

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

Кафедра инфокоммуникаций

Отчет по лабораторной работе №7
Декораторы функций в языке Python
По дисциплине «Технологии программирования и алгоритмизация»

Выполнил студент группы ИВТ-б-о-20-1

Галяс Д. И. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р. А. _____

(подпись)

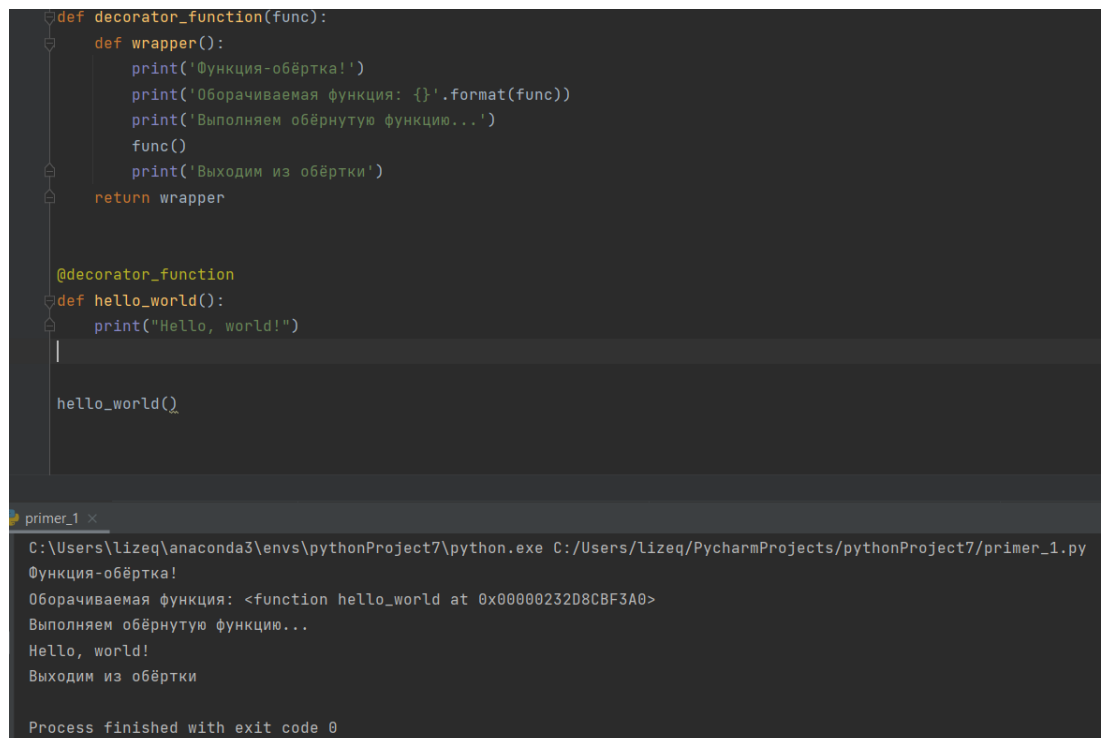
Ставрополь 2021

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

Ход работы: <https://github.com/DIMITRY-GALYAS1/Laba-2.12.git>

Ссылка на репозиторий:

1. Создал новый репозиторий на github, после клонировал его и создал в папке репозитория новый проект PyCharm.
2. Выполнил пример.



```
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()
```

primer_1 x

C:\Users\lizeq\anaconda3\envs\pythonProject7\python.exe C:/Users/lizeq/PycharmProjects/pythonProject7/primer_1.py

Функция-обёртка!

Оборачиваемая функция: <function hello_world at 0x00000232D8CBF3A0>

Выполняем обёрнутую функцию...

Hello, world!

Выходим из обёртки

Process finished with exit code 0

Рисунок 1. Пример 1

3. Выполнил второй пример.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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 2. Пример 2

4. Выполнил индивидуальное задание.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
На вход программы поступает строка из целых чисел, записанных через пробел. Напишите
функцию get_list, которая преобразовывает эту строку в список из целых чисел и
возвращает его. Определите декоратор для этой функции, который сортирует список чисел,
полученный из вызываемой в нем функции. Результат сортировки должен возвращаться при
вызове декоратора. Вызовите декорированную функцию get_list и отобразите полученный
отсортированный список на экране.
"""

def decorator_function(func):
    def sorting(z):
        return sorted(func(z))
    return sorting

@decorator_function
def get_list(z):
    return [int(i) for i in z.split()]

if __name__ == '__main__':
    print(get_list(input('Введите числа через пробел: ')))
```

Рисунок 3. Код индивидуального задания

```
C:\Users\lizeq\anaconda3\envs\pythonProject7\python.exe C:/Users/lizeq/PycharmProjects/pythonProject7/individual_1.py
Введите числа через пробел: 1 4 0 9 2
[1, 2, 4, 8, 9]
Process finished with exit code 0
```

Рисунок 4. Выполнение индивидуального задания

Контрольные вопросы:

1. Что такое декоратор?

Декоратор — это функция, которая позволяет обернуть другую функцию 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 позволяют 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 функция `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.