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»

Выполнил:
Евдаков Евгений Владимирович
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 функций в языке Python.

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

Ход работы:

Задание 1. Создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер.

```
C:\Users\Gaming-PC>git clone https://github.com/EvgenyEvdakov/Laba_2.12.git
Cloning into 'Laba_2.12'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
```

Рисунок 1. Клонирование репозитория

Задание 2. Организовал свой репозиторий в соответствие с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\Gaming-PC\Laba_2.12>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Gaming-PC/Laba_2.12/.git/hooks]
```

Рисунок 2. Модель ветвления git-flow

Задание 3. Создал проект PyCharm в папке репозитория. Приступил к работе с примером. Добавил новый файл primer.py.

Условие примера: посмотреть и понять как работает декоратор.

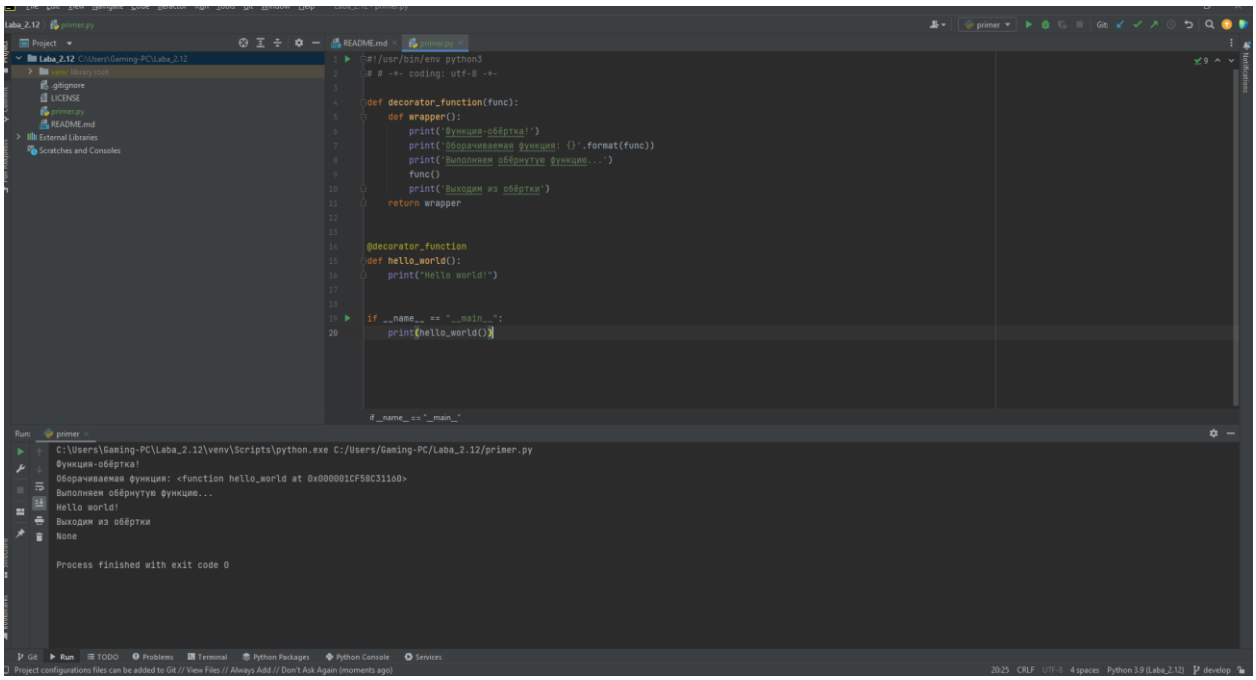

Рисунок 3. Реализация примера

Задание 4.

Индивидуальное задание

Вариант 9 (по списку группы)

Создал новый файл под названием individual.py.

Условие задания: 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.

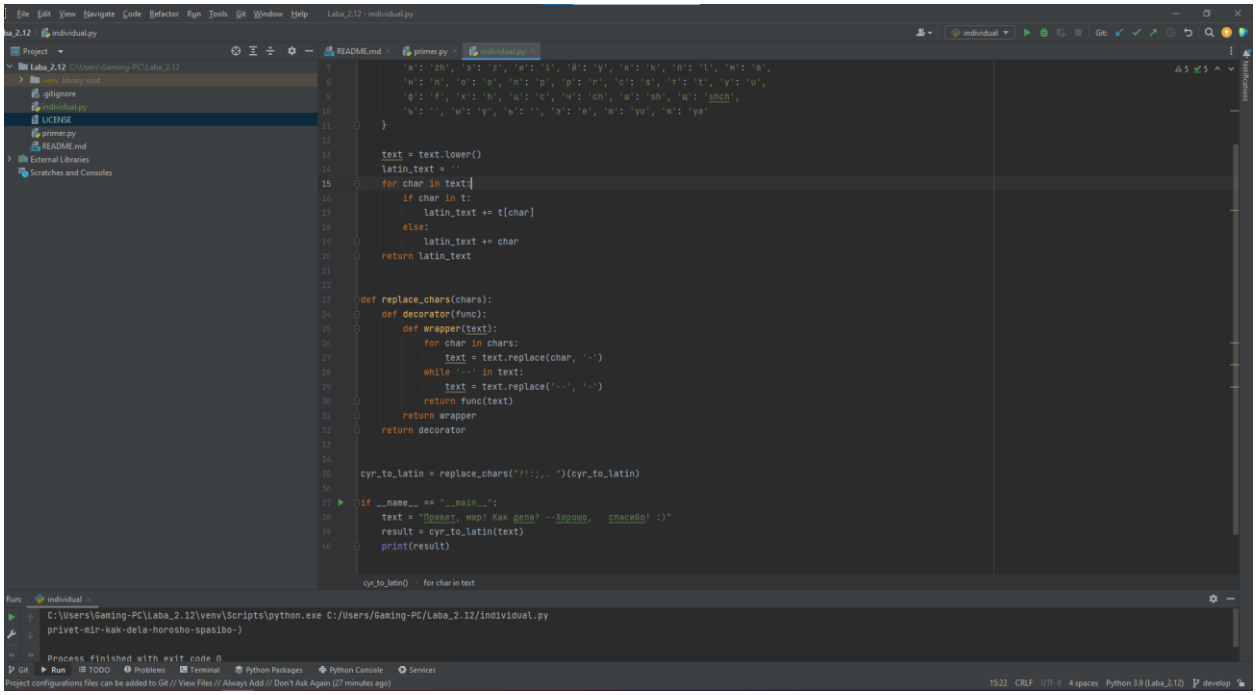

Рисунок 4. Программа индивидуального задания

Задание 5.

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер.

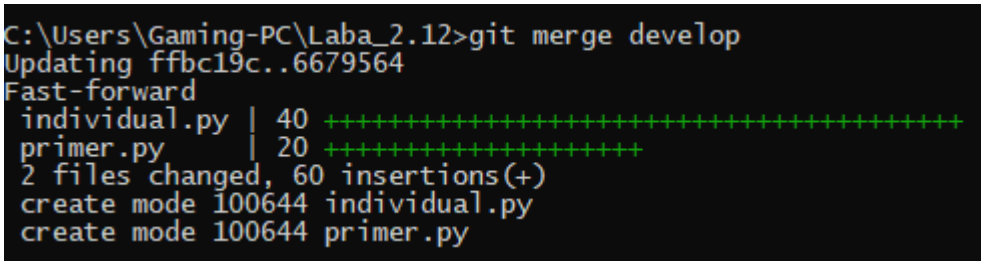

Рисунок 5. Слияние ветки develop и main

Ссылка:

Ответы на контрольные вопросы:

1. **Что такое декоратор?**

Декоратор - это функция, которая принимает другую функцию и расширяет ее функциональность, не изменяя ее исходный код. Декоратор позволяет добавлять новое поведение для функции, не затрагивая ее исходный код, что делает код более читаемым, понятным и модульным.

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

Функции являются объектами первого класса, потому что в Python они могут быть созданы, использованы и переданы как любой другой объект, например, числа, строки или списки. Функции могут быть присвоены переменной, переданы как аргументы в другую функцию, возвращены из функции и использованы в выражениях.

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

Функции высших порядков - это функции, которые принимают одну или несколько функций как аргументы, либо возвращают другую функцию в качестве результата. Назначение функций высших порядков заключается в создании абстракций, которые могут использоваться для управления поведением других функций, делая код более гибким и модульным.

4. **Как работают декораторы?**

При использовании декоратора, функция или метод класса передаются в качестве аргумента в другую функцию, которая и выполняет нужное действие, дополняя тем самым исходную функциональность. Для создания декоратора в Python используют функцию, которая принимает в качестве аргументов другую функцию и возвращает функцию, которая представляет из себя декорированную версию первоначальной функции.

5. **Какова структура декоратора функций?**

Структура декоратора функций в Python состоит из двух функций:

1) Внешняя функция, которая принимает в качестве аргумента функцию, которую необходимо декорировать, и возвращает внутреннюю функцию.

2) Внутренняя функция, которая принимает те же аргументы, что и декорируемая функция, и содержит вызов декорируемой функции. Эта функция может изменять аргументы, передаваемые декорируемой функции, а также результат, возвращаемый этой функцией.

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

В Python параметры декоратору могут быть переданы с помощью вложенной функции-обертки (wrapper), которая будет вызвана после декорирования функции, но перед выполнением ее кода.

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