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»
Вариант 5.

Выполнила:
Михеева Елена Александровна
2 курс, группа ИВТ-б-з-20-1,
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., канд. техн. 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.

1. Была создана программа, в которой изучается работа со стандартным пакетом timeit. Была проведена оценка скорости работы рекурсивных версий функций factorial и fib при использовании lru_cache.

```
Итеративная версия функции factorial: 0.057513175008352846
Рекурсивная версия функции factorial: 0.0006897240236867219
Итеративная версия функции fib: 0.0004716459952760488
Рекурсивная версия функции fib: 0.009399462025612593
Рекурсивная версия функции fib с lru_cache: 6.758299423381686e-05
Рекурсивная версия функции factorial с lru_cache: 6.455200491473079e-05
(base) mikheeva@MacBook-Pro-Elena Python_2.9 %
```

Рисунок 1. Выполнение программы task.py

Анализируя полученные результаты, можно сделать вывод, что итеративная функция поиска факториала работает гораздо медленнее рекурсивной. Рекурсивная функция поиска чисел Фибоначчи, наоборот работает медленнее, чем итеративная. А при использовании декоратора lru_cache, обе рекурсивные функции работают гораздо быстрее.

2. Было выполнено индивидуальное задание согласно варианту 5: создайте рекурсивную функцию, печатающую все подмножества множества $\{1, 2, \ldots, N\}$.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


def print_subsets(s, current_set=[]):
    """
    Рекурсивная функция, печатающая все подмножества множества.
    """
    if not s:
        print(current_set)
        return

    print_subsets(s[1:], current_set + [s[0]])
    print_subsets(s[1:], current_set)


if __name__ == "__main__":
    input_set = input("Введите элементы множества через пробел: ")
    example_set = input_set.split()

    print_subsets(example_set)

```

Рисунок 3. Программа для индивидуального задания

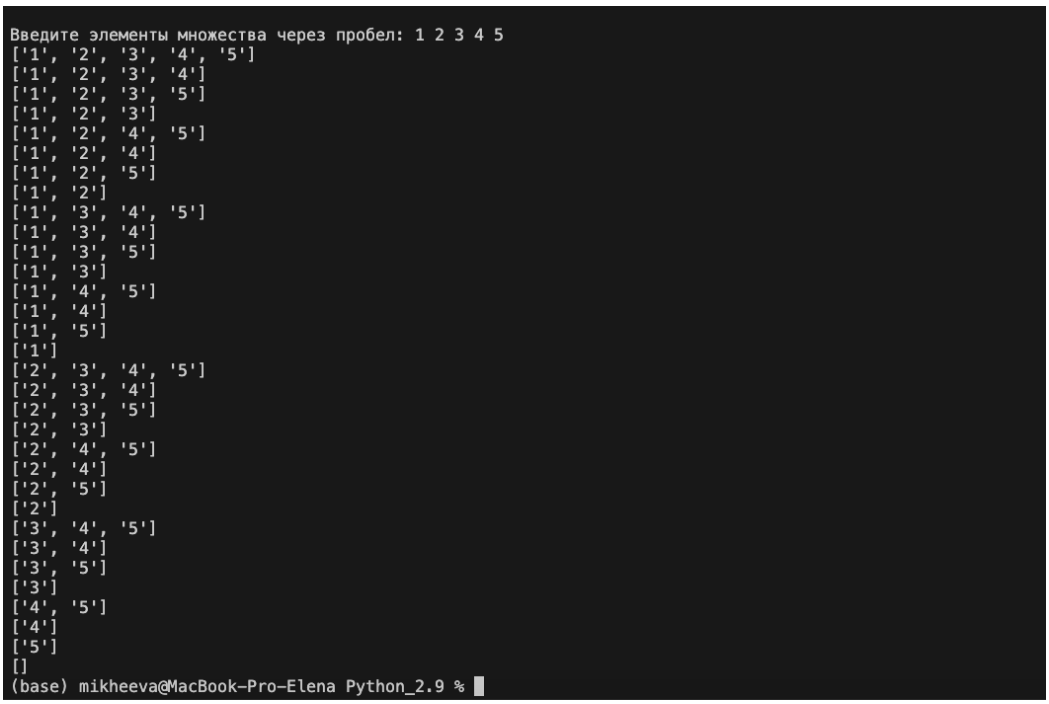

Рисунок 4. Результат выполнения программы

Ответы на контрольные вопросы.

1. Для чего нужна рекурсия?

Рекурсия используется в программировании для решения задач, которые могут быть разбиты на более мелкие подзадачи того же типа. Она позволяет функции вызывать саму себя, что упрощает решение сложных задач путем

разделения их на более простые подзадачи. Рекурсия также широко используется в алгоритмах, таких как алгоритмы обхода деревьев и графов.

2. Что называется базой рекурсии?

Условие, при котором рекурсивные вызовы функции прекращаются и начинается возврат из рекурсивных вызовов. Это базовый случай, который предотвращает бесконечное выполнение рекурсивной функции и обеспечивает завершение процесса.

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

Стек программы – это структура данных, которая хранит информацию о вызовах функций во время выполнения программы. При вызове функции, информация о текущем состоянии функции, такая как локальные переменные и адрес возврата, помещается в стек. Когда функция завершает выполнение,

информация извлекается из стека, и управление передается обратно вызывающей функции. Это позволяет программе возвращаться к предыдущему состоянию после завершения выполнения функции.

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

В языке Python можно получить текущее значение максимальной глубины рекурсии с помощью функции sys.getrecursionlimit(). Она возвращает текущее максимальное количество рекурсивных вызовов, которое может быть выполнено до возникновения ошибки "RecursionError".

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

Если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python, возникнет ошибка "RecursionError". Это произойдет, когда программа пытается выполнить больше рекурсивных вызовов, чем разрешено текущей максимальной глубиной рекурсии.

6. Как изменить максимальную глубину рекурсии в языке Python?

Максимальную глубину рекурсии в языке Python можно изменить с помощью функции sys.setrecursionlimit(). Однако, изменение этого значения должно быть осуществлено с осторожностью, так как слишком большая глубина рекурсии может привести к переполнению стека и ошибкам выполнения.

7. Каково назначение декоратора lru_cache ?

Используется для кеширования результатов вызовов функции с определенными аргументами. Он сохраняет результаты предыдущих вызовов функции, чтобы избежать повторных вычислений при повторных вызовах с теми же аргументами. Это может значительно улучшить производительность функций, особенно при выполнении тяжелых вычислений.

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

Это особый вид рекурсии, при котором рекурсивный вызов является последней операцией в функции. Оптимизация хвостовых вызовов заключается в том, что компилятор или интерпретатор может заменить рекурсивный вызов на цикл, что позволяет избежать увеличения стека вызовов. Это позволяет снизить использование памяти и улучшить производительность программы.