
timeutils Documentation

Release 0.2.3

Michal Ciesielczyk

Sep 26, 2017

Contents:

1	timeutils package	1
1.1	Submodules	1
1.1.1	timeutils.stopwatch module	1
1.1.2	timeutils.timespan module	3
1.2	Module contents	4
1.2.1	Examples	4
2	Indices and tables	5
	Python Module Index	7

Submodules

timeutils.stopwatch module

Stopwatch

class `timeutils.stopwatch.Stopwatch` (*start=False, verbose=False, label=None, logger=None, logger_level=None*)

Provides a set of methods and properties that you can use to accurately measure elapsed time.

Parameters

- **start** (*bool, optional*) – if set to *True*, immediately starts measuring the time (by calling *start()*). By default set to *False*.
- **verbose** (*bool, optional*) – if set to *True*, logs the elapsed time when the *stop()* method is called. By default set to *False*.
- **label** (*str, optional*) – Optional stopwatch label to be included in the log messages (only if *verbose* is *True*).
- **logger** (`logging.Logger`, optional) – logger object for logging stopwatch messages if *verbose* is *True*. If set to *None* (the default), the logger is set to `sys.stdout`.
- **logger_level** (*int, optional*) – logging level as defined in the build-in `logging` package (only if the *logger* object is set).

Examples

Simple time measurement:

```
sw = Stopwatch(start=True)
# code to be measured
sw.stop()
```

Getting the elapsed time:

```
print(sw.elapsed) # hh:mm:ss.ms
print(sw.elapsed.human_str()) # human-readable time
```

Restarting the stopwatch instance:

```
sw.restart()
```

Pausing and resuming the stopwatch:

```
sw.suspend()
# code block not included in the measurement
sw.resume()
```

Using a logger:

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
logger.addHandler(logging.FileHandler(filename='example.log'))

sw = Stopwatch(verbose=True, label='Example', logger=logger)
sw.start()
# code to be measured
sw.stop()
```

Note: *Stopwatch* methods are protected against inappropriate calls. It is an error to start or stop a *Stopwatch* object that is already in the desired state.

See also:

Documentation of the *TimeSpan* class.

elapsed

The total elapsed time measured by the current instance.

elapsed_seconds

The total elapsed time in fractions of a second measured by the current instance.

is_running

Indicates whether the *Stopwatch* instance is running.

is_suspended

Indicates whether the *Stopwatch* instance is suspended.

reset()

Stops time interval measurement and resets the *Stopwatch* instance. The time elapsed before reset is set to zero.

restart()

Stops time interval measurement, resets the *Stopwatch* instance, and starts measuring elapsed time. The time elapsed before restart is set to zero.

resume()

Resumes measuring elapsed time after calling *suspend()*.

start()

Starts measuring elapsed time for an interval.

start_time

The time at which the time measurement has been started.

stop()

Stops the time measurement. Returns the total elapsed time measured by the current instance.

stop_time

The time at which the time measurement has been stopped.

suspend()

Pauses the time measurement until the *Stopwatch* instance is resumed or stopped. Returns the total elapsed time measured by the current instance.

Call *resume()* to resume measuring elapsed time.

timeutils.timespan module

TimeSpan

class timeutils.timespan.**TimeSpan**

Bases: *datetime.timedelta*

Represents a time span.

All arguments are optional and default to 0. Arguments may be integers or floats, and may be positive or negative.

As in the base *datetime.timedelta* class, only days, seconds and microseconds are stored internally. Arguments are converted to those units:

- A millisecond is converted to 1000 microseconds.
- A minute is converted to 60 seconds.
- An hour is converted to 3600 seconds.

and days, seconds and microseconds are then normalized so that the representation is unique.

If any argument is a float and there are fractional microseconds, the fractional microseconds left over from all arguments are combined and their sum is rounded to the nearest microsecond using round-half-to-even tiebreaker.

Parameters

- **seconds** (*float, optional*) – number of seconds in the time span.
- **microseconds** (*float, optional*) – number of microseconds in the time span.
- **milliseconds** (*float, optional*) – number of milliseconds in the time span.
- **minutes** (*float, optional*) – number of minutes in the time span.
- **hours** (*float, optional*) – number of hours in the time span.
- **days** (*float, optional*) – number of days in the time span.

__str__

days

Number of days.

human_str (*trim_zeros=True*)

Returns a human-readable *TimeSpan* object, represented as time units such as days, hours, minutes, and seconds.

Parameters *trim_zeros* (*bool, optional*) – indicates whether the leading zeros in the result should be skipped

Returns human-readable time span

Return type *str*

microseconds

Number of microseconds (≥ 0 and less than 1 second).

seconds

Number of seconds (≥ 0 and less than 1 day).

total_hours ()

Total hours in the duration.

total_milliseconds ()

Total milliseconds in the duration.

total_minutes ()

Total minutes in the duration.

total_seconds ()

Total seconds in the duration.

Module contents

A set of methods and classes to accurately measure elapsed time.

See <https://gitlab.com/cmick/timeutils> for more information.

Examples

```
>>> from timeutils import Stopwatch
>>> sw = Stopwatch(start=True)
>>> sw.elapsed_seconds
16.282313108444214
>>> str(sw.stop())
'00:01:30.416'
>>> sw.elapsed.human_str()
'1 min, 30 secs'
```

See also:

Documentation of the *Stopwatch* class.

timeutils.current_time_millis ()

Returns the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

t

- `timeutils`, 4
- `timeutils.stopwatch`, 1
- `timeutils.timespan`, 3

Symbols

`__str__` (timeutils.timespan.TimeSpan attribute), 3

C

`current_time_millis()` (in module timeutils), 4

D

`days` (timeutils.timespan.TimeSpan attribute), 3

E

`elapsed` (timeutils.stopwatch.Stopwatch attribute), 2

`elapsed_seconds` (timeutils.stopwatch.Stopwatch attribute), 2

H

`human_str()` (timeutils.timespan.TimeSpan method), 3

I

`is_running` (timeutils.stopwatch.Stopwatch attribute), 2

`is_suspended` (timeutils.stopwatch.Stopwatch attribute), 2

M

`microseconds` (timeutils.timespan.TimeSpan attribute), 4

R

`reset()` (timeutils.stopwatch.Stopwatch method), 2

`restart()` (timeutils.stopwatch.Stopwatch method), 2

`resume()` (timeutils.stopwatch.Stopwatch method), 2

S

`seconds` (timeutils.timespan.TimeSpan attribute), 4

`start()` (timeutils.stopwatch.Stopwatch method), 2

`start_time` (timeutils.stopwatch.Stopwatch attribute), 3

`stop()` (timeutils.stopwatch.Stopwatch method), 3

`stop_time` (timeutils.stopwatch.Stopwatch attribute), 3

`Stopwatch` (class in timeutils.stopwatch), 1

`suspend()` (timeutils.stopwatch.Stopwatch method), 3

T

`TimeSpan` (class in timeutils.timespan), 3

`timeutils` (module), 4

`timeutils.stopwatch` (module), 1

`timeutils.timespan` (module), 3

`total_hours()` (timeutils.timespan.TimeSpan method), 4

`total_milliseconds()` (timeutils.timespan.TimeSpan method), 4

`total_minutes()` (timeutils.timespan.TimeSpan method), 4

`total_seconds()` (timeutils.timespan.TimeSpan method), 4