

Discover asyncio event loop



Pycon 2014, Lyon



Victor Stinner
victor.stinner@gmail.com

Victor Stinner



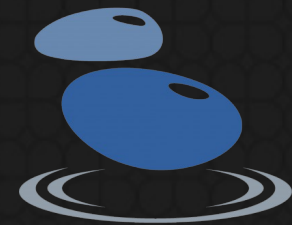
- Python core developer since 2010
- github.com/haypo/
- bitbucket.org/haypo/
- Working for **eNovance**

Disclaimer



- How asyncio is implemented
- Code snippets close to, but different, than asyncio
- No error handling
- No optimization
- (asyncio handles exceptions and is optimized)

Agenda



1. Callbacks
2. Timers
3. Generator and yield from
4. Coroutine
5. Task

Callbacks

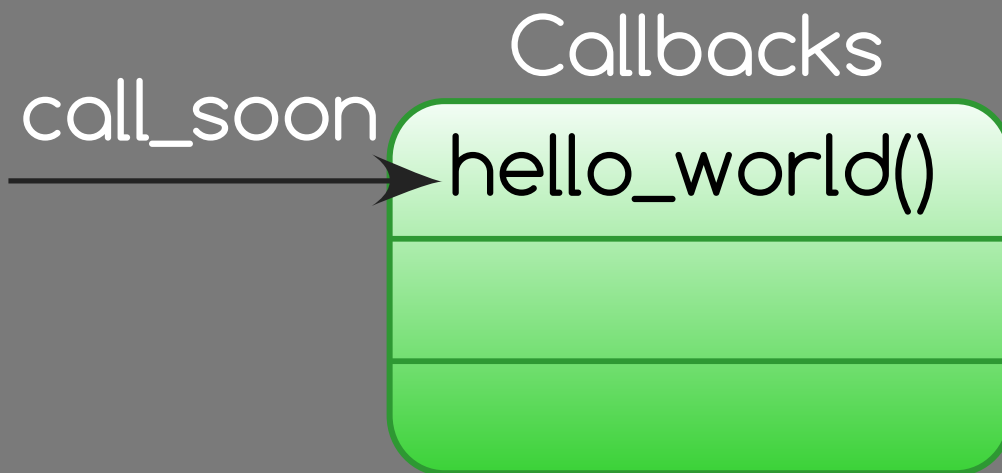
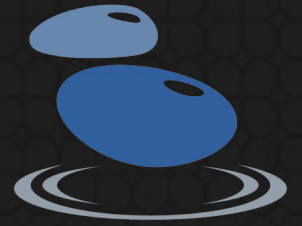


```
class CallbackEventLoop:
    def __init__(self):
        self.callbacks = []

    def call_soon(self, func):
        self.callbacks.append(func)

    def execute_callbacks(self):
        callbacks = self.callbacks
        self.callbacks = []
        for cb in callbacks:
            cb()
```

Callbacks

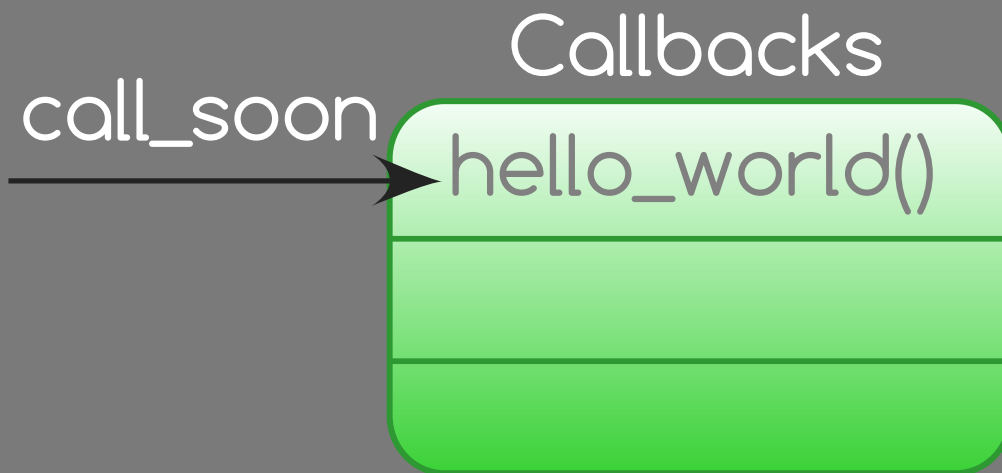


Code

```
.call_soon(hello_world)
```

Output

Callbacks



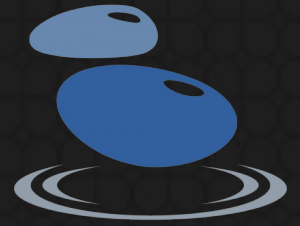
Code

```
.call_soon(hello_world)  
.execute_callbacks()
```

Output

Hello World!

Callbacks



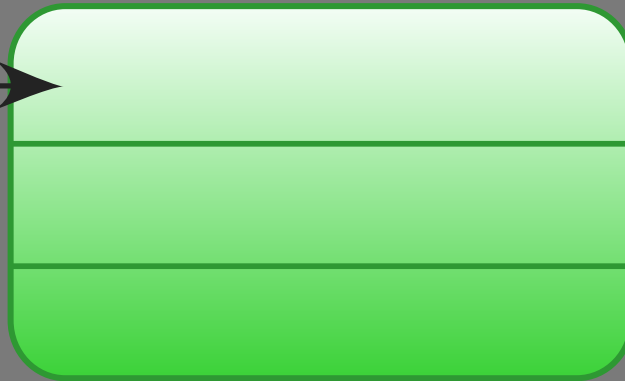
Code

```
.call_soon(hello_world)  
.execute_callbacks()
```

Output

Hello World!

call_soon → Callbacks



Timers



```
class TimerEventLoop(CallbackEventLoop):
```

```
    def __init__(self):  
        super().__init__()  
        self.timers = []
```

```
    def call_at(self, when, func):  
        self.timers.append((when, func))
```

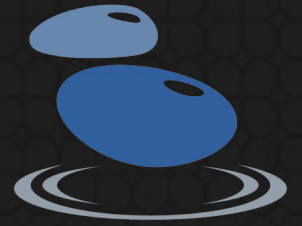
```
    ...
```

Timers



```
class TimerEventLoop(CallbackEventLoop):  
    ...  
    def execute_timers(self):  
        now = time.time()  
        new_timers = []  
        for when, func in self.timers:  
            if when <= now:  
                self.call_soon(func)  
            else:  
                new_timers.append((when, func))  
        self.timers = new_timers  
  
        self.execute_callback()
```

Timers



Timers

Code

call_at

(1, hello_world)

```
.call_at(1, hello_world)
```

Callbacks

call_soon

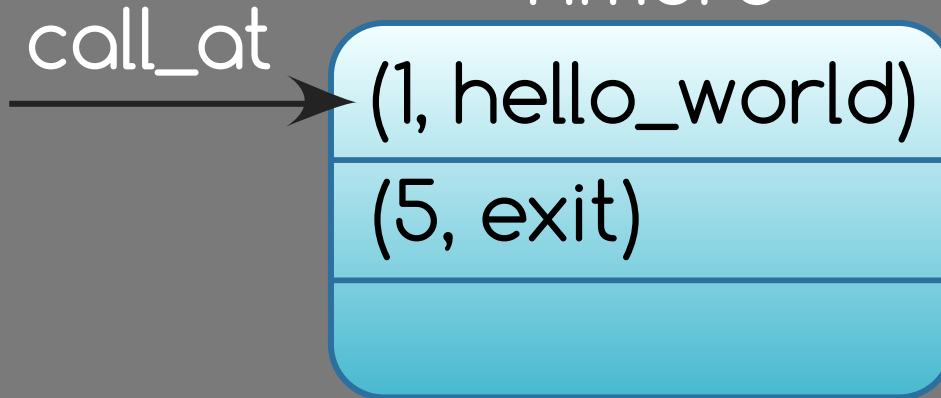
Output

Timers



Timers

call_at

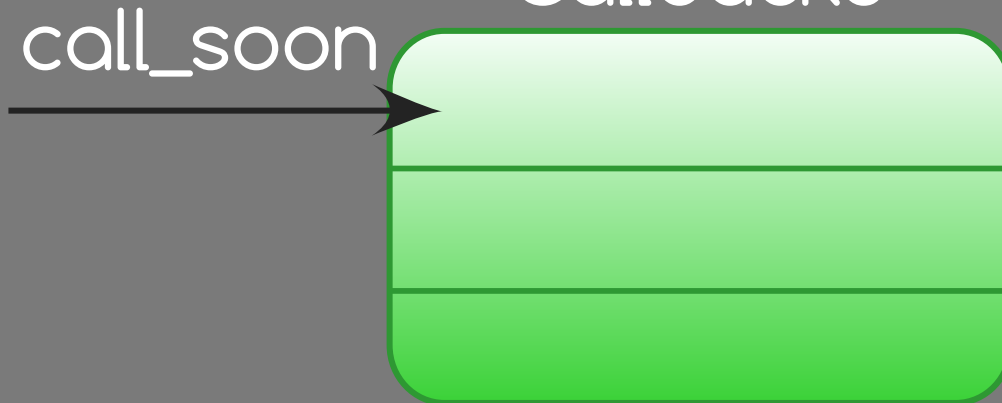


Code

```
.call_at(1, hello_world)  
.call_at(5, exit)
```

Callbacks

call_soon



Output



Timers



call_at

Timers

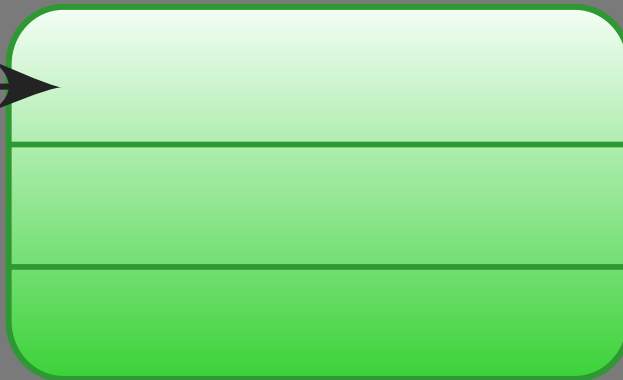
(1, hello_world)
(5, exit)
(2, good_bye)

Code

```
.call_at(1, hello_world)  
.call_at(5, exit)  
.call_at(2, good_bye)
```

call_soon

Callbacks



Output



Timers



call_at

Timers

(1, hello_world)
(5, exit)
(2, good_bye)

Code

```
.call_at(1, hello_world)  
.call_at(5, exit)  
.call_at(2, good_bye)  
.execute_timers()
```

call_soon

Callbacks

hello_world()
good_bye()

Output

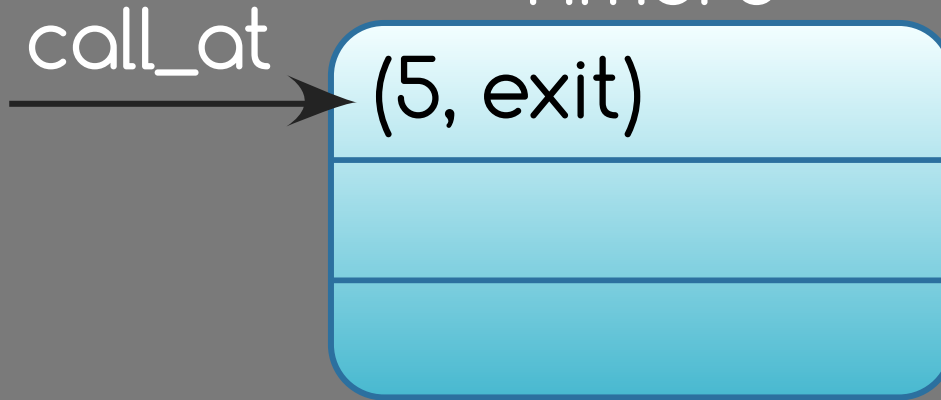


Timers



Timers

call_at



Callbacks

call_soon



Code

```
.call_at(1, hello_world)
.call_at(5, exit)
.call_at(2, good_bye)
.execute_timers()
```

Output

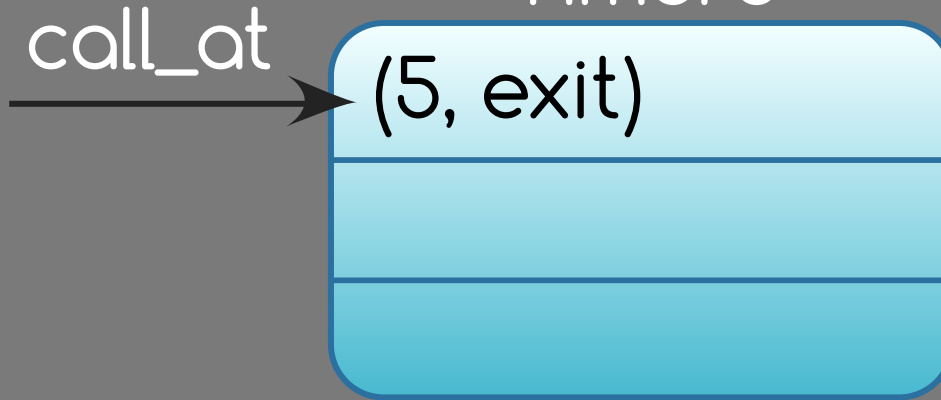
```
Hello World!
Good bye.
```

Timers



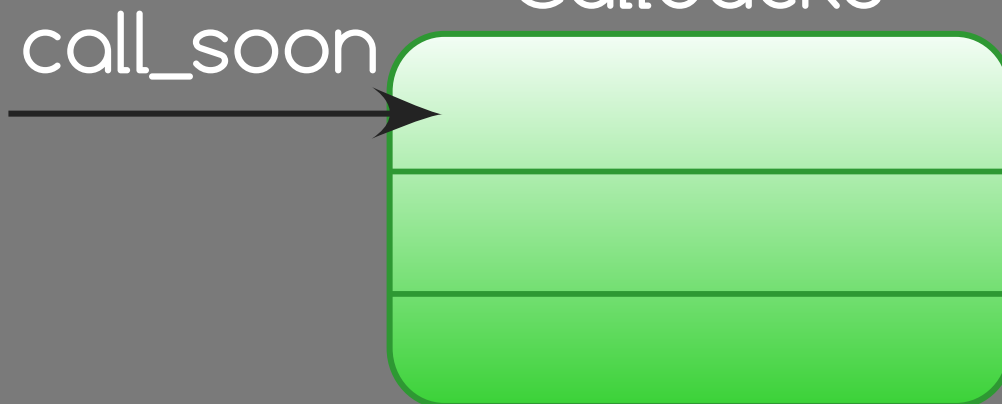
Timers

call_at



Callbacks

call_soon



Code

```
.call_at(1, hello_world)
.call_at(5, exit)
.call_at(2, good_bye)
.execute_timers()
```

Output

```
Hello World!
Good bye.
```


Thanks David Malcom
for the LibreOffice model

<http://dmalcolm.livejournal.com/>