taiste /

Going ~~concurrent~~ asynchronous with Python

Jyrki Pulliainen / 18.10.2011

/ Motivation

- Not threaded
- Good for I/O bound software
- Buzzword-compatible!

/ Two ways to achieve

- Coroutines
- Event loop

/ Coroutines

- Components, that are suspendable / resumable
- One scheduler to rule them all

/ Coroutines in Python

- Python 2.5 added support for coroutines

```
>>>def coroutine():
>>>     input = yield i
>>>     print i + 1

>>> a = coroutine()
>>> a.next()
>>> a.send(1)
2
```

/ Running multiple coroutines

```
def main_loop(self):
    while self.tasks_left():
        task = self.get_next_task()
        try:
            task.send(None)
        except StopIteration:
            # Do not schedule if the task is done
            self.remove_task(task)
```

taiste / Event loop

- Good for I/O
- Main idea: Wait for something to happen, then pass to a handler

/ Idea in a nutshell

```
class EventLoop(object):
    # ...
    def main_loop():
        while True:
            event = self.wait_for_event()
            handler = self.get_handler_for_event(handler)
            handler()
```

/ Finding events

- Usually you want to poll I/O
- Tools for that: select, kselect, epoll...

taiste /

CODE TIEM!!11

(actually, just looking at it)

Jyrki Pulliainen / 18.10.2011