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
# Декораторы функций в языке Python

Выполнил:
Боженко Александр Иванович
2 курс, группа ИТС-б-о-21-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»,
направленность (профиль) «Инфокоммуникационные системы и сети»,
очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А, канд. техн. наук, доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой_____ Дата защиты _____

Ставрополь, 2022 г.

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

**Краткий конспект**

Декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода. Вот почему декораторы можно рассматривать как практику метапрограммирования, когда программы могут работать с другими программами как со своими данными. Чтобы понять, как это работает, сначала разберёмся в работе функций в Python.

**Функции как объекты первого класса**

В Python всё является объектом, а не только объекты, которые вы создаёте из классов. В этом смысле он (Python) полностью соответствует идеям объектно-ориентированного программирования. Это значит, что в Python всё это — объекты:

- числа;
- строки;
- классы (да, даже классы!);
- функции (то, что нас интересует).

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.

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.

И тут в дело вступает функциональное программирование, а вместе с ним — декораторы.

**Функциональное программирование — функции высших порядков**

В Python используются некоторые концепции из функциональных языков вроде Haskell и OCaml. Пропустим формальное определение функционального языка и перейдём к двум его характеристикам, свойственным Python:

- функции являются объектами первого класса;
- следовательно, язык поддерживает функции высших порядков.

Функциональному программированию присущи и другие свойства вроде отсутствия побочных эффектов, но мы здесь не за этим. Лучше сконцентрируемся на другом — функциях высших порядков.

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

Из основ высшей математики, 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).

**Пример 1.**

```
>>> def hello_world():
...     print('Hello world!')
...
>>> type(hello_world)
<class 'function'>
>>> class Hello:
...     pass
...
>>> type(Hello)
<class 'type'>
>>> type(10)
<class 'int'>
>>> hello = hello_world
>>> hello()
Hello world!
```

Рисунок 1. Результат выполнения программы

**Пример 2.**

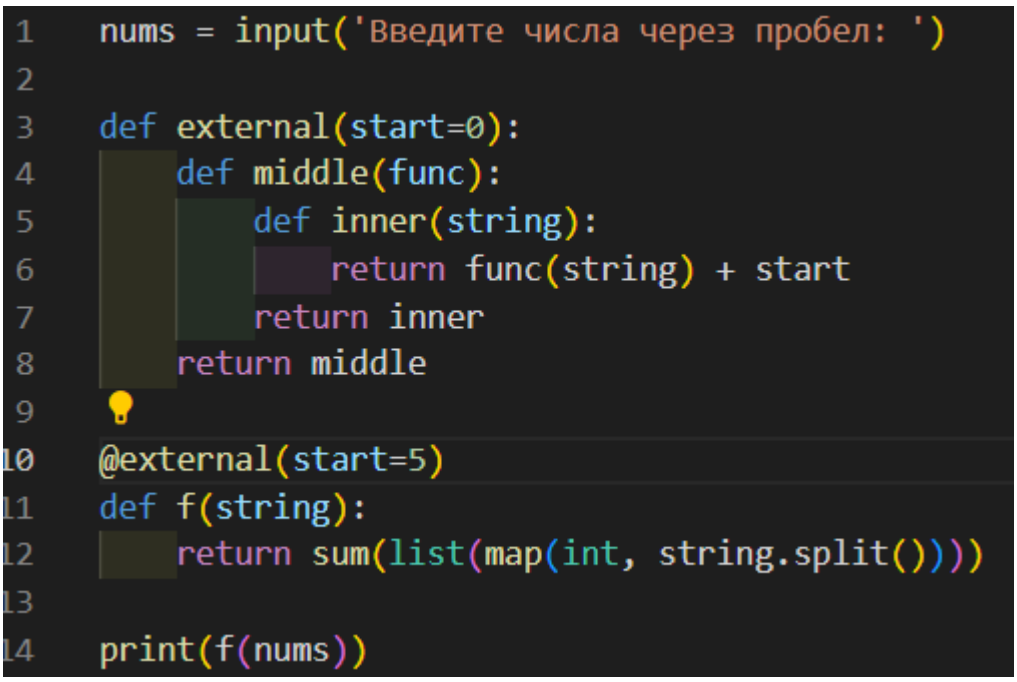
```
>>> def decorator_function(func):
...   def wrapper():
...    print('Функция-обёртка!')
...    print('Оборачиваемая функция: {}'.format(func))
...    print('Выполняем обёрнутую функцию...')
...    func()
...    print('Выходим из обёртки')
...   return wrapper
...
...
>>> @decorator_function
... def hello_world():
...   print('Hello world!')
...
>>> hello_world()
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x0000019900CAFAC0>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки
```

Рисунок 2. Результат выполнения программы

**Ход работы**

Задание 6

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.



```python
nums = input('Введите числа через пробел: ')

def external(start=0):
    def middle(func):
        def inner(string):
            return func(string) + start
        return inner
    return middle

@external(start=5)
def f(string):
    return sum(list(map(int, string.split())))

print(f(nums))
```

Рис 1. Код программы

```
Введите числа через пробел: 5 6
16
```

Рис 2. Результат программы

**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. Вот почему 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.

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 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
```

Здесь decorator_function() является функцией-декоратором. Как вы могли заметить, она является функцией высшего порядка, так как принимает функцию в качестве аргумента, а также возвращает функцию. Внутри decorator_function() мы определили другую функцию, обёртку, так сказать, которая обёртывает функцию-аргумент и затем изменяет её поведение. Декоратор возвращает эту обёртку

**Вывод:** в ходе выполнения лабораторной работы были приобретение навыков по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.