

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.12
дисциплины «Программирование на Python»

Выполнил:
Кенесбаев Хилол Куат улы
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Декораторы функций в языке Python

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

Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.

2. Проработал примеры лабораторной работы:

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe C:\Users\HP-PC\PycharmProjects\lab-15\Lab2.12\prog\prim1.py
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x000001D0293B4B80>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Process finished with exit code 0
```

Рисунок 1 Вывод примера 1

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe C:\Users\HP-PC\PycharmProjects\lab-15\Lab2.12\prog\prim2.py
[*] Время выполнения: 0.99349045753479 секунд.
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ru"><head><meta content

Process finished with exit code 0
```

Рисунок 2 Вывод примера 2

Выполнил индивидуальное задание: 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.

Код программы:

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

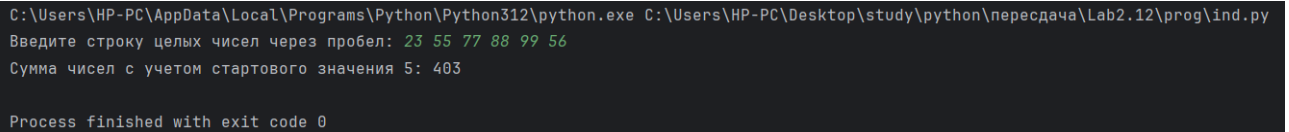
def sum_of_numbers(numbers):
    # Функция для подсчета суммы чисел
    nums = [int(num) for num in numbers.split()]
    total_sum = sum(nums)
    return total_sum

# Ввод строки целых чисел
numbers_string = input("Введите строку целых чисел через пробел: ")
```

```
# Получаем сумму чисел с учетом декоратора
```

```
result = sum_of_numbers(numbers_string)
```

```
print("Сумма чисел с учетом стартового значения 5:", result)
```



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe C:\Users\HP-PC\Desktop\study\python\пересдача\Lab2.12\prog\ind.py
Введите строку целых чисел через пробел: 23 55 77 88 99 56
Сумма чисел с учетом стартового значения 5: 403
Process finished with exit code 0
```

Рисунок 3 Вывод программы ind.py

Ответы на контрольные вопросы:

1. Что такое декоратор?

Декоратор – это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода. Декораторы можно рассматривать как практику метапрограммирования, когда программы могут работать с другими

программами как со своими данными.

2. Почему функции являются объектами 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. С функцией все это делать можно, поэтому ее и можно назвать объектом первого класса.

3. Каково назначение функций высших порядков?

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

4. Как работают декораторы?

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
```

Здесь `decorator_function()` является функцией-декоратором. Она является функцией высшего порядка, так как принимает функцию в качестве аргумента, а также возвращает функцию. Внутри `decorator_function()` определена другая функция, которая обёртывает функцию-аргумент и затем изменяет её поведение. Декоратор возвращает эту обёртку.

Перед функцией остается прописать `@decorator_function`.

Однако выражение с `@` является всего лишь синтаксическим сахаром для `hello_world = decorator_function(hello_world)`.

5. Какова структура декоратора функций?

```

def decorator(func):
    def wrapper(*args, **kwargs):
        # Код до вызова целевой функции
        result = func(*args, **kwargs)
        # Вызов целевой функции
        # Код после вызова целевой функции
        return result
    return wrapper

```

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

В Python можно передавать параметры декоратору, добавляя еще один уровень вложенности.

Например:

```

def decorator_with_parameters(param1, param2):
    def actual_decorator(func):
        def wrapper(*args, **kwargs):
            print(f"Decorator parameters: {param1 }, {param2}")
            result = func(*args, **kwargs)
            return result
        return wrapper
    return actual_decorator

```

Вызов декоратора с параметрами будет выглядеть так:

```
@decorator_with_parameters(p1, p2)
```

Вывод: в результате выполнения работы были приобретены навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.