

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 «Основы кроссплатформенного программирования»**

Выполнила:  
Мурашко Анастасия Юрьевна  
1 курс, группа ИТС-б-о-22-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи»,  
направленность (профиль)  
«Инфокоммуникационные системы и  
сети», очная форма обучения

---

(подпись)

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

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** Декораторы в языке Python

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

**Ход работы:**

**Пример №1:**

```
1  def decorator_function(func):
2      def wrapper():
3          print('Функция-обёртка!')
4          print('Оборачиваемая функция: {}'.format(func))
5          print('Выполняем обёрнутую функцию...')
6          func()
7          print('Выходим из обёртки')
8
9      return wrapper
10
11
12  @decorator_function
13  def hello_world():
14      print('Hello world!')
15
16
17  hello_world()
```

Рисунок 1. Код

```
↑ C:\Users\MonstR\PycharmProjects\pythonProject\venv\Scripts\python.exe "D:/User
↓
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x0000018F80C8DD30>
:U
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки
Process finished with exit code 0
```

Рисунок 2. Результат выполнения

Для следующих примеров, чтобы установить библиотеку requests, вводим в строку `pip install requests`.

### Пример №2:

```
4 def wrapper():
5     start = time.time()
6     func()
7     end = time.time()
8     print('[*] Время выполнения: {} секунд.'.format(end - start))
9
10    return wrapper
11
12
13    @benchmark
14    def fetch_webpage():
15        import requests
16        requests.get('https://google.com')
17
18    fetch_webpage()
```

Рисунок 3. Код

```
↑ C:\Users\MonstR\PycharmProjects\pythonProject\venv\Script
↓ [*] Время выполнения: 0.9621412754058838 секунд.
⏏ Process finished with exit code 0
```

Рисунок 4. Результат выполнения

### Пример №3:

```
1 def benchmark(func):
2     import time
3
4     def wrapper(*args, **kwargs):
5         start = time.time()
6         return_value = func(*args, **kwargs)
7         end = time.time()
8         print('[*] Время выполнения: {} секунд.'.format(end - start))
9         return return_value
10
11    return wrapper
12
13
14    @benchmark
15    def fetch_webpage(url):
16        import requests
17        webpage = requests.get(url)
18        return webpage.text
```

Рисунок 5. Код

```
C:\Users\MonstR\PycharmProjects\pythonProject\venv\Scripts\python.exe "D:/Users/Acer/Desktop
[*] Время выполнения: 0.9626438617706299 секунд.
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ru"><head><me
var f=this||self;var h,k=[];function l(a){for(var b;a&&(!a.getAttribute)||!(b=a.getAttribut
function n(a,b,c,d,g){var e="";c||-1!=b.search("&ei=")|| (e="&ei="+l(d),-1===b.search("&le
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.target){var c=
</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;overflow-y:scroll
var h=this||self;var k,l=null!=(k=h.mei)?k:1,n,p=null!=(n=h.sdo)?n:!0,q=0,r,t=google.erd,v
a.fileName;g&&(0<g.indexOf("-extension:/")&&(e=3),c+="%script="+b(g),f&&g===window.location
if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}
```

Рисунок 6. Результат выполнения

## Вариант №9

### Индивидуальное задание:

9. Объявите функцию, которая принимает строку на кириллице и преобразовывает ее в латиницу, используя следующий словарь для замены русских букв на соответствующее латинское написание:

```
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'}
```

Функция должна возвращать преобразованную строку. Замены делать без учета регистра (исходную строку перевести в нижний регистр – малые буквы). Определите декоратор с параметром `chars` и начальным значением `" !?," . "`, который данные символы преобразует в символ `"-"` и, кроме того, все подряд идущие дефисы (например, `"--"` или `"---"`) приводит к одному дефису. Полученный результат должен возвращаться в виде строки. Примените декоратор со значением `chars="?!:, . "` к функции и вызовите декорированную функцию. Результат отобразите на экране.

Рисунок 7. Условия задания

```

1  ▶ if __name__ == '__main__':
2      def f(func):
3          def g(b, chars=' !?'):
4              tmp = ''.join(map(lambda x: x if x not in chars else '-', func(b)
5                  while '--' in tmp:
6                      tmp = tmp.replace('--', '-')
7                  return tmp
8
9          return g
10
11
12      @f
13      def h(t):
14          t = {'ё': 'yo', 'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd', 'е':
15              'э': 'z', 'и': 'i', 'й': 'y', 'к': 'k', 'л': 'l', 'м': 'm', 'н':
16              'р': 'r', 'с': 's', 'т': 't', 'у': 'u', 'ф': 'f', 'х': 'h', 'ц':
17              'щ': 'shch', 'ъ': '', 'ы': 'y', 'ь': '', 'э': 'e', 'ю': 'yu', 'я'
18          return ''.join(map(lambda x: t.get(x, x), a.lower()))
19
20      print('Введите текст:')
21      a = input()
22      print(h(a, chars='?!:;,. '))

```

Рисунок 8. Код

```

C:\Users\MonstR\PycharmProjects\pythonProject\venv\Scripts\python.exe "D:/Users/
Введите текст:
Здравствуйте, это мой код, для выполнения--индивидуального--задания!.,?
zdravstvuyte-eto-moy-kod-dlya-vypolneniya-individualnogo-zadaniya-
Process finished with exit code 0

```

Рисунок 9. Результат выполнения

### Ответы на контрольные вопросы:

#### 1. Что такое декоратор?

Декоратор – это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственно изменения её кода.

#### 2. Почему функции являются объектами первого класса?

Мы можем сохранять функции в переменные, передавать их в качестве аргументов и возвращать из других функций. Можно даже определить одну функцию внутри другой.

#### 3. Каково назначение функций высших порядков?

Это такие функции, которые могут принимать в качестве аргументов и возвращать другие функции.

#### 4. Как работают декораторы?

Из определения: декоратор - это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода. На примере кода:

```
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
```

#### 5. Какова структура декоратора функций?

Функция декоратор, содержащая в себе декорируемую функцию.

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

Функциональный вызов `func(...)`, который вернет что-то тоже вызываемое или имя функции, или переменная или экземпляр класса.

**Вывод:** Приобрела навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.