МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования
Отчет по лабораторной работе №2.12

Декораторы функций в языке Python

Выполнил студент группы
ИТС-б-о-21-1
Гуляницкий А.Е. « »_____20__г.
Подпись студента_____
Работа защищена « »_____20__г.
Проверил к.т.н., доцент
Кафедры инфокоммуникаций
Воронкин Р.А.
_____
(подпись)

Ставрополь 2021

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/Alexander-its/2.12

Ход работы:

Пример 1. Создаём декоратор

```
File  Edit  Shell  Debug  Options  Window  Help
      Python 3.10.4 (tags/v3.10.4:9d38
      AMD64)] on win32
      Type "help", "copyright", "credi
>>> def hello_world():
...     print('Hello world!')
...
...
>>> type(hello_world)
      <class 'function'>
>>> class Hello:
...     pass
...
>>> type(Hello)
      <class 'type'>
>>> type(10)
      <class 'int'>
>>>
```

```
>> def decorator_function(func):
..     def wrapper():
..         print('Функция-обертка!')
..         print('Оборачиваемая функция: {}'.format(func))
..         print('Выполняем обернутую функцию...')
..         func()
..         print('ыходим из обертки')
..     return wrapper
..
>> @decorator_function
.. def hello_world():
..     print('hello_world!')
..
..
>> hello_world()
Функция-обертка!
Оборачиваемая функция: <function hello_world at 0x000002593F80FAC0>
Выполняем обернутую функцию...
hello_world!
ыходим из обертки
>>
```

Рисунок 1. Результат выполнения программы

Индивидуальное задание. 8 вариант.

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

if __name__ == '__main__':
    def print_info(func):
        def f(*args, **kwargs):
            result = func(*args, **kwargs)
            print(f'Площадь круга равна = {result:1.2f}')
        return f
    @print_info
    def area(x):
        return 3.1415926 * x * x

    r = float(input('Введите радиус'))
    area(r)
```

```
== RESTART: C:\Users\GO_PB\Desktop
Введите радиус78
Площадь круга равна = 19113.45
```

Рисунок 2. Результат выполнения программы

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?

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.

В Python всё является объектом, а не только объекты, которые вы создаёте из классов. Это значит, что в Python всё это — объекты:

- числа;
- строки;
- классы;
- функции.

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?

Декоратор – это функция, которая позволяет обернуть другую функцию с помощью символа «@»

5. Какова структура декоратора функций?

Сначала записывается функция – декоратор. Потом идет его вызов с помощью @, а затем основная функция, которую оборачивает декоратор.

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

Используя замыкание функций.

Вывод: В ходе лабораторной работы я приобрел навыки по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python .