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 №12
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: «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

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

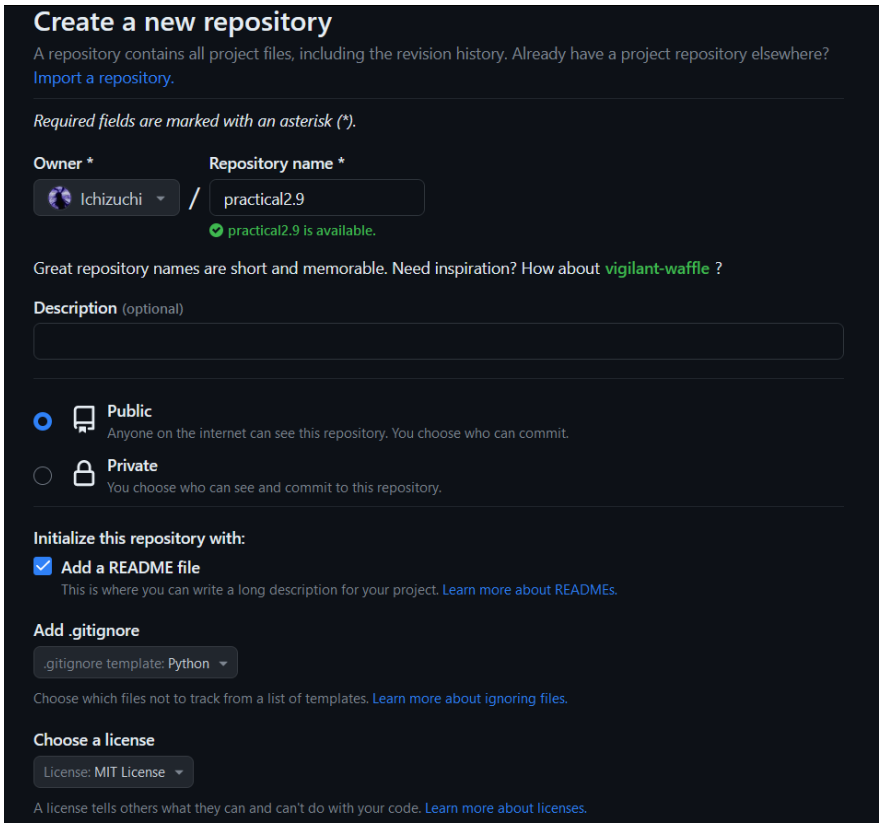


Рисунок 1. Новый репозиторий

2. Скопировала репозиторий на свой компьютер.

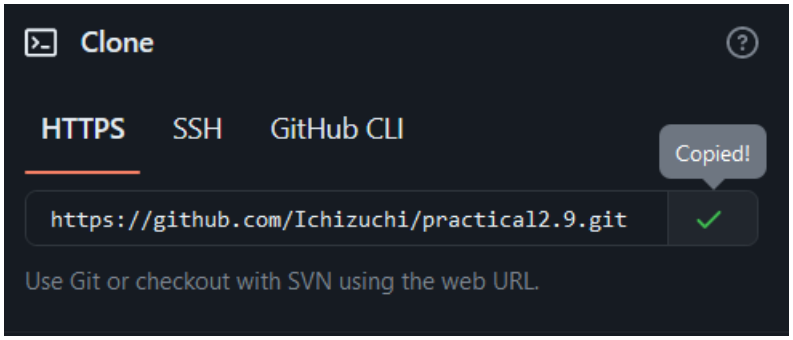
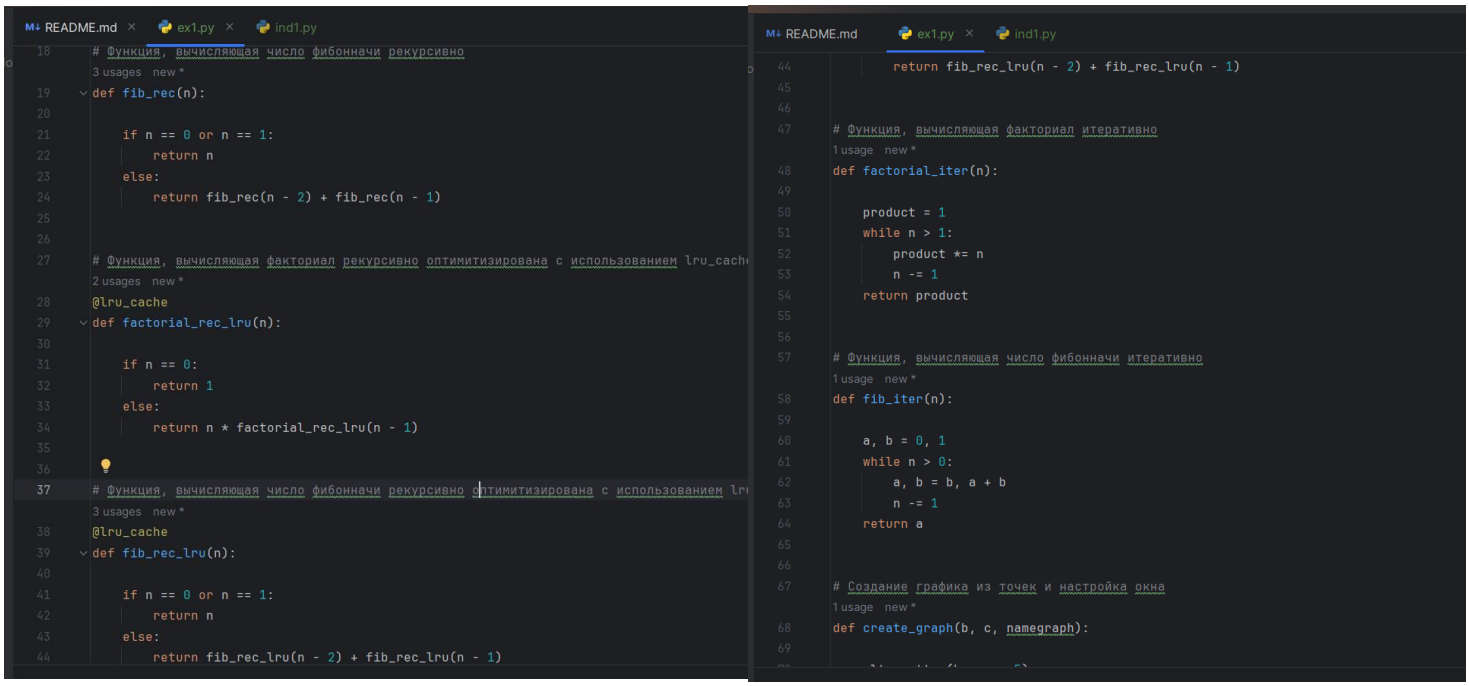


Рисунок 2. Клонирование репозитория

3. Использовала систему ветвления git-flow

Рисунок 3. Ветка develop

4. Задание №1: самостоятельно изучите работу со стандартным пакетом Python timeit. Оцените с помощью этого модуля скорость работы итеративной и рекурсивной версий функций factorial и fib. Во сколько раз измениться скорость работы рекурсивных версий функций factorial и fib при использовании декоратора lru_cache.



```python
# Функция, вычисляющая число фибонначи рекурсивно
def fib_rec(n):

    if n == 0 or n == 1:
        return n
    else:
        return fib_rec(n - 2) + fib_rec(n - 1)


# Функция, вычисляющая факториал рекурсивно оптимитизирована с использованием lru_cache
@lru_cache
def factorial_rec_lru(n):

    if n == 0:
        return 1
    else:
        return n * factorial_rec_lru(n - 1)

# Функция, вычисляющая число фибонначи рекурсивно оптимитизирована с использованием lru_cache
@lru_cache
def fib_rec_lru(n):

    if n == 0 or n == 1:
        return n
    else:
        return fib_rec_lru(n - 2) + fib_rec_lru(n - 1)
```

```python
        return fib_rec_lru(n - 2) + fib_rec_lru(n - 1)


# Функция, вычисляющая факториал итеративно
def factorial_iter(n):

    product = 1
    while n > 1:
        product *= n
        n -= 1
    return product


# Функция, вычисляющая число фибонначи итеративно
def fib_iter(n):

    a, b = 0, 1
    while n > 0:
        a, b = b, a + b
        n -= 1
    return a


# Создание графика из точек и настройка окна
def create_graph(b, c, namegraph):
```

Рисунок 4. Код программы

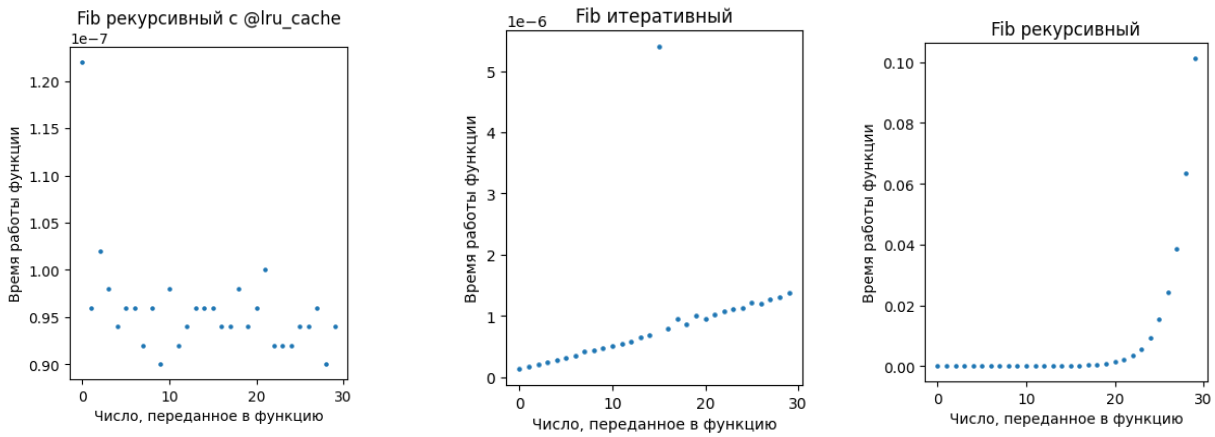


Рисунок 5. Сравнение функций fib

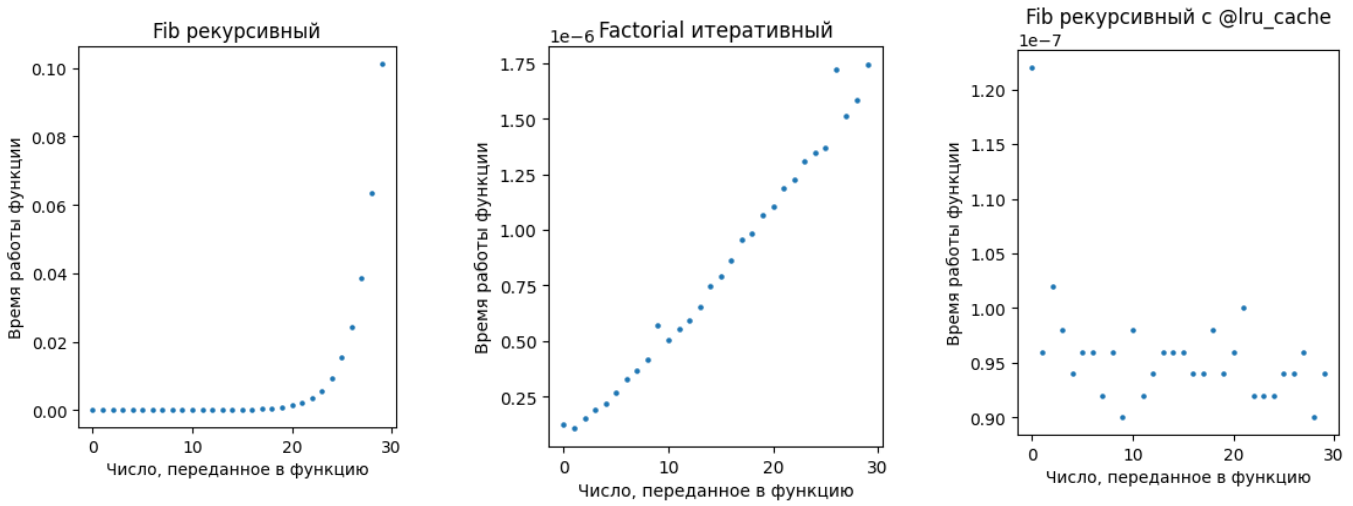


Рисунок 6. Сравнений функций factorial

5. Индивидуальное задание №1: создайте рекурсивную функцию, печатающую все возможные перестановки для целых чисел от 1 до N.

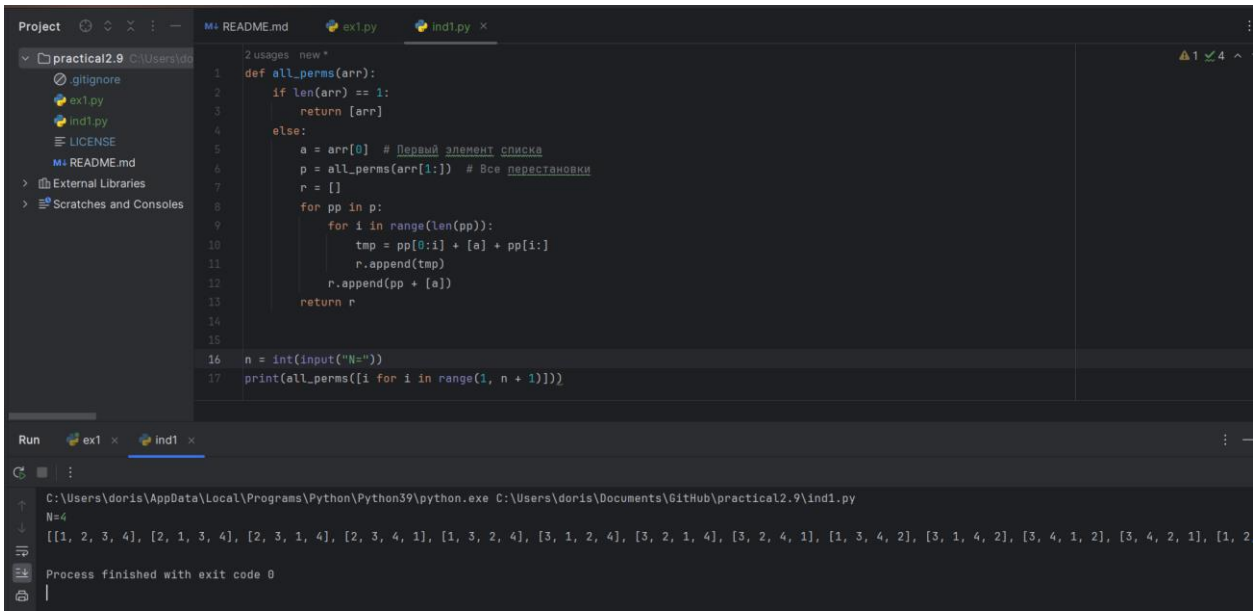


Рисунок 6. Программа и ее результат

6. Слила ветку develop с веткой main и отправила на удаленный сервер – Github.

Ответы на контрольные вопросы

1. Для чего нужна рекурсия?

Рекурсия используется в программировании для решения задач, которые могут быть разбиты на более простые подзадачи. Функция в рекурсивном вызове вызывает сама себя для решения подзадач, что делает код более читаемым и естественным.

2. Что называется базой рекурсии?

База рекурсии - это условие, которое определяет конец рекурсивного процесса. Когда выполняется база рекурсии, функция прекращает вызывать саму себя.

3. Самостоятельно изучите что является стеком программы. Как используется стек программы при вызове функций?

Стек программы - это структура данных, используемая для хранения локальных переменных и данных вызываемых функций. При вызове функции, ее локальные переменные и адрес возврата помещаются в стек. Когда функция завершается, данные извлекаются из стека.

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

Текущую максимальную глубину рекурсии можно получить с помощью sys.getrecursionlimit() из модуля sys.

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

Превышение максимальной глубины рекурсии приведет к ошибке RecursionError. Это может произойти из-за неэффективной рекурсии, и решением может быть переписывание кода с использованием циклов или оптимизированных подходов.

6. Как изменить максимальную глубину рекурсии в языке Python?

Максимальную глубину рекурсии можно изменить с помощью sys.setrecursionlimit(new_limit). Однако, изменение этого значения может повлиять на стабильность работы программы, поэтому следует использовать осторожно.

7. Каково назначение декоратора lru_cache?

Декоратор lru_cache используется для кэширования результатов вызовов функции. Это увеличивает производительность, избегая повторных вычислений для одних и тех же аргументов.

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

Хвостовая рекурсия — это форма рекурсии, при которой рекурсивный вызов является последней операцией в функции. Оптимизация хвостовых вызовов (tail call optimization, TCO) в Python официально не поддерживается, но в некоторых случаях интерпретатор может проводить оптимизации, основанные на конкретных условиях кода. Однако, в общем случае, Python не оптимизирует хвостовые вызовы так, как делают некоторые другие языки, такие как Scheme.

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