МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №15 по дисциплине основы программной инженерии

Выполнила:
Емельянова Яна
Александровна, 2 курс,
группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

1. Декораторы функций в языке Python

Примеры из методических указаний

e1.py e2.py

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


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


    print(type(Hello))
    print(type(10))
    hello = hello_world
    hello()
    wrapper_function()
    print(higher_order(hello_world))

```

```
<class 'function'>
<class 'type'>
<class 'int'>
Hello world!
Hello world!
Получена функция <function hello_world at 0x000001947735D1F0> в качестве аргумента
Hello world!
<function hello_world at 0x000001947735D1F0>

Process finished with exit code 0
```

Как работают декораторы

e1.py e2.py

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


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()
```

```
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x0000022458090310>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Process finished with exit code 0
```

e1.py e2.py e3.py

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


def benchmark(func):
    import time

    def wrapper():
        start = time.time()
        func()
        end = time.time()
        print('[*] Время выполнения: {} секунд.'.format(end - start))

    return wrapper


@benchmark
def fetch_webpage():
    import requests
    webpage = requests.get('https://google.com')


if __name__ == "__main__":
    fetch_webpage()

```

```
[*] Время выполнения: 1.2031803131103516 секунд.

Process finished with exit code 0
```

Используем аргументы и возвращаем значения

e1.py e2.py e3.py e4.py

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


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


if __name__ == "__main__":
    webpage = fetch_webpage('https://google.com')
    print(webpage)

```

```
[*] Время выполнения: 1.898144245147705 секунд.
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" la
var f=this||self;var h,k=[];function l(a){for(var b;a&&(!a.getAttribute||
function n(a,b,c,d,g){var e="";c||-1!==b.search("&ei=")||(e="&ei="+l(d),-
google.y={};google.sy=[];google.x=function(a,b){if(a)var c=a.id;else{do c
document.documentElement.addEventListener("submit",function(b){var a;if(a
</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;
var f=this||self;var g,h,k=null!==(g=f.mei)&&void 0!==g?g:1,l=null!==(h=f
```

1.1 Индивидуальное задание 1 (рис 1-3)

Вариант 9

ind_zad.py

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

def change_punct(chars=" !?"):
    def wrapper(func):
        def wrapped(*args):
            print(f"Входим в обёртку с символами {chars}")
            print(f"Выполняем обёртнутую функцию: {func}")
            string = args[0]
            for char in chars:
                string = string.replace(char, '-')
            print(string)
            return_str = func(string)
            print("Выходим из обёртки")
            return return_str

        return wrapped

    return wrapper


@change_punct(chars="?!:;,. ")
def rep_str(string):
    t = {'ё': 'yo', 'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd', 'е': 'e',
         'ж': 'zh',
         'з': 'z', 'и': 'i', 'й': 'y', 'к': 'k', 'л': 'l', 'м': 'm', 'н': 'n',
         'о': 'o', 'п': 'p',
         'р': 'r', 'с': 's', 'т': 't', 'у': 'u', 'ф': 'f', 'х': 'h', 'ц': 'c',
         'ч': 'ch', 'ш': 'sh',
         'щ': 'shch', 'ъ': '', 'ы': 'y', 'ь': '', 'э': 'e', 'ю': 'yu',
         'я': 'ya'}
    for k, v in t.items():
        string = string.replace(k, v)
    return string


if __name__ == "__main__":
    string = input("Введите строку для замены: ")
    print(f"Изменённая строка: {rep_str(string.lower())}")
```

Рисунок 1 – Код программы

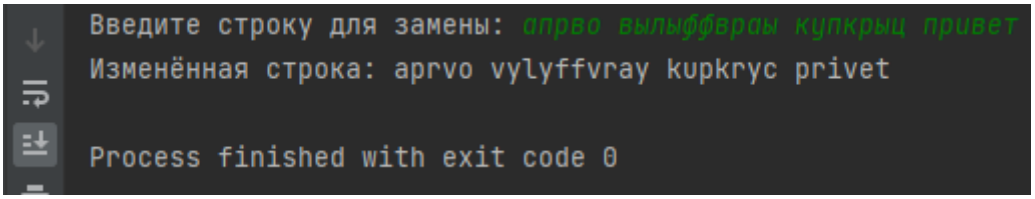

Рисунок 2 – Результат выполнения программы без декоратора

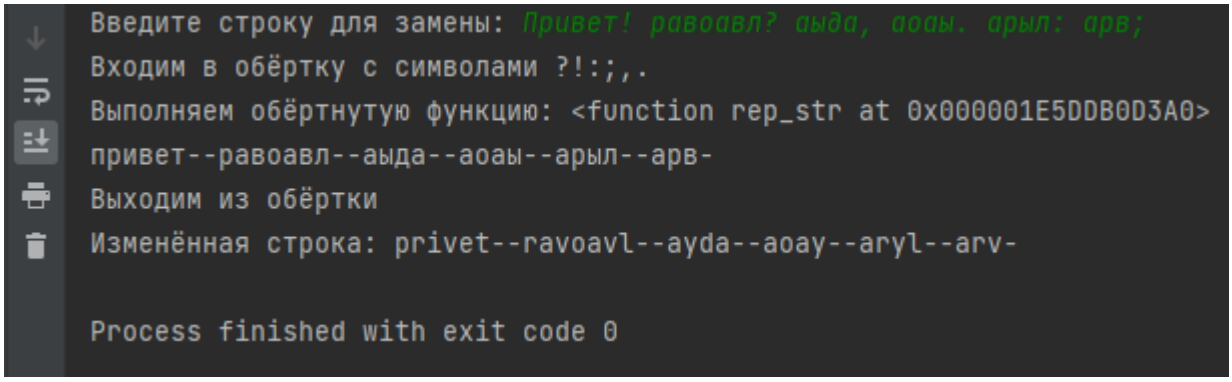

Рисунок 3 – Результат выполнения программы с использованием декоратора

2. 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.

2. Почему функции являются объектами первого класса?

Потому что с ними 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.

Они предназначены для передачи функций как параметры другим функциям, а также возвращать их как значения

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?

Объявляется декоратор, который принимает функцию как аргумент, внутри декоратора объявляется обёртка, которая выполняется функцию с изменёнными условиями.

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

Поместить декоратор внутри функции, которая будет принимать некоторые параметры, а затем использовать эти параметры в декораторе.