_strs\(\)](#) (*temci.misc.game.ProgramCategory method*), 70  
[get\\_max\\_input\\_num\(\)](#) (*temci.misc.game.Language method*), 68  
[get\\_means\\_rel\\_to\\_best\(\)](#) (*temci.misc.game.ProgramWithInput method*), 71  
[get\\_memory\\_page\\_size\(\)](#) (*in module temci.utils.util*), 283  
[get\\_pair\(\)](#) (*temci.report.stats.TestedPairsAndSingles method*), 238  
[get\\_program\\_ids\\_to\\_bench\(\)](#) (*temci.report.rundata.RunDataStatsHelper method*), 222  
[get\\_property\\_descriptions\(\)](#) (*temci.run.run\_driver.AbstractRunDriver method*), 76  
[get\\_property\\_descriptions\(\)](#) (*temci.run.run\_driver.ExecRunDriver method*), 82  
[get\\_property\\_descriptions\(\)](#) (*temci.run.run\_driver.ExecRunner method*), 83  
[get\\_property\\_descriptions\(\)](#) (*temci.run.run\_driver.OutputExecRunner method*), 86  
[get\\_property\\_descriptions\(\)](#) (*temci.run.run\_driver.RusageExecRunner method*), 90  
[get\\_property\\_descriptions\(\)](#) (*temci.run.run\_driver.TimeExecRunner method*), 93  
[get\\_random\\_filename\(\)](#) (*temci.report.report.HTMLReporter2 method*), 215  
[get\\_reduced\\_x\\_per\\_impl\(\)](#) (*temci.misc.game.BaseObject method*), 66  
[get\\_runner\(\)](#) (*temci.run.run\_driver.ExecRunDriver class method*), 83  
[get\\_scores\\_per\\_impl\(\)](#) (*temci.misc.game.Language method*), 68  
[get\\_scores\\_per\\_impl\(\)](#) (*temci.misc.game.ProgramCategory method*), 70  
[get\\_single\(\)](#) (*temci.misc.game.ProgramWithInput method*), 71  
[get\\_single\\_properties\(\)](#) (*temci.misc.game.ProgramWithInput method*), 71  
[get\\_single\\_properties\(\)](#) (*temci.report.rundata.RunData method*), 219  
[get\\_single\\_property\(\)](#) (*temci.misc.game.Implementation method*), 67  
[get\\_sphinx\\_doc\(\)](#) (*temci.utils.click\_helper.CmdOptionList method*), 243  
[get\\_stat\\_messages\(\)](#) (*temci.report.stats.BaseStatObject method*), 225  
[get\\_stat\\_messages\(\)](#) (*temci.report.stats.TestedPairsAndSingles method*), 238  
[get\\_statistical\\_properties\\_for\\_each\(\)](#) (*temci.misc.game.ProgramWithInput method*), 71  
[get\\_statistical\\_property\\_scores\(\)](#) (*temci.misc.game.Language method*), 68  
[get\\_statistical\\_property\\_scores\(\)](#) (*temci.misc.game.Program method*), 69  
[get\\_statistical\\_property\\_scores\(\)](#) (*temci.misc.game.ProgramCategory method*), 70  
[get\\_statistical\\_property\\_scores\\_per\\_impl\(\)](#) (*temci.misc.game.Language method*), 68  
[get\\_statistical\\_property\\_scores\\_per\\_impl\(\)](#) (*temci.misc.game.ProgramCategory method*), 70  
[get\\_statistical\\_property\\_scores\\_per\\_input\\_per\\_impl\(\)](#) (*temci.misc.game.Language method*), 68  
[get\\_statistical\\_property\\_scores\\_per\\_input\\_per\\_impl\(\)](#) (*temci.misc.game.Program method*), 69  
[get\\_statistical\\_property\\_scores\\_per\\_input\\_per\\_impl\(\)](#) (*temci.misc.game.ProgramCategory method*), 70  
[get\\_sub\\_set\(\)](#) (*temci.run.cpuset.CPUSet method*), 75  
[get\\_suited\\_vcs\(\)](#) (*temci.utils.vcs.VCSDriver class method*), 289  
[get\\_tester\(\)](#) (*temci.utils.registry.AbstractRegistry*

- class method*), 249
- `get_type_scheme()` (*temci.utils.settings.Settings method*), 265
- `get_used()` (*temci.utils.registry.AbstractRegistry class method*), 249
- `get_used_plugins()` (*temci.run.run\_driver.AbstractRunDriver method*), 76
- `get_used_plugins()` (*temci.run.run\_driver.ExecRunDriver method*), 83
- `get_valid_branches()` (*temci.utils.vcs.GitDriver method*), 287
- `get_valid_branches()` (*temci.utils.vcs.VCSDriver method*), 289
- `get_value()` (*temci.utils.typecheck.Info method*), 276
- `get_x_per_impl()` (*temci.misc.game.BaseObject method*), 67
- `get_x_per_impl()` (*temci.misc.game.Implementation method*), 67
- `get_x_per_impl()` (*temci.misc.game.ProgramWithInput method*), 71
- `get_x_per_impl_and_input()` (*temci.misc.game.Language method*), 68
- `get_x_per_impl_and_input()` (*temci.misc.game.ProgramCategory method*), 70
- `GitDriver` (*class in temci.utils.vcs*), 286
- H**
- `handler` (*in module temci.utils.util*), 284
- `has_completion_hints` (*temci.utils.click\_helper.CmdOption attribute*), 242
- `has_default` (*temci.utils.click\_helper.CmdOption attribute*), 242
- `has_default()` (*temci.utils.typecheck.Dict method*), 273
- `has_default()` (*temci.utils.typecheck.Type method*), 281
- `has_description` (*temci.utils.click\_helper.CmdOption attribute*), 243
- `has_error()` (*temci.report.rundata.RunData method*), 219
- `has_error()` (*temci.report.rundata.RunDataStatsHelper method*), 223
- `has_errors()` (*temci.report.stats.BaseStatObject method*), 225
- `has_key()` (*temci.utils.settings.Settings method*), 266
- `has_log_level()` (*temci.utils.settings.Settings method*), 266
- `has_pdflatex()` (*in module temci.utils.util*), 284
- `has_root_privileges()` (*in module temci.utils.util*), 284
- `has_short` (*temci.utils.click\_helper.CmdOption attribute*), 243
- `has_time_left()` (*temci.run.run\_worker\_pool.AbstractRunWorkerPool method*), 107
- `has_uncommitted()` (*temci.utils.vcs.FileDriver method*), 286
- `has_uncommitted()` (*temci.utils.vcs.GitDriver method*), 287
- `has_uncommitted()` (*temci.utils.vcs.VCSDriver method*), 289
- `has_value` (*temci.utils.typecheck.Info attribute*), 276
- `has_warnings()` (*temci.report.stats.BaseStatObject method*), 225
- `haskel_config()` (*in module temci.misc.game*), 72
- `header()` (*in module temci.run.run\_driver*), 95
- `hint` (*temci.report.stats.EffectToSmallWarning attribute*), 227
- `hint` (*temci.report.stats.NotEnoughObservationsWarning attribute*), 227
- `hint` (*temci.report.stats.SignificanceTooLowWarning attribute*), 228
- `hint` (*temci.report.stats.StatMessage attribute*), 233
- `hint` (*temci.report.stats.StdDeviationToHighWarning attribute*), 235
- `hints` (*temci.utils.typecheck.CompletionHint attribute*), 271
- `histogram()` (*temci.report.stats.BaseStatObject method*), 225
- `hostname()` (*in module temci.utils.mail*), 245
- `html_escape_property()` (*in module temci.report.report*), 216
- `html_escape_property()` (*temci.report.report.HTMLReporter2 class method*), 216
- `HTMLReporter` (*class in temci.report.report*), 214
- `HTMLReporter2` (*class in temci.report.report*), 214
- I**
- `id` (*temci.build.builder.BuilderQueueItem property*), 65
- `id` (*temci.build.builder.BuilderThread attribute*), 65
- `id` (*temci.run.run\_driver.RunProgramBlock attribute*), 89
- `id` (*temci.run.run\_worker\_pool.BenchmarkingThread attribute*), 108
- `id_program_filter()` (*in module temci.misc.game*), 72
- `id_type` (*temci.utils.vcs.VCSDriver attribute*), 289
- `img_filename_ending` (*temci.report.stats.BaseStatObject attribute*), 226
- `Implementation` (*class in temci.misc.game*), 67
- `in_standalone_mode` (*in module temci.utils.util*), 284
- `include_properties()` (*temci.report.rundata.RunData method*), 219
- `include_properties()` (*temci.report.rundata.RunDataStatsHelper method*), 223



`Info` (class in `temci.utils.typecheck`), 275

`init_from_dicts()` (`temci.report.rundata.RunDataStatsHelper` class method), 223

`init_from_file()` (`temci.report.rundata.RunDataStatsHelper` class method), 223

`init_settings()` (`temci.utils.number.FNumber` class method), 246

`Input` (class in `temci.misc.game`), 67

`InsertionTimeOrderedDict` (class in `temci.utils.util`), 282

`Int` (class in `temci.utils.typecheck`), 276

`INT` (`temci.report.stats.StatMessageValueFormat` attribute), 234

`iqr()` (`temci.report.stats.SingleProperty` method), 230

`is_enqueued` (`temci.run.run_driver.RunProgramBlock` attribute), 89

`is_equal()` (`temci.report.rundata.RunDataStatsHelper` method), 224

`is_equal()` (`temci.report.stats.TestedPairProperty` method), 237

`is_equal()` (`temci.report.testers.Tester` method), 240

`is_flag` (`temci.utils.click_helper.CmdOption` attribute), 243

`is_obsolete()` (`temci.utils.settings.Settings` method), 266

`is_obsolete()` (`temci.utils.typecheck.Dict` method), 273

`is_perf_available()` (in module `temci.run.run_driver`), 95

`is_single_valued()` (`temci.report.stats.BaseStatObject` method), 226

`is_single_valued()` (`temci.report.stats.SingleProperty` method), 230

`is_single_valued()` (`temci.report.stats.TestedPairProperty` method), 237

`is_suited_for_dir()` (`temci.utils.vcs.FileDriver` class method), 286

`is_suited_for_dir()` (`temci.utils.vcs.GitDriver` class method), 287

`is_suited_for_dir()` (`temci.utils.vcs.VCSDriver` class method), 290

`is_uncertain()` (`temci.report.rundata.RunDataStatsHelper` method), 224

`is_uncertain()` (`temci.report.testers.Tester` method), 240

`is_unequal()` (`temci.report.rundata.RunDataStatsHelper` method), 224

`is_unequal()` (`temci.report.testers.Tester` method), 240

`items()` (`temci.utils.util.InsertionTimeOrderedDict` method), 283

## J

`join_strs()` (in module `temci.utils.util`), 284

## K

`key_type` (`temci.utils.typecheck.Dict` attribute), 273

`keys()` (`temci.utils.util.InsertionTimeOrderedDict` method), 283

`KSTester` (class in `temci.report.testers`), 239

## L

`Language` (class in `temci.misc.game`), 67

`last_inputs()` (in module `temci.misc.game`), 72

`List` (class in `temci.utils.typecheck`), 277

`list_from_numbers()` (`temci.misc.game.Input` class method), 67

`ListOrTuple` (class in `temci.utils.typecheck`), 277

`load_file()` (`temci.utils.settings.Settings` method), 266

`load_files()` (`temci.utils.settings.Settings` method), 266

`load_from_config_dir()` (`temci.utils.settings.Settings` method), 266

`load_from_current_dir()` (`temci.utils.settings.Settings` method), 266

`load_from_dict()` (`temci.utils.settings.Settings` method), 266

`load_from_dir()` (`temci.utils.settings.Settings` method), 266

`load_plugins()` (in module `temci.utils.plugin`), 249

`log()` (`temci.build.builder.BuildError` method), 64

`log_program_error()` (in module `temci.run.run_driver`), 95

`long_properties()` (`temci.report.rundata.RunData` method), 220

`long_properties()` (`temci.report.rundata.RunDataStatsHelper` method), 224

## M

`make_descriptions_distinct()` (`temci.report.rundata.RunDataStatsHelper` method), 224

`make_scripts()` (in module `temci.setup.setup`), 210

`map()` (`temci.utils.number.ParenthesesMode` class method), 247

`max()` (`temci.report.stats.SingleProperty` method), 230

`max()` (`temci.report.stats.SinglesProperty` method), 232

`max_rel_std_dev()` (`temci.report.stats.TestedPairProperty` method), 237

`max_runs` (`temci.run.run_processor.RunProcessor` attribute), 105

`max_std_dev()` (`temci.report.stats.TestedPairProperty` method), 237

`maximum_of_max_runs()` (`temci.run.run_processor.RunProcessor` method), 105

`maximum_of_min_runs()` (`temci.run.run_processor.RunProcessor` method), 105

`mean()` (*temci.misc.game.Implementation method*), 67  
`mean()` (*temci.report.stats.SingleProperty method*), 230  
`mean_ci()` (*temci.report.stats.SingleProperty method*), 230  
`mean_diff()` (*temci.report.stats.TestedPairProperty method*), 237  
`mean_diff_ci()` (*temci.report.stats.TestedPairProperty method*), 237  
`mean_diff_per_dev()` (*temci.report.stats.TestedPairProperty method*), 237  
`mean_diff_per_mean()` (*temci.report.stats.TestedPairProperty method*), 237  
`mean_rel_std()` (*in module temci.misc.game*), 72  
`mean_rel_to_first` (*temci.misc.game.Mode attribute*), 69  
`mean_rel_to_one` (*temci.misc.game.Mode attribute*), 69  
`mean_score_column()` (*in module temci.misc.game*), 72  
`mean_score_std_column()` (*in module temci.misc.game*), 72  
`mean_std_dev()` (*temci.report.stats.TestedPairProperty method*), 238  
`median()` (*temci.report.stats.SingleProperty method*), 230  
`merge_different_versions_of_the_same()` (*temci.misc.game.Language class method*), 68  
`message` (*temci.report.stats.EffectToSmallWarning attribute*), 227  
`message` (*temci.report.stats.NotEnoughObservationsWarning attribute*), 228  
`message` (*temci.report.stats.SignificanceTooLowWarning attribute*), 228  
`message` (*temci.report.stats.StatMessage attribute*), 233  
`message` (*temci.report.stats.StdDeviationToHighWarning attribute*), 235  
`min()` (*temci.report.stats.SingleProperty method*), 230  
`min_observations()` (*temci.report.stats.TestedPairProperty method*), 238  
`min_runs` (*temci.run.run\_processor.RunProcessor attribute*), 106  
`min_values()` (*temci.report.rundata.RunData method*), 220  
`misc` (*temci.report.report.AbstractReporter attribute*), 211  
`misc` (*temci.run.run\_driver.ExecRunner attribute*), 83  
`misc_options` (*temci.run.run\_driver.CPUSpecExecRunner attribute*), 78  
`misc_options` (*temci.run.run\_driver.ExecRunner attribute*), 83  
`misc_options` (*temci.run.run\_driver.OutputExecRunner attribute*), 86  
`misc_options` (*temci.run.run\_driver.PerfStatExecRunner attribute*), 87  
`misc_options` (*temci.run.run\_driver.RusageExecRunner attribute*), 90  
`misc_options` (*temci.run.run\_driver.SpecExecRunner attribute*), 92  
`misc_options` (*temci.run.run\_driver.TimeExecRunner attribute*), 94  
`misc_settings` (*temci.report.testers.Tester attribute*), 240  
`misc_settings` (*temci.run.run\_driver.AbstractRunDriver attribute*), 76  
`Mode` (*class in temci.misc.game*), 69  
`modify_setting()` (*temci.utils.settings.Settings method*), 266  
`modify_type_scheme()` (*temci.utils.settings.Settings method*), 267  
`module`  
    *temci*, 290  
    *temci.build*, 66  
    *temci.build.build\_processor*, 62  
    *temci.build.builder*, 64  
    *temci.misc*, 74  
    *temci.misc.game*, 66  
    *temci.report*, 241  
    *temci.report.report*, 210  
    *temci.report.report\_processor*, 217  
    *temci.report.rundata*, 217  
    *temci.report.stats*, 225  
    *temci.report.testers*, 239  
    *temci.run*, 109  
    *temci.run.cpuset*, 74  
    *temci.run.run\_driver*, 75  
    *temci.run.run\_driver\_plugin*, 96  
    *temci.run.run\_processor*, 105  
    *temci.run.run\_worker\_pool*, 106  
    *temci.scripts*, 209  
    *temci.scripts.cli*, 110  
    *temci.scripts.temci\_completion*, 209  
    *temci.scripts.version*, 209  
    *temci.setup*, 210  
    *temci.setup.setup*, 210  
    *temci.utils*, 290  
    *temci.utils.click\_helper*, 241  
    *temci.utils.config\_utils*, 245  
    *temci.utils.library\_init*, 245  
    *temci.utils.mail*, 245  
    *temci.utils.number*, 245  
    *temci.utils.plugin*, 249  
    *temci.utils.registry*, 249  
    *temci.utils.settings*, 250  
    *temci.utils.sudo\_utils*, 269  
    *temci.utils.typecheck*, 269  
    *temci.utils.util*, 282  
    *temci.utils.vcs*, 285

[move\\_process\\_to\\_set\(\)](#) (*temci.run.cpuset.CPUSet* method), 75

## N

[name](#) (*temci.report.testers.AndersonTester* attribute), 239  
[name](#) (*temci.report.testers.KSTester* attribute), 239  
[name](#) (*temci.report.testers.Tester* attribute), 240  
[name](#) (*temci.report.testers.TTester* attribute), 239  
[name](#) (*temci.run.run\_driver.CPUSpecExecRunner* attribute), 78  
[name](#) (*temci.run.run\_driver.ExecRunner* attribute), 84  
[name](#) (*temci.run.run\_driver.OutputExecRunner* attribute), 86  
[name](#) (*temci.run.run\_driver.PerfStatExecRunner* attribute), 87  
[name](#) (*temci.run.run\_driver.RusageExecRunner* attribute), 90  
[name](#) (*temci.run.run\_driver.SpecExecRunner* attribute), 93  
[name](#) (*temci.run.run\_driver.TimeExecRunner* attribute), 94  
[name](#) (*temci.utils.typecheck.Bool* attribute), 270  
[name](#) (*temci.utils.typecheck.BoolOrNone* attribute), 270  
[name](#) (*temci.utils.typecheck.StrList* attribute), 279  
[name](#) (*temci.utils.typecheck.ValidTimeSpan* attribute), 281  
[native\\_type](#) (*temci.utils.typecheck.T* attribute), 280  
[NaturalNumber\(\)](#) (in module *temci.utils.typecheck*), 277  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.AbstractRunDriverPlugin* attribute), 96  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.CPUGovernor* attribute), 97  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.CPUSet* attribute), 97  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.DisableAmdTurbo* attribute), 98  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.DisableASLR* attribute), 97  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.DisableCPUCaches* attribute), 98  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.DisableHyperThreading* attribute), 98  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.DisableIntelTurbo* attribute), 99  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.DisableSwap* attribute), 99  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.DisableTurboBoost* attribute), 99  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.DropFSCaches* attribute), 100  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.FlushCPUCaches* attribute), 101  
[needs\\_root\\_privileges](#) (*temci.run.run\_driver\_plugin.NicePlugin* attribute), 102  
[NEW\\_ROOT\\_SET](#) (in module *temci.run.cpuset*), 75  
[next\\_block\\_timeout\(\)](#) (*temci.run.run\_worker\_pool.AbstractRunWorkerPool* method), 107  
[NicePlugin](#) (class in *temci.run.run\_driver\_plugin*), 101  
[NO\\_ERROR](#) (*temci.scripts.cli.ErrorCode* attribute), 110  
[NonErrorConstraint](#) (class in *temci.utils.typecheck*), 278  
[NonExistent](#) (class in *temci.utils.typecheck*), 278  
[normality\(\)](#) (*temci.report.stats.SingleProperty* method), 230  
[NoSuchRevision](#), 288  
[NoSuchVCSError](#), 288  
[NotEnoughObservationsError](#) (class in *temci.report.stats*), 227  
[NotEnoughObservationsWarning](#) (class in *temci.report.stats*), 227  
[Number](#) (in module *temci.report.rundata*), 217  
[Number](#) (in module *temci.report.stats*), 228  
[Number](#) (in module *temci.run.run\_driver*), 85  
[Number](#) (in module *temci.utils.number*), 247  
[number](#) (*temci.build.builder.Builder* attribute), 64  
[number](#) (*temci.build.builder.BuilderQueueItem* property), 65  
[number\\_of\\_revisions\(\)](#) (*temci.utils.vcs.FileDriver* method), 286  
[number\\_of\\_revisions\(\)](#) (*temci.utils.vcs.GitDriver* method), 287  
[number\\_of\\_revisions\(\)](#) (*temci.utils.vcs.VCSDriver* method), 290  
[number\\_of\\_singles\(\)](#) (*temci.report.stats.TestedPairsAndSingles* method), 238  
[observations\(\)](#) (*temci.report.stats.SingleProperty* method), 230  
[obsoleteness\\_reason\(\)](#) (*temci.utils.settings.Settings* method), 267

`obsoleteness_reason()` (*temci.utils.typecheck.Dict method*), 273

`on_apple_os()` (in module *temci.utils.util*), 284

`option_name` (*temci.utils.click\_helper.CmdOption attribute*), 243

`Optional` (class in *temci.utils.typecheck*), 278

`options` (*temci.utils.click\_helper.CmdOptionList attribute*), 243

`ORDER_OF_MAGNITUDE` (*temci.utils.number.ParenthesesMode attribute*), 247

`OtherNicePlugin` (class in *temci.run.run\_driver.plugin*), 102

`out` (*temci.build.build\_processor.BuildProcessor attribute*), 63

`out` (*temci.setup.setup.ExecError attribute*), 210

`outliers()` (*temci.report.stats.SingleProperty method*), 230

`OutputExecRunner` (class in *temci.run.run\_driver*), 85

`overridden()` (*temci.report.stats.StatMessage class method*), 233

`overridden()` (*temci.report.stats.StatWarning class method*), 234

## P

`pairs` (*temci.report.stats.TestedPairsAndSingles property*), 239

`parallel` (*temci.run.cpuset.CPUSet attribute*), 75

`parallel_number` (*temci.run.cpuset.CPUSet attribute*), 75

`parallel_number` (*temci.run.run\_worker\_pool.AbstractRunWorkerPool attribute*), 107

`ParallelRunWorkerPool` (class in *temci.run.run\_worker\_pool*), 108

`parent` (*temci.report.stats.SingleProperty attribute*), 230

`parent` (*temci.report.stats.StatMessage attribute*), 233

`parent` (*temci.report.stats.TestedPairProperty attribute*), 238

`ParenthesesMode` (class in *temci.utils.number*), 247

`parse_processes()` (*temci.run.run\_driver\_plugin.StopStartPlugin method*), 104

`parse_result()` (*temci.run.run\_driver.ExecRunner method*), 84

`parse_result_impl()` (*temci.run.run\_driver.CPUSpecExecRunner method*), 78

`parse_result_impl()` (*temci.run.run\_driver.ExecRunner method*), 84

`parse_result_impl()` (*temci.run.run\_driver.OutputExecRunner method*), 86

`parse_result_impl()` (*temci.run.run\_driver.PerfStatExecRunner method*), 87

`parse_result_impl()` (*temci.run.run\_driver.RusageExecRunner method*), 90

`parse_result_impl()` (*temci.run.run\_driver.SpecExecRunner method*), 93

`parse_result_impl()` (*temci.run.run\_driver.TimeExecRunner method*), 94

`parse_timespan()` (in module *temci.utils.util*), 284

`PERCENT` (*temci.report.stats.StatMessageValueFormat attribute*), 234

`percentile()` (*temci.report.stats.SingleProperty method*), 230

`PerfStatExecRunner` (class in *temci.run.run\_driver*), 86

`plugin_paths()` (in module *temci.utils.plugin*), 249

`plugin_synonym` (*temci.report.report.ReporterRegistry attribute*), 216

`plugin_synonym` (*temci.report.testers.TesterRegistry attribute*), 241

`plugin_synonym` (*temci.run.run\_driver.AbstractRunDriver attribute*), 76

`plugin_synonym` (*temci.run.run\_driver.RunDriverRegistry attribute*), 88

`plugin_synonym` (*temci.utils.registry.AbstractRegistry attribute*), 249

`pool` (*temci.run.run\_processor.RunProcessor attribute*), 106

`pool` (*temci.run.run\_worker\_pool.BenchmarkingThread attribute*), 108

`PositiveInt()` (in module *temci.utils.typecheck*), 278

`prefix_inputs()` (in module *temci.misc.game*), 72

`prefs` (*temci.utils.settings.Settings attribute*), 267

`PreheatPlugin` (class in *temci.run.run\_driver\_plugin*), 102

`preprocess_build_blocks()` (*temci.build.build\_processor.BuildProcessor class method*), 63

`print_help()` (in module *temci.scripts.temci\_completion*), 209

`print_report()` (*temci.run.run\_processor.RunProcessor method*), 106

`proc_wait_with_rusage` (class in *temci.utils.util*), 284

`process()` (in module *temci.misc.game*), 72

`process_result_file()` (*temci.misc.game.Language method*), 68

`produce_ttest_comparison_table()` (in module *temci.misc.game*), 73

`Program` (class in *temci.misc.game*), 69

`PROGRAM_ERROR` (*temci.scripts.cli.ErrorCode attribute*), 110

`ProgramCategory` (class in *temci.misc.game*), 69

`ProgramFilterFunc` (in module *temci.misc.game*), 70



- ProgramWithInput (class in *temci.misc.game*), 70  
 properties (*temci.report.rundata.RunData* attribute), 220  
 properties (*temci.report.stats.Single* attribute), 229  
 properties (*temci.report.stats.StatMessage* attribute), 233  
 properties (*temci.report.stats.TestedPair* attribute), 236  
 properties() (*temci.report.rundata.RunDataStatsHelper* method), 224  
 properties() (*temci.report.stats.TestedPairsAndSingles* method), 239  
 properties() (*temci.run.run\_driver.BenchmarkingResultBlock* method), 77  
 property (*temci.report.stats.SingleProperty* attribute), 231  
 property (*temci.report.stats.SinglesProperty* attribute), 232  
 property (*temci.report.stats.TestedPairProperty* attribute), 238  
 property\_descriptions\_scheme (*temci.report.rundata.RunData* attribute), 220  
 property\_filter\_half() (in module *temci.misc.game*), 73
- ## Q
- quartiles() (*temci.report.stats.SingleProperty* method), 231
- ## R
- range (*temci.utils.typecheck.Int* attribute), 277  
 recorded\_error() (*temci.run.run\_processor.RunProcessor* method), 106  
 RecordedError (class in *temci.report.rundata*), 217  
 RecordedInternalError (class in *temci.report.rundata*), 217  
 RecordedProgramError (class in *temci.report.rundata*), 217  
 recursive\_exec\_for\_leafs() (in module *temci.utils.util*), 284  
 ReduceFunc (in module *temci.misc.game*), 71  
 ref() (in module *temci.misc.game*), 73  
 register() (in module *temci.utils.registry*), 250  
 register() (*temci.run.run\_driver.RunDriverRegistry* class method), 88  
 register() (*temci.utils.registry.AbstractRegistry* class method), 249  
 register\_runner() (*temci.run.run\_driver.ExecRunDriver* class method), 83  
 registry (*temci.report.report.ReporterRegistry* attribute), 216  
 registry (*temci.report.testers.TesterRegistry* attribute), 241  
 registry (*temci.run.run\_driver.AbstractRunDriver* attribute), 76  
 registry (*temci.run.run\_driver.ExecRunDriver* attribute), 83  
 registry (*temci.run.run\_driver.RunDriverRegistry* attribute), 88  
 registry (*temci.utils.registry.AbstractRegistry* attribute), 250  
 rel\_mean\_func() (in module *temci.misc.game*), 73  
 rel\_mean\_property() (in module *temci.misc.game*), 73  
 rel\_std\_dev\_func() (in module *temci.misc.game*), 73  
 rel\_std\_dev\_to\_min\_func() (in module *temci.misc.game*), 73  
 rel\_std\_property() (in module *temci.misc.game*), 73  
 replace() (*temci.misc.game.Input* method), 67  
 replace\_run\_with\_build\_cmd() (in module *temci.misc.game*), 73  
 report() (*temci.report.report.AbstractReporter* method), 211  
 report() (*temci.report.report.Codespeed2Reporter* method), 212  
 report() (*temci.report.report.CodespeedReporter* method), 213  
 report() (*temci.report.report.ConsoleReporter* method), 214  
 report() (*temci.report.report.CSVReporter* method), 212  
 report() (*temci.report.report.HTMLReporter* method), 214  
 report() (*temci.report.report.HTMLReporter2* method), 216  
 report() (*temci.report.report.VelcomReporter* method), 216  
 report() (*temci.report.report\_processor.ReportProcessor* method), 217  
 reporter (*temci.report.report\_processor.ReportProcessor* attribute), 217  
 ReporterRegistry (class in *temci.report.report*), 216  
 ReportProcessor (class in *temci.report.report\_processor*), 217  
 reset() (*temci.utils.settings.Settings* method), 267  
 reset\_plt() (*temci.report.stats.BaseStatObject* method), 226  
 result (*temci.build.builder.BuilderKeyboardInterrupt* attribute), 65  
 result\_queue (*temci.run.run\_worker\_pool.AbstractRunWorkerPool* attribute), 107  
 ResultGenerator (in module *temci.run.run\_worker\_pool*), 109  
 results() (*temci.run.run\_worker\_pool.AbstractRunWorkerPool* method), 107  
 results() (*temci.run.run\_worker\_pool.ParallelRunWorkerPool* method), 108  
 results() (*temci.run.run\_worker\_pool.RunWorkerPool* method), 108

- method*), 109
- revision* (*temci.build.builder.Builder* attribute), 64
- run()* (*temci.build.builder.BuilderThread* method), 65
- run()* (*temci.run.run\_worker\_pool.BenchmarkingThread* method), 108
- run\_block\_size* (*temci.run.run\_processor.RunProcessor* attribute), 106
- run\_blocks* (*temci.run.run\_processor.RunProcessor* attribute), 106
- run\_driver* (*temci.run.run\_worker\_pool.AbstractRunWorkerPool* attribute), 107
- run\_driver\_class* (*temci.run.run\_driver.RunProgramBlock* attribute), 89
- RunData* (class in *temci.report.rundata*), 218
- rundata* (*temci.report.stats.Single* attribute), 229
- rundata* (*temci.report.stats.SingleProperty* attribute), 231
- RunDataStatsHelper* (class in *temci.report.rundata*), 220
- RunDriverRegistry* (class in *temci.run.run\_driver*), 87
- runners* (*temci.run.run\_driver.ExecRunDriver* attribute), 83
- RunProcessor* (class in *temci.run.run\_processor*), 105
- RunProgramBlock* (class in *temci.run.run\_driver*), 88
- runs* (*temci.report.rundata.RunDataStatsHelper* attribute), 224
- runs* (*temci.run.run\_processor.RunProcessor* attribute), 106
- runs\_benchmarks* (*temci.run.run\_driver.AbstractRunDriver* attribute), 76
- runs\_benchmarks* (*temci.run.run\_driver.ShellRunDriver* attribute), 92
- RunWorkerPool* (class in *temci.run.run\_worker\_pool*), 109
- rusage* (*temci.run.run\_driver.ExecRunDriver.ExecResult* property), 81
- rusage\_header()* (in module *temci.utils.util*), 284
- RusageExecRunner* (class in *temci.run.run\_driver*), 89
- rust\_config()* (in module *temci.misc.game*), 73
- S**
- scipy\_stat\_method* (*temci.report.testers.AndersonTester* attribute), 239
- scipy\_stat\_method* (*temci.report.testers.KSTester* attribute), 239
- scipy\_stat\_method* (*temci.report.testers.Tester* attribute), 240
- scipy\_stat\_method* (*temci.report.testers.TTester* attribute), 240
- script\_relative()* (in module *temci.setup.setup*), 210
- second* (*temci.report.stats.TestedPair* attribute), 236
- second* (*temci.report.stats.TestedPairProperty* attribute), 238
- sem()* (*temci.report.stats.SingleProperty* method), 231
- send\_mail()* (in module *temci.utils.mail*), 245
- serialize()* (*temci.report.rundata.RunDataStatsHelper* method), 224
- set()* (*temci.utils.settings.Settings* method), 267
- set\_branch()* (*temci.utils.vcs.FileDriver* method), 286
- set\_branch()* (*temci.utils.vcs.GitDriver* method), 287
- set\_branch()* (*temci.utils.vcs.VCSDriver* method), 290
- set\_difference\_from\_two\_result\_dicts()* (*temci.misc.game.Language* method), 68
- set\_merged\_run\_data\_from\_result\_dict()* (*temci.misc.game.Language* method), 69
- set\_run\_data\_from\_result\_dict()* (*temci.misc.game.Language* method), 69
- set\_short()* (*temci.utils.click\_helper.CmdOptionList* method), 243
- set\_value()* (*temci.utils.typecheck.Info* method), 276
- Settings* (class in *temci.utils.settings*), 250
- settings* (*temci.utils.number.FNumber* attribute), 246
- settings\_format* (*temci.utils.number.FNumber* attribute), 246
- settings\_key* (*temci.utils.click\_helper.CmdOption* attribute), 243
- settings\_key\_path* (*temci.report.report.ReporterRegistry* attribute), 216
- settings\_key\_path* (*temci.report.testers.TesterRegistry* attribute), 241
- settings\_key\_path* (*temci.run.run\_driver.AbstractRunDriver* attribute), 76
- settings\_key\_path* (*temci.run.run\_driver.ExecRunDriver* attribute), 83
- settings\_key\_path* (*temci.run.run\_driver.RunDriverRegistry* attribute), 88
- settings\_key\_path* (*temci.utils.registry.AbstractRegistry* attribute), 250
- SettingsError*, 269
- setup()* (*temci.run.run\_driver.AbstractRunDriver* method), 76
- setup()* (*temci.run.run\_driver\_plugin.AbstractRunDriverPlugin* method), 96
- setup()* (*temci.run.run\_driver\_plugin.CPUGovernor* method), 97
- setup()* (*temci.run.run\_driver\_plugin.CPUSet* method), 97
- setup()* (*temci.run.run\_driver\_plugin.DisableAmdTurbo* method), 98
- setup()* (*temci.run.run\_driver\_plugin.DisableASLR* method), 97
- setup()* (*temci.run.run\_driver\_plugin.DisableCPUCaches* method), 98
- setup()* (*temci.run.run\_driver\_plugin.DisableHyperThreading* method), 98
- setup()* (*temci.run.run\_driver\_plugin.DisableIntelTurbo* method), 99
- setup()* (*temci.run.run\_driver\_plugin.DisableSwap*

- method), 99
- setup() (temci.run.run\_driver\_plugin.DisableTurboBoost method), 99
- setup() (temci.run.run\_driver\_plugin.DiscardedRuns method), 100
- setup() (temci.run.run\_driver\_plugin.NicePlugin method), 102
- setup() (temci.run.run\_driver\_plugin.OtherNicePlugin method), 102
- setup() (temci.run.run\_driver\_plugin.PreheatPlugin method), 103
- setup() (temci.run.run\_driver\_plugin.StopStartPlugin method), 104
- setup\_block() (temci.run.run\_driver.CPUSpecExecRunner method), 78
- setup\_block() (temci.run.run\_driver.ExecRunner method), 84
- setup\_block() (temci.run.run\_driver.OutputExecRunner method), 86
- setup\_block() (temci.run.run\_driver.PerfStatExecRunner method), 87
- setup\_block() (temci.run.run\_driver.RusageExecRunner method), 90
- setup\_block() (temci.run.run\_driver.SpecExecRunner method), 93
- setup\_block() (temci.run.run\_driver.TimeExecRunner method), 94
- setup\_block() (temci.run.run\_driver\_plugin.AbstractRunDriverPlugin method), 96
- setup\_block() (temci.run.run\_driver\_plugin.DropFSCaches method), 100
- setup\_block() (temci.run.run\_driver\_plugin.PreheatPlugin method), 103
- setup\_block() (temci.run.run\_driver\_plugin.SleepPlugin method), 103
- setup\_block\_run() (temci.run.run\_driver\_plugin.AbstractRunDriverPlugin method), 96
- setup\_block\_run() (temci.run.run\_driver\_plugin.EnvRandomizePlugin method), 101
- setup\_block\_run() (temci.run.run\_driver\_plugin.FlushCPUCache method), 101
- setup\_block\_run() (temci.run.run\_driver\_plugin.SyncPlugin method), 104
- ShellRunDriver (class in temci.run.run\_driver), 90
- short (temci.utils.click\_helper.CmdOption attribute), 243
- show\_report (temci.run.run\_processor.RunProcessor attribute), 106
- shuffle (temci.run.run\_processor.RunProcessor attribute), 106
- SignificanceTooLowError (class in temci.report.stats), 228
- SignificanceTooLowWarning (class in temci.report.stats), 228
- Single (class in temci.report.stats), 229
- SingleProperty (class in temci.report.stats), 229
- singles (temci.report.stats.SinglesProperty attribute), 232
- singles (temci.report.stats.TestedPairsAndSingles attribute), 239
- singles\_properties (temci.report.stats.TestedPairsAndSingles attribute), 239
- SinglesProperty (class in temci.report.stats), 231
- Singleton (class in temci.utils.util), 283
- skewedness() (temci.report.stats.SingleProperty method), 231
- SleepPlugin (class in temci.run.run\_driver\_plugin), 103
- SpecExecRunner (class in temci.run.run\_driver), 92
- sphinx\_doc() (in module temci.utils.util), 284
- start\_time (temci.run.run\_processor.RunProcessor attribute), 106
- StatError (class in temci.report.stats), 232
- StatisticalPropertyFunc (in module temci.misc.game), 71
- StatMessage (class in temci.report.stats), 232
- StatMessageType (class in temci.report.stats), 234
- StatMessageValueFormat (class in temci.report.stats), 234
- StatProperty (in module temci.misc.game), 71
- stats (temci.report.report.AbstractReporter attribute), 211
- stats\_helper (temci.report.report.AbstractReporter attribute), 211
- stats\_helper (temci.run.run\_processor.RunProcessor attribute), 106
- StatWarning (class in temci.report.stats), 234
- std() (temci.report.stats.SingleProperty method), 231
- std\_dev() (temci.report.stats.SingleProperty method), 231
- std\_dev\_ci() (temci.report.stats.SingleProperty method), 231
- std\_dev\_per\_mean() (temci.report.stats.SingleProperty method), 231
- std\_devs() (temci.report.stats.SingleProperty method), 231
- std\_error\_mean() (temci.report.stats.SingleProperty method), 231
- StdDeviationToHighError (class in temci.report.stats), 234
- StdDeviationToHighWarning (class in temci.report.stats), 235
- stderr (temci.run.run\_driver.ExecRunDriver.ExecResult property), 81
- stdout (temci.run.run\_driver.ExecRunDriver.ExecResult property), 81
- stop (temci.build.builder.BuilderThread attribute), 65
- stop (temci.run.run\_worker\_pool.BenchmarkingThread

attribute), 108  
 StopStartPlugin (class in temci.run.run\_driver\_plugin), 103  
 store() (temci.run.run\_processor.RunProcessor method), 106  
 store\_and\_teardown() (temci.run.run\_processor.RunProcessor method), 106  
 store\_erroneous() (temci.run.run\_processor.RunProcessor method), 106  
 store\_example\_config() (temci.build.build\_processor.BuildProcessor class method), 64  
 store\_example\_config() (temci.run.run\_driver.AbstractRunDriver class method), 76  
 store\_figure() (temci.report.stats.BaseStatObject method), 226  
 store\_files (temci.run.run\_driver.AbstractRunDriver attribute), 76  
 store\_files (temci.run.run\_driver.ShellRunDriver attribute), 92  
 store\_html() (temci.misc.game.Language method), 69  
 store\_into\_file() (temci.utils.settings.Settings method), 267  
 store\_often (temci.run.run\_processor.RunProcessor attribute), 106  
 Str (class in temci.utils.typecheck), 279  
 string\_representation() (temci.utils.typecheck.Constraint method), 271  
 string\_representation() (temci.utils.typecheck.Dict method), 273  
 string\_representation() (temci.utils.typecheck.Type method), 281  
 StrList (class in temci.utils.typecheck), 279  
 SUB\_BENCH\_SET (in module temci.run.cpuset), 75  
 sub\_core\_number (temci.run.cpuset.CPUSet attribute), 75  
 submit() (temci.run.run\_worker\_pool.AbstractRunWorkerPool method), 107  
 submit() (temci.run.run\_worker\_pool.ParallelRunWorkerPool method), 108  
 submit() (temci.run.run\_worker\_pool.RunWorkerPool method), 109  
 submit\_queue (temci.build.builder.BuilderThread attribute), 65  
 submit\_queue (temci.run.run\_worker\_pool.AbstractRunWorkerPool attribute), 107  
 supports\_parsing\_out (temci.run.run\_driver.ExecRunner attribute), 84  
 supports\_parsing\_out (temci.run.run\_driver.PerfStatExecRunner attribute), 87  
 supports\_parsing\_out (temci.run.run\_driver.TimeExecRunner attribute), 94  
 swap() (temci.report.stats.TestedPair method), 236  
 swap() (temci.report.stats.TestedPairProperty method), 238  
 SyncPlugin (class in temci.run.run\_driver\_plugin), 104  
**T**  
 T (class in temci.utils.typecheck), 279  
 table\_html\_for\_vals\_per\_impl() (temci.misc.game.BaseObject method), 67  
 teardown() (temci.run.cpuset.CPUSet method), 75  
 teardown() (temci.run.run\_driver.AbstractRunDriver method), 76  
 teardown() (temci.run.run\_driver.ExecRunDriver method), 83  
 teardown() (temci.run.run\_driver.ShellRunDriver method), 92  
 teardown() (temci.run.run\_driver\_plugin.AbstractRunDriverPlugin method), 96  
 teardown() (temci.run.run\_driver\_plugin.CPUGovernor method), 97  
 teardown() (temci.run.run\_driver\_plugin.DisableASLR method), 97  
 teardown() (temci.run.run\_driver\_plugin.DisableCPUCaches method), 98  
 teardown() (temci.run.run\_driver\_plugin.DisableHyperThreading method), 98  
 teardown() (temci.run.run\_driver\_plugin.DisableSwap method), 99  
 teardown() (temci.run.run\_driver\_plugin.DisableTurboBoost method), 99  
 teardown() (temci.run.run\_driver\_plugin.NicePlugin method), 102  
 teardown() (temci.run.run\_driver\_plugin.OtherNicePlugin method), 102  
 teardown() (temci.run.run\_driver\_plugin.StopStartPlugin method), 104  
 teardown() (temci.run.run\_processor.RunProcessor method), 106  
 teardown() (temci.run.run\_worker\_pool.AbstractRunWorkerPool method), 107  
 teardown() (temci.run.run\_worker\_pool.ParallelRunWorkerPool method), 109  
 teardown() (temci.run.run\_worker\_pool.RunWorkerPool method), 109  
 teardown\_block() (temci.run.run\_driver\_plugin.AbstractRunDriverPlugin method), 96  
 temci module, 290  
 temci.build module, 66



---

|                                |   |
|--------------------------------|---|
| temci.build.build_processor    | temci.utils.number  |
| module, 62                     | module, 245   |
| temci.build.builder            | temci.utils.plugin  |
| module, 64                     | module, 249   |
| temci.misc                     | temci.utils.registry  |
| module, 74                     | module, 249   |
| temci.misc.game                | temci.utils.settings  |
| module, 66                     | module, 250   |
| temci.report                   | temci.utils.sudo_utils  |
| module, 241                    | module, 269   |
| temci.report.report            | temci.utils.typecheck   |
| module, 210                    | module, 269   |
| temci.report.report_processor  | temci.utils.util  |
| module, 217                    | module, 282   |
| temci.report.rundata           | temci.utils.vcs   |
| module, 217                    | module, 285   |
| temci.report.stats             | temci__build() (in module temci.scripts.cli), 110               |
| module, 225                    | temci__clean() (in module temci.scripts.cli), 112               |
| temci.report.testers           | temci__completion() (in module temci.scripts.cli), 113          |
| module, 239                    | temci__completion__bash() (in module temci.scripts.cli), 113    |
| temci.run                      | temci__completion__zsh() (in module temci.scripts.cli), 114     |
| module, 109                    | temci__exec() (in module temci.scripts.cli), 115                |
| temci.run.cpuset               | temci__format() (in module temci.scripts.cli), 135              |
| module, 74                     | temci__init() (in module temci.scripts.cli), 137                |
| temci.run.run_driver           | temci__init__build_config() (in module temci.scripts.cli), 137  |
| module, 75                     | temci__init__run_config() (in module temci.scripts.cli), 138    |
| temci.run.run_driver_plugin    | temci__init__settings() (in module temci.scripts.cli), 139      |
| module, 96                     | temci__report() (in module temci.scripts.cli), 140              |
| temci.run.run_processor        | temci__setup() (in module temci.scripts.cli), 148               |
| module, 105                    | temci__shell() (in module temci.scripts.cli), 148               |
| temci.run.run_worker_pool      | temci__short() (in module temci.scripts.cli), 167               |
| module, 106                    | temci__short__exec() (in module temci.scripts.cli), 167         |
| temci.scripts                  | temci__short__shell() (in module temci.scripts.cli), 188        |
| module, 209                    | temci__version() (in module temci.scripts.cli), 208             |
| temci.scripts.cli              | TEMCI_ERROR (temci.scripts.cli.ErrorCode attribute), 110        |
| module, 110                    | temci_in_base_set (temci.run.cpuset.CPUSet attribute), 75       |
| temci.scripts.temci_completion | test() (temci.report.testers.Tester method), 240                |
| module, 209                    | TestedPair (class in temci.report.stats), 235                   |
| temci.scripts.version          | TestedPairProperty (class in temci.report.stats), 236           |
| module, 209                    | TestedPairsAndSingles (class in temci.report.stats), 238        |
| temci.setup                    | Tester (class in temci.report.testers), 240                     |
| module, 210                    | tester (temci.report.rundata.RunDataStatsHelper attribute), 224 |
| temci.setup.setup              |   |
| module, 210                    |   |
| temci.utils                    |   |
| module, 290                    |   |
| temci.utils.click_helper       |   |
| module, 241                    |   |
| temci.utils.config_utils       |   |
| module, 245                    |   |
| temci.utils.library_init       |   |
| module, 245                    |   |
| temci.utils.mail               |   |
| module, 245                    |   |

- tester (*temci.report.stats.TestedPair* attribute), 236
- tester (*temci.report.stats.TestedPairProperty* attribute), 238
- TesterRegistry (class in *temci.report.testers*), 241
- threads (*temci.run.run\_worker\_pool.ParallelRunWorkerPool* attribute), 109
- time (*temci.run.run\_driver.ExecRunDriver.ExecResult* property), 81
- time\_file() (in module *temci.run.run\_driver*), 95
- time\_left() (*temci.run.run\_worker\_pool.AbstractRunWorkerPool* method), 107
- TimeExecRunner (class in *temci.run.run\_driver*), 93
- TimeoutException, 94
- tmp\_build\_dir (*temci.build.builder.BuilderQueueItem* property), 65
- tmp\_dir (*temci.build.builder.BuilderQueueItem* property), 65
- to\_dict() (*temci.misc.game.Input* method), 67
- to\_dict() (*temci.report.rundata.RunData* method), 220
- to\_dict() (*temci.run.run\_driver.RunProgramBlock* method), 89
- to\_long\_prop\_dict (*temci.report.report.AbstractReporter* attribute), 211
- ttest\_rel\_to\_first\_property() (in module *temci.misc.game*), 73
- ttest\_summarize() (in module *temci.misc.game*), 73
- ttest\_to\_first() (in module *temci.misc.game*), 73
- TTester (class in *temci.report.testers*), 239
- Tuple (class in *temci.utils.typecheck*), 280
- Type (class in *temci.utils.typecheck*), 280
- type (*temci.report.stats.EffectToSmallError* attribute), 226
- type (*temci.report.stats.NotEnoughObservationsError* attribute), 227
- type (*temci.report.stats.SignificanceTooLowError* attribute), 228
- type (*temci.report.stats.StatError* attribute), 232
- type (*temci.report.stats.StatMessage* attribute), 233
- type (*temci.report.stats.StatWarning* attribute), 234
- type (*temci.report.stats.StdDeviationToHighError* attribute), 235
- type\_scheme (*temci.run.run\_driver.RunProgramBlock* attribute), 89
- type\_scheme (*temci.utils.click\_helper.CmdOption* attribute), 243
- type\_scheme (*temci.utils.settings.Settings* attribute), 267
- type\_scheme\_option() (in module *temci.utils.click\_helper*), 244
- typecheck() (in module *temci.utils.typecheck*), 281
- typecheck\_default (*temci.utils.typecheck.Type* attribute), 281
- typecheck\_locals() (in module *temci.utils.typecheck*), 282
- types (*temci.utils.typecheck.All* attribute), 270
- types (*temci.utils.typecheck.Either* attribute), 274
- ## U
- uncertainty\_range (*temci.report.testers.Tester* attribute), 241
- unknown\_keys (*temci.utils.typecheck.Dict* attribute), 273
- update\_env\_info() (*temci.report.rundata.RunDataStatsHelper* method), 224
- use\_key (*temci.report.report.ReporterRegistry* attribute), 216
- use\_key (*temci.report.testers.TesterRegistry* attribute), 241
- use\_key (*temci.run.run\_driver.AbstractRunDriver* attribute), 76
- use\_key (*temci.run.run\_driver.ExecRunDriver* attribute), 83
- use\_key (*temci.run.run\_driver.RunDriverRegistry* attribute), 88
- use\_key (*temci.utils.registry.AbstractRegistry* attribute), 250
- use\_list (*temci.report.report.ReporterRegistry* attribute), 216
- use\_list (*temci.report.testers.TesterRegistry* attribute), 241
- use\_list (*temci.run.run\_driver.AbstractRunDriver* attribute), 76
- use\_list (*temci.run.run\_driver.ExecRunDriver* attribute), 83
- use\_list (*temci.run.run\_driver.RunDriverRegistry* attribute), 88
- use\_list (*temci.utils.registry.AbstractRegistry* attribute), 250
- used\_plugins (*temci.run.run\_driver.AbstractRunDriver* attribute), 76
- used\_rel\_mean\_property() (in module *temci.misc.game*), 74
- used\_std\_property() (in module *temci.misc.game*), 74
- used\_summarize\_mean() (in module *temci.misc.game*), 74
- used\_summarize\_mean\_std() (in module *temci.misc.game*), 74
- ## V
- valid\_runs() (*temci.report.rundata.RunDataStatsHelper* method), 224
- validate() (in module *temci.utils.click\_helper*), 244
- validate() (*temci.run.run\_driver.ExecValidator* method), 85
- validate() (*temci.utils.settings.Settings* method), 269
- validate\_key\_path() (*temci.utils.settings.Settings* method), 269
- validate\_revision() (*temci.utils.vcs.FileDriver* method), 286

[validate\\_revision\(\)](#) (*temci.utils.vcs.GitDriver method*), 287  
[validate\\_revision\(\)](#) (*temci.utils.vcs.VCSDriver method*), 290  
[ValidCPUCoreNumber\(\)](#) (*in module temci.utils.settings*), 269  
[ValidPerfStatPropertyList](#) (*class in temci.run.run\_driver*), 94  
[ValidPropertyList](#) (*class in temci.run.run\_driver*), 94  
[ValidRusagePropertyList](#) (*class in temci.run.run\_driver*), 95  
[ValidTimePropertyList](#) (*class in temci.run.run\_driver*), 95  
[ValidTimeSpan](#) (*class in temci.utils.typecheck*), 281  
[ValidYamlFileName](#) (*class in temci.utils.typecheck*), 281  
[value](#) (*temci.utils.typecheck.Info attribute*), 276  
[value\\_format](#) (*temci.report.stats.EffectToSmallWarning attribute*), 227  
[value\\_format](#) (*temci.report.stats.NotEnoughObservationsWarning attribute*), 228  
[value\\_format](#) (*temci.report.stats.StatMessage attribute*), 233  
[value\\_format](#) (*temci.report.stats.StdDeviationToHighWarning attribute*), 235  
[value\\_name](#) (*temci.utils.typecheck.Info attribute*), 276  
[value\\_type](#) (*temci.utils.typecheck.Dict attribute*), 273  
[values](#) (*temci.report.stats.StatMessage attribute*), 234  
[values\(\)](#) (*temci.utils.util.InsertionTimeOrderedDict method*), 283  
[variance\(\)](#) (*temci.report.stats.SingleProperty method*), 231  
[vcs\\_driver](#) (*temci.build.builder.Builder attribute*), 64  
[VCSDriver](#) (*class in temci.utils.vcs*), 288  
[VCSError](#), 290  
[VelcomReporter](#) (*class in temci.report.report*), 216  
[verbose\\_isinstance\(\)](#) (*in module temci.utils.typecheck*), 282  
[version](#) (*in module temci.scripts.version*), 209

## W

[warn\\_for\\_pdflatex\\_non\\_existence\\_once\(\)](#) (*in module temci.utils.util*), 284  
[WARNING](#) (*temci.report.stats.StatMessageType attribute*), 234  
[warnings\(\)](#) (*temci.report.stats.BaseStatObject method*), 226  
[whiskers\(\)](#) (*temci.report.stats.SingleProperty method*), 231  
[wrap\(\)](#) (*temci.utils.typecheck.Info method*), 276

## Y

[YAML\\_FILE\\_COMPLETION\\_HINT](#) (*in module temci.utils.typecheck*), 281