МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №2.12 по дисциплине основы программной инженерии

Выполнила:
Лобанова Елизавета Евгеньевна,
2 курс, группа ПИЖ-б-о-20-1

Проверил:
Доцент кафедры прикладной математики и компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2021 г.

Ход работы

```python
def higher_order(func):
    print('Получена функция {} в качестве аргумента'.format(func))
    func()
    return func


def wrapper_function():
    def hello_world():
        print('Hello world!')
    hello_world()


def hello_world():
    print('Hello world!')


if __name__ == "__main__":
    print(type(hello_world))


    class Hello:

        pass
```

Пример 1

```
<class 'function'>
<class 'type'>
<class 'int'>
Hello world!
Hello world!
Получена функция <function hello_world at 0x00000202D2D3C310> в качестве аргумента
Hello world!
<function hello_world at 0x00000202D2D3C310>

Process finished with exit code 0
```

Результат 1 программы (1)

```python
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!')


if __name__ == "__main__":
    hello_world()
```

Пример 2

```
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x000002A604A2C3A0>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Process finished with exit code 0
```

Результат 2 программы (1)

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

def dec_fun(func):
    def sorting(z):
        return sorted(func(z))
    return sorting


@dec_fun
def get_list(z):
    return [int(i) for i in z.split()]


if __name__ == '__main__':
    print(get_list(input('Введите числа через пробел: ')))
```

Индивидуальное задание 1

```
Введите числа через пробел: -90 54 23 -7 25 0
[-90, -7, 0, 23, 25, 54]

Process finished with exit code 0
```

Результат индивидуального задания 1 (1)

Ответы на 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?**
   «Функции первого класса» (FCF) – это функции, которые рассматриваются как так называемые «первоклассные граждане» (FCC). FCC на языке программирования – это объекты (здесь очень свободно используется термин «объекты»), которые:
   • Может использоваться как параметры
   • Может использоваться как возвращаемое значение
   • Может быть присвоен переменным
   • Может храниться в структурах данных, таких как хеш-таблицы, списки, …
3. **Каково назначение функций высших порядков?**
   В языках, где функции можно принимать и передавать в качестве значений, функции называются гражданами первого сорта (first-class citizen). А функции, которые принимают в качестве аргументов другие функции и/или возвращают функции в качестве результата, принято называть функциями высшего порядка (или же функциями высших порядков, ФВП, high-order functions). Таким образов функции высших порядков нужны для расширения функционала других функций

4. **Как работают декораторы?**
   выражение @decorator_function вызывает decorator_function() с hello_world в качестве аргумента и присваивает имени hello_world возвращаемую функцию.
5. **Какова структура декоратора функций?**

```
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)
```

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

```
import functools


def decoration(*args):
    def dec(func):
        @functools.wraps(func)
        def decor():
            func()
            print(*args)
        return decor
    return dec


@decoration('This is *args')
def func_ex():
    print('Look at that')

if __name__ == '__main__':
    func_ex()
```

```
Look at that
This is *args

Process finished with exit code 0
```