

**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

Романов Платон Дмитриевич

«__»_____20__г.

Подпись студента_____

Проверил: Доцент, к.т.н, доцент

кафедры инфокоммуникаций

Воронкин Роман Александрович

(подпись)

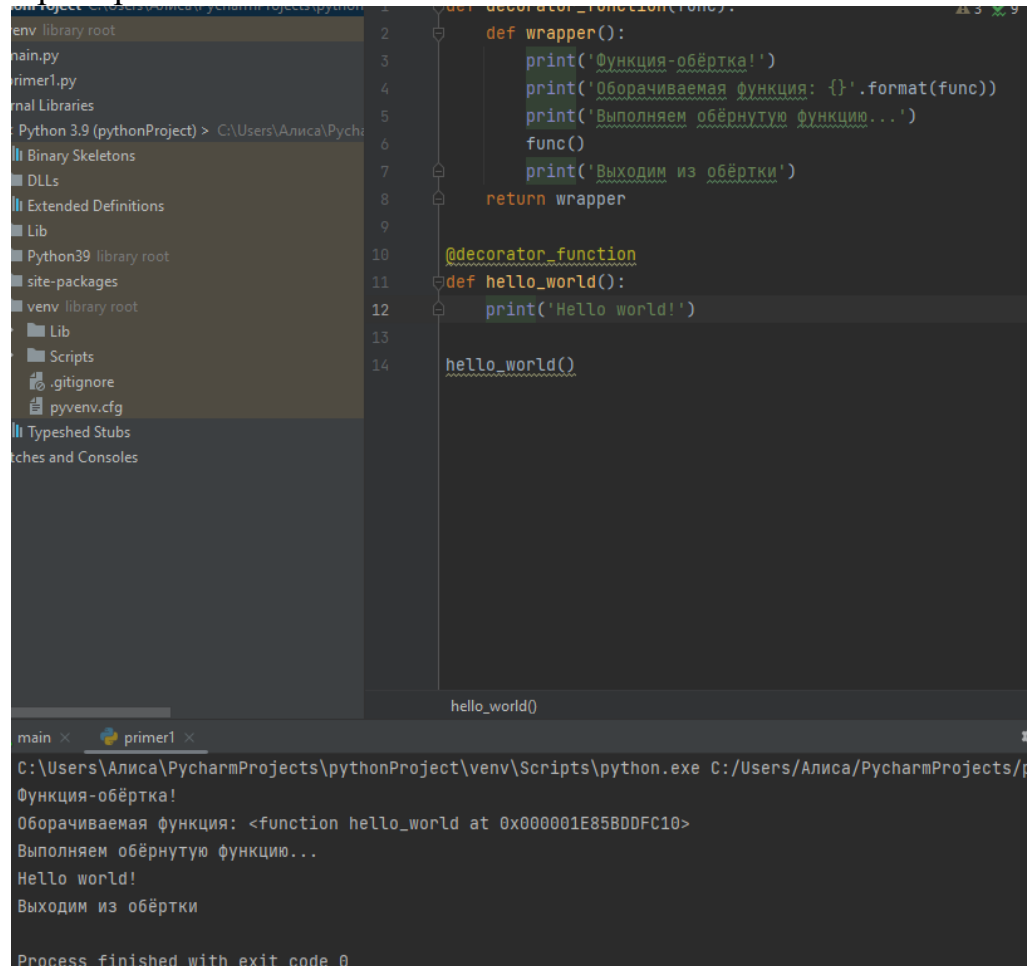
Ставрополь, 2022

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/lesnaya1shelupon/3sem2laba>

Ход работы:

Пример №1:



```
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()
```

main x primer1 x
C:\Users\Алиса\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Алиса/PycharmProjects/p
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x000001E85BDDFC10>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки
Process finished with exit code 0

Рисунок 1. Код и результат выполнения

Установка библиотеки requests:

В версиях Python выше 3, уже установлен pip (pip — система управления пакетами, которая используется для установки и управления программными пакетами, написанными на Python.). Поэтому, вводим в строку `pip install requests`.

```
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()
```

Рисунок 2. Код и результат выполнения

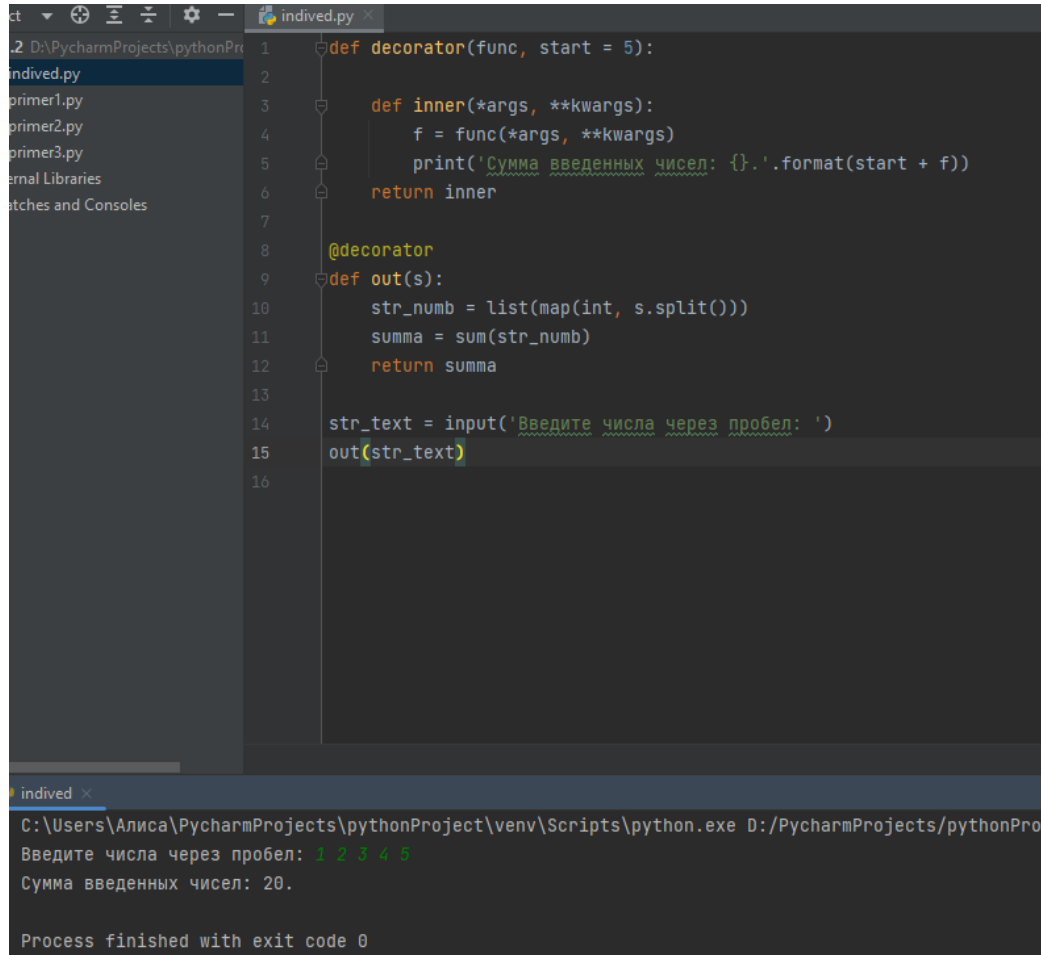
[illegible]

Рисунок 3. Код и результат выполнения

Вариант 5

Индивидуальное задание:

5. Вводится строка целых чисел через пробел. Напишите функцию, которая преобразовывает эту строку в список чисел и возвращает их сумму. Определите декоратор для этой функции, который имеет один параметр `start` – начальное значение суммы. Примените декоратор со значением `start=5` к функции и вызовите декорированную функцию. Результат отобразите на экране.



```
1 def decorator(func, start = 5):
2
3     def inner(*args, **kwargs):
4         f = func(*args, **kwargs)
5         print('Сумма введенных чисел: {}'.format(start + f))
6         return inner
7
8     @decorator
9     def out(s):
10         str_numb = list(map(int, s.split()))
11         summa = sum(str_numb)
12         return summa
13
14     str_text = input('Введите числа через пробел: ')
15     out(str_text)
16
```

indived

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

Введите числа через пробел: 1 2 3 4 5

Сумма введенных чисел: 20.

Process finished with exit code 0

Рисунок 4. Код и результат выполнения

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 обернутую 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: приобрел навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.