МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №4

Рекурсия в языке Python

По дисциплине «Технологии программирования и алгоритмизация»

Выполнил студент группы ИВТ-б-о-20-1

Галяс Д. И. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2021

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

Ход работы:

Ссылка на репозиторий: https://github.com/DIMITRY-GALYAS1/Rabota-11.git

1. Создал новый репозиторий на github, после клонировал его и создал в папке репозитория новый проект PyCharm.

2. Выполнил первое задание. С помощью пакета timeit оценил скорость работы итеративной и рекурсивной версий функций factorial и fib. Также проверил скорость работы с использованием декоратора lru_cache. Быстрее всего вычисления выполняются с декоратором. На основании проделанной работы можно сделать вывод о том, что декоратор позволяет увеличить скорость работы программы.

```
C:\Users\lizeq\anaconda3\envs\pythonProject4\python.exe C:/Users/lizeq/PycharmProjects/pythonProject4/zadanie_1.py
Время выполнения рекрусивной функции числа Фибоначи:  0.00015640000000000098
Время выполнения итеративной функции числа Фибоначи:  0.00010500000000000093
Время выполнения рекрусивной функции числа Фибоначи с использованием декоратора lru_cache:  0.012756
Время выполнения рекрусивной функции факториала:  0.00016199999999999548
Время выполнения итеративной функции факториала:  0.000516599999999999
Время выполнения рекрусивной функции факториала с использованием декоратора lru_cache:  0.0222368

Process finished with exit code 0
```

Рисунок 1. Первое задание

3. Выполнил второе задание. Проработал пример с оптимизацией хвостовых вызовов в Python. С помощью пакета timeit оценил скорость работы функций factorial и fib с использованием интроспекции стека и без использования интроспекции стека. Использование интроспекции стека сильно уменьшает время выполнения программы.

```
C:\Users\lizeq\anaconda3\envs\pythonProject4\python.exe C:/Users/lizeq/PycharmProjects/pythonProject4/zadanie_2.py
Время выполнения функции factorial():  0.0008055000000000007
Время выполнения функции factorial() с использованием интроспекции стека:  0.0484608
Время выполнения функции fib():  0.0010754999999999931
Время выполнения функции fib() с использованием интроспекции стека:  0.055956000000000006

Process finished with exit code 0
```

Рисунок 2. Второе задание

4. Выполнил индивидуальное задание.

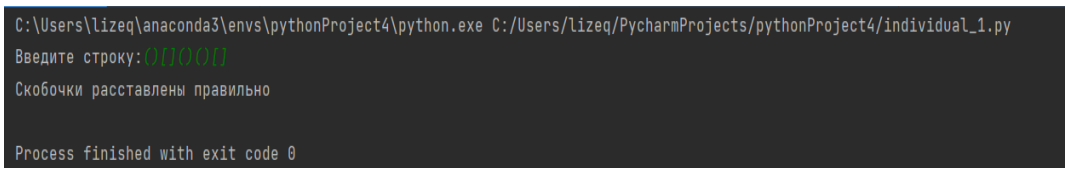

Рисунок 3. Индивидуальное задание

```
"""
В строке могут присутствовать скобки как круглые, так и квадратные скобки.
Каждой открывающей скобке соответствует закрывающая того же типа
(круглой - круглая,квадратной- квадратная). Напишите рекурсивную функцию,
проверяющую правильность расстановки скобок в этом случае.
"""


def check_par(s):
    """Проверка скобочек"""
    if len(s) == 0:
        return True
    else:
        f = s.find('()')
        x = s.find('[]')
        s = s.replace('()', '')
        s = s.replace('[]', '')
        if f == -1 and x == -1:
            return False
        else:
            return check_par(s[0:-1])


if __name__ == '__main__':

    if check_par(input('Введите строку:')):
        print('Скобочки расставлены правильно')
    else:
        print('Скобочки расставлены не правильно')
```

Рисунок 4. Код индивидуального задания

Контрольные вопросы:

1. Для чего нужна рекурсия?

Функция может содержать вызов других функций. В том числе процедура может вызвать саму себя. Никакого парадокса здесь нет – компьютер лишь последовательно выполняет встретившиеся ему в программе команды и, если встречается вызов процедуры, просто начинает выполнять эту функцию. Без разницы, какая функция дала команду это делать.

2. Что называется базой рекурсии?

База рекурсии – это такие аргументы функции, которые делают задачу настолько простой, что решение не требует дальнейших вложенных вызовов.

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

Стек в Python – это линейная структура данных, в которой данные расположены объектами друг над другом. Он хранит данные в режиме LIFO (Last in First Out). Данные хранятся в том же порядке, в каком на кухне тарелки располагаются одна над другой. Мы всегда выбираем последнюю тарелку из стопки тарелок. В стеке новый элемент вставляется с одного конца, и элемент может быть удален только с этого конца.

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

Чтобы проверить текущие параметры лимита, нужно запустить: sys.getrecursionlimit().

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

Существует предел глубины возможной рекурсии, который зависит от реализации Python. Когда предел достигнут, возникает исключение RuntimeError.

6. Как изменить максимальную глубину рекурсии в языке Python?

Изменить максимальную глубину рекурсии можно с помощью sys.setrecursionlimit().

7. Каково назначение декоратора lru_cache ?

Декоратор lru_cache является полезным инструментом, который можно использовать для уменьшения количества лишних вычислений. Декоратор оборачивает функцию с переданными в нее аргументами и запоминает возвращаемый результат, соответствующий этим аргументам.

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. Подобный вид рекурсии примечателен тем, что может быть легко

заменён на итерацию путём формальной и гарантированно корректной перестройки кода функции. Оптимизация хвостовой рекурсии путём преобразования её в плоскую итерацию реализована во многих оптимизирующих компиляторах. В некоторых функциональных языках программирования спецификация гарантирует обязательную оптимизацию хвостовой рекурсии.

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