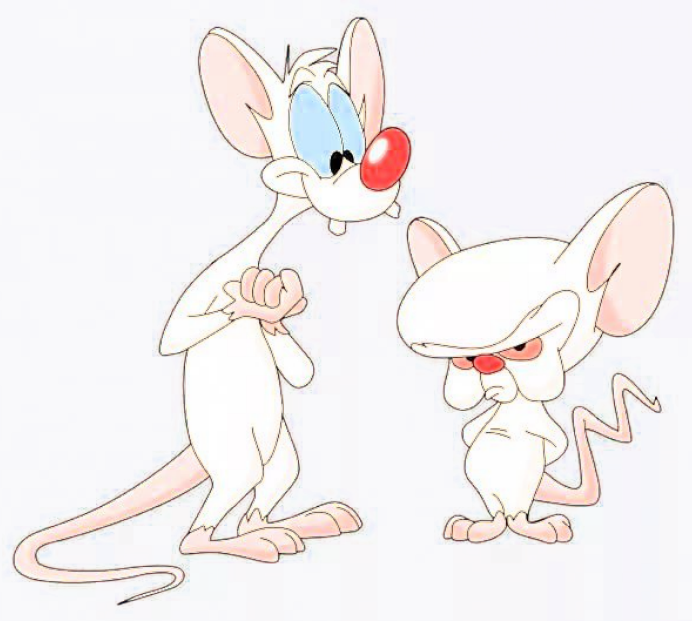
Клиент-серверные приложения на Python

Урок 6

Декораторы и продолжение работы с сетью

Декоратор. Декоратор с параметром. Сетевое программирование (продолжение).

Цели урока



1. Погрузиться в тему декораторов;
2. Продолжить изучение особенностей сетевого взаимодействия.

Декораторы

@property

@login_required

@x.setter

@require_http_methods

@classmethod

@permission_required

@staticmethod

@pytools.fixture

@pytest.yield_fixture

@functools.wraps

@pytest.mark.parametrize

Декораторы

Функция

```
def func_z()
    код
```

Декоратор

```
def wrap(func)
    def deco(*args, **kwargs)
        код
    return deco
```

Декораторы

Функция

```
def func_z()
    код
```

Декоратор

```
def wrap(func)
```

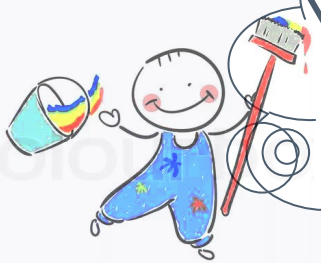
Функция

```
def deco(*args, **kwargs)
    код
```

```
return deco
```

Применение декоратора

```
@wrap
def func_z()
    код
```



Декораторы

Функция

```
def func_z()
    код
```

Декоратор

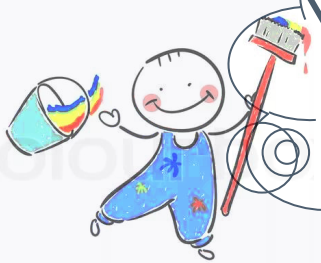
```
def wrap(func)
```

Функция

```
def deco(*args, **kwargs)
    код
```

```
return deco
```

Применение декоратора



```
@wrap
def func_z()
    код
```

Никакой магии, только
“синтаксический сахар”

Декораторы

Функция

```
def func_z()
    код
```

Декоратор

```
def wrap(func)
```

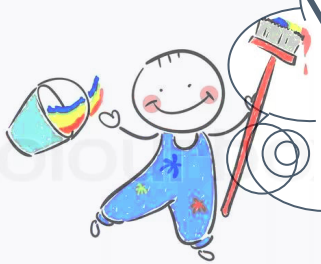
Функция

```
def deco(*args, **kwargs)
    код
```

```
return deco
```

Применение декоратора

```
@wrap
def func_z()
    код
```



```
def func_z()
    код
```

```
func_z = wrap(func_z)
```

Никакой магии, только
“синтаксический сахар”

Декораторы

Функция

```
def func_z()
    код
```

Декоратор

```
def wrap(func)
```

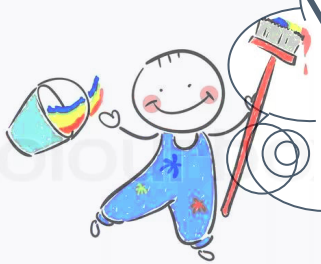
Функция

```
def deco(*args, **kwargs)
    код
```

```
return deco
```

Применение декоратора

```
@wrap
def func_z()
    код
```



```
def func_z()
    код
func_z = wrap(func_z)
```

Теперь при вызове...

```
func_z()
```

Никакой магии, только
“синтаксический сахар”

Декораторы

Функция

```
def func_z()
    код
```

Декоратор

```
def wrap(func)
```

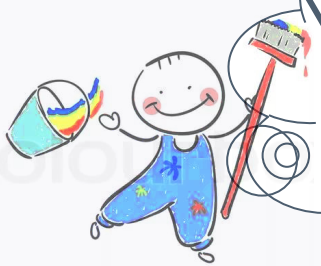
Функция

```
def deco(*args, **kwargs)
    код
```

```
return deco
```

Применение декоратора

```
@wrap
def func_z()
    код
```



Никакой магии, только **“синтаксический сахар”**

```
def func_z()
    код
```

```
func_z = wrap(func_z)
```

Теперь при вызове...

```
func_z()
```

На самом деле происходит обращение к

```
deco()
```

Декоратор с параметром

Функция

```
def func_z():
    код
```

Декоратор с параметром

```
def matrish(param):
```

Декоратор

```
def wrap(func):
```

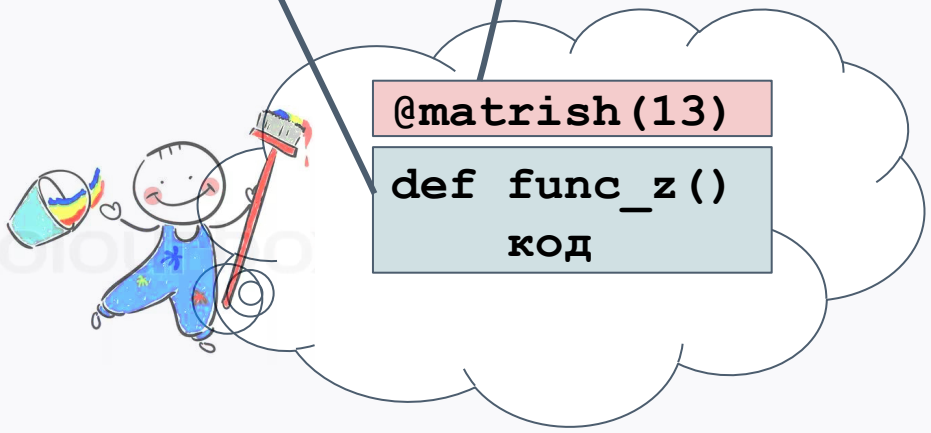
Функция

```
def deco(*args, **kwargs):
    код
```

```
return deco
```

```
return wrap
```

Применение декоратора



```
@matrish(13)
def func_z()
    код
```

```
def func_z()
    код
```

```
func_z = matrish(13)(func_z)
```

Декоратор с параметром

Функция

```
def func_z():
    код
```

Декоратор с параметром

```
def matrish(param):
```

Декоратор

```
def wrap(func):
```

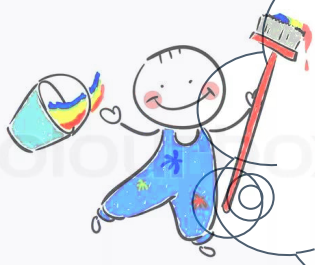
Функция

```
def deco(*args, **kwargs):
    код
```

```
return deco
```

```
return wrap
```

Применение декоратора



```
@matrish(13)
def func_z()
    код
```

```
def func_z()
    код
func_z = matrish(13)(func_z)
```

Декоратор с параметром

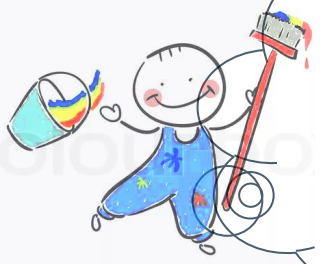
Функция

```
def func_z():
    код
```

Декоратор с параметром

```
def matrish(param):
    def wrap(func):
        def deco(*args, **kwargs):
            код
        return deco
    return wrap
```

Применение декоратора



```
@matrish(13)
def func_z()
    код
```

```
def func_z()
    код
func_z = matrish(13)(func_z)
```

Декоратор с параметром — это...

Декоратор с параметром — это...

Фабрика

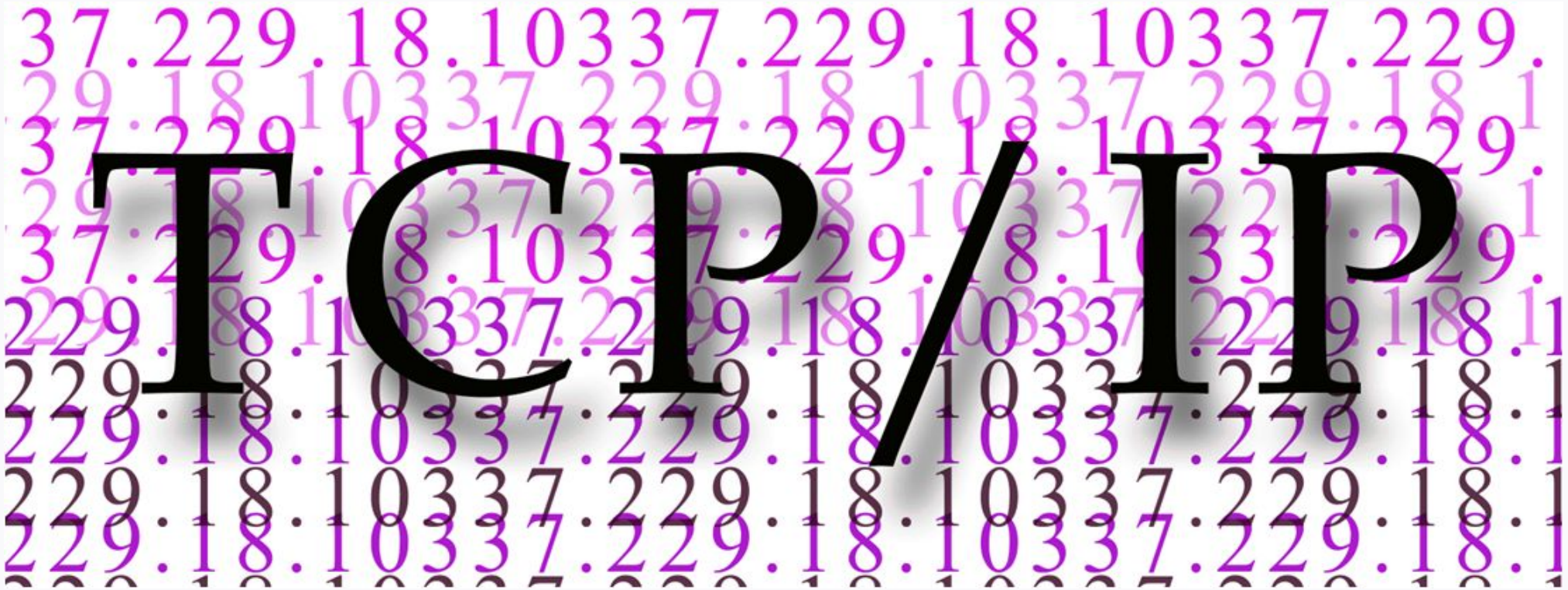
Декоратор с параметром — это...

Фабрика по производству декораторов

Декораторы. Итоги

- Подменяют функцию и её атрибуты (`__doc__`, `__name__`);
- Выполняются на этапе импорта!
- Декоратор с параметром не так страшен, как кажется;
- Аккуратней с рекурсивными функциями.

Сетевое программирование (продолжение)



Семейства адресов

Для функций из модуля `socket`

Константа	Описание
AF_BLUETOOTH	Протокол Bluetooth
AF_INET	Протокол IPv4 (TCP, UDP)
AF_INET6	Протокол IPv6 (TCP, UDP)
AF_NETLINK	Протокол Netlink взаимодействия процессов
AF_PACKET	Пакеты канального уровня
AF_TIPC	Прозрачный протокол взаимодействия процессов (Transparent Inter-Process Communication protocol, TIPC)
AF_UNIX	Протоколы домена UNIX

Типы сокетов

Константа	Описание
SOCK_STREAM	Поток байтов с поддержкой логического соединения, обеспечивающего надежность передачи данных (TCP)
SOCK_DGRAM	Дейтаграммы (UDP)
SOCK_RAW	Простой сокет
SOCK_RDM	Дейтаграммы с надежной доставкой
SOCK_SEQPACKET	Обеспечивает последовательную передачу записей с поддержкой логических соединений

Модуль socket

Методы класса socket.socket

```
s.accept()
s.bind(address)
s.close()
s.connect(address)
s.fileno()
s.getpeername()
s.getsockname()
s.getsockopt(level, optname [, buflen])
s.setsockopt(level, optname, value)
s.gettimeout()
s.settimeout(timeout)
```

```
s.listen(backlog)
s.recv(bufsize [, flags])
s.send(string [, flags])
s.sendall(string [, flags])
s.setblocking(flag)
s.shutdown(how)
```

Практическое задание

Практическое задание

1. Продолжая задачу логирования, реализовать декоратор **@log**, фиксирующий обращение к декорируемой функции. Он сохраняет ее имя и аргументы.
2. В декораторе **@log** реализовать фиксацию функции, из которой была вызвана декорированная.

Дополнительные материалы

1. Logging Cookbook: https://docs.python.org/3/howto/logging-cookbook.html;
2. Python Decorators Library: https://wiki.python.org/moin/PythonDecoratorLibrary;
3. Awesome Python Decorator – A curated list of awesome python decorator resources: https://github.com/lord63/awesome-python-decorator;
4. Понимаем декораторы в Python, шаг за шагом. Шаг 1: https://habrahabr.ru/post/141411/.