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 ПРАКТИЧЕСКОЙ РАБОТЕ №15
дисциплины «Программирование на Python»

Выполнил:
Лейс Алексей Вячеславович
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»

_____
(подпись)

Руководитель практики: кандидат тех. наук доцент кафедры инфокоммуникаций: Воронкин Р.А

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Декораторы функций в языке Python

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

Порядок выполнения работы:

Пример из лабораторной работы с декораторами

```python
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)
```

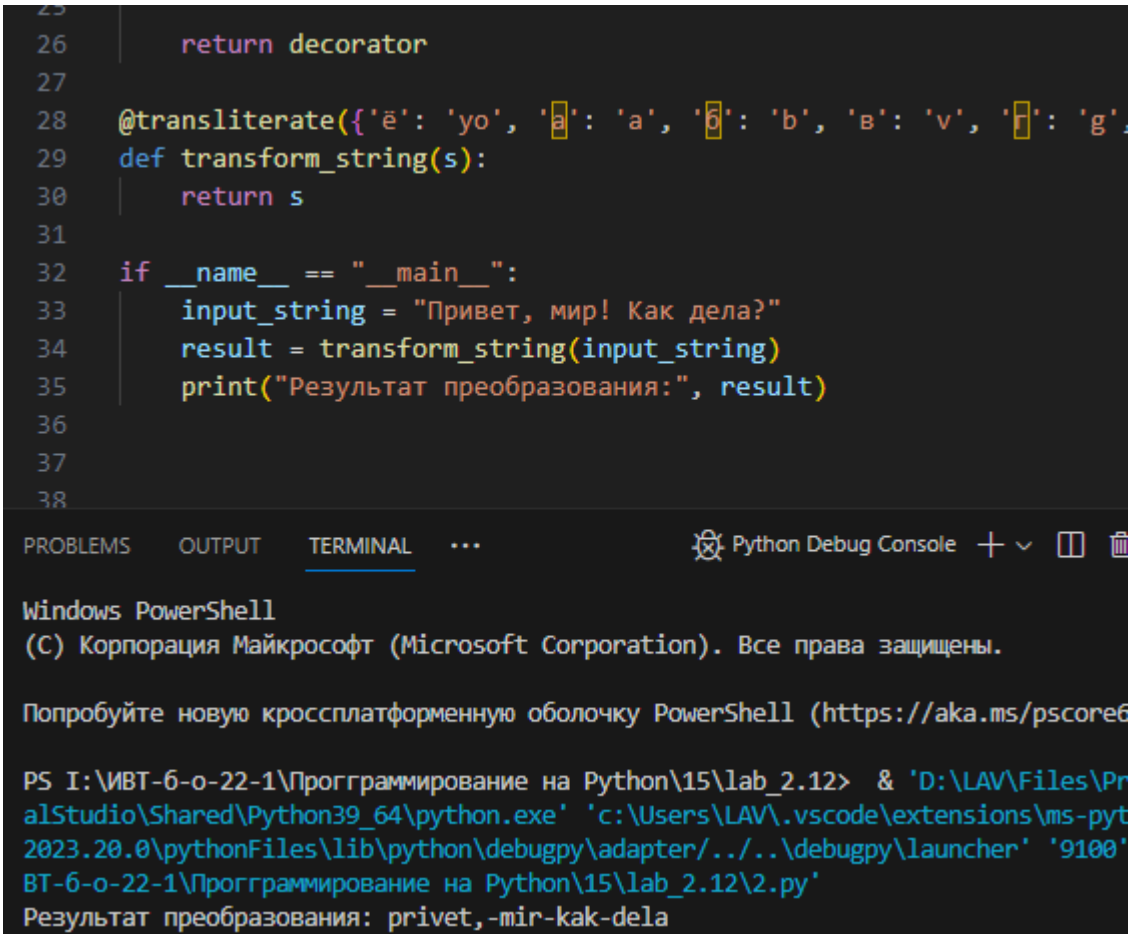
Индивидуальное задание:

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'} 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.



Ответы на вопросы:

1. **Что такое декоратор?**

- Декоратор в Python - это функция, которая принимает другую функцию и расширяет ее функциональность, обычно без изменения ее кода. Декораторы часто используются для изменения поведения функций или добавления какой-то логики до или после их выполнения.

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

- В Python функции считаются объектами первого класса, потому что они могут быть присвоены переменным, переданы как аргументы в другие функции, возвращены из функций и использованы в любых выражениях.

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

- Функции высших порядков в Python могут принимать одну или несколько функций в качестве аргументов и/или возвращать функцию в

качестве результата. Это позволяет создавать более абстрактные и гибкие конструкции, такие как декораторы или функции, работающие с другими функциями.

4. **Как работают декораторы?**

- Декораторы работают, оборачивая функцию, которую они декорируют, внутри дополнительной функции. Эта обертка может выполнять какие-то действия до или после вызова оригинальной функции. Декораторы применяются к функциям с использованием символа **@**.

5. **Какова структура декоратора функций?**

- Структура декоратора состоит из определения функции-декоратора и применения его к целевой функции с использованием символа **@**. Пример:

pythonCopy code

def my_decorator(func): def wrapper(): print("Something is happening before the function is called.") func() print("Something is happening after the function is called.") return wrapper @my_decorator def say_hello(): print("Hello!") say_hello()

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

- Параметры могут быть переданы декоратору, используя вложенные функции. В следующем примере **my_decorator** принимает параметр, который затем передается внутренней функции-обертке:

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