

# Углубленный Python

Лекция 6

Потоки, GIL, процессы

Кандауров Геннадий



образование

# Напоминание отметиться на портале

+ ОСТАВИТЬ ОТЗЫВ

vk образование

БлогиЛюдиПрограммаВакансииРасписание

Q<

VK

Техно

Открыт приём заявок!

чт, 8 сентября

Нет занятий

пт, 9 сентября

18:00 Углубленный Py... с3

Введение в Python, основные понятия, тестирование

Г. Кандауров

сб, 10 сентября

Нет занятий

вс, 11 сентября

Нет занятий

пн, 12 сентября

Нет занятий

Углубленный Python

↓ 0 ↑

Блог для материалов по курсу "Углубленный Python"

57 читателей, 2 топика

Подписаться

Создать топик

Поиск по авторам, заголовку и тексту топика...

Найти

Добро пожаловать на курс!

Углубленный Python

Изменить

Удалить

Всем привет и добро пожаловать на курс по углубленному изучению Python!

Прямой эфир

МоиВсе

Геннадий Кандауров час назад

Углубленный Python → Добро пожаловать на курс! 0

Екатерина Черкасова 7 дней назад

Стажировка → Приглашаем мобильных, фронтенд- и бэкэнд-разработчиков на Weekend Offer! 0

Дарья Вовченко 9 дней назад

Углубленный Python → Добро пожаловать в образовательные проекты VK Образование! 0

Дарья Вовченко 9 дней назад

Разработка веб-сервисов на

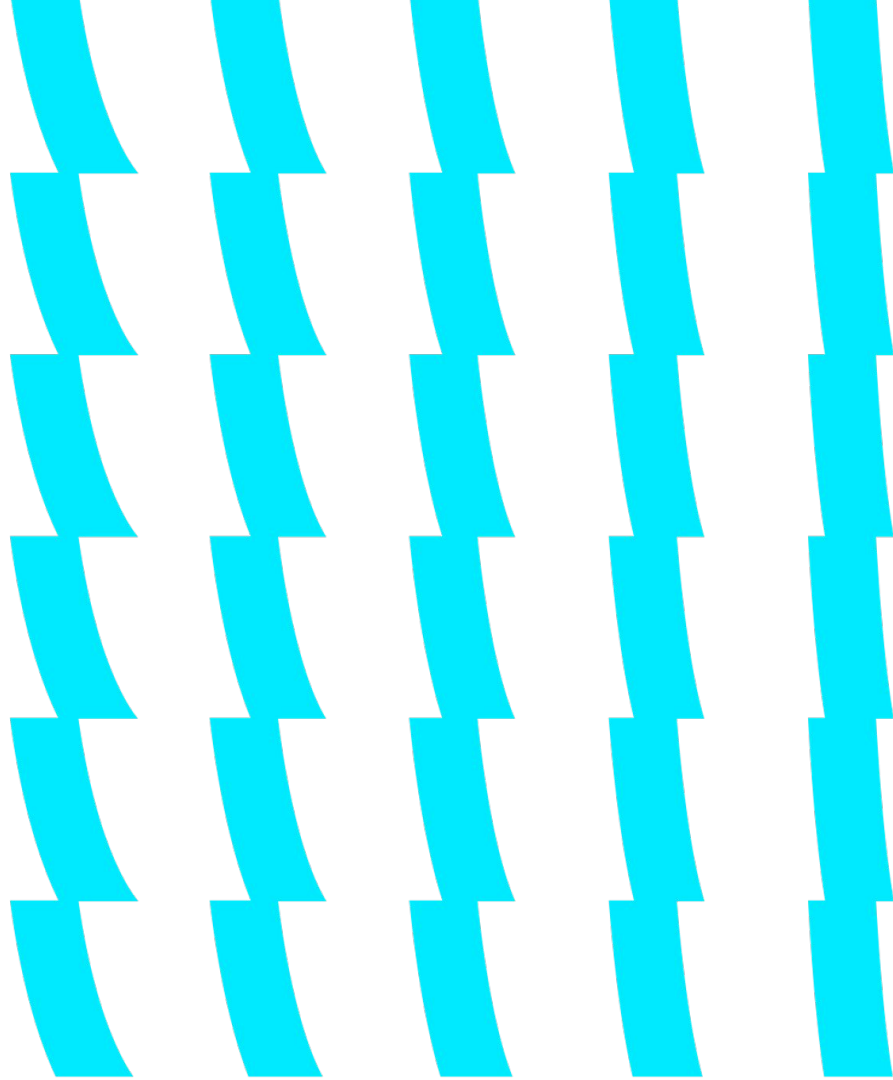
## Квиз про прошлой лекции



# Содержание занятия

1. Потоки, GIL
2. Процессы
3. IPC
4. subprocess

# Потоки (Threads)



# Потоки

**Thread (поток)** - это сущность операционной системы, процесс выполнения на процессоре набора инструкций, а именно программного кода.

# Потоки: создание и запуск

```
class threading.Thread(  
    group=None, target=None, name=None, args=(), kwargs={}, *, daemon=None  
)
```

- `th.start()`
- `th.join(timeout=None)`
- `th.run()`
- `th.is_alive()`
- `th.name`
- `th.ident`
- `th.native_id`
- `th.daemon`

# Потоки: создание и запуск

```
import threading
```

```
1. class CustomThread(threading.Thread):  
    def run(self):  
        func()
```

```
    th = CustomThread()
```

```
2. th = threading.Thread(target=func)
```

```
th.start()
```

```
th.join()
```



# Потоки: синхронизация

- `threading.Lock`
- `threading.RLock`
- `threading.Semaphore`
- `threading.BoundedSemaphore`
- `threading.Event`
- `threading.Timer`
- `threading.Barrier`

Дополнительно:

`queue (Queue, LifoQueue, PriorityQueue)`

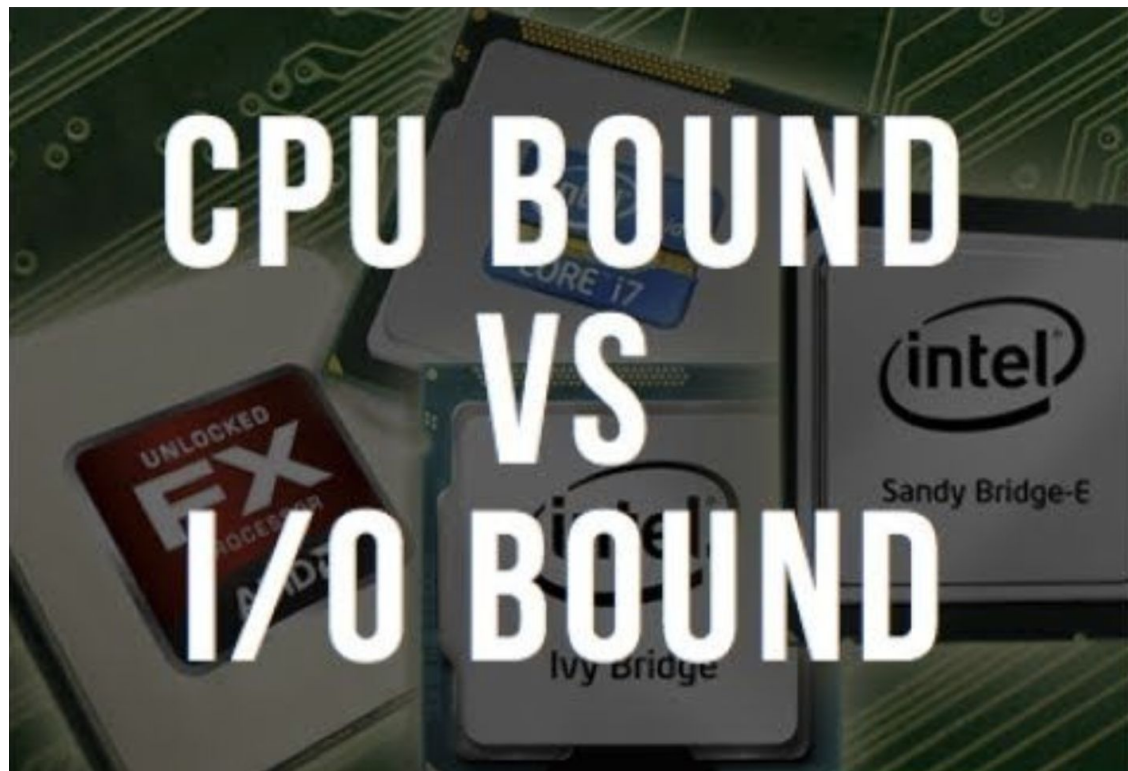
## Потоки: local

```
import threading
```

```
my_data = threading.local()
```

```
my_data.x = 42
```

Потоки



# GIL

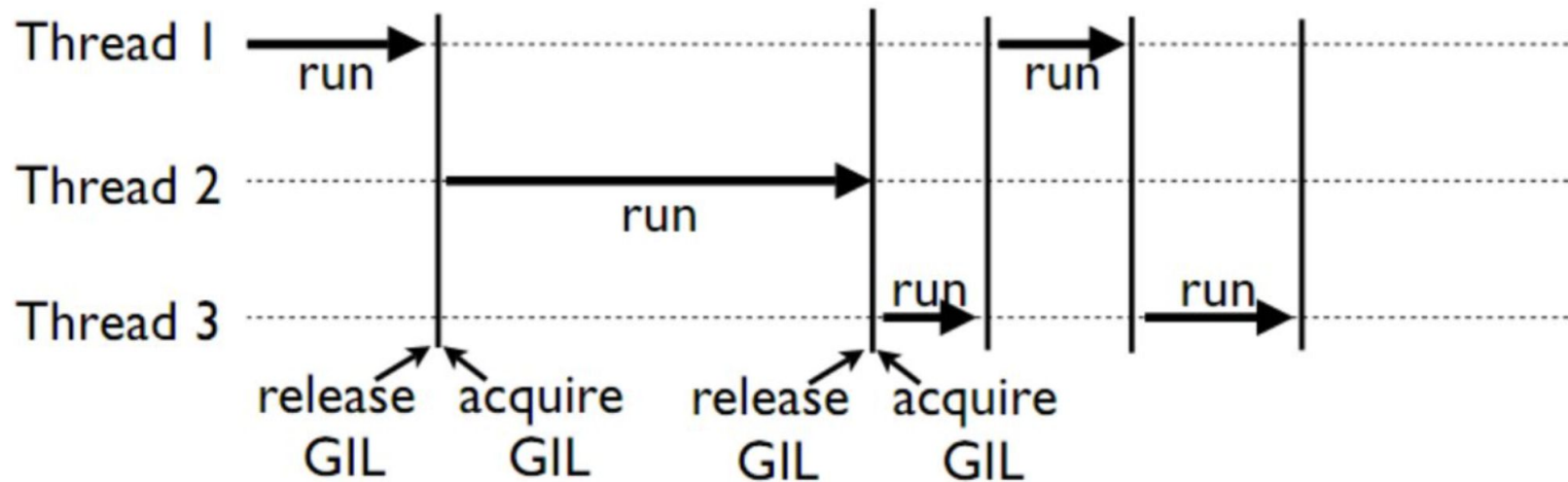
**Global Interpreter Lock (GIL)** — это способ синхронизации потоков, который используется в некоторых интерпретируемых языках программирования.

Mutex, который разрешает только одному потоку использовать интерпретатор python

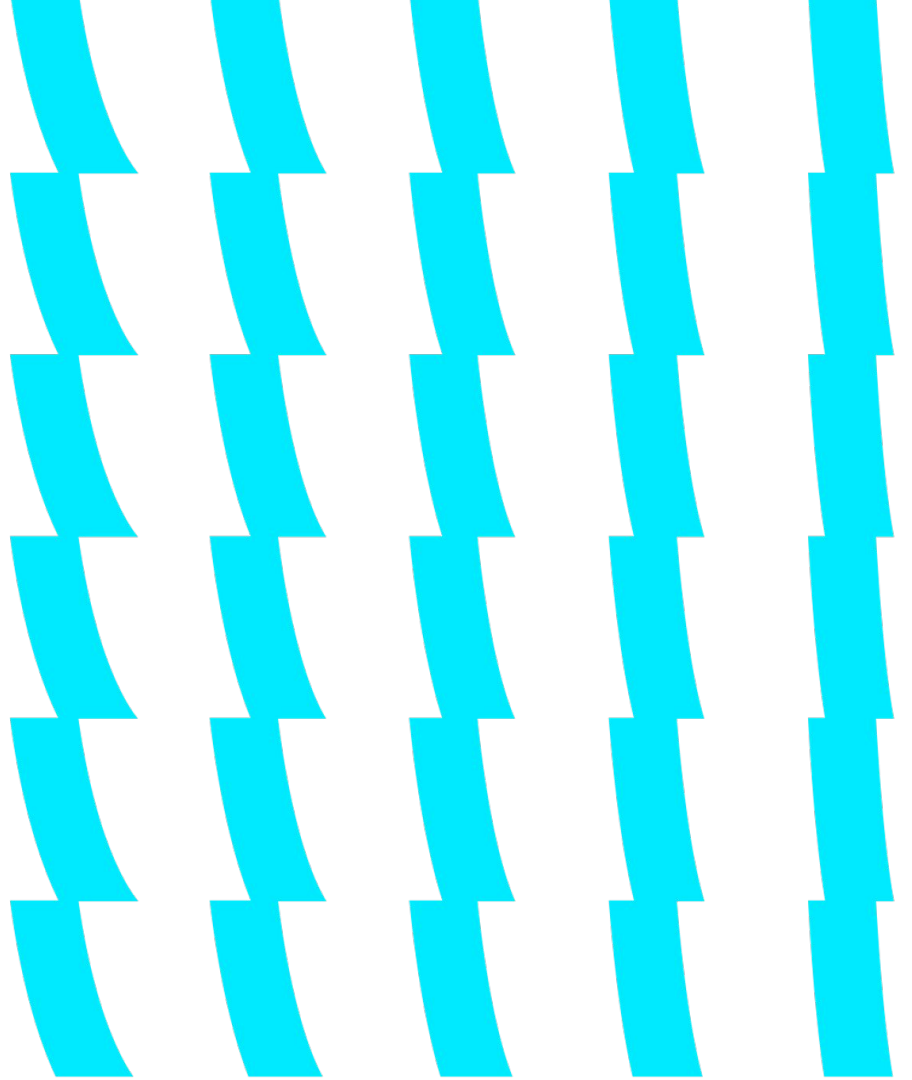
# GIL

- Что решает? - Race conditions
- Почему глобальный? - Deadlocks, производительность
- Выбран в качестве решения из-за C extensions
- Изначально вводился для I/O bound потоков

# GIL



# Multiprocessing



# Multiprocessing

**Процесс** - абстракция, которая инкапсулирует в себе все ресурсы процесса: открытые файлы, отображенные в память файлы, дескрипторы, потоки и тд.

Составные части:

1. Образ машинного кода;
2. Область памяти, в которую включается исполняемый код, данные процесса (входные и выходные данные, стек вызовов и куча для хранения динамически создаваемых данных);
3. Дескрипторы ОС, например, файловые;
4. Состояние процесса.



# Multiprocessing

```
import os
from multiprocessing import Process

def print_info(name):
    print(f"Process {name}, pid={os.getpid()}, parent pid={os.getppid()}")

if __name__ == "__main__":
    print_info("main")
    processes = [
        Process(target=print_info, args=(f"child{i}",))
        for i in range(1, 5)
    ]
    for proc in processes:
        proc.start()
    for proc in processes:
        proc.join()
```

# Multiprocessing: Pool

```
import multiprocessing
import time

def countdown(n):
    while n > 0:
        n -= 1

if __name__ == '__main__':
    t1 = time.time()
    with multiprocessing.Pool(2) as p:
        p.apply_async(countdown, (100000000,))
        p.apply_async(countdown, (100000000,))
        p.close()
        p.join()
    t2 = time.time()
    print(t2 - t1)
```

# Multiprocessing: синхронизация

- Lock, Semaphore, Event и тп

- Value

```
result = multiprocessing.Value("i")
```

- Array

```
result = multiprocessing.Array("i", 4)
```

- Manager

```
with multiprocessing.Manager() as manager:
```

```
    records = manager.list([])
```

- Queue

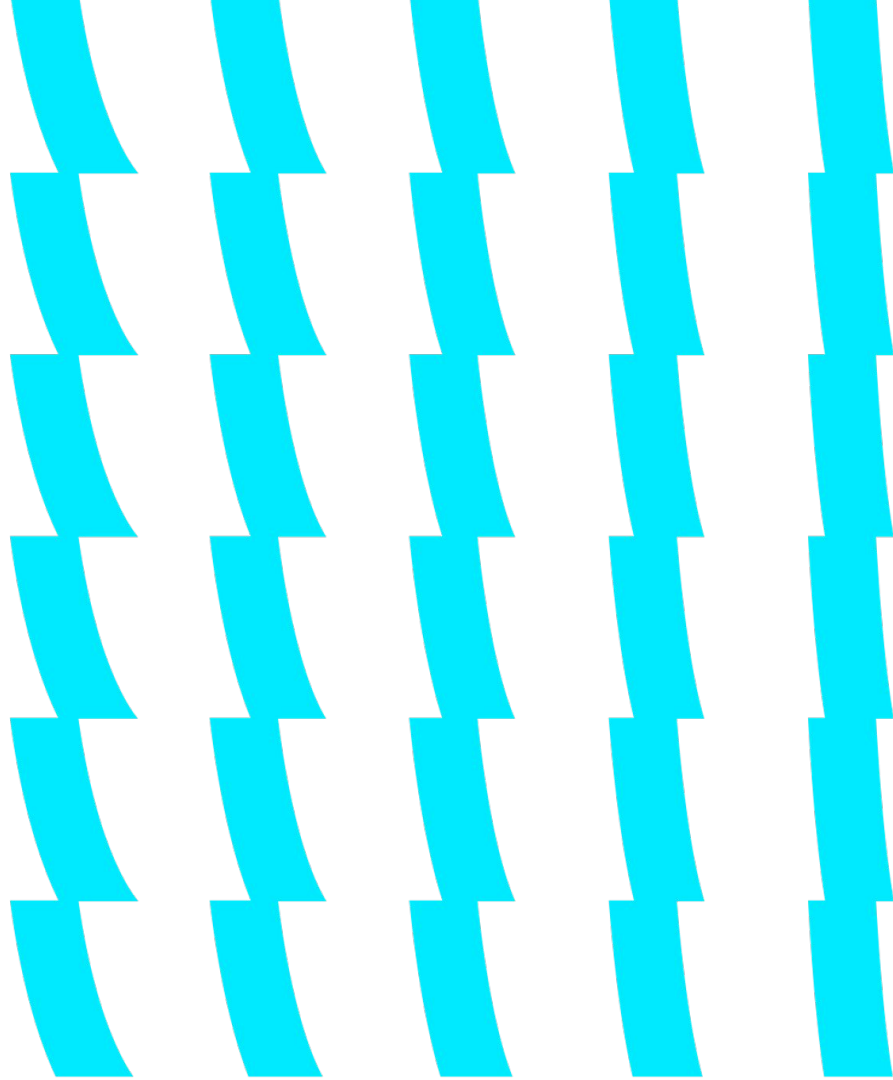
```
q = multiprocessing.Queue()
```

- Pipe

```
parent_conn, child_conn = multiprocessing.Pipe()
```

# IPC

**Inter Process Communications**  
**(межпроцессное взаимодействие)**



# IPC

ОС предоставляют механизмы для IPC:

- механизмы обмена сообщениями
- механизмы синхронизации
- механизмы разделения памяти
- механизмы удаленных вызовов (RPC)

## IPC: виды

- файл
- сигнал
- сокет
- каналы (именованные/неименованные)
- семафор
- разделяемая память
- обмен сообщениями
- проецируемый в памяти файл
- очередь сообщений
- почтовый ящик

# IPC: сигналы

```
import os, time, signal

def signal_handler(signal_num, frame):
    print(f"Handle signal {signal_num}")

if __name__ == "__main__":
    signal.signal(signal.SIGUSR1, signal_handler)
    signal.signal(signal.SIGUSR2, signal_handler)

    print(f"pid={os.getpid()}")
    while True:
        time.sleep(0.5)
```

# IPC: сокеты

```
import socket
```

```
server = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)  
server.bind("/tmp/py_unix_example")  
data = server.recv(1024)
```

```
client = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)  
client.connect("/tmp/py_unix_example")  
client.send(data.encode())
```



# IPC: каналы (pipe)

```
# sender.py
```

```
import os
```

```
fpath = "/tmp/example.fifo"  
os.mkfifo(fpath)
```

```
fifo = open(fpath, "w")  
fifo.write("Hello!\n")  
fifo.close()
```

```
# receiver.py
```

```
import os  
import sys
```

```
fpath = "/tmp/example.fifo"
```

```
fifo = open(fpath, "r")  
for line in fifo:  
    print(f"Recv: {line}")  
fifo.close()
```

# IPC: mmap

```
import mmap

with open("data.txt", "w") as f:
    f.write("Hello, python!\n")

with open("data.txt", "r+") as f:
    map = mmap.mmap(f.fileno(), 0)
    print(map.readline()) # Hello, python!
    print(map[:5]) # Hello
    map[7:] = "world!\n"
    map.seek(0)
    print(map.readline()) # Hello, world!
    map.close()
```

# subprocess

```
subprocess.run(args, **kwargs)
```

```
subprocess.run(["ls", "-l", "/dev/null"], capture_output=True)  
CompletedProcess(args=['ls', '-l', '/dev/null'], returncode=0,  
stdout=b'crw-rw-rw- 1 root root 1, 3 Jan 23 16:23 /dev/null\n', stderr=b'')
```

```
subprocess.Popen(args, **kwargs)
```

## Домашнее задание #06

- Клиент и сервер для обкачки урлов и подсчета наиболее употребимых слов
- +тесты
- flake8 + pylint перед сдачей

# Напоминание отметиться на портале Vol 2

+ ОСТАВИТЬ ОТЗЫВ ПОСЛЕ ЛЕКЦИИ

vk образование

БлогиЛюдиПрограммаВакансииРасписание

Q<

VK

Техно

Открыт приём заявок!

чт, 8 сентября

Нет занятий

пт, 9 сентября

18:00 Углубленный Py... с3  
Введение в Python, основные  
понятия, тестирование  
Г. Кандауров

сб, 10 сентября

Нет занятий

вс, 11 сентября

Нет занятий

пн, 12 сентября

Нет занятий

Углубленный Python

↓ 0 ↑

Блог для материалов по курсу "Углубленный Python"

57 читателей, 2 топика

Подписаться

Создать топик

Поиск по авторам, заголовку и тексту топика...

Найти

Добро пожаловать на курс!

Углубленный Python

Изменить

Удалить

Всем привет и добро пожаловать на курс по углубленному изучению Python!

Прямой эфир

МоиВсе

Геннадий Кандауров час назад  
Углубленный Python → Добро пожаловать  
на курс! 0

Екатерина Черкасова 7 дней назад  
Стажировка → Приглашаем мобильных,  
фронтенд- и бэкэнд-разработчиков на  
Weekend Offer! 0

Дарья Вовченко 9 дней назад  
Углубленный Python → Добро пожаловать  
в образовательные проекты VK  
Образование! 0

Дарья Вовченко 9 дней назад  
Разработка веб-сервисов на

Спасибо за  
внимание



образование