

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ  
ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Дисциплина: «Языки программирования»

Отчет по лабораторной работе №10

**Декораторы функции в языке Python**

Выполнил студент группы ИТС-б-о-21-1

Тимофеева Марина Сергеевна

«\_\_»\_\_\_\_\_20\_\_г.

Подпись студента\_\_\_\_\_

Проверил: Доцент, к.т.н, доцент

кафедры инфокоммуникаций

Воронкин Роман Александрович

---

(подпись)

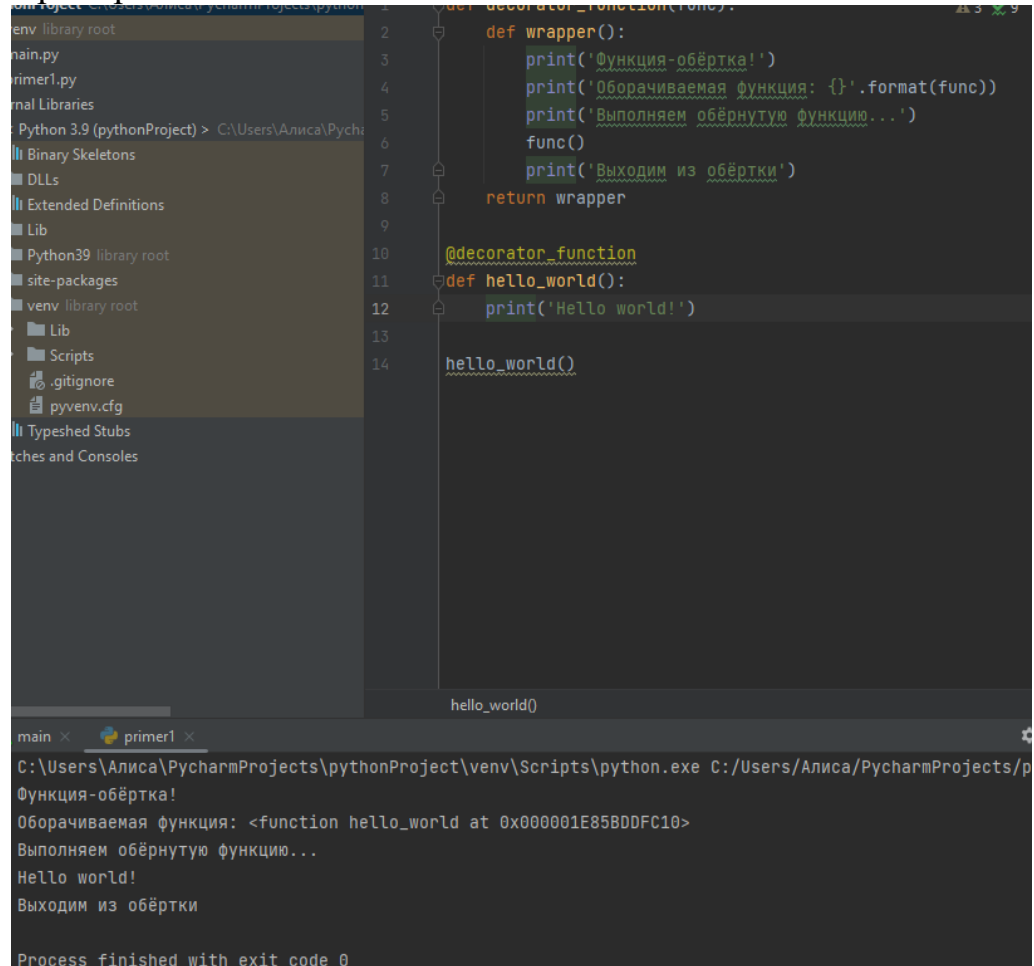
Ставрополь, 2022

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

Ссылка на репозиторий - <https://github.com/AlisaL1sa/primer1.2>

**Ход работы:**

**Пример №1:**



```
1 def decorator_function(func):
2     def wrapper():
3         print('Функция-обёртка!')
4         print('Оборачиваемая функция: {}'.format(func))
5         print('Выполняем обёрнутую функцию...')
6         func()
7         print('Выходим из обёртки')
8     return wrapper
9
10 @decorator_function
11 def hello_world():
12     print('Hello world!')
13
14 hello_world()
```

main × primer1 ×

C:\Users\Алиса\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Алиса/PycharmProjects/p

Функция-обёртка!  
Оборачиваемая функция: <function hello\_world at 0x000001E85B0D0FC10>  
Выполняем обёрнутую функцию...  
Hello world!  
Выходим из обёртки

Process finished with exit code 0

Рисунок 1. Код и результат выполнения программой примера 1

Для следующих примеров, чтобы установить библиотеку requests, вводим в строку `pip install requests`.

### Пример №2:

The screenshot shows the PyCharm IDE interface. On the left is the Project Explorer showing the file structure of a project named 'pythonProject'. The main editor displays a Python script with the following code:

```

1 def benchmark(func):
2     import time
3
4     def wrapper():
5         start = time.time()
6         func()
7         end = time.time()
8         print('[*] Время выполнения: {} секунд.'.format(end-start))
9     return wrapper
10
11 @benchmark
12 def fetch_webpage():
13     import requests
14     webpage = requests.get('https://google.com')
15
16     fetch_webpage()

```

At the bottom, the Run tool window shows the output of the script:

```

main x primer2 x
C:\Users\Алиса\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Алиса\PycharmProjects\pythonProject\primer2.py
[*] Время выполнения: 0.4358799457550049 секунд.
Process finished with exit code 0

```

Рисунок 2. Код и результат выполнения программой примера 2

### Пример №3:

The screenshot shows the PyCharm IDE interface. On the left is the Project Explorer showing a project named 'pythonProject' located at 'C:\Users\Алиса\PycharmProjects\pythonProject'. The file explorer lists several files: 'main.py', 'primer1.py', 'primer2.py', 'primer3.py', 'pyproject.toml', 'venv\library root', 'site-packages', 'venv\Scripts', '.gitignore', and 'pyvenv.cfg'. The main editor window displays the code for 'primer3.py'. The code defines a 'benchmark' decorator that uses 'time.time()' to measure execution time. It wraps a 'fetch\_webpage' function that makes a GET request to 'https://google.com/'. The output at the bottom shows the execution time as approximately 0.427 seconds.

```

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

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

```

Output:

```

[*] Время выполнения: 0.42742308033569336 секунд.

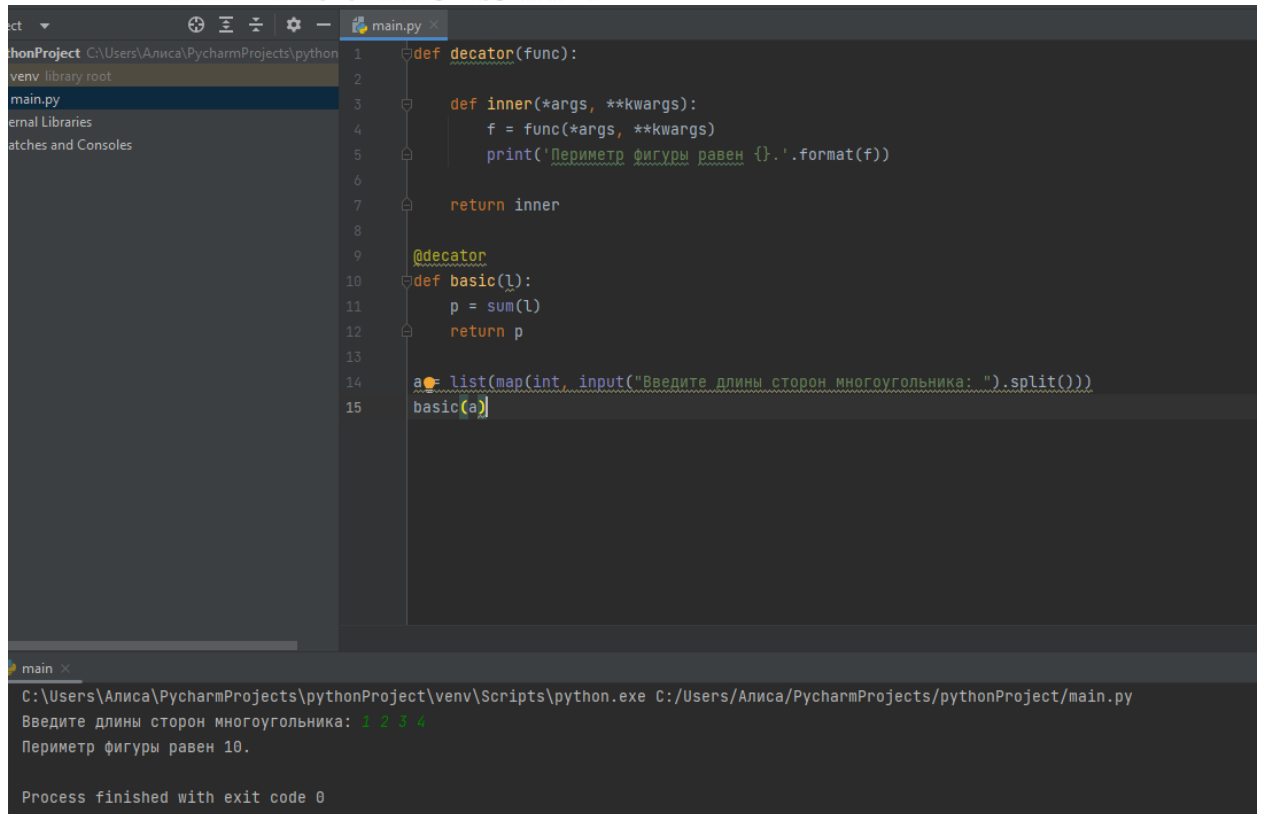
```

Рисунок 3. Код и результат выполнения программой примера 3

## Вариант 17(7)

### Индивидуальное задание:

7. Объявите функцию, которая вычисляет периметр многоугольника и возвращает вычисленное значение. Длины сторон многоугольника передаются в виде коллекции (списка или кортежа). Определите декоратор для этой функции, который выводит на экран сообщение: «Периметр фигуры равен = <число>». Примените декоратор к функции и вызовите декорированную функцию.



```
1 def decator(func):
2
3     def inner(*args, **kwargs):
4         f = func(*args, **kwargs)
5         print('Периметр фигуры равен {}'.format(f))
6
7     return inner
8
9 @decator
10 def basic(l):
11     p = sum(l)
12     return p
13
14 a = list(map(int, input("Введите длины сторон многоугольника: ").split()))
15 basic(a)
```

main x

C:\Users\Алиса\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Алиса/PycharmProjects/pythonProject/main.py

Введите длины сторон многоугольника: 1 2 3 4

Периметр фигуры равен 10.

Process finished with exit code 0

Рисунок 4. Код и результат выполнения программой 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?

Мы можем сохранять функции в переменные, передавать их в качестве аргументов и возвращать из других функций. Можно даже определить одну функцию внутри другой.

3. Каково назначение функций высших порядков?

Это такие функции, которые могут принимать в качестве аргументов и возвращать другие функции.

4. Как работают декораторы?

Из определения: декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода. На примере кода:

```
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
```

5. Какова структура декоратора функций?

Функция декоратор, содержащая в себе декорируемую функцию.

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

Функциональный вызов `func(...)`, который вернет что-то тоже вызываемое или имя функции, или переменная или экземпляр класса

**Вывод:** приобрела навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.