

Углубленный Python

Лекция 7

Асинхронное программирование

Кандауров Геннадий



образование

Напоминание отметиться на портале

+ ОСТАВИТЬ ОТЗЫВ ПОСЛЕ ЛЕКЦИИ

mail

БлогиЛюдиПрограммаВакансииРасписание

python

сб, 16 октября	вс, 17 октября	пн, 18 октября	вт, 19 октября	ср, 20 октября	чт, 21 октября
Занятий нет	Занятий нет	18:00 Back-end разработка ...	Занятий нет	Занятий нет	Занятий нет

Backend разработка на Python

Привет!
Это блог курса Backend разработка на Python.
Все занятия проходят в зуме согласно расписанию, по ссылке:
<https://mailru.zoom.us/j/96845327537?pwd=SkFxQ0FmVXowQnR4dlh2eWM3ZmZmRdz09>

Записи:
0 Вебинар. Организационное собрание. - [ссылка](#) (нужно смотреть/скачать через облако mail)

82 читателя, 3 топика

ПодписатьсяСоздать топик

Поиск по авторам, заголовку и тексту топика...Найти

Материалы к первой лекции

Backend разработка на PythonСмешанное занятие 1

Прямой эфир

МоиВсе

Сергей Шаленко 2 дня назад
[Лекция 1. Знакомство. Введение в Linux. Работа с файлами. Просмотр ресурсов сервера.](#) 1

Сергей Шаленко 3 дня назад
[Linux + Лекция 1. Знакомство. Введение в Linux. Работа с файлами. Просмотр ресурсов сервера.](#) 1

Сергей Шаленко 3 дня назад
[Linux + Добро пожаловать на борту!](#) 0

Артур Сардарян 3 дня назад
[Разработка приложений на iOS | Осень 2021 + Рубежный контроль](#) 1 0

Константин Ермаков 3 дня назад
[Автоматизированное тестирование | Осень 2021 + Итоги 4 лекции \(семинар\)](#) 0

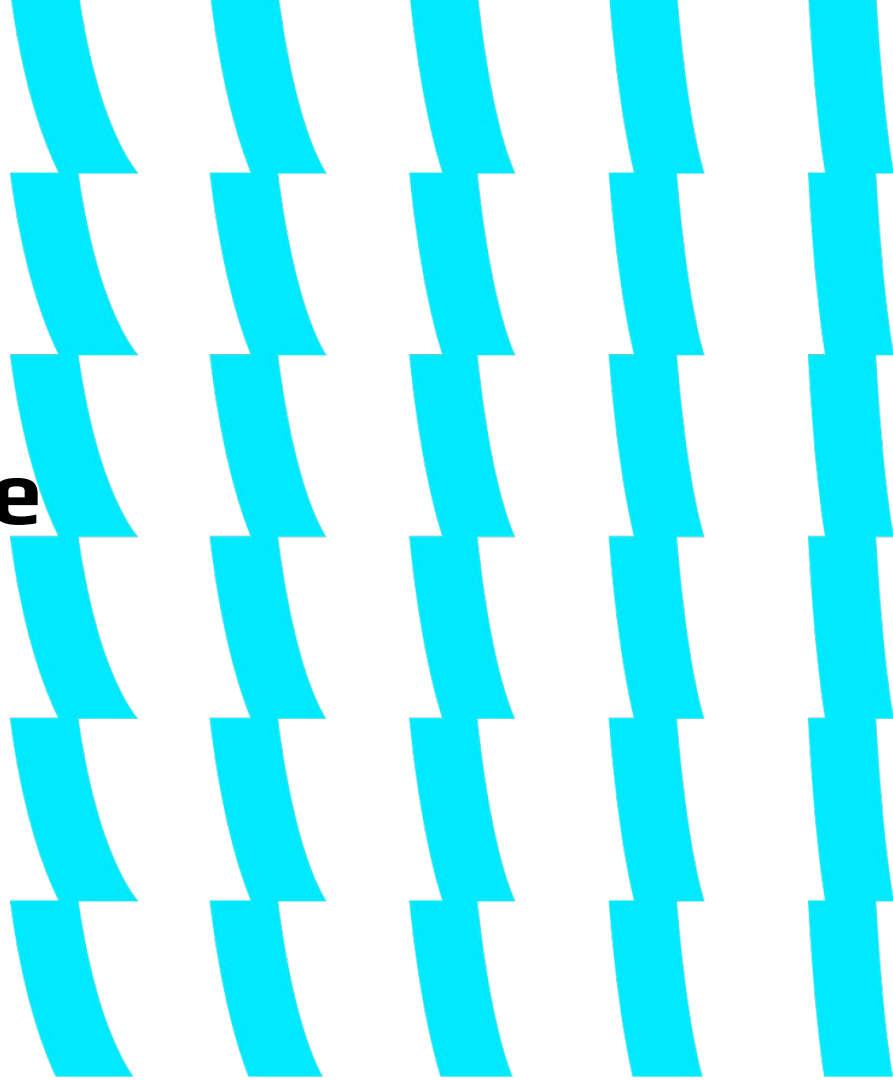
Квиз про прошлой лекции



Содержание занятия

1. Цикл событий
2. Корутины, нативные корутины
3. `asyncio`

Асинхронное программирование



Блокирующие операции

```
import socket
server_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_sock.bind(("localhost", 15000))
server_sock.listen()
```

```
while True:
    client_sock, addr = server_sock.accept()
    while True:
        data = client_sock.recv(4096)
        if not data:
            break
        else:
            client_sock.send(data.decode().upper().encode())
    client_sock.close()
```

Блокирующие операции

- connect, accept, recv, send - блокирующие операции
- C10k problem, <http://kegel.com/c10k.html>
- Поток дорого стоит (CPU & RAM)
- Поток простаивает часть времени

Неблокирующие операции

Системные вызовы:

- `select` (man 2 `select`)
- `poll` (man 2 `poll`)
- `epoll` (man 7 `epoll`)
- `kqueue`

python:

- `select`
- `selectors`

select

```
from select import select
```

```
def event_loop():
```

```
    while True:
```

```
        ready_to_read, _, _ = select(to_monitor, [], [])
```

```
        for sock in ready_to_read:
```

```
            if sock is server_sock:
```

```
                accept_conn(sock)
```

```
            else:
```

```
                respond(sock)
```

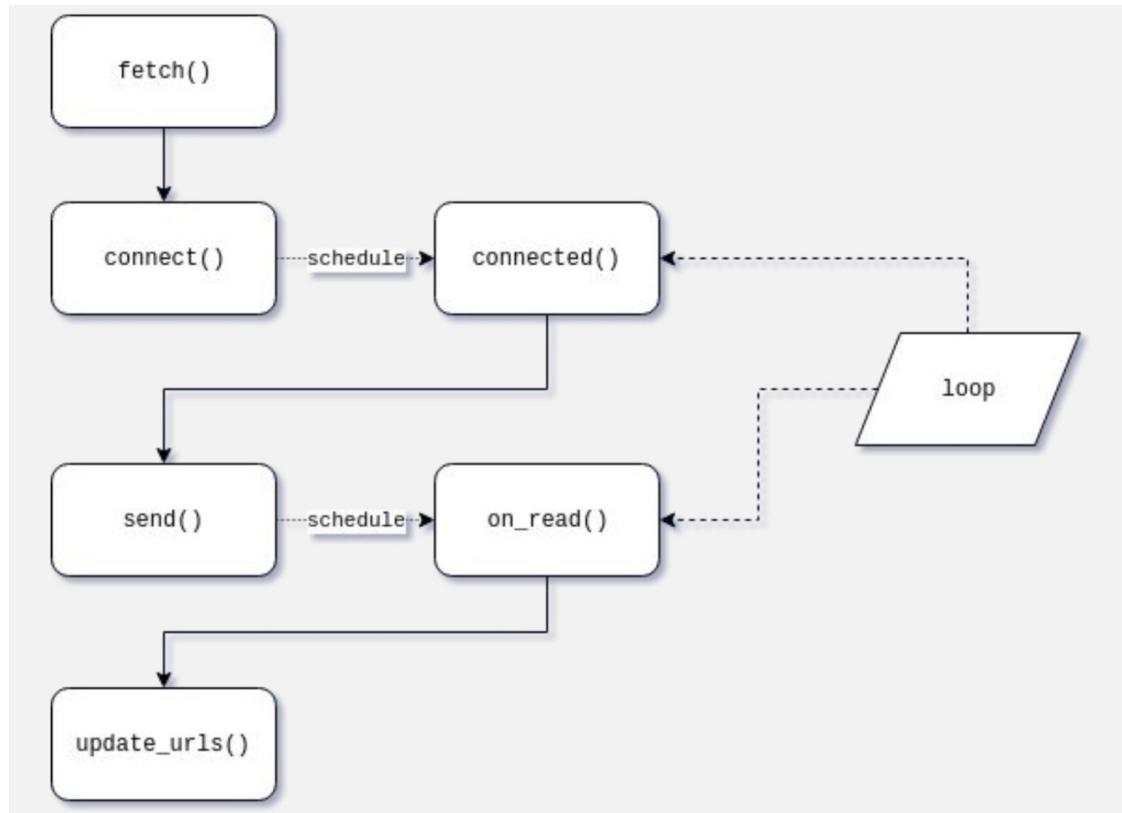
selectors

```
import selectors
```

```
selector = selectors.DefaultSelector()  
selector.register(server_sock, selectors.EVENT_READ, accept_conn)
```

```
def event_loop():  
    while True:  
        events = selector.select() # (key, events_mask)  
  
        for key, _ in events:  
            # key: NamedTuple(fileobj, events, data)  
            callback = key.data  
            callback(key.fileobj)  
            # selector.unregister(key.fileobj)
```

Callback hell



Generator based event loop

Дэвид Бизли (David Beazley), "Python Concurrency From the Ground Up: LIVE!"

```
def event_loop():
    while any([tasks, to_read, to_write]):
        while not tasks:
            ready_to_read, ready_to_write, _ = select(to_read, to_write, [])
            for sock in ready_to_read:
                tasks.append(to_read.pop(sock))
            for sock in ready_to_write:
                tasks.append(to_write.pop(sock))
        try:
            task = tasks.pop(0)
            op_type, sock = next(task)
            if op_type == "read":
                to_read[sock] = task
            elif op_type == "write":
                to_write[sock] = task
        except StopIteration:
            pass
```

Корутины

```
def grep(pattern):  
    print("start grep for", pattern)  
    while True:  
        s = yield  
        if pattern in s:  
            print("found!", s)  
        else:  
            print("no %s in %s" % (pattern, s))
```

```
g = grep("python")  
next(g)  
g.send("data")  
g.send("deep python")
```

```
$ python grep_python.py  
start grep for python  
no python in data  
found! deep python
```

Корутины

- использование **yield** более обобщенно определяет корутину
- не только генерируют значения
- потребляют данные, отправленные через **.send**
- отправленные данные возвращаются через **data = yield**

Нативные корутины

coroutine

Coroutines are a more generalized form of subroutines. Subroutines are entered at one point and exited at another point. Coroutines can be entered, exited, and resumed at many different points.

They can be implemented with the `async def` statement.

See also **PEP 492**.

Нативные корутины

```
async def say_after(delay, what):  
    await asyncio.sleep(delay)  
    print(what)
```

```
async def main():  
    print(f"started at {time.strftime('%X')}")  
    await say_after(1, 'hello')  
    await say_after(2, 'world')  
    print(f"finished at {time.strftime('%X')}")
```

```
asyncio.run(main())
```

```
>run.py
```

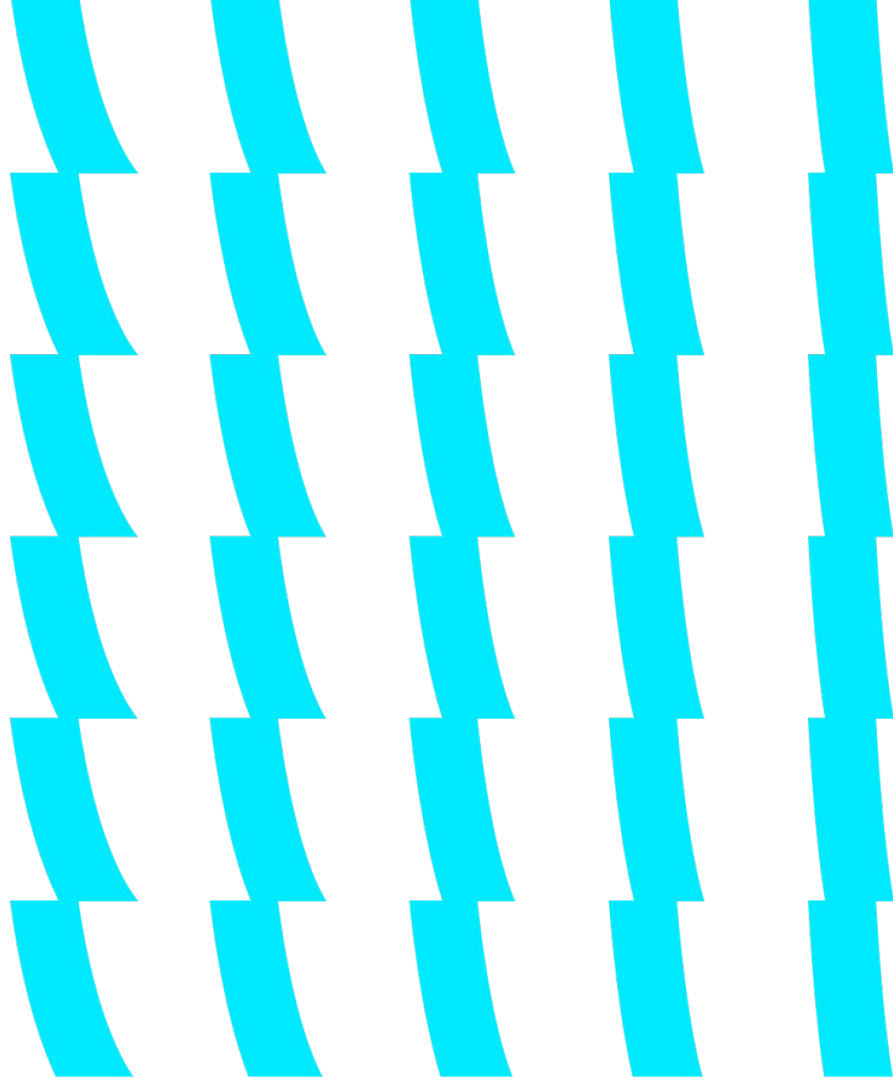
```
started at 16:42:46
```

```
hello
```

```
world
```

```
finished at 16:42:49
```


asincio



asyncio

- 1 процесс
- 1 поток
- кооперативная многозадачность (vs вытесняющая)
- передача управления в event loop на ожидающих операциях
- `async/await` это API Python, а не часть `asyncio`

asyncio

Event loop:

coroutine > Task (Future)

- **Future** представляет ожидаемый в будущем (eventual) результат асинхронной операции;
- **Task** это **Future-like** объект, запускающий корутины в событийном цикле;
- **Task** используется для запуска нескольких корутин в событийном цикле параллельно.

asyncio

High-level APIs

- Coroutines and Tasks
- Streams
- Synchronization Primitives
- Subprocesses
- Queues
- Exceptions

asyncio

Low-level APIs

- Event Loop
- Futures
- Transports and Protocols
- Policies
- Platform Support

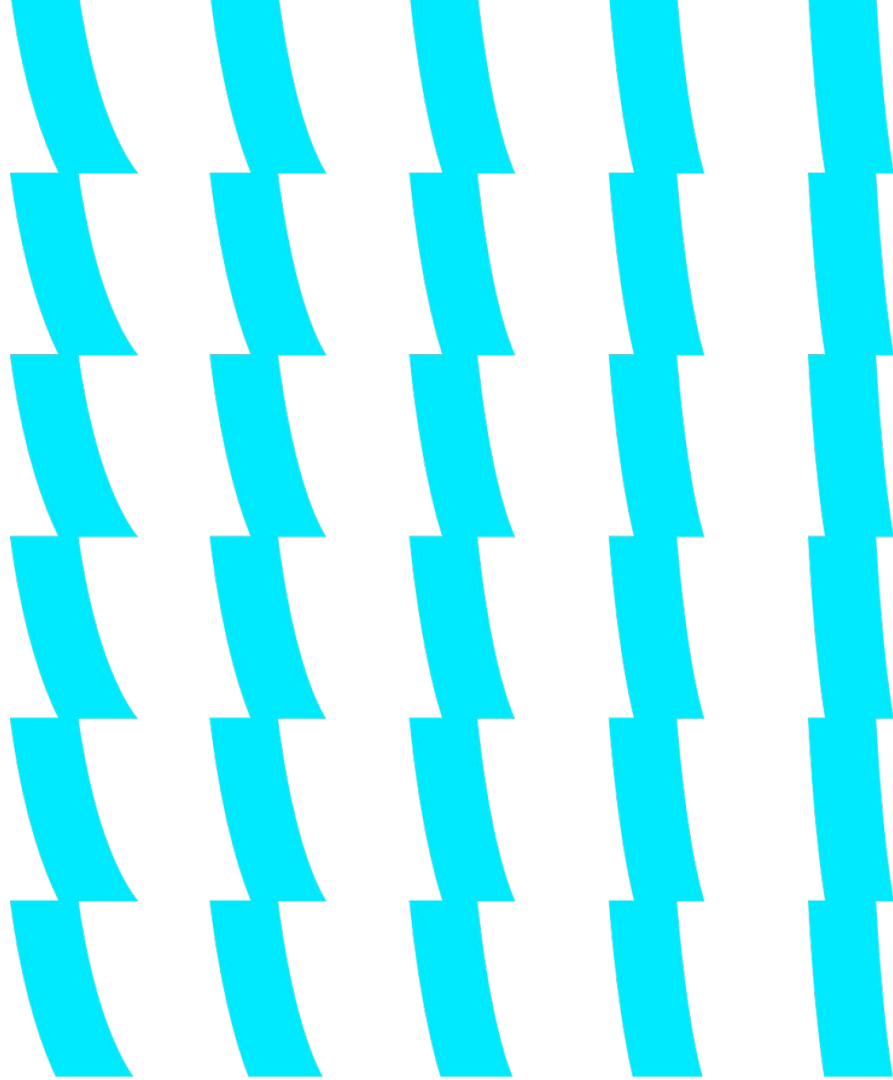
asyncio

Вспомогательное API

- `asyncio.create_task`
- `asyncio.sleep`
- `asyncio.gather`
- `asyncio.shield`
- `asyncio.wait_for`
- `asyncio.wait`
- `asyncio.Queue`
- `asyncio.Lock`
- `asyncio.Event`

Домашнее задание #7

- Асинхронный сервер для равномерной обкачки и парсинга веб-страниц
- +тесты
- flake8 + pylint перед сдачей



Напоминание отметиться на портале Vol 2

+ ОСТАВИТЬ ОТЗЫВ

mail

БлогиЛюдиПрограммаВакансииРасписание

python

сб, 16 октября	вс, 17 октября	пн, 18 октября	вт, 19 октября	ср, 20 октября	чт, 21 октября
Занятий нет	Занятий нет	18:00 Back-end разработка ...	Занятий нет	Занятий нет	Занятий нет

Backend разработка на Python

Привет!
Это блог курса Backend разработка на Python.
Все занятия проходят в зуме согласно расписанию, по ссылке:
<https://mailru.zoom.us/j/96845327537?pwd=SkFxQ0FmVXowQnR4dlh2eWM3ZmZ Rdz09>

Записи:
0 Вебинар. Организационное собрание. - [ссылка](#) (нужно смотреть/скачать через облако mail)

82 читателя, 3 топика

ПодписатьсяСоздать топик

Поиск по авторам, заголовку и тексту топика...Найти

Материалы к первой лекции

Backend разработка на PythonСмешанное занятие 1

Прямой эфир

МоиВсе

Сергей Шаленко 2 дня назад
[Лекция 1. Знакомство. Введение в Linux. Работа с файлами. Просмотр ресурсов сервера.](#) 1

Сергей Шаленко 3 дня назад
[Linux + Лекция 1. Знакомство. Введение в Linux. Работа с файлами. Просмотр ресурсов сервера.](#) 1

Сергей Шаленко 3 дня назад
[Linux + Добро пожаловать на борту!](#) 0

Артур Сардарян 3 дня назад
[Разработка приложений на iOS | Осень 2021 + Рубежный контроль](#) 1 0

Константин Ермаков 3 дня назад
[Автоматизированное тестирование | Осень 2021 + Итоги 4 лекции \(семинар\)](#) 0

Спасибо за
внимание



образование