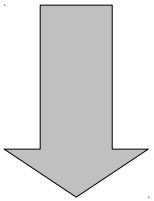
Мультипоточность

Основы

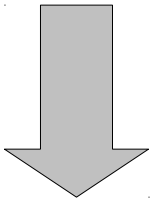
Андрей Светлов

Процесс



- Файлы
- Память
- Соединения к базе данных

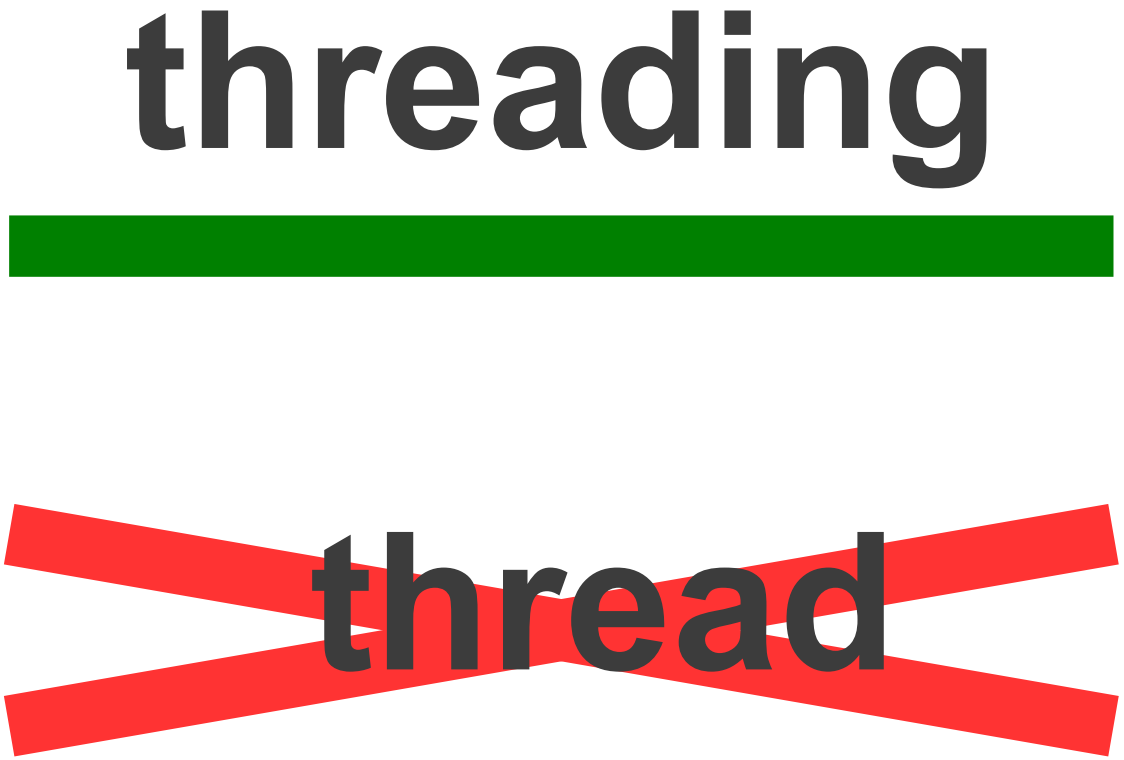
Поток



Стек

Потоки

Создание
Завершение
Ожидание


threading
thread

Создание потока

```
def f(a, b):
    pass

th.threading.Thread(target=f, args=(1, 2))
th.start()
```

Не нужно наследоваться от threading.Thread

Завершение

Не существует правильного способа завершить поток снаружи

Ожидание

просигнализировать потоку
просьбу о завершении

th.join()

Потоки-демоны

th.setDaemon(True)

Не стоит их использовать, потому что довольно опасны

Объекты синхронизации

- Rlock
- Condition Variable

- Semaphore
- Lock
- Event

Блокировки

```
def __init__(self):
    self.lock = threading.RLock()
    self.val = 0

def get(self):
    with self.lock:
        return self.val

def set(self, val):
    with self.lock:
        self.val = val
```

Область блокирования

Условные переменные

```
class Queue(object):
    SIZE = 5

    def __init__(self):
        self.q = []
        self.empty = threading.Condition()
        self.full
```

Условные переменные 2

```
def put(self, val):
    with self.full:
        while len(self.q) > self.MAX:
            self.full.wait()
        self.q.append(val)
        self.empty.notify()
```

Условные переменные 3

```
def get(self):
    with self.empty:
        while len(self.q) == 0:
            self.empty.wait()
        val = self.q.pop(0)
        self.full.notify()
        return val
```

Вопросы