

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

Кафедра инфокоммуникаций

Отчет по лабораторной работе №9

Рекурсия в языке Python.

по дисциплине «Технологии программирования и алгоритмизации»

Выполнил студент группы ИВТ-б-о-21-1

Лысенко И.А. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

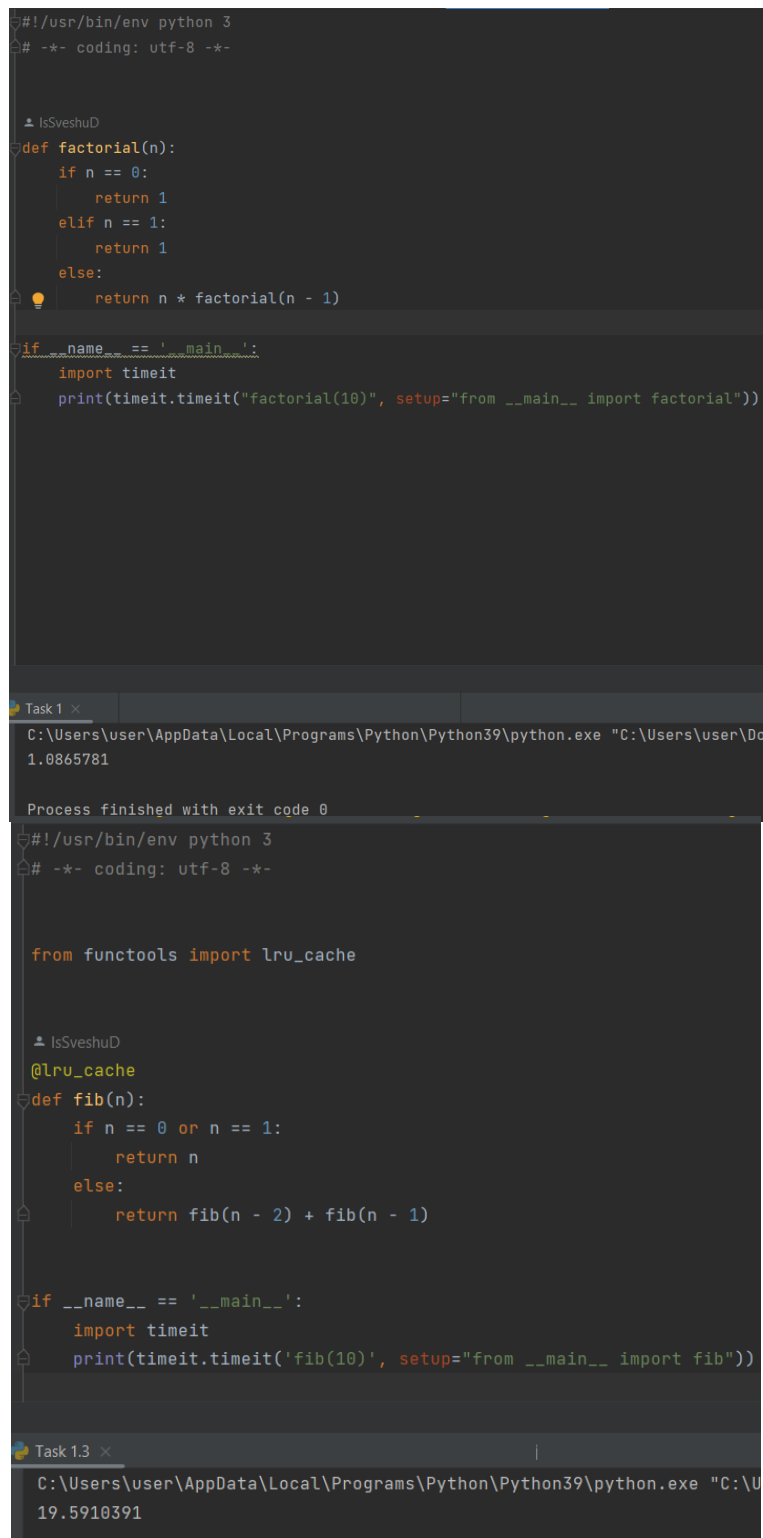
(подпись)

Ставрополь 2022

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

Ход работы:

1. Изучил пакет timeit.



```
#!/usr/bin/env python 3
# -*- coding: utf-8 -*-

IsSveshuD
def factorial(n):
    if n == 0:
        return 1
    elif n == 1:
        return 1
    else:
        return n * factorial(n - 1)

if __name__ == '__main__':
    import timeit
    print(timeit.timeit("factorial(10)", setup="from __main__ import factorial"))

Task 1 x
C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\user\Documents\Task 1.py"
1.0865781

Process finished with exit code 0

#!/usr/bin/env python 3
# -*- coding: utf-8 -*-

from functools import lru_cache

IsSveshuD
@lru_cache
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 2) + fib(n - 1)

if __name__ == '__main__':
    import timeit
    print(timeit.timeit('fib(10)', setup="from __main__ import fib"))

Task 1.3 x
C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\user\Documents\Task 1.3.py"
19.5910391
```

Рисунок 1 – Пакет timeit

2. Изучил и проработал приведённый пример.

```
import sys

IsSveshuD
class TailRecurseException:
    IsSveshuD
    def __init__(self, args, kwargs):
        self.args = args
        self.kwargs = kwargs

IsSveshuD
def tail_call_optimized(g):
    """
    Эта программа показывает работу декоратора, который производит оптимизацию
    хвостового вызова. Он делает это, вызывая исключение, если оно является его
    прародителем, и перехватывает исключения, чтобы подделать оптимизацию хвоста.

    Эта функция не работает, если функция декоратора не использует хвостовой вызов.
    """

    IsSveshuD
    def func(*args, **kwargs):
        f = sys._getframe()
        if f.f_back and f.f_back.f_back and f.f_back.f_back.f_code == f.f_code:
            raise TailRecurseException(args, kwargs)
        else:
            while True:
                try:
                    return g(*args, **kwargs)
                except TailRecurseException as e:
                    args = e.args
                    kwargs = e.kwargs
    func.__doc__ = g.__doc__
    return func

IsSveshuD
@tail_call_optimized
def factorial(n, acc=1):
    """calculate a factorial"""
    if n == 0:
        return acc
    return factorial(n-1, n*acc)

if __name__ == '__main__':
    print(factorial(10000))
```

Рисунок 2 – Пример

3. Выполнил индивидуальное задание согласно своему варианту.

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

def fum(m, n):
    if m == 0:
        return n + 1
    if m > 0 and n == 0:
        return fum(m - 1, 1)
    if m > 0 and n > 0:
        return fum(m - 1, fum(m, n - 1))

m = int(input("m = "))
n = int(input("n = "))

if __name__ == '__main__':
    print(fum(m, n))
```

Рисунок 3 – Индивидуальное задание

4. Результат выполнения индивидуального задания.

```
m = 4
n = 0
13
Process finished with exit code 0
```

Рисунок 4 – Результат выполнения индивидуального задания

Ответы на вопросы:

1. Для чего нужна рекурсия?

Для вызова функции до завершения первоначального вызова функции.

2. Что называется базой рекурсии?

База рекурсии – это тривиальный случай, при котором решение задачи очевидно, то есть не требуется обращение функции к себе.

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. Мы можем выполнять две операции в стеке – PUSH и POP. Операция PUSH – это когда мы добавляем элемент, а операция POP – когда мы удаляем элемент.

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?

Возникает исключение `RuntimeError`

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 хвостовых вызовов?

Хвостовой вызов – это просто вызов рекурсивной функции, который является последней операцией и должна быть выполнена перед возвратом значения. Чтобы было понятно, `return foo (n - 1)` – это хвост вызова, но `return foo (n - 1) + 1` не является (поскольку операция сложения будет последней операцией).

Оптимизация хвостового вызова – это способ автоматического сокращения рекурсии в рекурсивных функциях. Можно провести оптимизацию вручную и с помощью декоратора `lru_cache`.

Вывод: в ходе лабораторной работы были изучены все аспекты работы с

рекурсиями в языке Python, стеки программ.