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 «Теории программирования и алгоритмизации»

Выполнил студент группы ИВТ-б-о-20-1

Плотников Д. В. « »_____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

1. Создал новый собственный репозиторий. Ссылка на репозиторий: https://github.com/Dmitry-15/11_laba.git.

2. С помощью команды git clone клонировал удаленный репозиторий на свой ПК. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

3. Проработал два примера лабораторной работы.

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

import timeit
from functools import lru_cache


@lru_cache
def non_rec_factorial(n=5):
    for i in range(1, n+1):
        n += i


@lru_cache
def rec_factorial(n=5):
    if n == 1:
        return 1
    return n + rec_factorial(n - 1)


print(timeit.timeit(stmt=non_rec_factorial, number=10000))
print(timeit.timeit(stmt=rec_factorial, number=10000))
```

Рисунок 1. Код первого примера с использованием декоратора lru_cache

```
C:\ProgramData\Anaconda3\python.exe C:/Users/Дмитрий/11_laba/Zadaniy/primer1.py
0.0010500000000000231
0.0013838000000000183

Process finished with exit code 0
```

Рисунок 2. Выполнение первого примера с использованием декоратора lru_cache

```
C:\ProgramData\Anaconda3\python.exe C:/Users/Дмитрий/11_laba/Zadaniy/primer1.py
0.008464799999999995
0.013525700000000002

Process finished with exit code 0
```

Рисунок 3. Выполнение первого примера без использования декоратора lru_cache

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

from timeit import timeit


code1 = """
def factorial(n, acc=1):
    if n == 0:
        return acc
    return factorial(n-1, n*acc)
"""
code2 = """
def fib(i, current = 0, next = 1):
    if i == 0:
        return current
    else:
        return fib(i - 1, next, current + next)
"""
code3 = """
class TailRecurseException:
```

Рисунок 4. Код второго примера

```
C:\ProgramData\Anaconda3\python.exe C:/Users/Дмитрий/11_laba/primer2.py
Время выполнения функции factorial():  0.00011060000000000236
Время выполнения функции factorial() с использованием интроспекции стека:  0.00011120000000000574
Время выполнения функции fib():  0.00011060000000000236
Время выполнения функции fib() с использованием интроспекции стека:  0.00011060000000000236

Process finished with exit code 0
```

Рисунок 5. Выполнение второго примера

Индивидуальное задание

Вариант 14

1. Выполнил индивидуальное задание.

14. Напишите рекурсивную функцию, которая вычисляет $y = \sqrt[k]{x}$ по следующей формуле:

$$y_0 = 1; y_{n+1} = y_n + \frac{x/y_n^{k-1} - y_n}{k},$$

$n = 0, 1, 2, \ldots$. За ответ принять приближение, для которого выполняется $|y_n - y_{n+1}| < \varepsilon$, где $\varepsilon = 0,0001$.

Рисунок 6. Условие

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


def sqr_k(x, k, yp=1.0, eps=1.0e-5):
    yc = yp + (x / yp ** (k-1) - yp) / k
    if abs(yc - yp) <= eps:
        return yc
    else:
        return sqr_k(x, k, yc)


if __name__ == '__main__':
    x1 = float(input("Введите x - "))
    k1 = int(input("Введите степень k - "))
    sqr = sqr_k(x1, k1)
    print("Корень степени", k1, "из", x1, "=", sqr)
    print("Проверка:", sqr, "**", k1, "=", sqr ** k1)
```

Рисунок 7. Код задания

```
C:\ProgramData\Anaconda3\python.exe C:/Users/Дмитрий/11_laba/individ.py
Введите x - 25
Введите степень k - 2
Корень степени 2 из 25.0 = 5.0
Проверка - 5.0 ** 2 = 25.0

Process finished with exit code 0
```

Рисунок 8. Выполнение 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 – аргументы, для которых значения функции определены (элементарные задачи).

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

При вызове подпрограммы или возникновении прерывания, в стек заносится адрес возврата – адрес в памяти следующей инструкции приостановленной программы и управление передается подпрограмме или подпрограмме-обработчику.

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: Maximum Recursion Depth Exceeded.

6. Как изменить максимальную глубину рекурсии в языке Python?

Можно изменить предел глубины рекурсии с помощью вызова: sys.setrecursionlimit(limit).

7. Каково назначение декоратора lru_cache?

Декоратор lru_cache можно использовать для уменьшения количества лишних вычислений.

8. Что такое хвостовая рекурсия? Как проводится оптимизация хвостовыхвызовов?

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 успешно приобрел навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python3.