

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 №15**  
**дисциплины «Программирование на Python»**  
**Вариант 29**

Выполнил:  
Саенко Андрей Максимович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

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

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_  
Ставрополь, 2023 г.

Тема: Декораторы функций в языке Python

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

Порядок выполнения работы:

Вариант 29 (9)

1. Выполнено индивидуальное задание

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.

Код программы:

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
def decorator_parameters(chars = " !?"):  
    def decorator_translate(function):  
        import re  
  
        def wrapper(n):  
            result = function(n)  
  
            for item in set(result):  
                if item in chars:  
                    result = result.replace(item, "-")  
  
            result = re.sub(r'-+', '-', result)
```

```

        return result

    return wrapper

return decorator_translate

@decorator_parameters("?!:;,. ")
def translate(n):
    result = n.lower()
    elements = set(result)

    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'}

    keys_lst = list(t.keys())

    for item in elements:
        if item in keys_lst:
            result = result.replace(item, t[item])

    return result

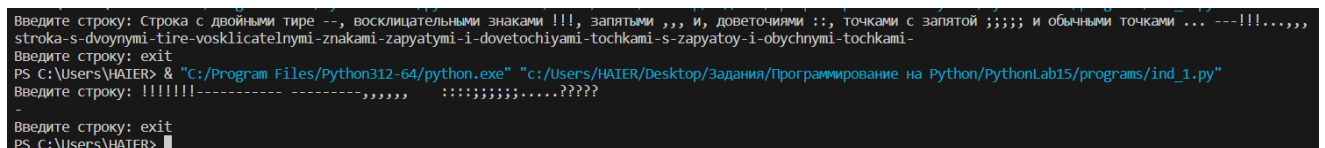
if __name__ == "__main__":
    while True:
        n = input("Введите строку: ")

        if n == "exit":
            break

    print(translate(n))

```

### Результат работы программы:



```

Введите строку: Строка с двойными тире --, восклицательными знаками !!!, запятыми,,, и, довоочиями ::, точками с запятой ;;;; и обычными точками ... ---!!!...,,,
stroka-s-dvoynymi-tire-vosklicatel'nyimi-znakami-zapyatymi-i-dovetochiyami-tochkami-s-zapyatoy-i-obychnymi-tochkami-
Введите строку: exit
PS C:\Users\HAIER> & "C:/Program Files/Python312-64/python.exe" "c:/Users/HAIER/Desktop/Задания/Программирование на Python/PythonLab15/programs/ind_1.py"
Введите строку: !!!!!!!-----,,,,,, :::::::::::.....?????
-
Введите строку: exit
PS C:\Users\HAIER>

```

Рисунок 1 – Результат работы 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.

2. Почему функции являются объектами первого класса?

Функции в языке программирования Python являются объектами первого класса, потому что они могут быть переданы в качестве параметра в другую функцию, присвоены переменной или возвращены из другой функции.

3. Каково назначение функций высших порядков?

Назначение функций высших порядков — принимать в качестве аргументов и(или) возвращать другие функции.

4. Как работают декораторы?

Декораторы в Python принимают функцию в качестве аргумента, добавляют к ней некоторую дополнительную функциональность и возвращают функцию с измененным поведением.

5. Какова структура декоратора функций?

Декоратор является функцией, которая содержит вложенную функцию и принимает другую функцию как аргумент. Вложенная функция модифицирует поведение функции, переданной в качестве параметра, а внешняя функция декоратора возвращает вложенную. Для применения декоратора используется строка вида `@<имя_декоратора>[(<параметры>)]`, которая добавляется перед определением функции, поведение которой нужно модифицировать.

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

Для того, чтобы передать параметры декоратору, а не декорируемой функции, надо обернуть декоратор ещё одной функцией, принимающей

необходимые параметры и возвращающей этот декоратор. Для передачи параметров при применении декоратора используется уже не его изначальное имя, а имя функции, которой он обёрнут, кроме того, добавляются круглые скобки, в которых должны быть параметры. В результате строка для применения декоратора принимает вид:

@<имя\_декоратора>(<параметры>),

#### Вывод

В ходе выполнения работы были приобретены навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.