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 №2.9
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: Рекурсия в языке Python.

Цель: приобретение навыков по работе с рекурсивными 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_2.9.git
Cloning into 'Laba_2.9'...
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_2.9>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_2.9/.git/hooks]
```

Рисунок 2. Модель ветвления git-flow

Задание 3. Создал проект PyCharm в папке репозитория. Приступил к работе с примером. Добавил новый файл primer1.py.

Условие примера: Если бы мы хотели узнать сумму чисел от 1 до, где - натуральное число, то могли бы посчитать вручную 1 + 2 + 3 + 4 + ... + (несколько часов спустя) +. А можно просто написать цикл for.

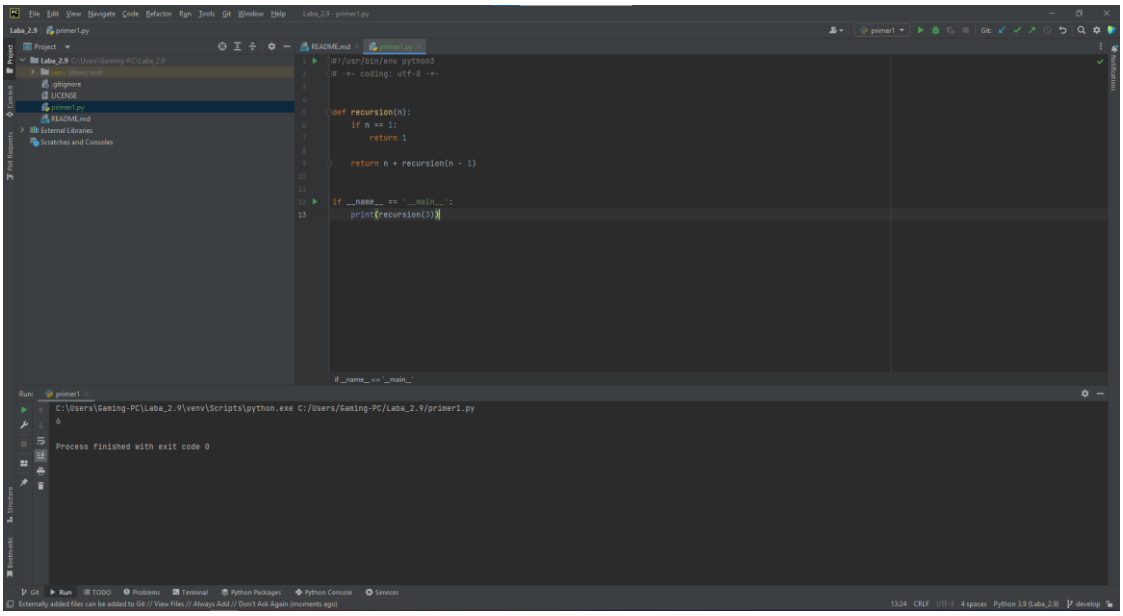

Рисунок 3. Реализация примера 1

Задание 4. Создал новый файл под названием primer2.py.

Условие примера:

Вы также можете иметь «параллельные» рекурсивные вызовы функций. Например, рассмотрим последовательность Фибоначчи, которая определяется следующим образом:

- Если число равно 0, то ответ равен 0.
- Если число равно 1, то ответ равен 1.

В противном случае ответ представляет собой сумму двух предыдущих чисел Фибоначчи.

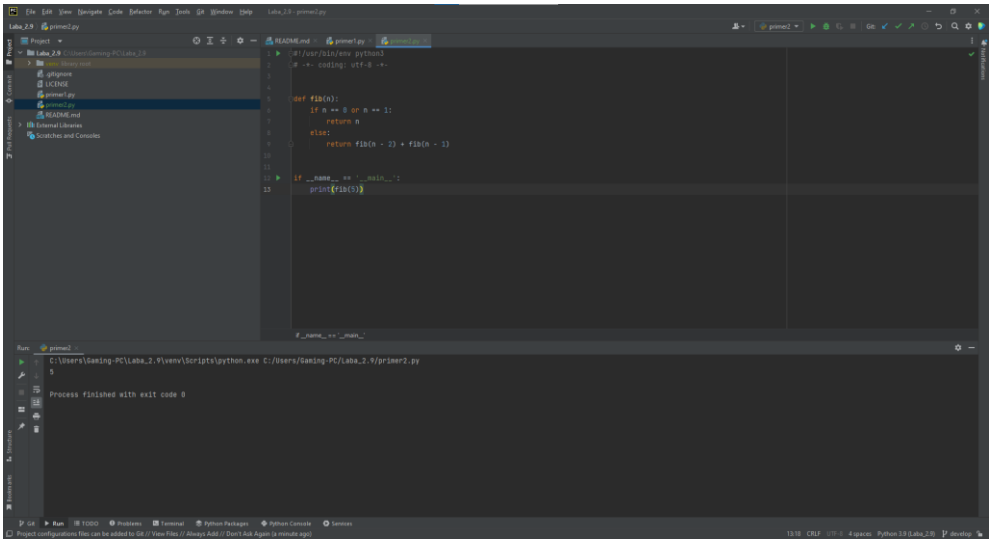

Рисунок 4. Реализация метода Фибоначчи

Задание 5.

Индивидуальное задание

Вариант 9 (по списку группы)

Создал новый файл под названием individual.py.

Условие задания:

9. Даны целые числа m и n, где $0 \leq m \leq n$, вычислить, используя рекурсию, число сочетаний C_n^m по формуле: $C_n^0 = C_n^n = 1, C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ при $0 \leq m \leq n$. Воспользовавшись формулой

$$C_n^m = \frac{n!}{m!(m-n)!} \quad (1)$$

можно проверить правильность результата.

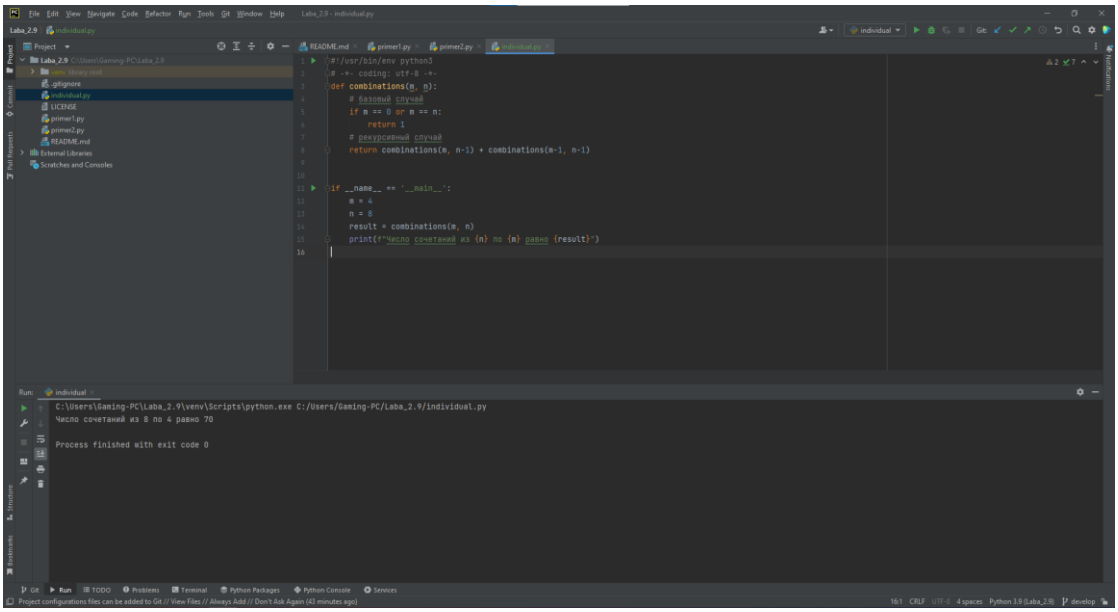

Рисунок 5. Программа индивидуального задания

Задание 6.

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

```
C:\Users\Gaming-PC\Laba_2.9>git merge develop
Updating 5423611..5c4f16b
Fast-forward
 individual.py | 15 +++++++++++++++
 primer1.py    | 13 +++++++++++++
 primer2.py    | 13 +++++++++++++
 3 files changed, 41 insertions(+)
 create mode 100644 individual.py
 create mode 100644 primer1.py
 create mode 100644 primer2.py
```

Рисунок 6. Слияние ветки develop и main

Ссылка: https://github.com/EvgenyEvdakov/Laba_2.9

Ответы на контрольные вопросы:

1. **Для чего нужна рекурсия?**

Рекурсия - это процесс, в котором функция вызывает сама себя. Она используется для решения задач, которые могут быть разбиты на более мелкие задачи, которые могут быть решены тем же алгоритмом. Она может быть использована везде, где можно использовать циклы.

2. **Что называется базой рекурсии?**

Базой рекурсии называется условие, при котором рекурсивный процесс прекращается. Она определяет базовый случай, который не требует дальнейшего рекурсивного вызова функции.

3. **Как используется стек программы при вызове функций?**

Стек программы - это структура данных, которая используется для хранения информации о вызовах функций. Каждый раз, когда функция вызывается, информация о ней сохраняется в стеке. Когда функция завершается, информация о ней удаляется из стека. Стек используется для поддержки вызова функций и возврата из функций.

4. **Как получить текущее значение максимальной глубины рекурсии в языке Python?**

Для получения текущего значения максимальной глубины рекурсии в языке Python можно использовать функцию sys.getrecursionlimit(). Функция возвращает максимальную глубину рекурсии в качестве целого числа.

5. **Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python?**

Если число рекурсивных вызовов превысит максимальную глубину рекурсии в Python, то возникнет исключение RuntimeError: maximum recursion depth exceeded.

6. **Как изменить максимальную глубину рекурсии в языке Python?**

Для установки нового значения максимальной глубины рекурсии используется функция sys.setrecursionlimit(). Однако, изменение максимальной глубины рекурсии может быть опасно, так как это может привести к переполнению стека вызовов и привести к аварийному завершению программы.

7. **Каково назначение декоратора lru_cache?**

Декоратор lru_cache используется для кэширования результатов вызовов функции. Он позволяет сохранять результаты предыдущих вызовов функции в кэше и использовать их при повторных вызовах функции с теми же аргументами. Это может ускорить выполнение функции при повторных вызовах с теми же аргументами.

8. **Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов?**

Хвостовая рекурсия - это разновидность рекурсии, при которой рекурсивный вызов происходит в самом конце функции. В таком случае можно провести оптимизацию хвостовых вызовов, при которой рекурсивный вызов заменяется на цикл. Эта оптимизация позволяет избежать переполнения стека и ускоряет выполнение функции.

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