

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 РАБОТЕ №2.9
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Кондратенко Даниил Витальевич
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., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: рекурсия в языке Python.

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

Порядок выполнения работы:

Задание 1.

Изучил теоретический материал работы, создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами.

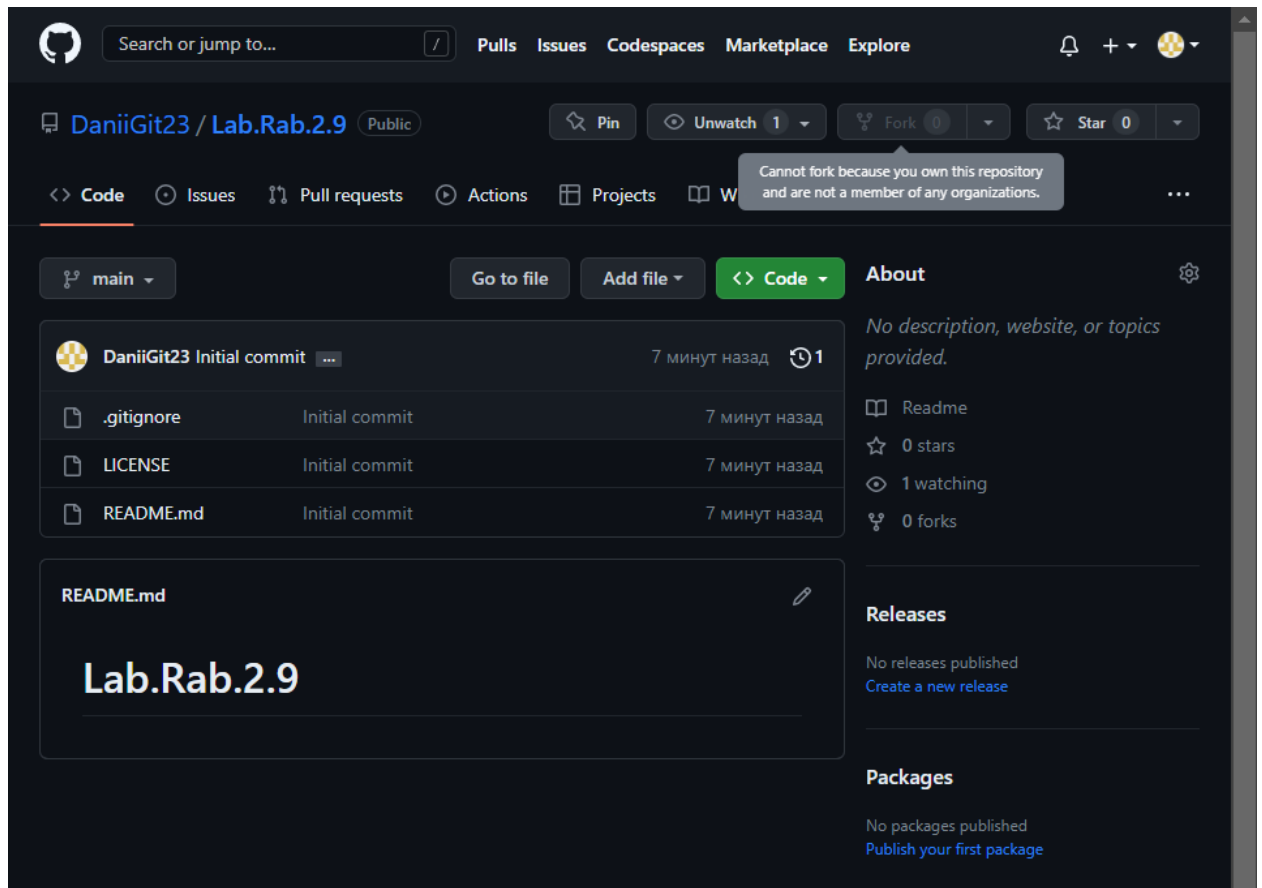


Рисунок 1 - Новый репозиторий

Задание 2.

Проклонировал свой репозиторий на свой компьютер.

Организовал свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop.

```
C:\Users\HUAWEI>git clone https://github.com/DaniiGit23/Lab.Rab.2.9.git
Cloning into 'Lab.Rab.2.9'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\HUAWEI>cd C:\Users\HUAWEI\Lab.Rab.2.9

C:\Users\HUAWEI\Lab.Rab.2.9>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\HUAWEI\Lab.Rab.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? [] t
Hooks and filters directory? [C:/Users/HUAWEI/Lab.Rab.2.9/.git/hooks]

C:\Users\HUAWEI\Lab.Rab.2.9>
```

Рисунок 2 – Клонирование и модель ветвления git-flow

Реализовывал примеры и индивидуальные задания на основе ветки develop, без создания дополнительной ветки feature/(название ветки) по указанию преподавателя.

Задание 3.

Создал проект PyCharm в папке репозитория.

Работа с примером №1.

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 $1 + 2 + 3 + 4 + \dots +$ (несколько часов спустя) $+$. А можно просто написать цикл for.

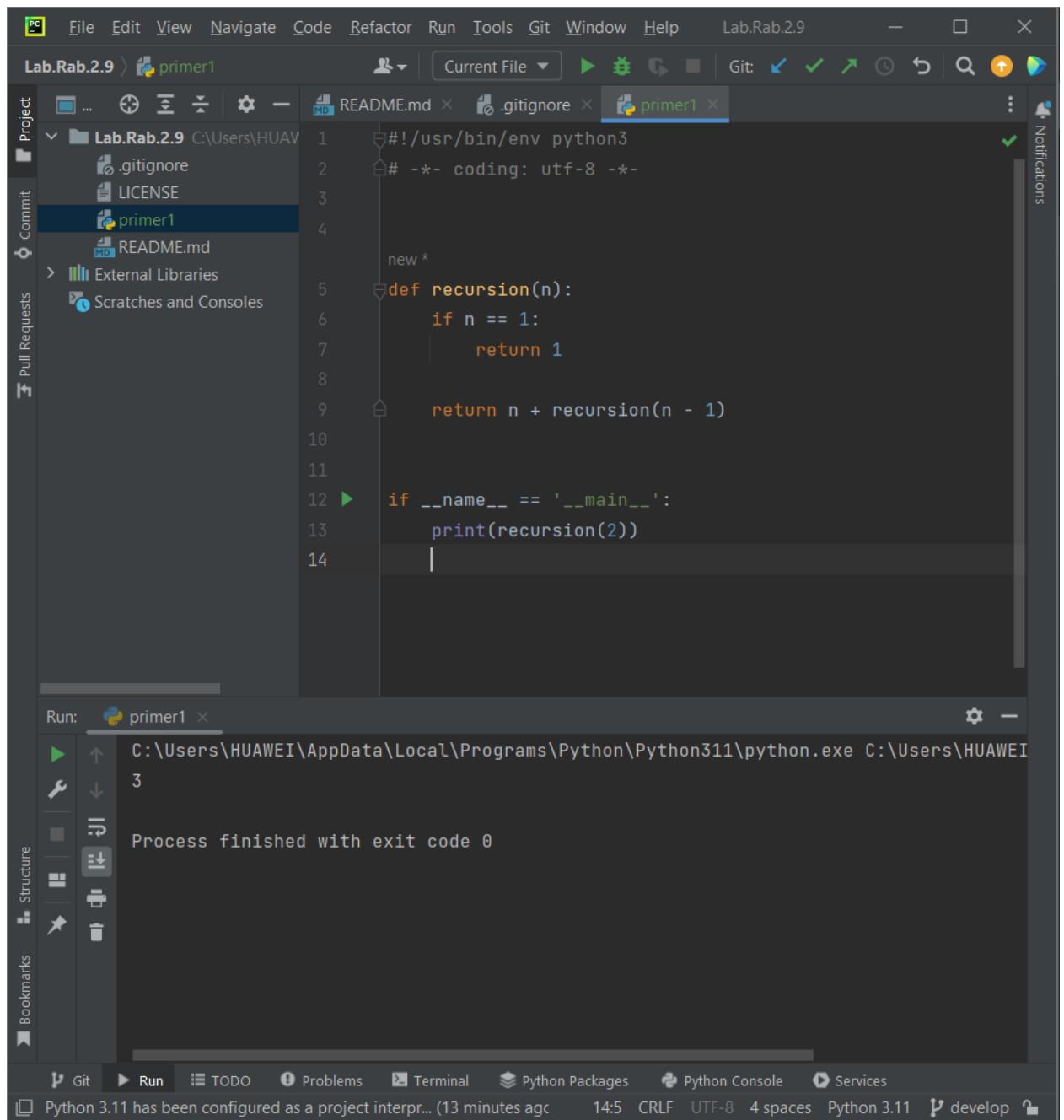


Рисунок 3 – Выполнение программы

Зафиксировал данные изменения.

Задание 4.

Выполнение примера 2.

Работа с последовательностью Фибоначчи.

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.

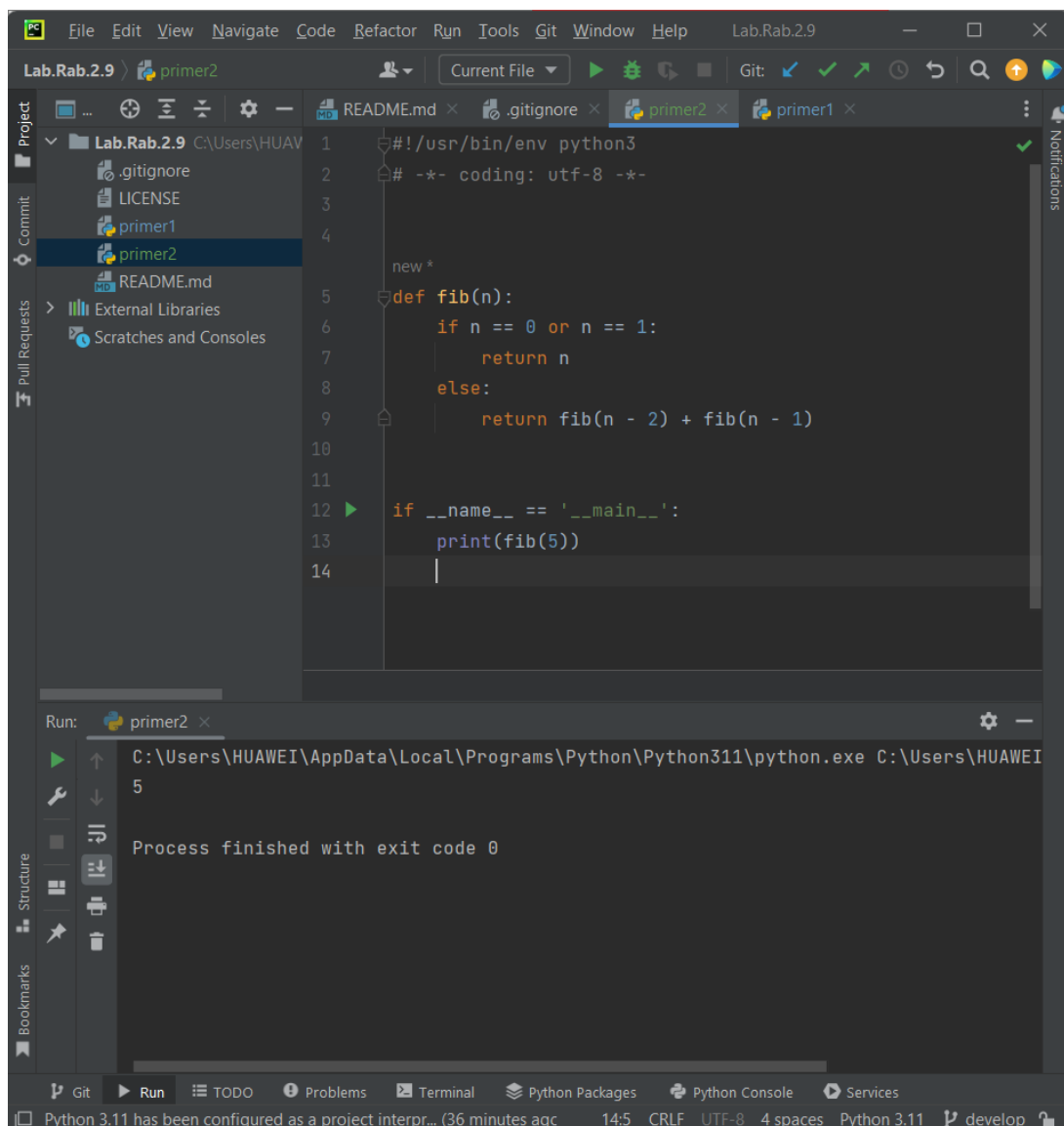


Рисунок 4 – Выполнение примера.

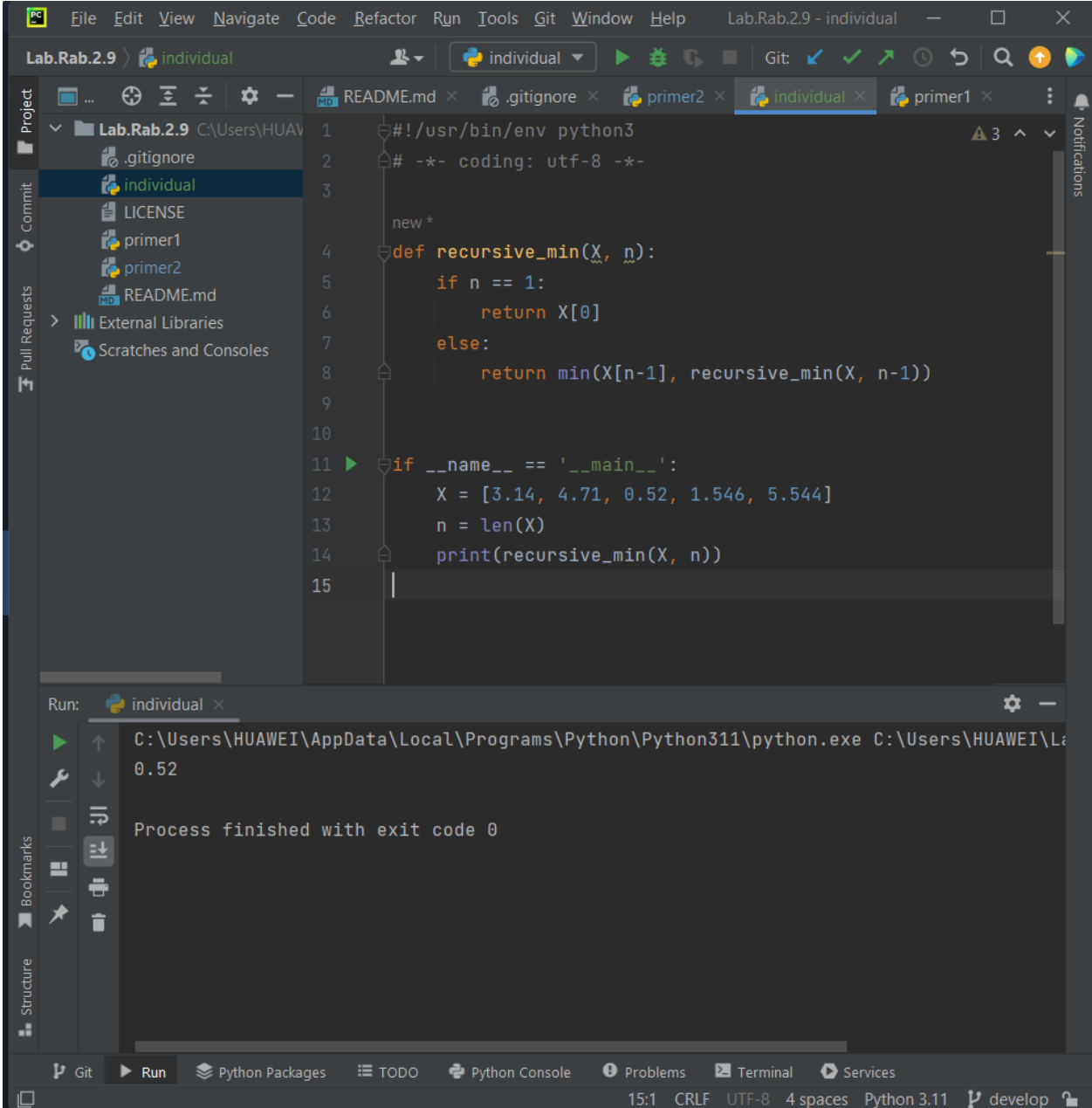
Зафиксировал данные изменения.

Задание 5.

Выполнение индивидуального задания.

Создал новый файл под названием *individual.py*

Условие индивидуального задания: дан список *X* из *n* вещественных чисел. Найти минимальный элемент списка, используя вспомогательную рекурсивную функцию, находящую минимум среди последних элементов списка *X*, начиная с *n*-го.



The screenshot shows an IDE window titled "Lab.Rab.2.9 - individual". The left sidebar displays the project structure with files: `.gitignore`, `individual` (selected), `LICENSE`, `primer1`, `primer2`, and `README.md`. The main editor displays the content of `individual.py`:

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  new *
5  def recursive_min(X, n):
6      if n == 1:
7          return X[0]
8      else:
9          return min(X[n-1], recursive_min(X, n-1))
10
11  if __name__ == '__main__':
12      X = [3.14, 4.71, 0.52, 1.546, 5.544]
13      n = len(X)
14      print(recursive_min(X, n))
15
```

At the bottom, the "Run" panel shows the execution command: `C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab.2.9\individual.py`. The output is `0.52`, and the status indicates "Process finished with exit code 0".

Рисунок 5 – Выполнение программы

Зафиксировал данные изменения.

Задание 6.

Слил ветку develop с веткой main и отправил на удаленный сервер.

```
C:\Users\HUAWEI\Lab.Rab.2.9>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

C:\Users\HUAWEI\Lab.Rab.2.9>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\HUAWEI\Lab.Rab.2.9>git merge develop
Merge made by the 'ort' strategy.
 individual | 14 ++++++
 primer1    | 13 ++++++
 primer2    | 13 ++++++
 3 files changed, 40 insertions(+)
 create mode 100644 individual
 create mode 100644 primer1
 create mode 100644 primer2
```

Рисунок 6 – Слияние веток

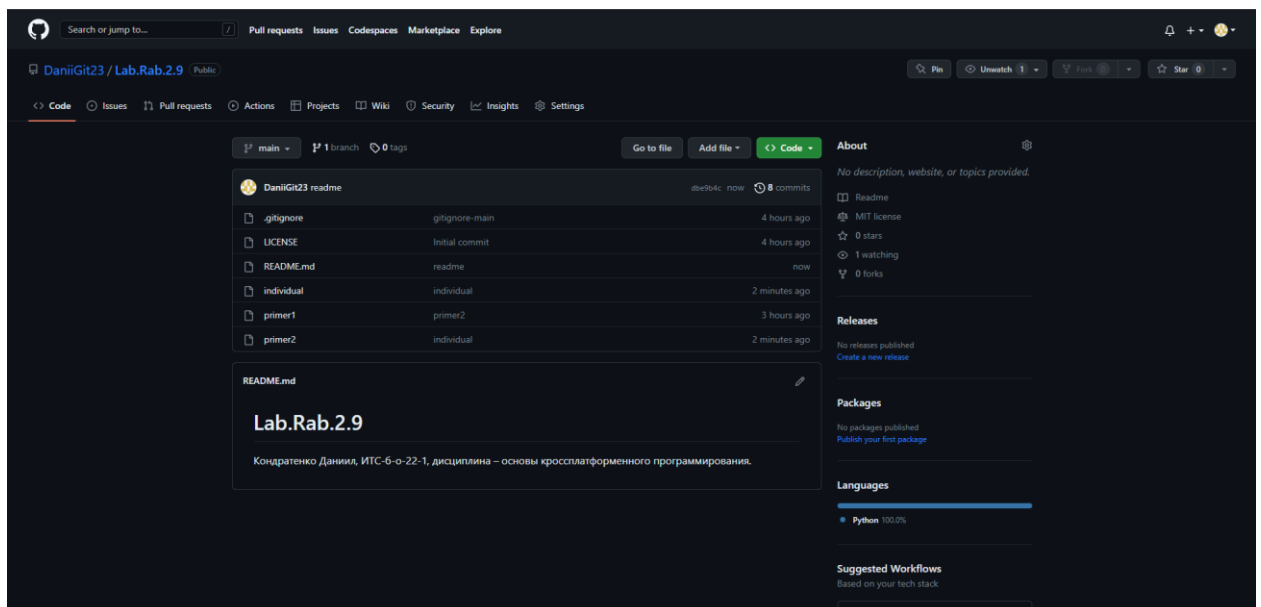


Рисунок 7 – Удаленный сервер

Ссылка на репозиторий: <https://github.com/DaniiGit23/Lab.Rab.2.9.git>

Контрольные вопросы:

1) Для чего нужна рекурсия?

Рекурсия — это процесс, когда функция вызывает саму себя, чтобы решить задачу. Она может быть использована везде, где можно использовать циклы.

Такой подход к программированию удобен, если задача может быть разбита на меньшие подзадачи, которые сводятся к аналогичному, но более простому (или наоборот) решению. Также рекурсия позволяет написать более читабельный и понятный код.

2) Как используется стек программы при вызове функций?

Стек программы - это область памяти, которая хранит временные данные о вызовах функций и прерываниях в работе программы.

При вызове функции, текущее место выполнения сохраняется в стеке, а управление передается вызываемой функции. Вызываемая функция выполняется, и после ее завершения, управление возвращается обратно в оригинальный код.

При каждом вызове функции, в стеке программы создается новый блок памяти, который содержит информацию об аргументах функции, локальных переменных и адрес возврата.

3) Что называется базой рекурсии?

База рекурсии в Python - это условие, при котором рекурсивная функция прекращает свою работу и возвращает результат без дополнительных вызовов самой себя. Обычно базой рекурсии является простой случай, который не требует дополнительных операций. Например, базой для функции вычисления факториала может быть случай, когда передается аргумент 0 или 1, а базой для функции поиска минимального элемента в списке - случай, когда список содержит только один элемент.

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` позволяет сохранять результат выполнения функции с определенным набором аргументов в кэше, чтобы в дальнейшем возвращать его, не вычисляя повторно. Это позволяет повысить эффективность программы, если функция вызывается несколько раз с одним и тем же набором аргументов.

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.

Оптимизация хвостовых вызовов проводится при помощи декоратора `@tail_recursion`, который заменяет рекурсивный вызов на цикл. В Python этот декоратор не встроен и требует отдельной установки через `pip`.

Вывод: в ходе данной лабораторной работы я приобрел навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.