

---

# **timeutils Documentation**

***Release 0.2.2***

**Michal Ciesielczyk**

**Sep 20, 2017**



---

## Contents:

---

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



## 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*

Stores a time span.

**Parameters** **seconds** (*float*) – precise number of seconds in the time span.

**human\_str** (*trim\_zeros=True*)

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

**total\_hours**()

**total\_milliseconds**()

**total\_minutes**()

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



## C

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

## 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

## R

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

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

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

## S

`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), 3

`timeutils.stopwatch` (module), 1

`timeutils.timespan` (module), 3

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

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

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