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 9

Выполнил:
Дудкин Константин Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»
направление «Программное
обеспечение средств вычислительной
техники и автоматизированных
систем»,
очная форма обучения

_____
(подпись)

Руководитель практики:
Кандидат технических наук, доцент
кафедры инфокомуникаций, доцент
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Декораторы функций в языке Python

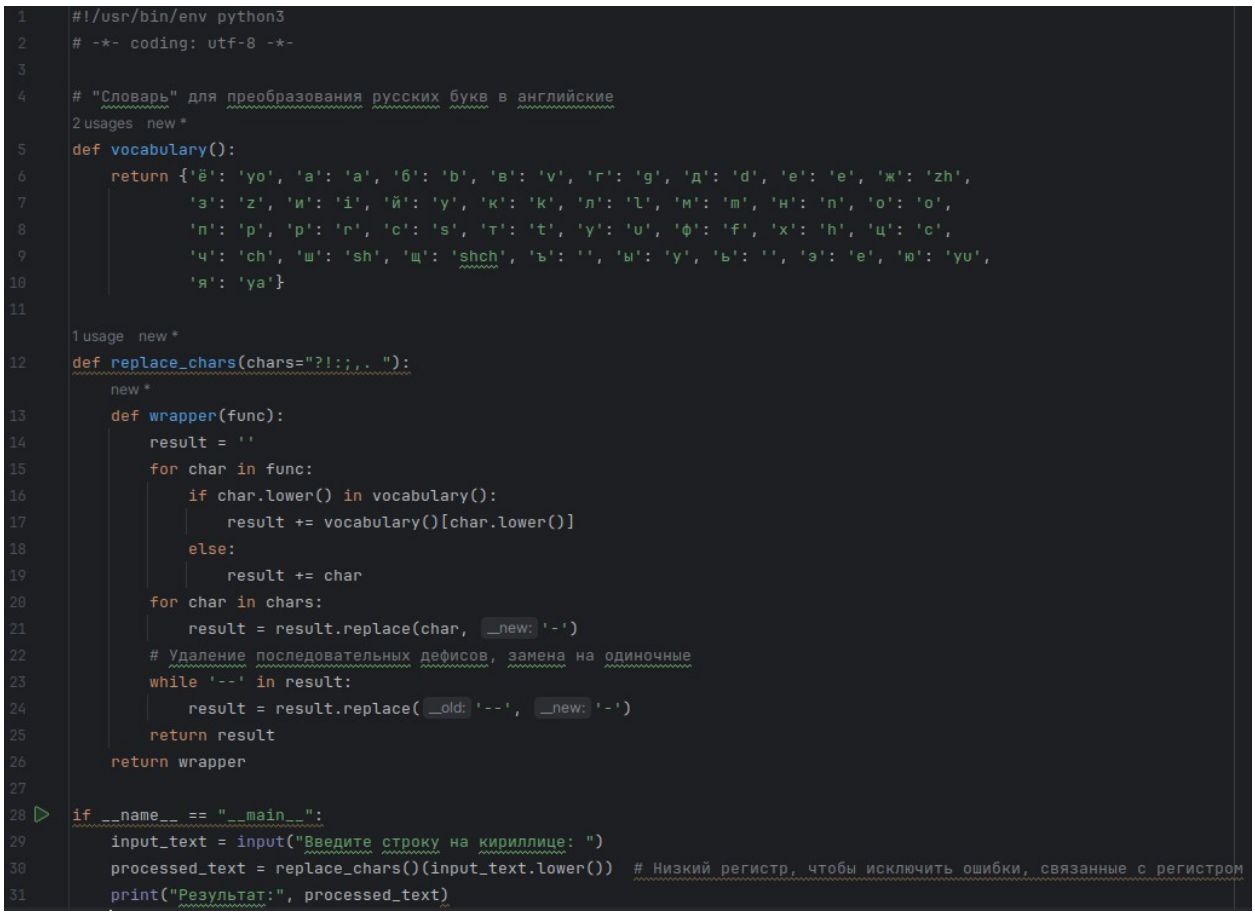
Цель: Приобретение навыков по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x

Порядок выполнения работы

1. Выполнил 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.



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

# "Словарь" для преобразования русских букв в английские
2 usages  new *
def vocabulary():
    return {'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'}

1 usage  new *
def replace_chars(chars="?!:;,. "):
    new *
    def wrapper(func):
        result = ''
        for char in func:
            if char.lower() in vocabulary():
                result += vocabulary()[char.lower()]
            else:
                result += char
        for char in chars:
            result = result.replace(char, _new: '-')
        # Удаление последовательных дефисов, замена на одиночные
        while '--' in result:
            result = result.replace( _old: '--',  _new: '-')
        return result
    return wrapper

if __name__ == "__main__":
    input_text = input("Введите строку на кириллице: ")
    processed_text = replace_chars()(input_text.lower())  # Низкий регистр, чтобы исключить ошибки, связанные с регистром
    print("Результат:", processed_text)
```

Рисунок 1. Код программы задачи

```
/usr/bin/python3.11 /home/code_ra1der/git/Python_LW15/Individual.py
Введите строку на кириллице: Шла Саша по шоссе и сосала сущку
Результат: shla-sasha-po-shosse-i-sosala-sushchku

Process finished with exit code 0
```

Рисунок 2. Результат выполнения программы

Ответы на вопросы

1. Декоратор - это функция, которая принимает другую функцию в качестве аргумента и возвращает новую функцию, обычно расширяя или изменяя поведение оригинальной функции.

2. Функции являются объектами первого класса, потому что они могут быть присвоены переменной, переданы в качестве аргументов другим функциям, возвращены из других функций и сохранены в структурах данных.

3. Функции высших порядков используются для работы с другими функциями как с данными, позволяя создавать более абстрактные и гибкие решения.

4. Декораторы работают путем обертывания (или декорирования) функций, позволяя добавлять дополнительное поведение до, после или вокруг вызова оригинальной функции, не изменяя её исходный код.

5. Структура декоратора функций обычно состоит из определения декоратора (функции), которая принимает функцию в качестве аргумента, определяет новую функцию внутри себя, которая может включать вызов оригинальной функции, и затем возвращает эту новую функцию.

6. Для передачи параметров декоратору, а не декорируемой функции, можно воспользоваться дополнительной оберткой или использовать функции с переменным числом аргументов (*args, **kwargs) для передачи всех параметров в декоратор.

Вывод: В ходе выполнения работы были приобретены навыки по работе с декораторами функций при написании программ на языке программировании Python версии 3.x