

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 №2.8
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 _____

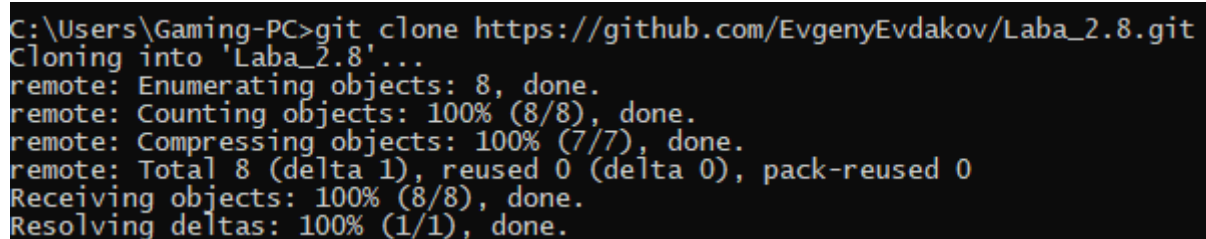
Ставрополь, 2023 г.

Тема: Работа с функциями в языке Python.

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

Ход работы:

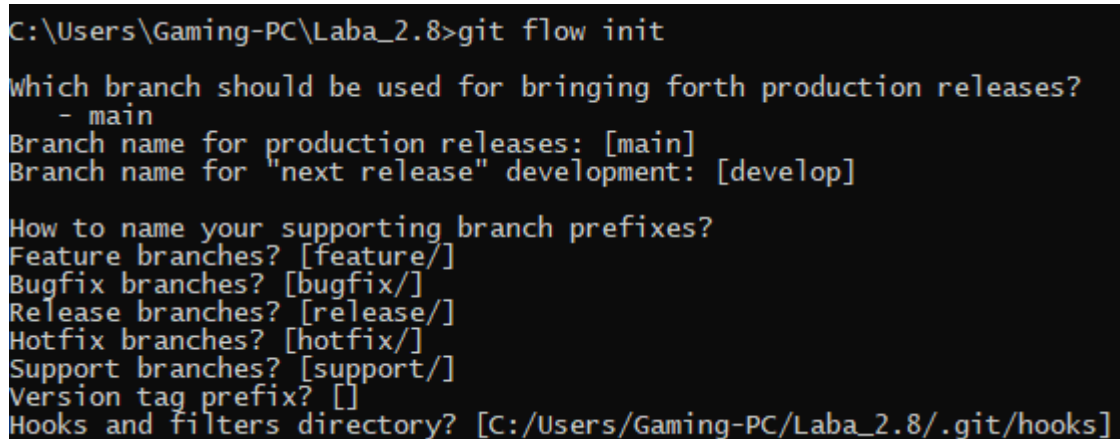
Задание 1. Создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер.



```
C:\Users\Gaming-PC>git clone https://github.com/EvgenyEvdakov/Laba_2.8.git
Cloning into 'Laba_2.8'...
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 соответствии с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.



```
C:\Users\Gaming-PC\Laba_2.8>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_2.8/.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: Для примера 1 лабораторной работы 2.6, оформить каждую команду в виде вызова отдельной функции.

```
114 # Получить требуемый стаж.
115 period = int(parts[1])
116
117 # Выбрать работников с заданным стажем.
118 selected = select_workers(workers, period)
119 # Отобразить выбранных работников.
120 display_workers(selected)
121
122 elif command == 'help':
123     # Вывести справку о работе с программой.
124     print("Список команд:\n")
125     print("add - добавить работника")
126     print("list - вывести список работников")
127     print("select <стаж> - запросить работников со стажем")
128     print("help - отобразить справку")
129     print("exit - завершить работу с программой.")
130
131 else:
132     print(f"Неизвестная команда {command}", file=sys.stderr)
133
134 if __name__ == '__main__':
135     main()
136
137 if __name__ == '__main__':
```

Run: primer

C:\Users\Gaming-PC\Lab2.8\venv\scripts\python.exe C:/Users/Gaming-PC/Lab2.8/primer.py

```
>>> add
Введите инициалы? Е.В.
Должность? студент
Год поступления? 2022
>>> list
```

No	И.И.О.	Должность	Год
1	Евдаков Е.В.	Студент	2022

```
>>>
```

Рисунок 3. Реализация примера

Задание 4. Создал новый файл под названием ObshZadanie1. Приступил к работе с общим заданием №1.

Условие примера: основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции `test()` и инструкции `if __name__ == '__main__':`. В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция `positive()`, тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция `negative()`, ее тело содержит выражение вывода на экран слова "Отрицательное". Понятно, что вызов `test()` должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения `positive()` и `negative()` предшествовать `test()` или могут следовать после него?

Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.

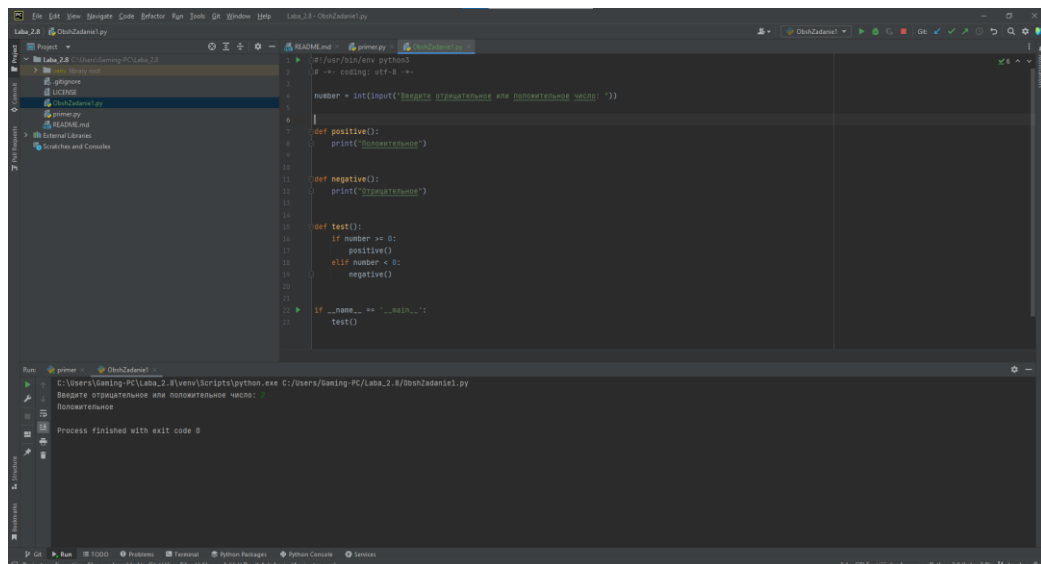


Рисунок 4. Реализация общего задания 1

Задание 5. Создал новый файл под названием ObshZadanie2. Приступил к работе с общим заданием №2.

Условие примера: в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле . В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле , или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.

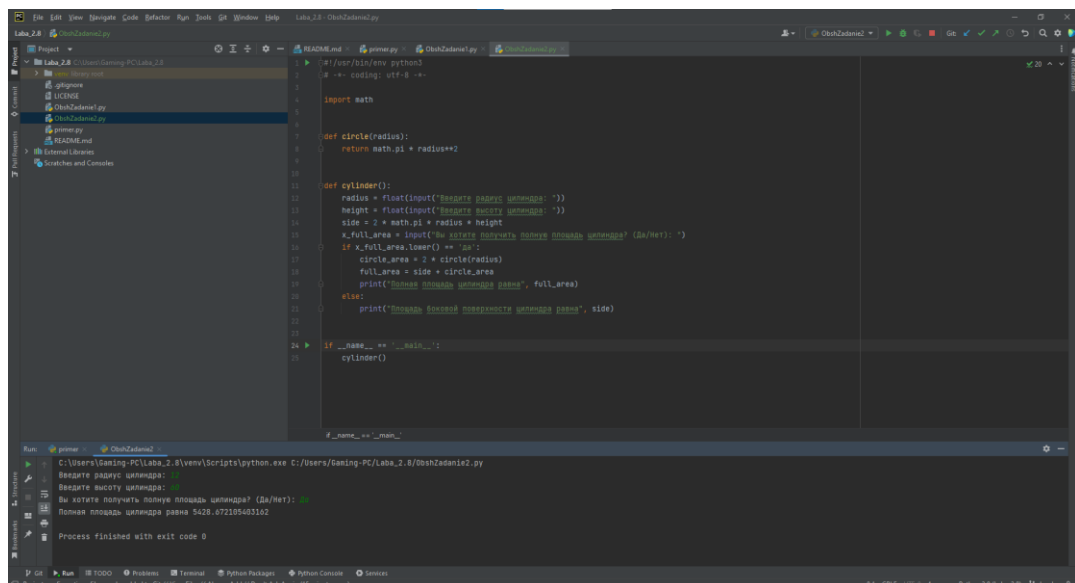
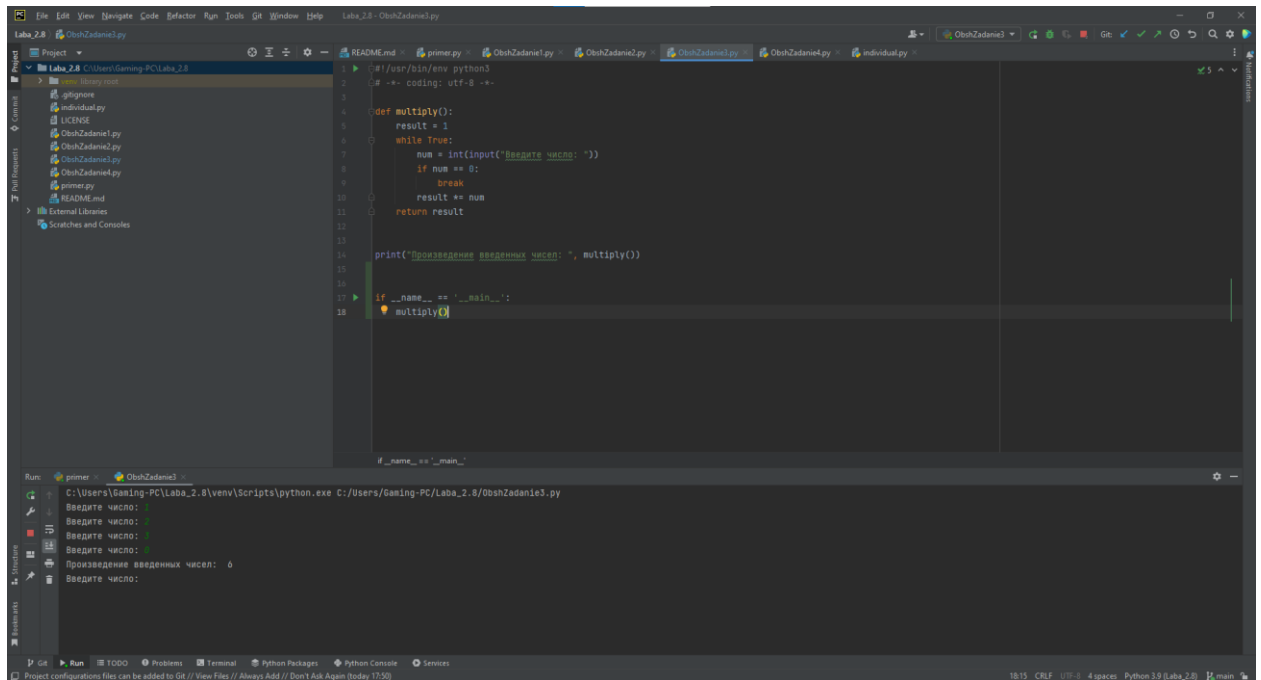


Рисунок 5. Реализация общего задания 2

Задание 6. Создал новый файл под названием ObshZadanie3. Приступил к работе с общим заданием №3.

Условие примера: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def multiply():
5     result = 1
6     while True:
7         num = int(input("Введите число: "))
8         if num == 0:
9             break
10        result *= num
11    return result
12
13 print("Общее задание: Введите число: ", multiply())
14
15 if __name__ == '__main__':
16     multiply()
```

Run: primer - ObshZadanie3 - C:\Users\Gaming-PC\Laba_2.8\Scripts\python.exe C:/Users/Gaming-PC/Laba_2.8/ObshZadanie3.py

Введите число: 5
Введите число: 2
Введите число: 0
Произведение введенных чисел: 0
Введите число:

Рисунок 6. Реализация общего задания 3

Задание 7. Создал новый файл под названием ObshZadanie4. Приступил к работе с общим заданием №4.

Условие примера: напишите программу, в которой определены следующие четыре функции:

1. Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.
2. Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.
3. Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.

4. Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула `True`, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.

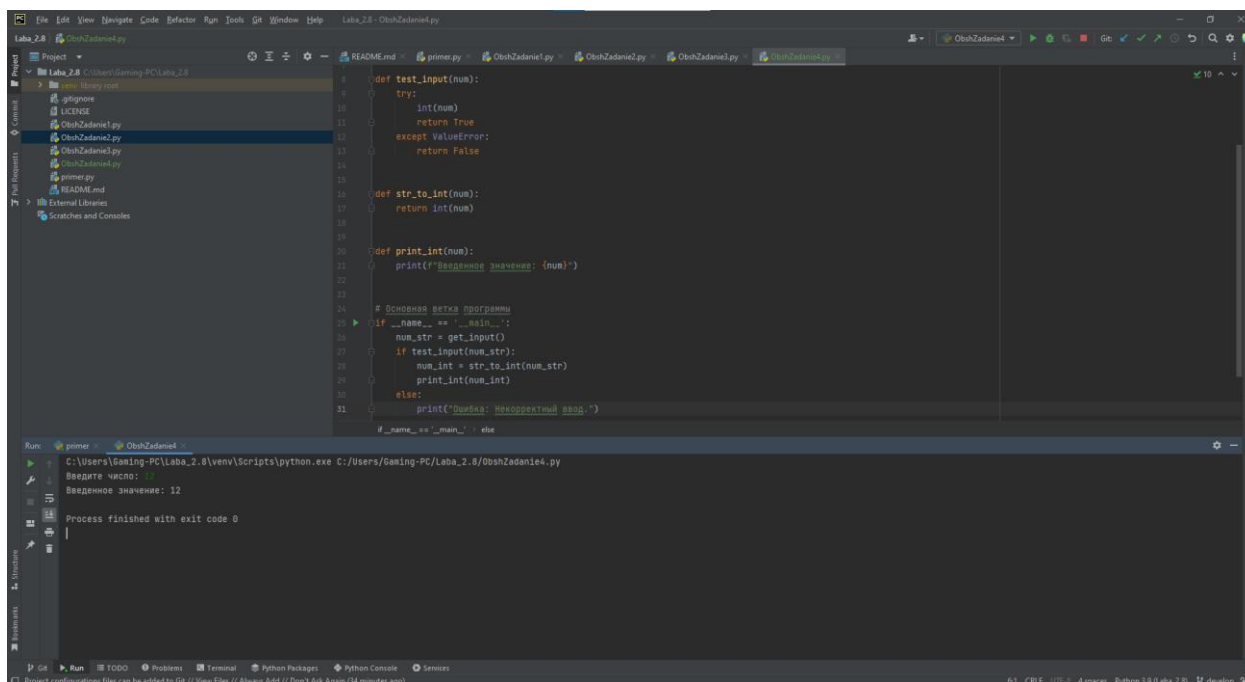


Рисунок 7. Реализация общего задания 4

Задание 8.

Индивидуальное задание

Вариант 9 (по списку группы)

Создал новый файл под названием `individual.py`.

Условие задания: Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

Условие задание 2.6: Использовать словарь, содержащий следующие ключи: название начального пункта маршрута; название конечного пункта маршрута; номер маршрута. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по номерам маршрутов; вывод на экран информации о маршруте, номер которого введен

с клавиатуры; если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

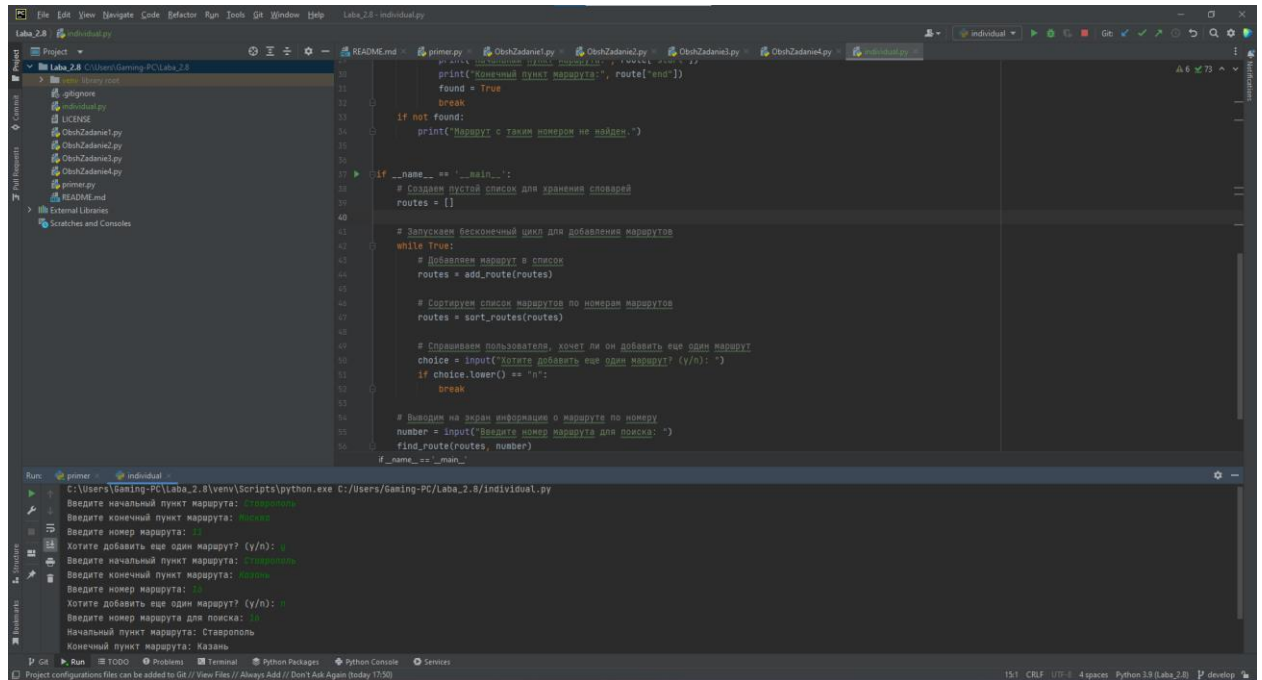


Рисунок 6. Программа индивидуального задания

Задание 7.

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

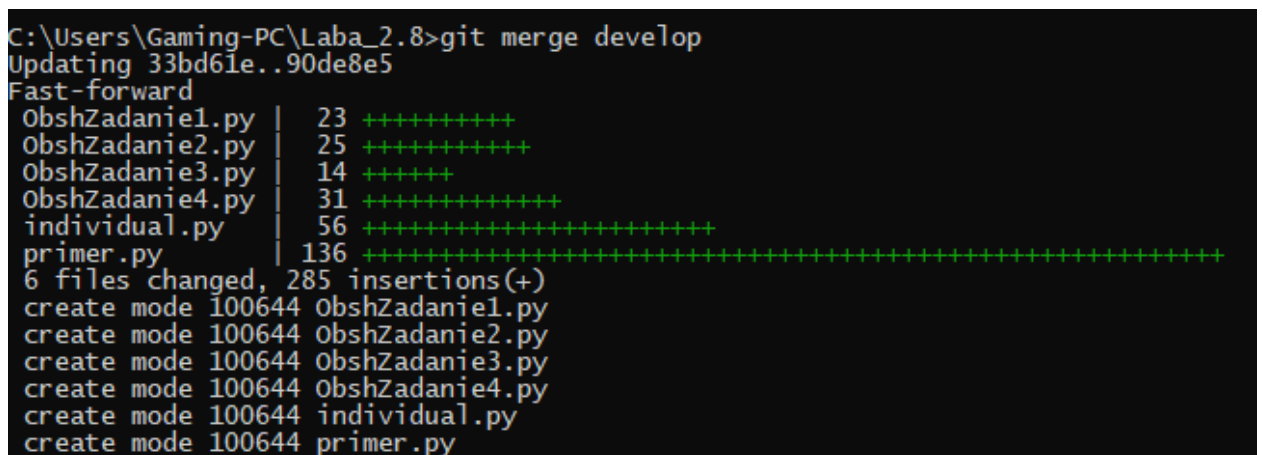


Рисунок 7. Слияние ветки develop и main

Ссылка: https://github.com/EvgenyEvdakov/Laba_2.8

Ответы на контрольные вопросы:

1. Каково назначение функций в языке программирования Python?

Функции в языке программирования Python используются для организации кода в отдельные блоки, которые можно вызывать и использовать многократно в программе. Функции позволяют повысить читаемость, упростить отладку и сократить количество дублирующегося кода. Также функции могут быть использованы для создания модулей, которые могут быть использованы в различных проектах.

2. Каково назначение операторов def и return?

Оператор def используется для определения функции, а оператор return используется для возврата значения из функции. При вызове функции, ее тело выполняется, а затем значение, указанное в операторе return, возвращается в вызывающую программу.

3. Каково назначение локальных и глобальных переменных при написании функций в Python?

Локальные переменные создаются внутри функции и существуют только в рамках этой функции. Глобальные переменные создаются вне функций и могут быть доступны из любой части программы. При написании функций в Python важно учитывать область видимости переменных и использовать их правильно.

Назначение локальных переменных - это изолировать код функции от других частей программы, чтобы избежать изменений переменных из других частей программы, которые могут негативно повлиять на работу функции.

Назначение глобальных переменных - это обеспечить доступ к данному объекту из любой части программы. Однако, существует опасность перезаписи глобальных переменных, и использование глобальных переменных следует использовать с осторожностью.

4. Как вернуть несколько значений из функции Python?

В Python можно вернуть несколько значений из функции, используя кортеж. Для этого возвращаемые значения перечисляются через запятую внутри круглых скобок. В выбираемые значения можно обратиться по индексу.

5. Какие существуют способы передачи значений в функцию?

В языке программирования Python значения могут быть переданы в функцию несколькими способами: позиционные аргументы (передача аргументов в порядке их следования); именованные аргументы (передача аргументов с указанием их имени); аргументы по умолчанию (передача аргументов со значениями по умолчанию); распаковывание списков и словарей.

6. Как задать значение аргументов функции по умолчанию?

Для того, чтобы задать значение аргументов функции по умолчанию в Python, нужно указать это значение после имени аргумента в определении функции.

7. Каково назначение lambda-выражений в языке Python?

Lambda-выражения в языке Python представляют собой способ создания анонимных функций без явного определения имени функции. В lambda-выражениях объединяются три элемента: аргументы, оператор-разделитель и тело функции.

8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в Python осуществляется согласно рекомендациям PEP257. Для документирования функций и модулей используются строки документации (docstrings), которые должны быть помещены в начало определения функции или модуля. Для того чтобы оформить документацию в соответствии с PEP257, используют следующие рекомендации:

- 1) Docstring должен начинаться с однострочного описания того, что делает объект (функция, класс и т.д.). Это описание следует начинать с заглавной буквы и заканчивать точкой.
- 2) За однострочным описанием должна следовать пустая строка.
- 3) Если это функция или метод, то следует указать, какие аргументы она принимает, какие они должны быть по типу и за что они отвечают.

4) Если функция или метод что-то возвращает, то это также должно быть указано в документации.

5) Если объект имеет какие-то особенности или ограничения, то их нужно описать.

6) Если есть примеры использования, то их нужно привести.

9. В чем особенность однострочных и многострочных форм строк документации?

Однострочная форма документации заключается в одном ряду и применяется для краткого описания. Она начинается со знака # и двух пробелов, специально размещенных после знака #.

Многострочная форма документации позволяет вставлять более детальные описания. Она начинается и заканчивается тройными кавычками и предоставляет возможность размещения внутри описания более одного абзаца. Эта форма чаще применяется при описании функций и модулей.

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