МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 2.12

Декораторы функций в языке Python

По дисциплине «Технологии программирования и алгоритмизация»

Выполнил студент группы ИВТ-б-о-20-1

Погорелов Д.Н. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2021

Цель работы: приобретение навыков по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.

Ход работы:

1. Создал общедоступный репозиторий, клонировал его локальный сервер.

2. Изучил теоретический материал и проработал примеры.

```
def decorator_function(func):
    def wrapper():
        print('Функция-обёртка!')
        print('Оборачиваемая функция: {}'.format(func))
        print('Выполняем обёрнутую функцию...')
        func()
        print('Выходим из обёртки')
    return wrapper


@decorator_function
def hello_world():
    print("Hello, world!")


hello_world()
```

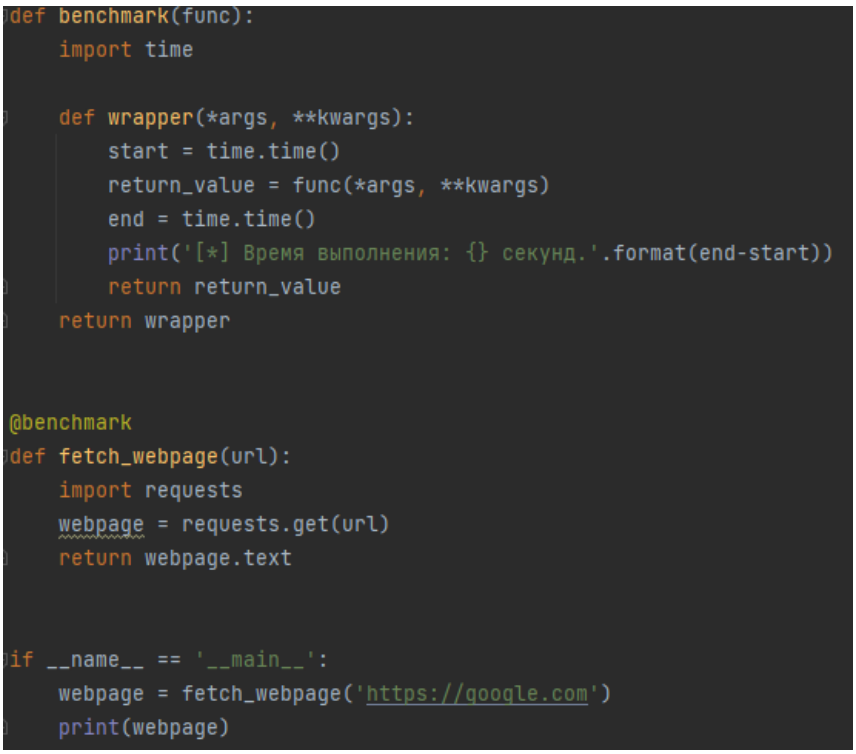
Рисунок 1 – Пример кода с декораторами

```
Функция-обёртка!
Оборачиваемая функция: <function hello
Выполняем обёрнутую функцию...
Hello, world!
Выходим из обёртки

Process finished with exit code 0
```

Рисунок 2 – Результат работы кода

3. Проработал второй пример с использованием декоратора.



```python
def benchmark(func):
    import time

    def wrapper(*args, **kwargs):
        start = time.time()
        return_value = func(*args, **kwargs)
        end = time.time()
        print('[*] Время выполнения: {} секунд.'.format(end-start))
        return return_value
    return wrapper


@benchmark
def fetch_webpage(url):
    import requests
    webpage = requests.get(url)
    return webpage.text


if __name__ == '__main__':
    webpage = fetch_webpage('https://google.com')
    print(webpage)
```

Рисунок 3 – Код второго примера

4. Приступил к выполнению индивидуального задания.

Индивидуальное задание. Вариант 1.

Условие: Объявите функцию с именем get_sq , которая вычисляет площадь прямоугольника по двум параметрам: width и height – ширина и высота прямоугольника и возвращает результат. Определите декоратор для этой функции с именем (внешней функции) func_show , который отображает результат на экране в виде строки (без кавычек): "Площадь прямоугольника: <значение>". Вызовите декорированную функцию get_sq.

1. Объявил внешнюю функцию func_show, в ней вызвал функцию обертку, где обратился к оригинальной функции для получения результата.

2. Объявил декоратор с именем get_sq.

3. Сделал вызов декоративной функции.

```python
def func_show(func):
    def wrapper(width, height):
        result = func(width, height)
        print(f"Площадь прямоугольника: {result}")
        return result

    return wrapper


@func_show
def get_sq(width, height):
    return width * height


if __name__ == '__main__':
    get_sq(width=12, height=7)
```

Рисунок 4 – Код выполненного результата

Рисунок 5 – Результат выполнения кода

Контрольные вопросы:

1. Что такое декоратор?

Декоратор — это функция, которая позволяет обернуть другуюфункцию для расширения её функциональности без непосредственного изменения её кода.

2. Почему функции являются объектами первого класса?

Потому что с ними можно работать как с переменными, могут быть переданы как аргумент процедуры, могут быть возвращены как результат выполнения процедуры, могут быть включены в другие структуры данных.

3. Каково назначение функций высших порядков?

Основной задачей функций высших порядков является возможность принимать в качестве аргументов и возвращать другие функции.

4. Как работают декораторы?

Они берут декорируемую функцию в качестве аргумента и позволяет совершать с ней какие-либо действия до и после того, что сделает эта функция,

не изменяя её.

5. Какова структура декоратора функций?

Функция decorator принимает в качестве аргумента функцию func, внутри функции decorator другая функций wrapper. В конце декоратора происходит возвращение функции wrapper.

6. Самостоятельно изучить как можно передать параметры декоратору, а не декорируемой функции?

Достаточно обернуть функцию декоратор в другую функцию, которая будет принимать аргументы. И сделать вывод функций wrapper и decorator.

Вывод: в ходе выполнения лабораторной работы приобрел навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.