



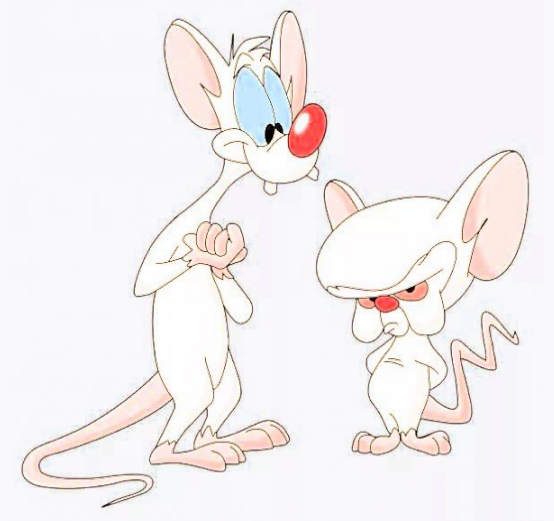
Клиент-серверные приложения на Python

## Урок 7

# Модуль select, слоты

Особенности использования модуля select.  
Слоты, их назначение и применение.

# Цели урока



1. Продолжить работу с сетью (модуль **select**);
2. Изучить назначение и особенности применения слотов.



# Сетевое программирование (продолжение)

ТСР/ІР



# Модуль `select`

Предоставляет функции `select()` и `poll()`, работающие на уровне ОС.



# Модуль `select`

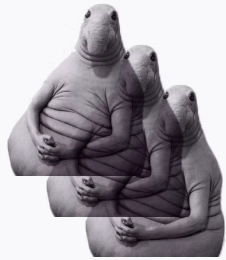
```
readst, writest, errorst = select(rlist, wlist, xlist[ , timeout])
```

## Опрос ожидающих объектов



# Модуль select

```
readst, writest, errorst = select(rlist, wlist, xlist[ , timeout])
```



Ожидающие  
чтения



# Модуль select

```
readst, writest, errorst = select(rlist, wlist, xlist[ , timeout])
```



Ожидающие  
чтения



Ожидающие  
записи



# Модуль select

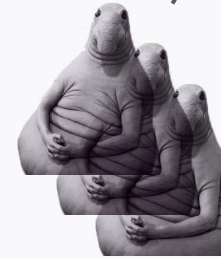
```
readst, writest, errorst = select(rlist, wlist, xlist[ , timeout])
```



Ожидающие  
чтения



Ожидающие  
записи



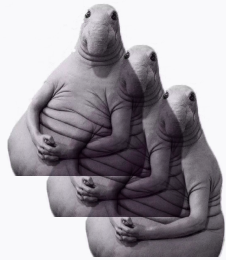
Ожидающие  
исключений





# Модуль select

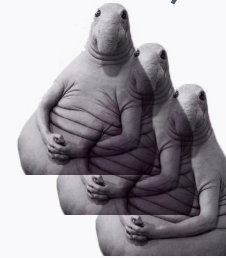
```
readst, writest, errorst = select(rlist, wlist, xlist[ , timeout])
```



Ожидающие  
чтения



Ожидающие  
записи



Ожидающие  
исключений

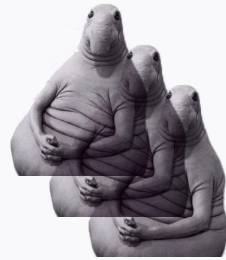


Таймаут  
ожидания

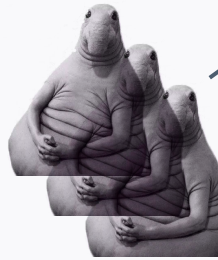


# Модуль select

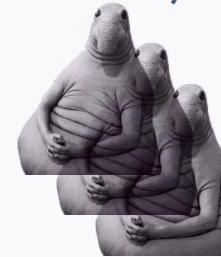
```
readst, writest, errorst = select(rlist, wlist, xlist[ , timeout])
```



Ожидающие  
чтения



Ожидающие  
записи



Ожидающие  
исключений



Таймаут  
ожидания



стандартный ввод (stdin), канал (popen), сокет (socket)



# Модуль select

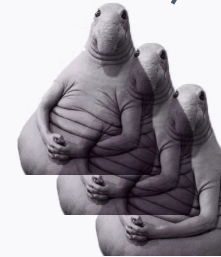
```
readst, writest, errorst = select(rlist, wlist, xlist[ , timeout])
```



Ожидающие  
чтения



Ожидающие  
записи



Ожидающие  
исключений



Таймаут  
ожидания



стандартный ввод (`stdin`), канал (`popen`), сокет (`socket`)

## Note

File objects on Windows are **not acceptable**, but sockets are. On Windows, the underlying `select()` function is provided by the WinSock library, and does not handle file descriptors that don't originate from WinSock.





# slots



# \_\_slots\_\_

```
class StrictClass:  
    __slots__ = ('x', 'y')  
  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y
```

Экземпляры такого класса смогут иметь атрибуты только с указанными именами



# `__slots__`

```
class StrictClass:
```

```
    __slots__ = ('x', 'y')
```

- Атрибут класса
- Кортеж с именами атрибутов

```
    def __init__(self, x, y):
```

```
        self.x = x
```

```
        self.y = y
```

Производный класс также  
должен объявлять атрибут `__slots__`



# `__slots__`

```
class StrictClass:
```

```
    __slots__ = ('x', 'y')
```

```
    def __init__(self, x, y):  
        self.x = x  
        self.y = y
```

- Атрибут класса
- Кортеж с именами атрибутов





# Практическое задание





# Практическое задание

Продолжаем работу над проектом «Мессенджер»:

1. Реализовать обработку нескольких клиентов на сервере, используя функцию **select**. Клиенты должны общаться в «общем чате»: каждое сообщение участника отправляется всем, подключенным к серверу.
2. Реализовать функции отправки/приема данных на стороне клиента. Чтобы упростить разработку на данном этапе, пусть клиентское приложение будет либо только принимать, либо только отправлять сообщения в общий чат. Эти функции надо реализовать в рамках отдельных скриптов.



# Дополнительные материалы

1. Что такое select: <https://toster.ru/q/311579>;
2. Разбираемся с устройством асинхронных фреймворков для Python: <https://xakep.ru/2015/02/21/python-tornado/>;
3. Программирование на Python. Специальные методы и атрибуты классов:  
[https://www.ibm.com/developerworks/ru/library/l-python\\_part\\_7/index.html](https://www.ibm.com/developerworks/ru/library/l-python_part_7/index.html);
4. Использование `__slots__`:  
<http://qaru.site/questions/5007/usage-of-slots>.

