Академия
Аналитиков
Авито

ДЕКОРАТОРЫ В PYTHON

Сергей Нинуа

Правила игры 📖

- Пожалуйста, включите камеру 🙏
- не стесняйтесь спрашивать: пишите в чате или поднимайте руку
- держите наготове IDE или редактор
- экспериментируйте!

План занятия 🧭 :

- Освежим знания о функциях – базисе для декораторов:
  - First-class objects
  - Вложенные функции
- Узнаём, какие бывают декораторы и как они работают:
  - Простой декоратор
  - Параметрический декоратор
  - Наложение нескольких декораторов

Академия
Аналитиков
Авито

ДЕМО

Академия
Аналитиков
Авито

FIRST-CLASS OBJECTS

First-class objects

- Функция без скобок – ссылка на объект функции (не результат выполнения)
- Функции подобны другим объектам в Python, например числам, строкам, спискам и т.д.:
  - Можно передавать как параметр в другие функции
  - Можно возвращать как результат функции
- Во всех описанных случаях используется ссылка на объект функции, сама функция не вызывается

First-class objects — демо

```python
from typing import Callable

def hail_someone(name: str) -> str:
    return f'Hail to you, {name}'

def say_hi(name: str) -> str:
    return f'Hi, {name}'

def greetings(greet_func: Callable) -> str:
    return greet_func('Sergey')

if __name__ == '__main__':
    print(greetings(hail_someone))
```

Академия
Аналитиков
Авито

ВЛОЖЕННЫЕ ФУНКЦИИ

Вложенные функции

- Функция, определённая внутри другой функции:
  - "Видна" только внутри той функции, где определена
  - Имеет доступ к контексту внешней по отношению к ней функции
  - Существует как локальный объект-ссылка
- Можно возвращать как результат

Вложенные функции

```
def the_father():
    print("I am the father")

    def the_son():
        print("I am the son")

    def the_daughter():
        print("I am the daughter")

    the_son()
    the_daughter()


if __name__ == '__main__':
    the_father()
    the_son()
```

Академия
Аналитиков
Авито

Вложенные функции

```python
from typing import Callable


def produce_child(name: str) -> Callable:
    print("I am the father")

    def the_son():
        print("The son was born")

    def the_daughter():
        print("The daughter was born")

    return the_son if name == 'Bob' else the_daughter


if __name__ == '__main__':
    child_func = produce_child(name='Alice')
    child_func()
```

Академия Аналитиков Авито

Академия
Аналитиков
Авито

ДЕКОРАТОРЫ

Декораторы

Декоратор - это особая функция, которая:

- Принимает функцию и дополняет её поведение
- Возвращает модифицированную версию
- Можно управлять "добавками" с помощью параметров
- Использует специальный синтаксис – `@decorator`

Декораторы

Декоратор - это особая функция, которая:

- Принимает функцию и дополняет её поведение
- Возвращает модифицированную версию
- Можно управлять "добавками" с помощью параметров
- Использует специальный синтаксис – `@decorator`

```
def oops_decorator(func: Callable) -> Callable:
    def wrapper():
        print('I say oops')
        func()
        print('I say oops again')
    return wrapper
```

Академия Аналитиков Авито

Декораторы "без сахара"

```python
from typing import Callable

def oops_decorator(func: Callable) -> Callable:
    def wrapper():
        print('I say oops')
        func()
        print('I say oops again')
    return wrapper

def say_hello_world() -> str:
    print('Hello world')

if __name__ == '__main__':
    oops_decorator(say_hello_world)()
```

Академия Аналитиков Авито

Синтаксис декоратора

```python
from typing import Callable


def oops_decorator(func: Callable) -> Callable:
    def wrapper():
        print('I say oops')
        func()
        print('I say oops again')
    return wrapper


@oops_decorator
def say_hello_world() -> str:
    print('Hello world')


if __name__ == '__main__':
    say_hello_world()
```

Академия Аналитиков Авито

Проброс аргументов

```python
from typing import Callable


# проброс аргументов декоратору
def enclose_with_tags(func: Callable) -> Callable:
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return '<Entity>{}</Entity>'.format(result)
    return wrapper


@enclose_with_tags
def say_hi(name: str) -> str:
    return 'Hi, {}'.format(name)


if __name__ == '__main__':
    print(say_hi('Bob'))
```

Параметризация декораторов

```
def apply_transform(transform):
    def decorator(func):
        def wrapper(username):
            return func(transform(username))
        return wrapper
    return decorator


def custom_upper(text):
    return str.upper(text)

@apply_transform(custom_upper)
def greet(username):
    return f'Hi, {username}'


if __name__ == '__main__':
    print(greet('Fedor'))
```

Академия Аналитиков Авито

Стекинг декораторов

```python
def wrap_with_tag(tag_name: str):
    def decorator(func):
        def wrapper(message: str):
            result = func('Hello')
            return f'<{tag_name}>{result}</{tag_name}>'
        return wrapper

    return decorator


@wrap_with_tag('p')
@wrap_with_tag('strong')
def greet(username):
    return f'Hi, {username}'

if __name__ == '__main__':
    print(greet('Fedor'))
```

Декораторы

- Логирование поведения функции
- Сбор различных метрик (например, время выполнения)
- Валидация входных данных
- Преобразование данных к нужному формату "налету"

Академия
Аналитиков
Авито

ПЕРЕРЫВ

10 минут

Академия
Аналитиков
Авито

ПРАКТИКА

Практика

- Работаем в командах
- Нечётные номера команд решают задачу 1, чётные - 2.
- После решения, отправьте ответ сюда: https://forms.gle/s3qhtVno6dUz9Ahk9
- Примеры кода с лекции: https://github.com/ffix/decorators
- Задачи лежат по ссылке: http://bit.ly/decorators-17-nov-22-tasks

Академия
Аналитиков
Авито

РАЗБОР ПРАКТИКИ

Академия
Аналитиков
Авито

ЕЩЁ НЕМНОГО О ДЕКОРАТОРАХ

Встроенные декораторы

- `property`, `classmethod`, `staticmethod`, `abc.abstractmethod` – управление доступом к атрибутам/методам класса.
- `cached_property` – аналог `property`, но кэширует результаты (доступен с 3.8)
- `lru_cache` – кэширует результаты вызова функции/метода
- `singledispatch` – позволяет задать несколько реализаций функции/метода в зависимости от типа параметров

Полезные ссылки

- https://www.geeksforgeeks.org/decorators-in-python/
- https://www.datacamp.com/community/tutorials/decorators-python
- https://realpython.com/primer-on-python-decorators/
- https://blog.miguelgrinberg.com/post/the-ultimate-guide-to-python-decorators-part-i-function-registration
- https://medium.com/better-programming/decorators-in-python-72a1d578eac4

Итого

Мы узнали, что декораторы:

- широко используется в питоне - как пример, много встроенных в язык
- удобный синтаксис (`@decorator`)
- просто реализовать свои
- можно использовать сразу несколько
- можно параметризировать
- часто спрашивают на собеседованиях, например, реализовать простейший

Академия
Аналитиков
Авито

ВОПРОСЫ?

Академия
Аналитиков
Авито

СПАСИБО!

Пожалуйста, пройдите опрос.

bit.ly/aaa-decorators-17-nov-22