

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»

Выполнил:
Гайчук Дарья Дмитриевна
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.-доцент кафедры
инфокоммуникаций

(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

1. Создала общедоступный репозиторий на GitHub, в котором будет использоваться лицензия MIT.

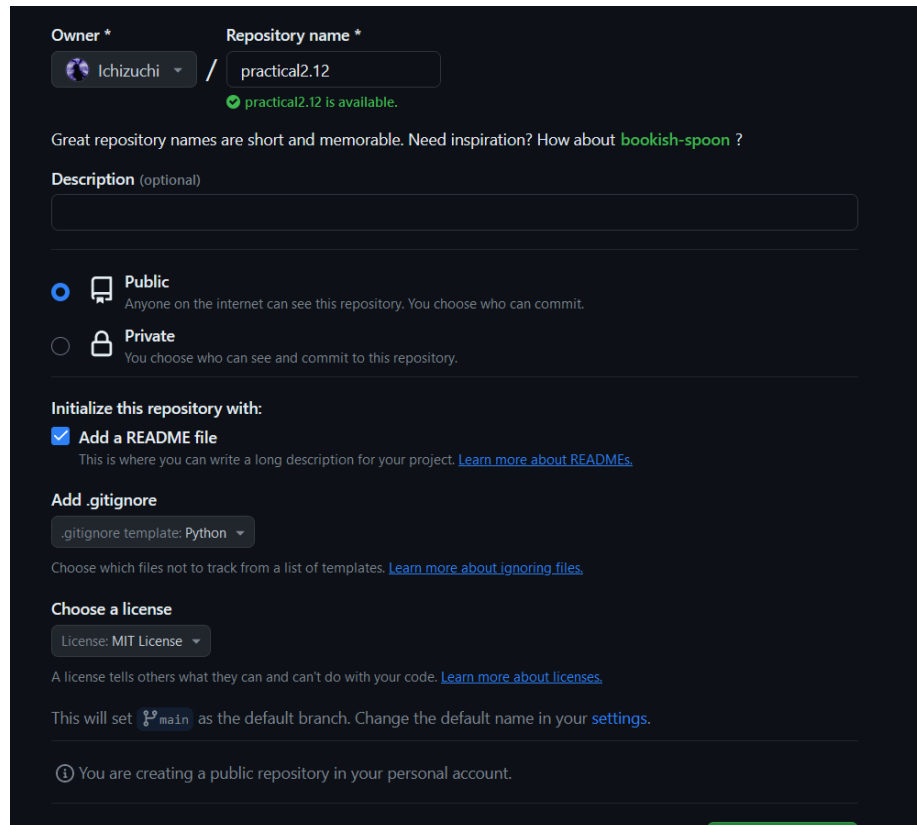


Рисунок 1. Новый репозиторий

2. Скопировала репозиторий на свой компьютер.

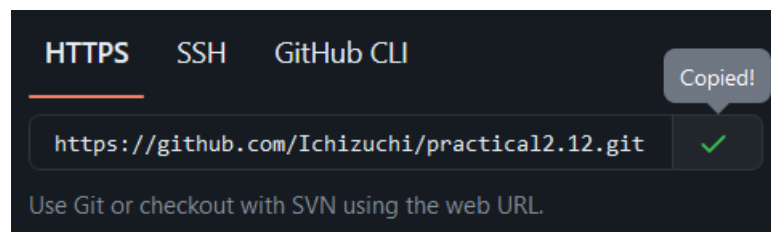


Рисунок 2. Клонирование репозитория

3. Использовала систему ветвления git-flow

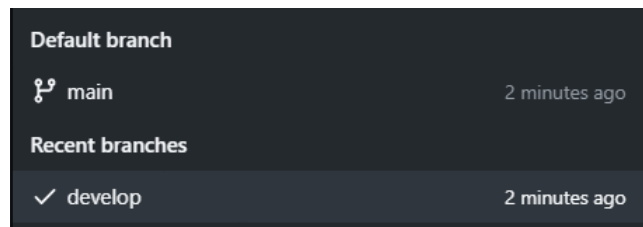


Рисунок 3. Ветка develop

4. Решить пример №1: создаём декоратор, измеряющий время выполнения функции. Далее мы используем его на функции, которая делает GET-запрос к главной странице Google. Чтобы измерить скорость, мы сначала сохраняем время перед выполнением обёрнутой функции, выполняем её, снова сохраняем текущее время и вычитаем из него начальное.

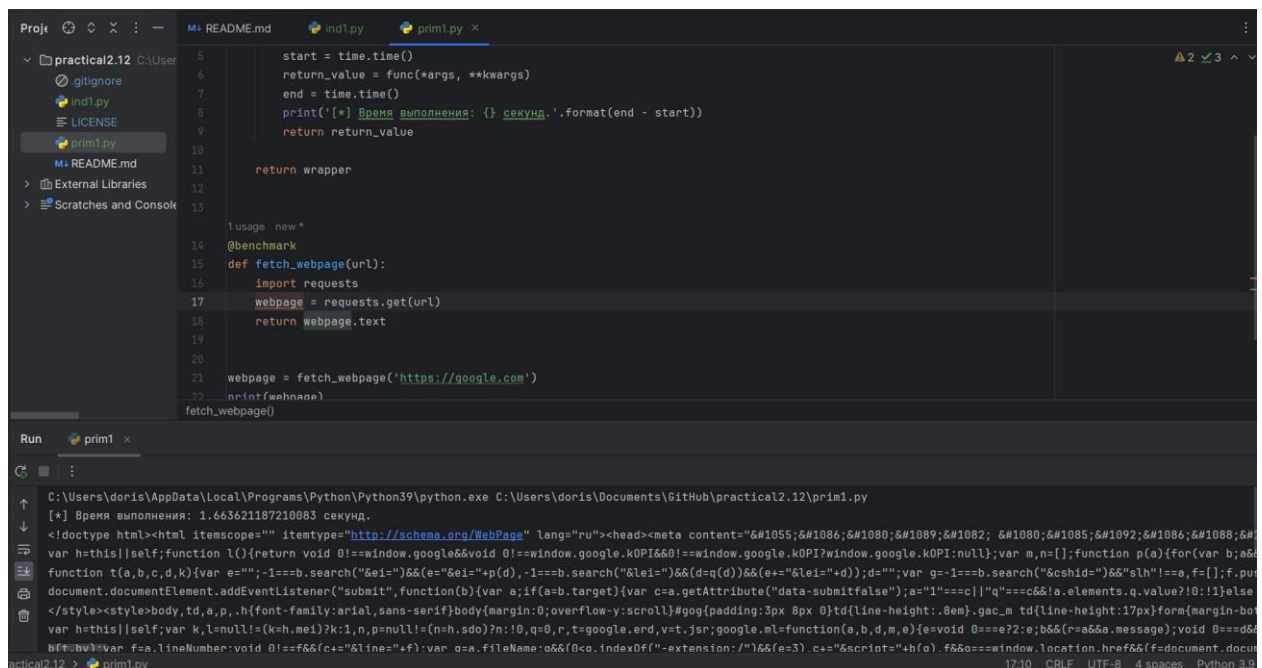
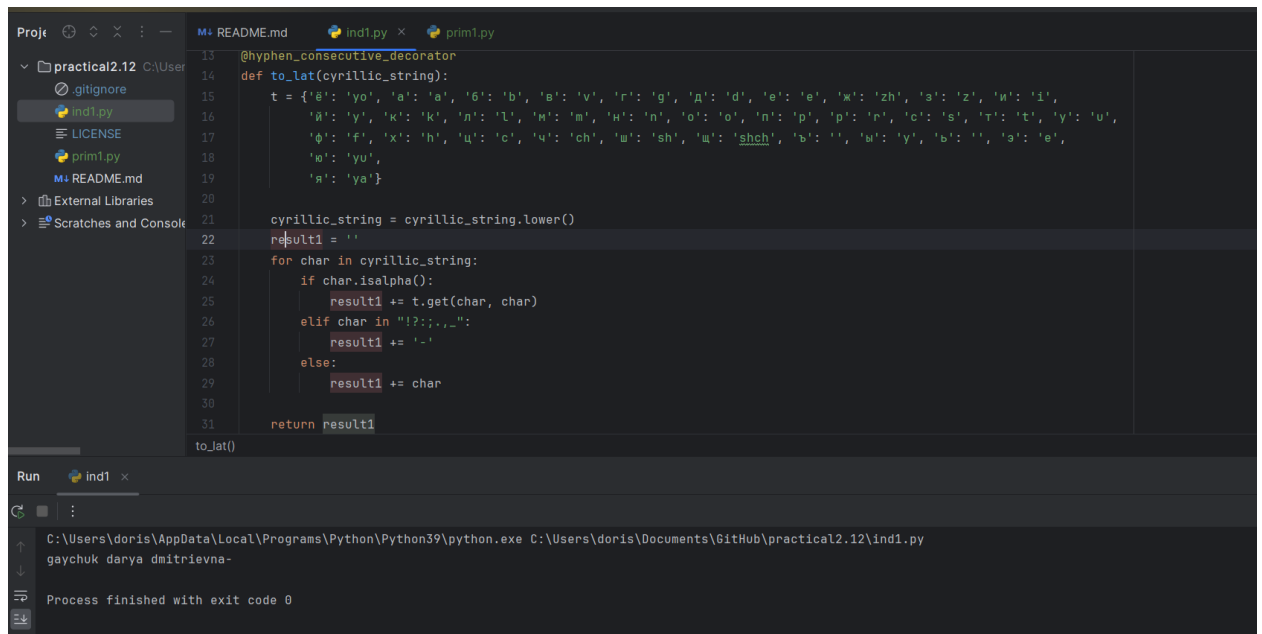


Рисунок 4. Программа и е результат

5. Выполнила индивидуальное задание (Вариант №4): объявите функцию с именем `to_lat`, 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). Все небуквенные символы "

?:,._" превращать в символ '-' (дефиса). Определите декоратор для этой функции, который несколько подряд идущих дефисов, превращает в один дефис. Полученная строка должна возвращаться при вызове декоратора. Примените декоратор к функции `to_lat` и вызовите ее. Результат работы декорированной функции отобразите на экране.



```
13 @hyphen_consecutive_decorator
14 def to_lat(cyrillic_string):
15     t = {'ё': 'yo', 'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd', 'е': 'e', 'ж': 'zh', 'з': 'z', 'и': 'i',
16         'й': 'y', 'к': 'k', 'л': 'l', 'м': 'm', 'н': 'n', 'о': 'o', 'п': 'p', 'р': 'r', 'с': 's', 'т': 't', 'у': 'u',
17         'ф': 'f', 'х': 'h', 'ц': 'c', 'ч': 'ch', 'ш': 'sh', 'щ': 'shch', 'ъ': '', 'ы': 'y', 'ь': '', 'я': 'a',
18         'а': 'yu',
19         'я': 'ya'}
20
21     cyrillic_string = cyrillic_string.lower()
22     result1 = ''
23     for char in cyrillic_string:
24         if char.isalpha():
25             result1 += t.get(char, char)
26         elif char in " !?:,._-":
27             result1 += '-'
28         else:
29             result1 += char
30
31     return result1
32
33 to_lat()
```

Run ind1 x

C:\Users\doris\AppData\Local\Programs\Python\Python39\python.exe C:\Users\doris\Documents\GitHub\practical2.12\ind1.py
gaychuk darya dmitrievna-

Process finished with exit code 0

Рисунок 5. Программа и ее результат

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 они могут быть присвоены переменной, переданы в качестве аргумента другой функции, возвращены из функции и сохранены в структурах данных.

3. Каково назначение функций высших порядков? могут принимать другие функции в качестве аргументов или возвращать их в качестве результатов. Они позволяют абстрагировать действия и оперировать функциями, как данными.

4. Как работают декораторы?

Работают путем обертывания другой функции, позволяя добавлять новое поведение к этой функции без изменения ее кода. Декораторы принимают функцию в качестве аргумента, возвращают другую функцию и обычно используются с символом @.

5. Какова структура декоратора функций?

Структура декоратора функций обычно выглядит следующим образом:

```
def decorator_function(original_function):  
    def wrapper_function(*args,  
        **kwargs):  
        # Добавление нового поведения  
        return original_function(*args,  
            **kwargs)  
    return wrapper_function
```

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

Параметры могут быть переданы декоратору, а не декорируемой функции, путем добавления еще одного уровня вложенной функции в декораторе. Этот уровень может принимать параметры и передавать их в обернутую функцию.

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