

**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 (2.12)
Декораторы функций в языке Python.**

по дисциплине «Технологии программирования и алгоритмизации»

Выполнила студентка группы ИВТ-б-о-20-1

Новикова В.С. « » _____ 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 3х.

Ход работы:

Ссылка на репозиторий: https://github.com/Valentina1502/LABA_7

Пример 1. (рис. 1).

Пример использования декоратора.

Создаём декоратор, измеряющий время выполнения функции. Далее мы используем его на функции, которая делает GET-запрос к главной странице Google. Чтобы измерить скорость, мы сначала сохраняем время перед выполнением обернутой функции, выполняем её, снова сохраняем текущее время и вычитаем из него начальное.

Код:

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

""" Модифицирование декоратора, измеряющего
время выполнения функции (используем аргументы
и возвращаем значения) """

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://pinterest.ru')
    print(webpage)
```

```
pr3_modify x
C:\Users\Valentina\AppData\Local\Programs\Python\Python38\python.exe C:/U
[*] Время выполнения: 2.8574891090393066 секунд
<!DOCTYPE html><html class="en" lang="en"><head><meta charset="utf-8"/><m

Process finished with exit code 0
```

Рисунок 1 – Пример 1

Задание 1. Вариант 11 (1) (рис. 2):

Составить программу с использованием замыканий для решения задачи.

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:». Вызовите декорированную функцию `get_sq`.

Код:

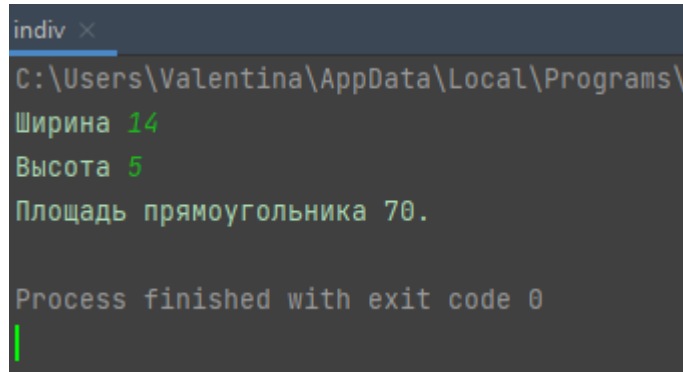
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
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
'''

def func_show(func):
    def wrapper(width, height):
        sq = func(width, height)
        print("Площадь прямоугольника {}".format(sq))
        return sq
    return wrapper

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

if __name__ == '__main__':
    width = int(input("Ширина "))
    height = int(input("Высота "))
    get_sq(width, height)
```



```
indiv x
C:\Users\Valentina\AppData\Local\Programs\I
Ширина 14
Высота 5
Площадь прямоугольника 70.

Process finished with exit code 0
```

Рисунок 2 – Задание 1

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?

Тот факт, что всё является объектами, открывает перед нами множество возможностей. Мы можем сохранять функции в переменные, передавать их в качестве аргументов и возвращать из других функций. Можно даже определить одну функцию внутри другой. Иными словами, функции — это объекты первого класса.

Объектами первого класса в контексте конкретного языка программирования называются элементы, с которыми можно делать всё то же, что и с любым другим объектом: передавать как параметр, возвращать из функции и присваивать переменной

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 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: при выполнении заданий были приобретены навыки по работе с декораторами при написании программ с помощью языка программирования Python.