



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



Декораторы

Функция

```
def func_z()  
    код
```

Декоратор

```
def wrap(func)
```

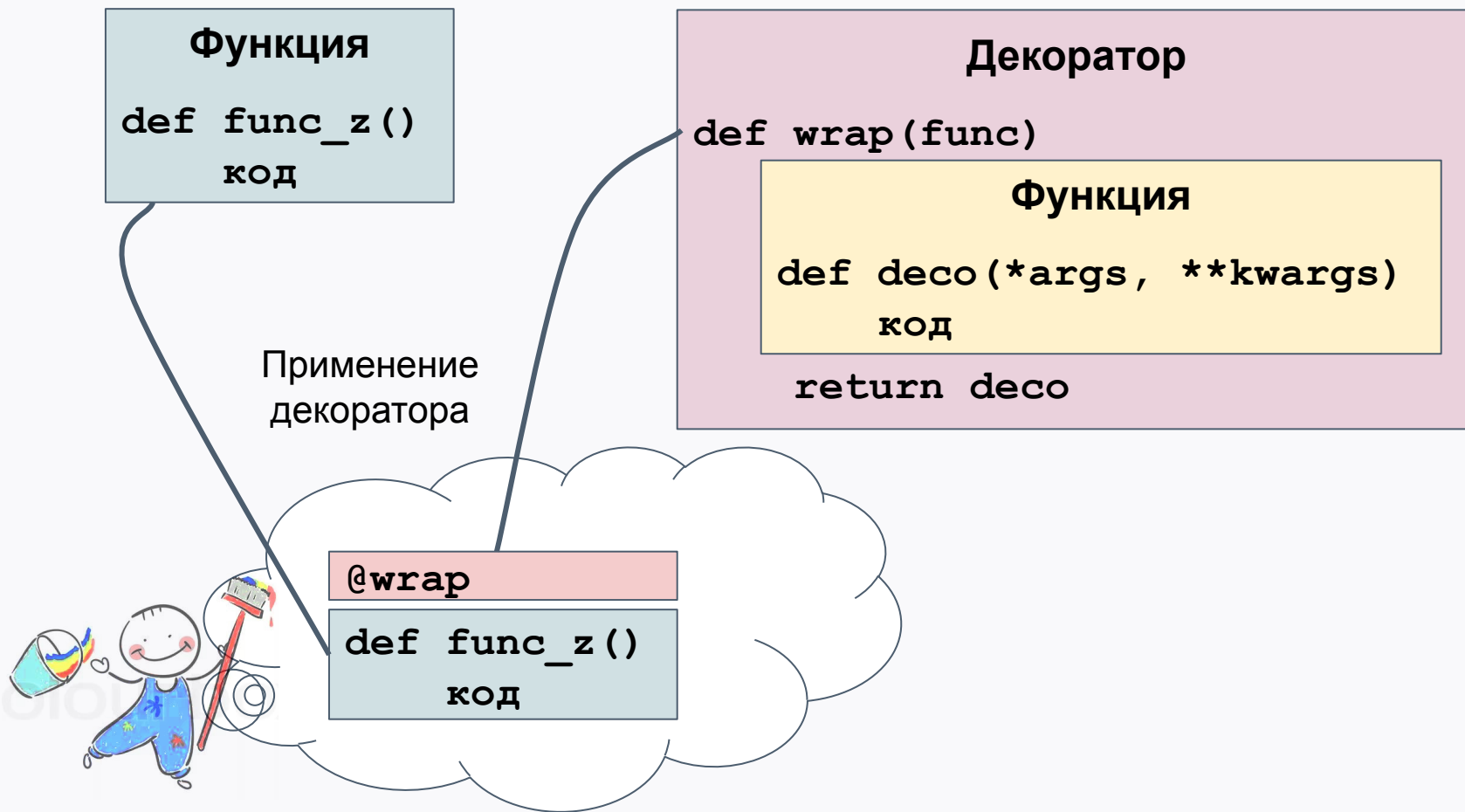
Функция

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

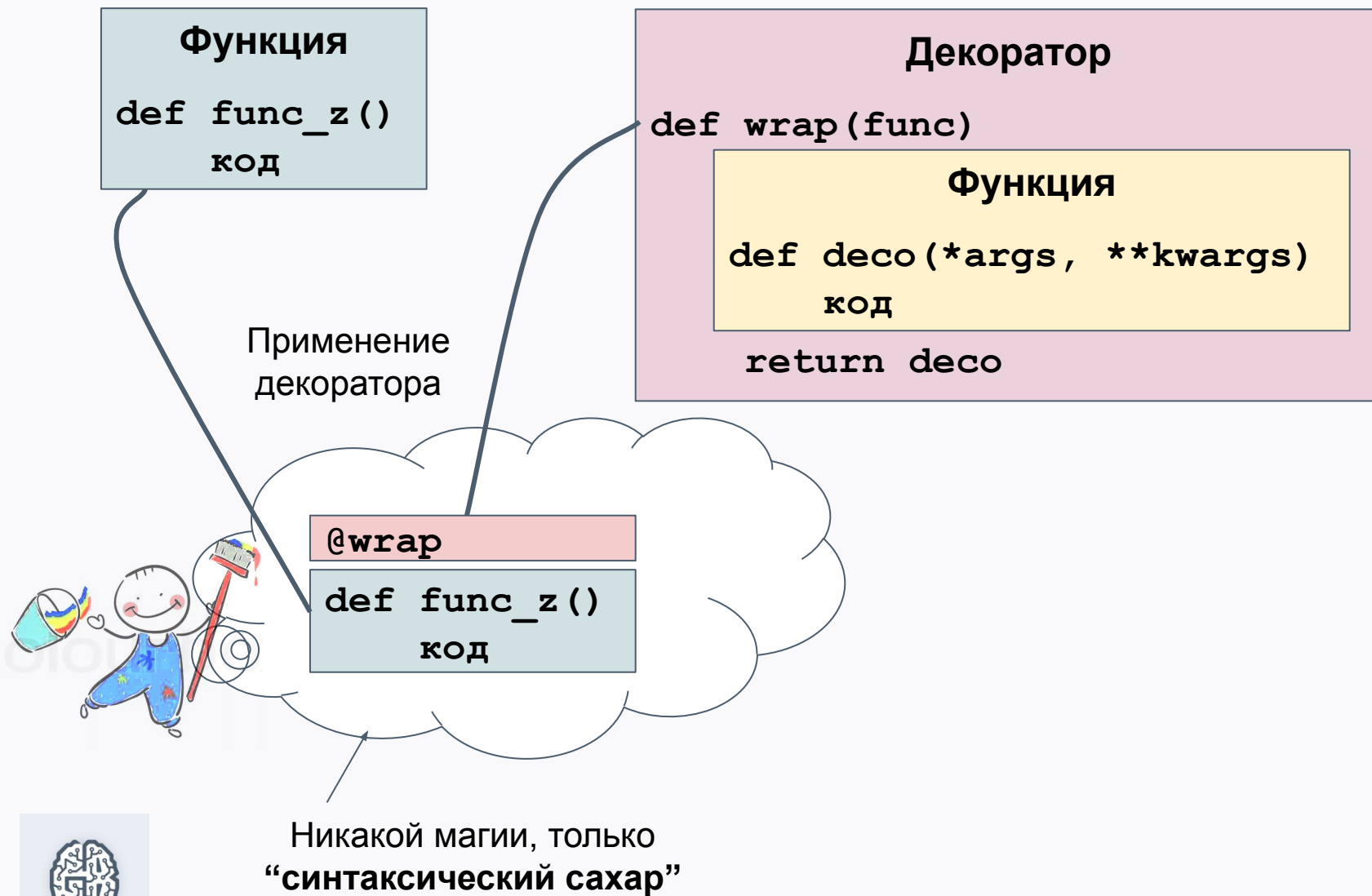
```
return deco
```



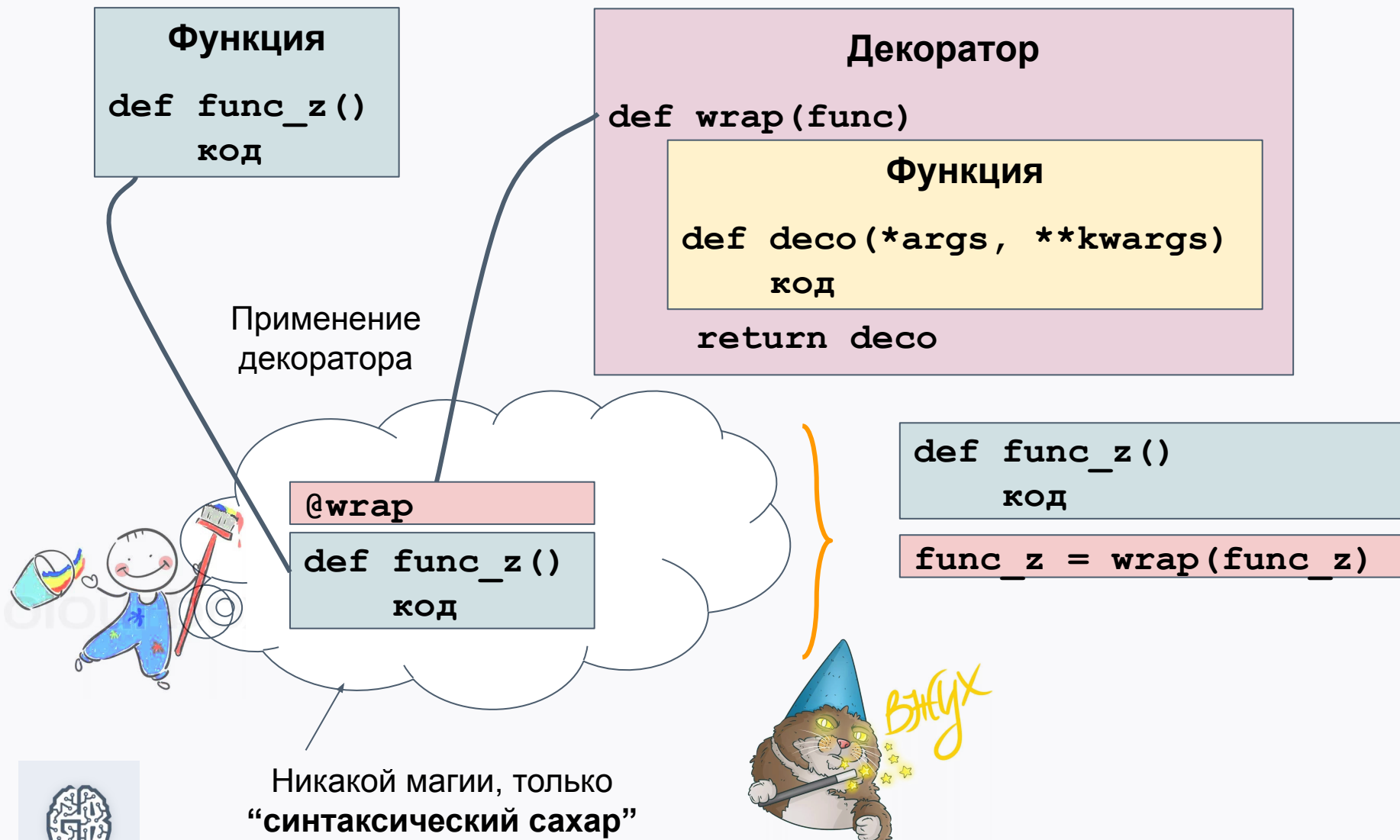
Декораторы



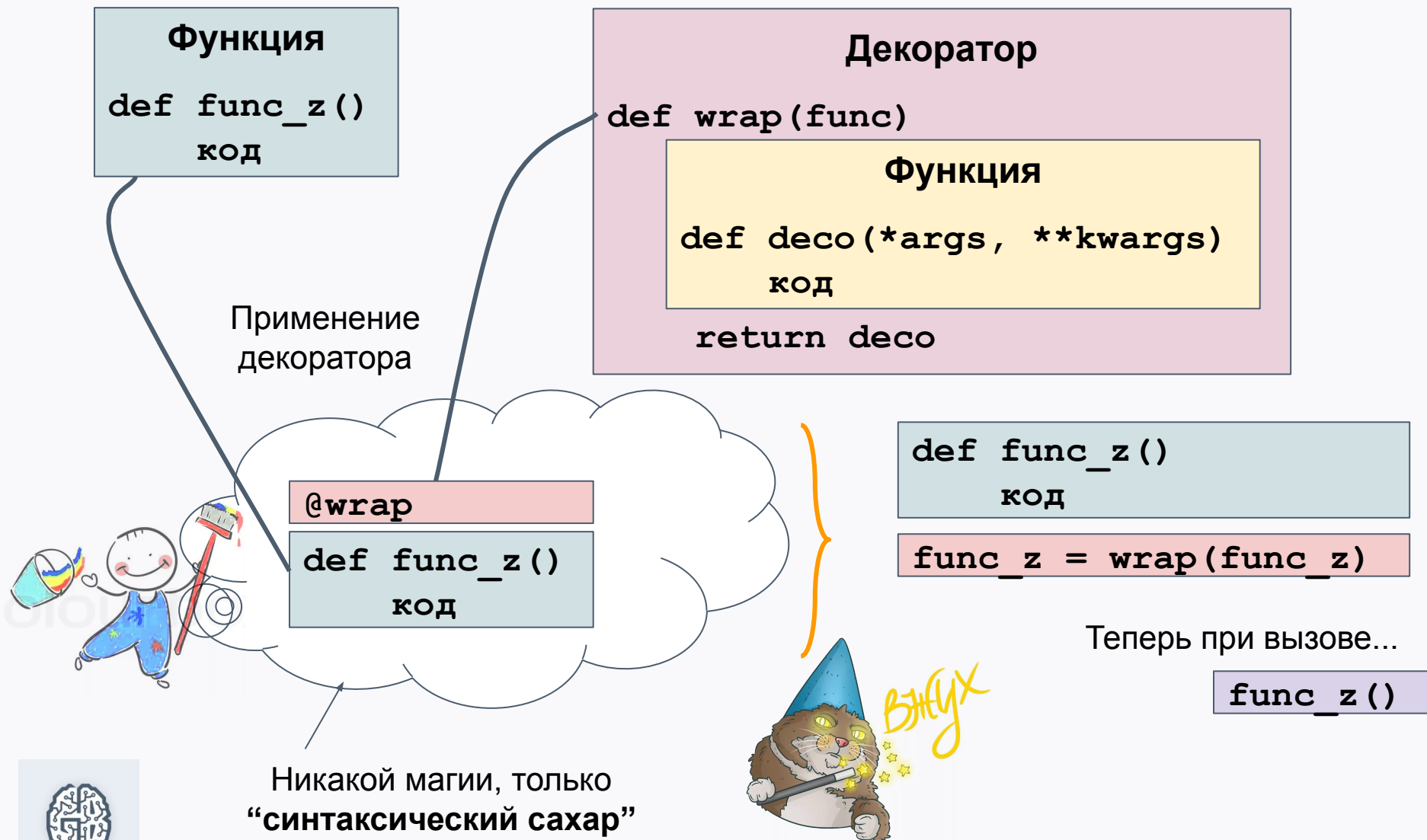
Декораторы



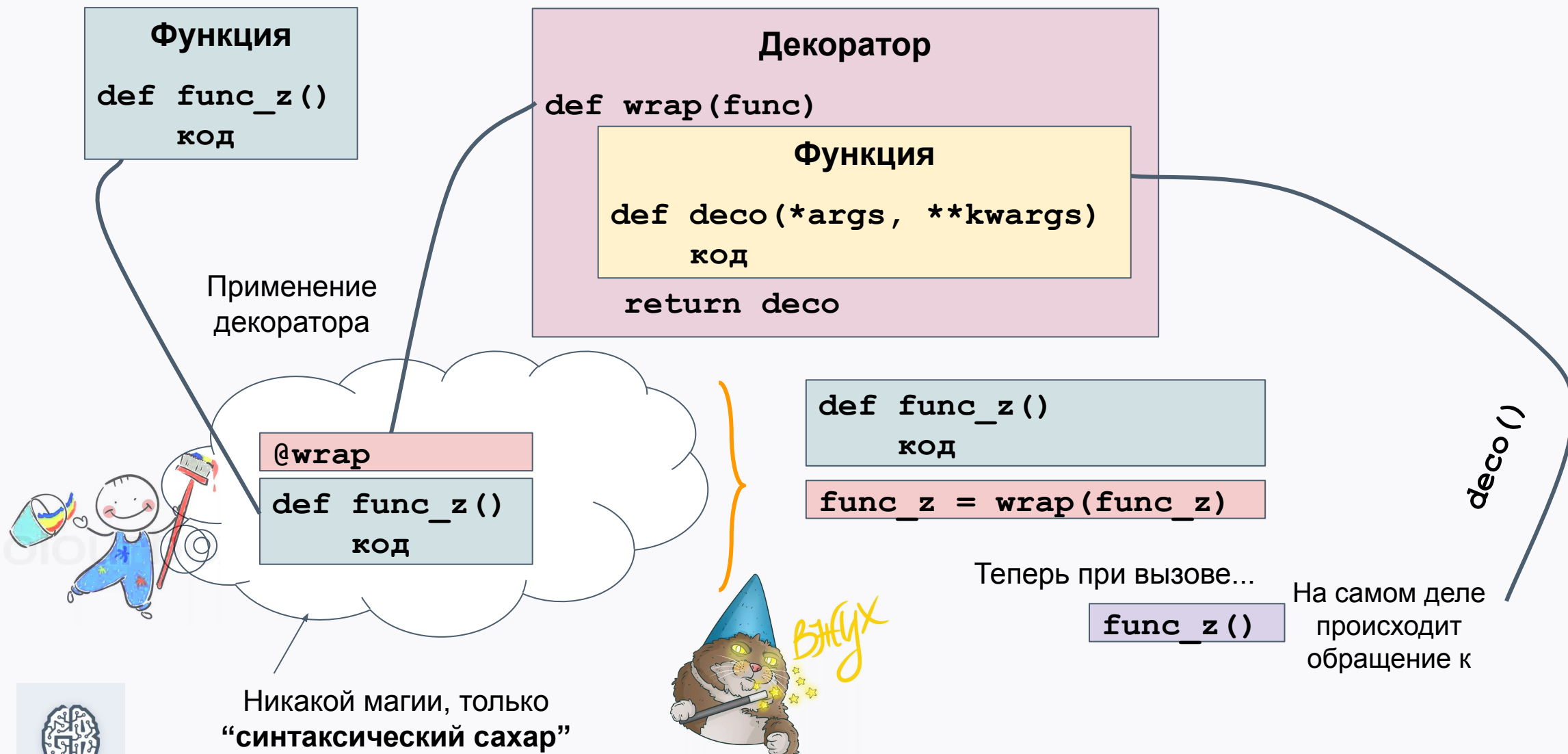
Декораторы



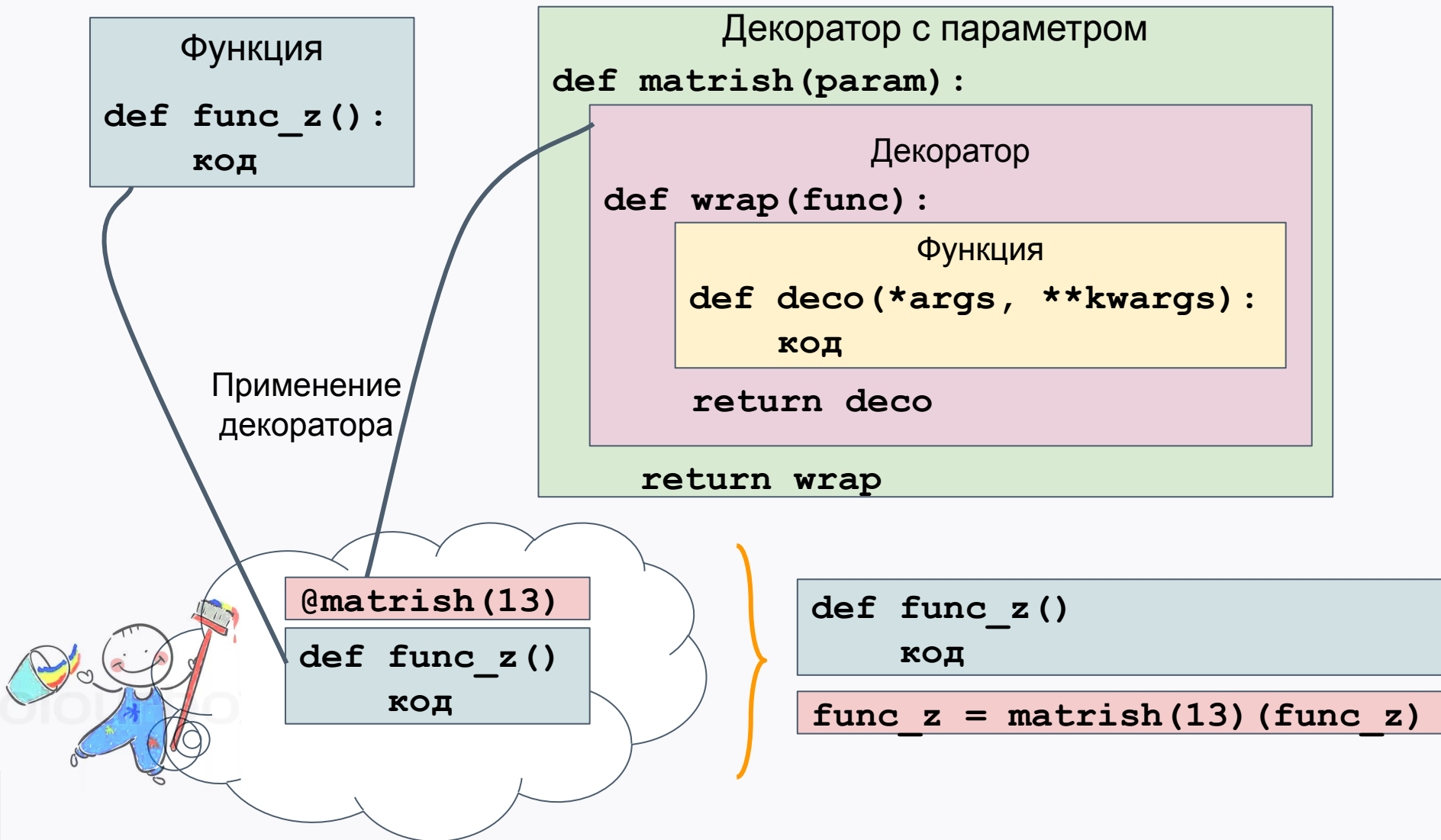
Декораторы



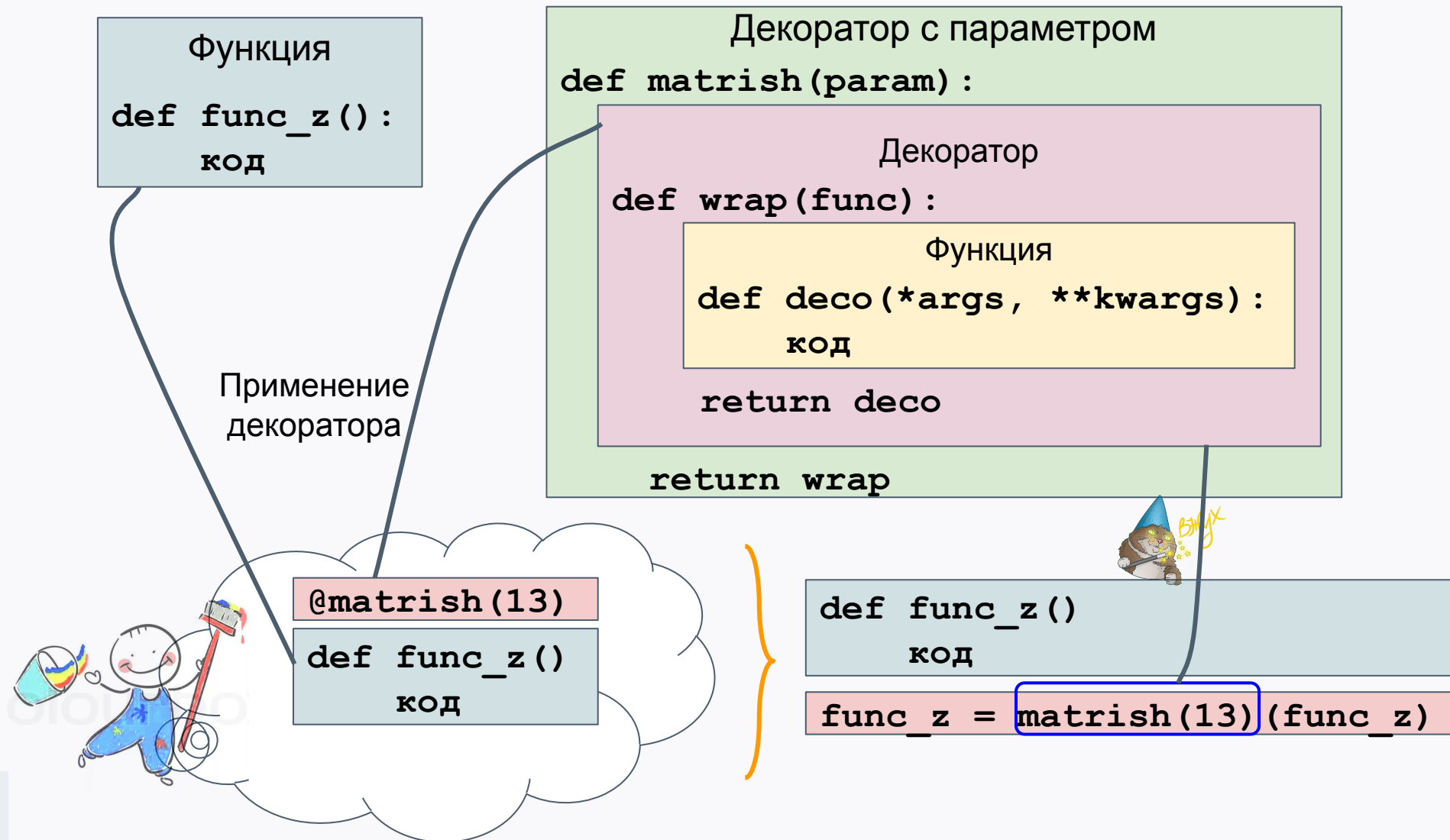
Декораторы



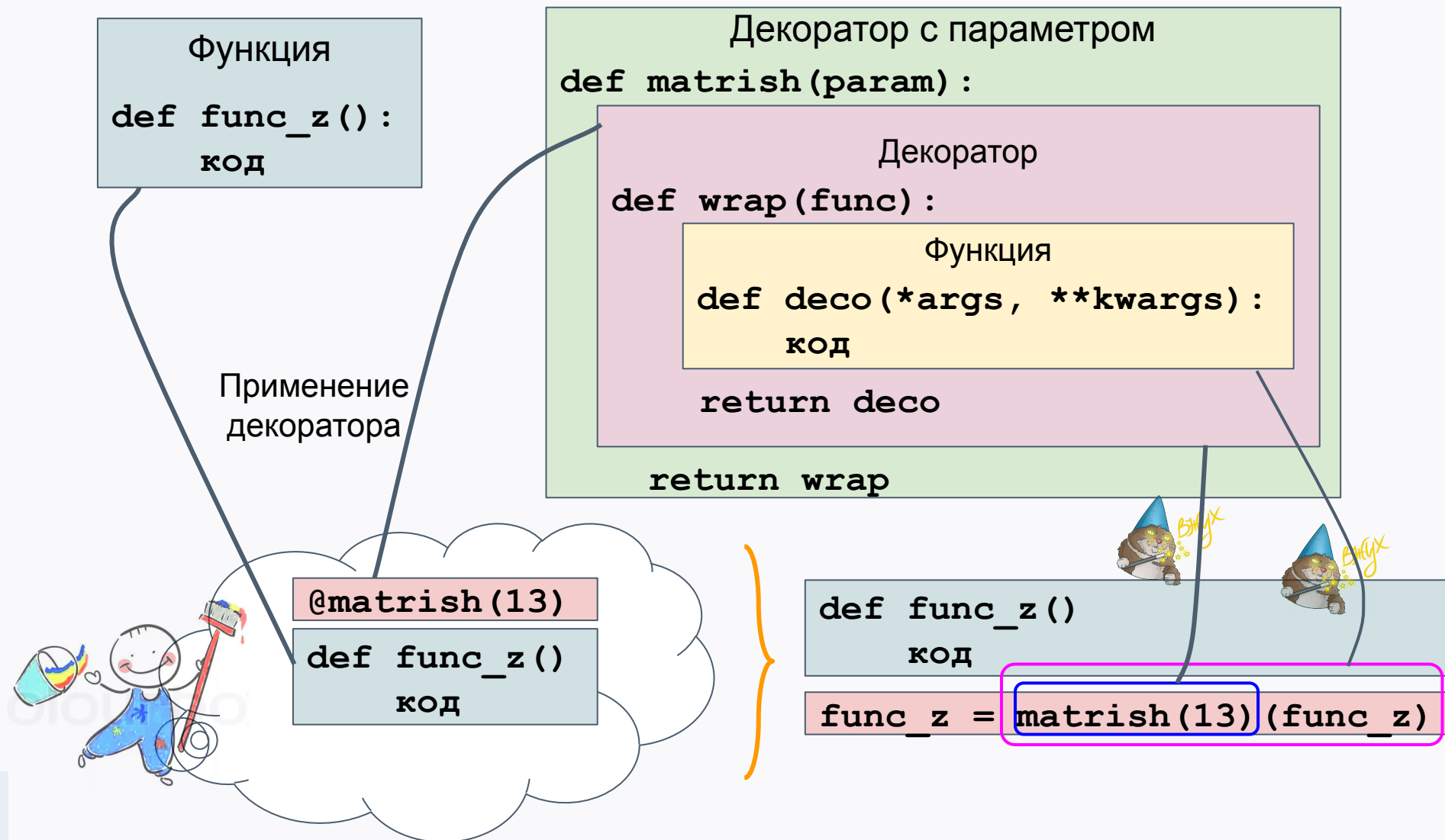
Декоратор с параметром



Декоратор с параметром



Декоратор с параметром



Декоратор с параметром — это...



Декоратор с параметром — это...



Декоратор с параметром — это...



Вжух

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)

ТСР/ІР



Семейства адресов

Для функций из модуля `socket`

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



Типы сокетов

Константа	Описание
<i>SOCK_STREAM</i>	Поток байтов с поддержкой логического соединения, обеспечивающего надежность передачи данных (TCP)
<i>SOCK_DGRAM</i>	Дейтаграммы (UDP)
<i>SOCK_RAW</i>	Простой сокет
<i>SOCK_RDM</i>	Дейтаграммы с надежной доставкой
<i>SOCK_SEQPACKET</i>	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/>.

