Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Декораторы функций в языке Python»

ОТЧЕТ
по лабораторной работе №15
дисциплины
«Основы программной инженерии»

Выполнил:
Борсуков Владислав Олегович
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия», направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:

_____
(подпись)

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2022 г.

Пример:

```python
#!/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)
```

```
benchmark()
Run:  Example
C:\Users\Borsukov\Desktop\LR15\PyCharm\venv\Scripts\python.exe C:\Users\Borsukov\Desktop\LR15\PyCharm\Examp
[*] Время выполнения: 0.9572186470031738 секунд.
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ru"><head><meta content="&#105
var f=this||self;var h,k=[];function l(a){for(var b;a&&(!a.getAttribute||!(b=a.getAttribute("eid")));)a=a.p
function n(a,b,c,d,g){var e="";c||-1!==b.search("&ei=")||(e="&ei="+l(d),-1===b.search("&lei=")&&(d=m(d))&&(
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.target){var c=a.getAttribute("
</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;overflow-y:scroll}#gog{padding:3px
var h=this||self;var k,l=null!=(k=h.mei)?k:1,n,p=null!=(n=h.sdo)?n:!0,q=0,r,t=google.erd,v=t.jsr;google.ml=
a.fileName;g&&(0<g.indexOf("-extension:/")&&(e=3),c+="&script="+b(g),f&&g===window.location.href&&(f=docume
if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}
```

Рисунок 1 – Код и результат работы примера

Индивидуальное задание: 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.

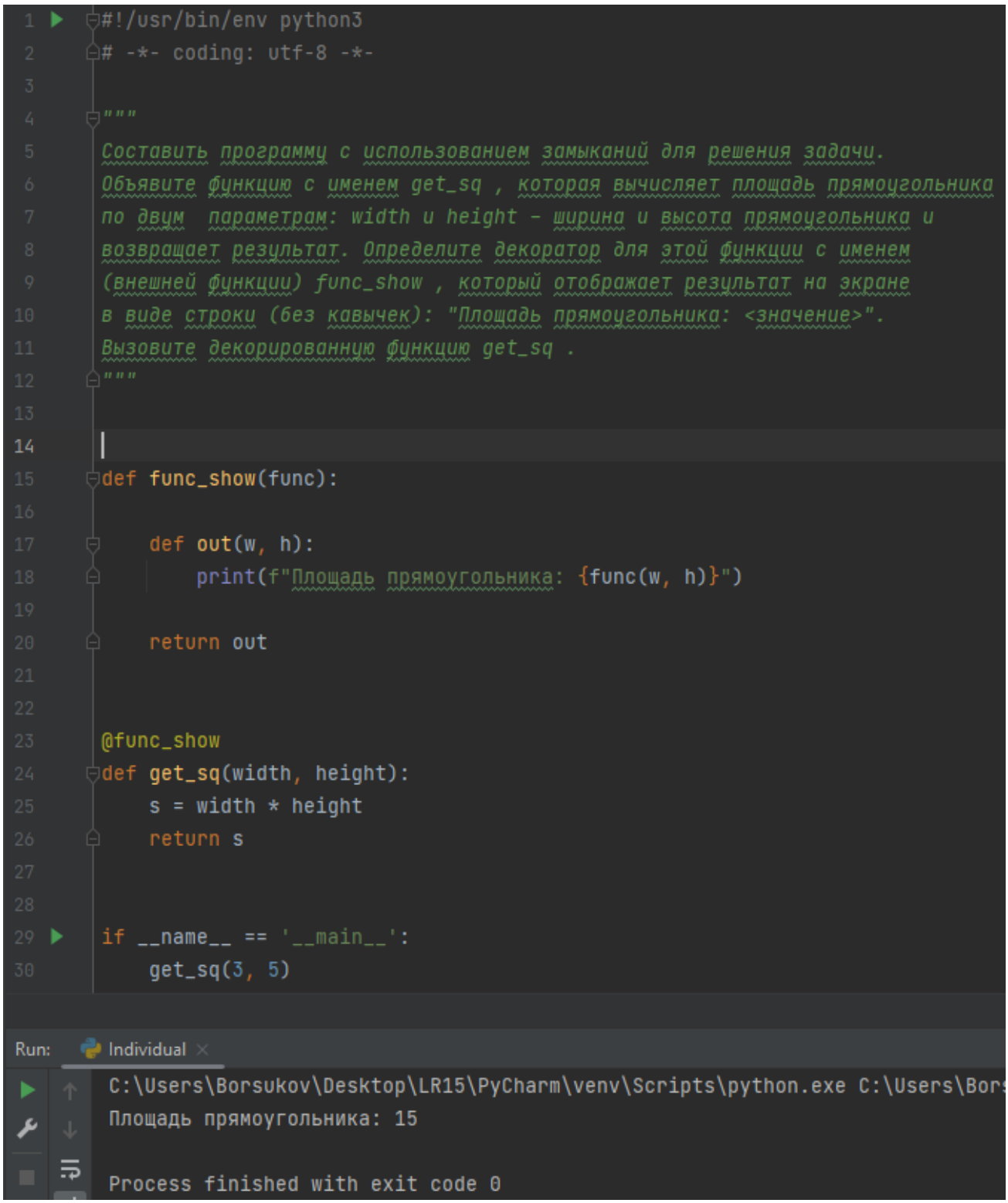

Рисунок 2 – Код и результат программы индивидуального задания

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?

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. Из определения в Википедии: 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?

Функции высших порядков — это такие функции, которые могут принимать в качестве аргументов и возвращать другие функции.

Если вы знакомы с основами высшей математики, то вы уже знаете некоторые математические функции высших порядков порядка вроде дифференциального оператора . Он принимает на входе функцию и возвращает другую функцию, производную от исходной. Функции высших порядков в программировании работают точно так же — они либо принимают функцию(и) на входе и/или возвращают функцию(и).

4. Как работают декораторы?

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

5. Какова структура декоратора функций?

```
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
```

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

Мы также можем создавать декораторы, которые принимают аргументы. Посмотрим на пример:

```
def benchmark(iters):
    def actual_decorator(func):
        import time

        def wrapper(*args, **kwargs):
            total = 0
            for i in range(iters):
                start = time.time()
                return_value = func(*args, **kwargs)
                end = time.time()
                total = total + (end-start)
            print('[*] Среднее время выполнения: {} секунд.'.format(total/iters))
            return return_value

        return wrapper
    return actual_decorator


@benchmark(iters=10)
def fetch_webpage(url):
    import requests
    webpage = requests.get(url)
    return webpage.text
```

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

Здесь мы модифицировали наш старый декоратор таким образом,чтобы он выполнял декорируемую функцию iters раз, а затем выводил среднее время выполнения. Однако чтобы добиться этого, пришлось воспользоваться природой функций в Python.

Функция benchmark() на первый взгляд может показаться декоратором,но на самом деле таковым не является. Это обычная функция, которая принимает аргумент iters, а затем возвращает декоратор. В свою очередь, он декорирует функцию fetch_webpage(). Поэтому мы использовали не выражение @benchmark, а @benchmark(iters=10) — это означает, что тут вызывается функция benchmark() (функция со скобками после неё обозначаетвызов функции), после чего она возвращает сам декоратор.

Да, это может быть действительно сложно уместить в голове, поэтомудержите правило:

Декоратор принимает функцию в качестве аргумента и возвращаетфункцию.

В нашем примере benchmark() не удовлетворяет этому условию, так какона не принимает функцию в качестве аргумента. В то время как функция actual_decorator(), которая возвращается benchmark(), является декоратором.