# sched — Event scheduler

**Source code:** Lib/sched.py

---

The sched module defines a class which implements a general purpose event scheduler:

*class* sched.**scheduler**(*timefunc=time.monotonic, delayfunc=time.sleep*)

> The scheduler class defines a generic interface to scheduling events. It needs two functions to actually deal with the "outside world" — *timefunc* should be callable without arguments, and return a number (the "time", in any units whatsoever). The *delayfunc* function should be callable with one argument, compatible with the output of *timefunc*, and should delay that many time units. *delayfunc* will also be called with the argument `0` after each event is run to allow other threads an opportunity to run in multi-threaded applications.
>
> *Changed in version 3.3: timefunc and delayfunc parameters are optional.*
>
> *Changed in version 3.3:* scheduler class can be safely used in multi-threaded environments.

Example:

```
>>> import sched, time
>>> s = sched.scheduler(time.time, time.sleep)
>>> def print_time(a='default'):
...     print("From print_time", time.time(), a)
...
>>> def print_some_times():
...     print(time.time())
...     s.enter(10, 1, print_time)
...     s.enter(5, 2, print_time, argument=('positional',))
...     s.enter(5, 1, print_time, kwargs={'a': 'keyword'})
...     s.run()
...     print(time.time())
...
>>> print_some_times()
930343690.257
From print_time 930343695.274 positional
From print_time 930343695.275 keyword
From print_time 930343700.273 default
930343700.276
```

## Scheduler Objects

scheduler instances have the following methods and attributes:

scheduler.**enterabs**(*time, priority, action, argument=(), kwargs={}*)

Schedule a new event. The *time* argument should be a numeric type compatible with the return value of the *timefunc* function passed to the constructor. Events scheduled for the same *time* will be executed in the order of their *priority*. A lower number represents a higher priority.

Executing the event means executing `action(*argument, **kwargs)`. *argument* is a sequence holding the positional arguments for *action*. *kwargs* is a dictionary holding the keyword arguments for *action*.

Return value is an event which may be used for later cancellation of the event (see `cancel()`).

*Changed in version 3.3: argument* parameter is optional.

*Changed in version 3.3: kwargs* parameter was added.

scheduler.**enter**(*delay*, *priority*, *action*, *argument=()*, *kwargs={}*)

Schedule an event for *delay* more time units. Other than the relative time, the other arguments, the effect and the return value are the same as those for `enterabs()`.

*Changed in version 3.3: argument* parameter is optional.

*Changed in version 3.3: kwargs* parameter was added.

scheduler.**cancel**(*event*)

Remove the event from the queue. If *event* is not an event currently in the queue, this method will raise a `ValueError`.

scheduler.**empty**()

Return `True` if the event queue is empty.

scheduler.**run**(*blocking=True*)

Run all scheduled events. This method will wait (using the `delayfunc()` function passed to the constructor) for the next event, then execute it and so on until there are no more scheduled events.

If *blocking* is false executes the scheduled events due to expire soonest (if any) and then return the deadline of the next scheduled call in the scheduler (if any).

Either *action* or *delayfunc* can raise an exception. In either case, the scheduler will maintain a consistent state and propagate the exception. If an exception is raised by *action*, the event will not be attempted in future calls to `run()`.

If a sequence of events takes longer to run than the time available before the next event, the scheduler will simply fall behind. No events will be dropped; the calling code is responsible for canceling events which are no longer pertinent.

*Changed in version 3.3:* *blocking* parameter was added.

scheduler.**queue**

Read-only attribute returning a list of upcoming events in the order they will be run. Each event is shown as a named tuple with the following fields: time, priority, action, argument, kwargs.