Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

**«Рекурсия в языке Python»**

**ОТЧЕТ**

**по лабораторной работе №12**

**дисциплины**

**«Основы программной инженерии»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Мизин Глеб Егорович  2 курс, группа ПИЖ-б-о-21-1,  09.03.04 «Программная инженерия», направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверил:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2022 г.

**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? Приведите в отчет и обоснуйте полученные результаты.



Рисунок 1 – Код задания №1 (поиск факториала)

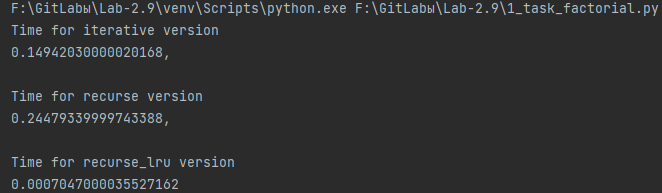


Рисунок 2 – Результат работы кода задания №1 (поиск факториала)



Рисунок 3 – Код задания №1 (числа Фибоначчи)

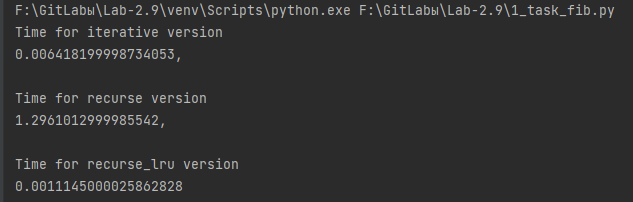


Рисунок 4 – Результат работы кода задания №1 (числа Фибоначчи)

**Задание №2**: самостоятельно проработайте пример с оптимизацией хвостовых вызовов в Python. С помощью пакета timeit оцените скорость работы функций factorial и fib с использованием интроспекции стека и без использования интроспекции стека. Приведите полученные результаты в отчет.

**Индивидуальное задание**: Создайте рекурсивную функцию, печатающую все подмножества множества 

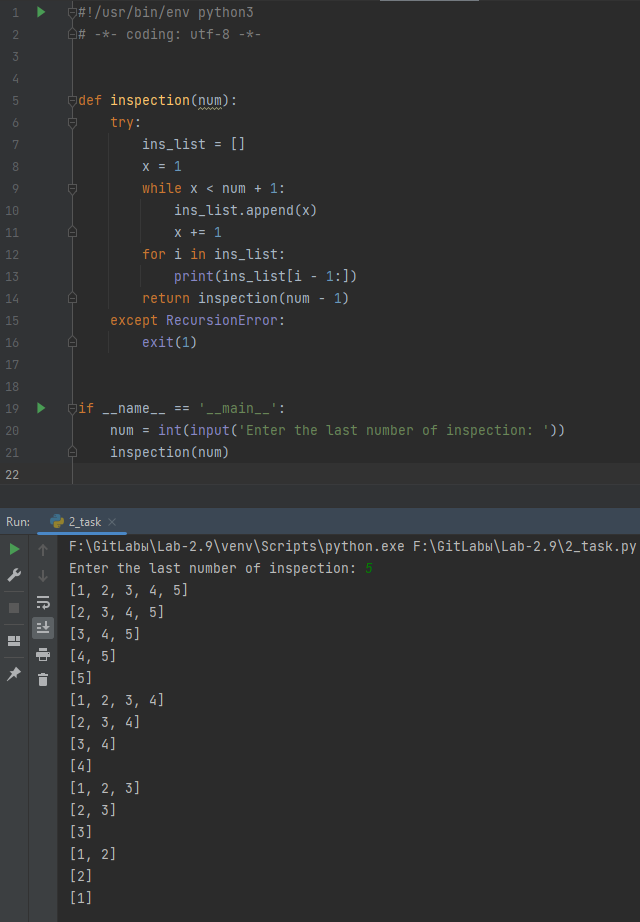
\

Рисунок – Код и результат работы программы индивидуального задания

**Контрольные вопросы**

1. Для чего нужна рекурсия?

Функция может содержать вызов других функций. В том числе процедура может вызвать саму себя.

2. Что называется базой рекурсии?

У рекурсии, как и у математической индукции, есть база — аргументы, для которых значения функции определены

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

Максимальная глубина рекурсии ограничена движком JavaScript. Точно можно рассчитывать на 10000 вложенных вызовов, некоторые интерпретаторы допускают и больше, но для большинства из них 100000 вызовов – за пределами возможностей. Существуют автоматические оптимизации, помогающие избежать переполнения стека вызовов («оптимизация хвостовой рекурсии»), но они ещё не поддерживаются везде и работают только для простых случаев.

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

Функция sys. getrecursionlimit() возвращает текущее значение предела рекурсии, максимальную глубину стека интерпретатора Python.

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

Ошибка RunTimeError

6. Как изменить максимальную глубину рекурсии в языке Python?

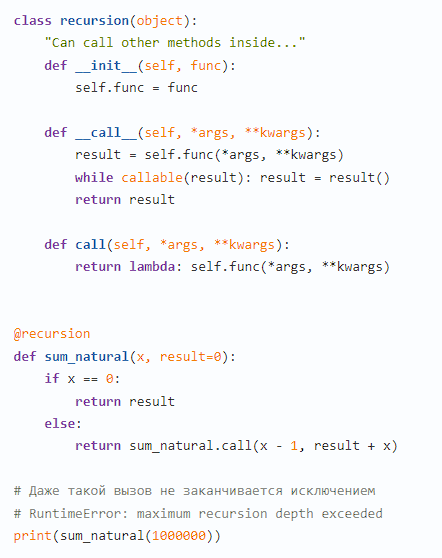
C помощью функции setrecursionlimit() модуля sys

7. Каково назначение декоратора lru\_cache ?

Декоратор @lru\_cache() модуля functools оборачивает функцию с переданными в нее аргументами и запоминает возвращаемый результат соответствующий этим аргументам. Такое поведение может сэкономить время и ресурсы, когда дорогая или связанная с вводом/выводом функция периодически вызывается с одинаковыми аргументами.

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

Хвостовая рекурсия — частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции. Оптимизация хвостовой рекурсии выглядит так:



**Прочесть про lru\_cache и запустить код для 2 задания на python 2.4**