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

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 11

Рекурсия в языке Python

по дисциплине «Технологии программирования и алгоритмизации»

Выполнил студент группы ИВТ-б-о-20-1
Пушкин Н.С. « »_____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. После изучения теоретической части методических указаний, разобрал пример, описанный в документе.

2. Затем приступил к выполнению общих заданий

```
factorial3 = """
from functools import lru_cache
@lru_cache
def factorial3(n):
    if n == 0:
        return 1
    else:
        return n * factorial3(n-1)
"""
if __name__ == '__main__':
    print("Время выполнения рекрусивной функции числа Фибоначи: ",
          timeit(setup=fib1, number=1000))
    print("Время выполнения итеративной функции числа Фибоначи: ",
          timeit(setup=fib2, number=1000))
    print("Время выполнения рекрусивной функции числа Фибоначи с"
          " использованием декоратора lru_cache: ",
          timeit(setup=fib3, number=1000))
    print("Время выполнения рекрусивной функции факториала: ",
          timeit(setup=factorial1, number=1000))
    print("Время выполнения итеративной функции факториала: ",
          timeit(setup=factorial2, number=10000))
    print("Время выполнения рекрусивной функции факториала с"
          " использованием декоратора lru_cache: ",
          timeit(setup=factorial3, number=10000))
```

Рисунок 11.1 – Код первого общего задания

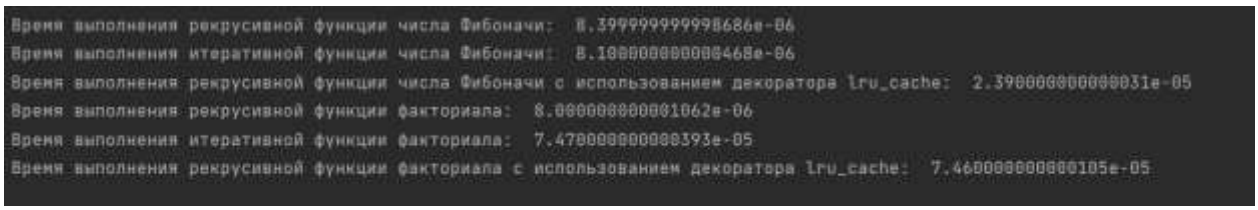

Рисунок 11.2 – Результат выполнения кода

```
                try:
                    return g(*args, **kwargs)
                except TailRecurseException as e:
                    args = e.args
                    kwargs = e.kwargs

    func.__doc__ = g.__doc__
    return func


@tail_call_optimized
def fib(i, current = 0, next = 1):
    if i == 0:
        return current
    else:
        return fib(i - 1, next, current + next)
"""
if __name__ == '__main__':
    print("Время выполнения функции factorial(): ",
          timeit(setup=code1, number=10000))
    print("Время выполнения функции factorial() с"
          " использованием интроспекции стека: ",
          timeit(setup=code3, number=10000))
    print("Время выполнения функции fib(): ",
          timeit(setup=code2, number=10000))
    print("Время выполнения функции fib() с"
          " использованием интроспекции стека: ",
          timeit(setup=code4, number=10000))
```

Рисунок 11.3 – Код второго общего задания

```
Время выполнения функции factorial():  0.00019620000000009075
Время выполнения функции factorial() с использованием интроспекции стека:  0.00015249999999999986
Время выполнения функции fib():  0.00014979999999997773
Время выполнения функции fib() с использованием интроспекции стека:  0.00016780000000005124
```

Рисунок 11.4 – Результат выполнения кода

Индивидуальное задание

Вариант 2

3. После выполнения общих заданий приступил к работе с индивидуальным.

```python
def clear_str(stri):
    res = ""
    for s in stri:
        if s in "()[]{}<>":
            res = res + s
    return res


# Проверка

def check_par(stri):
    if len(stri) == 0:
        return True
    else:
        f = stri[0]
        l = stri[-1]
        kf = "([{<".find(f)
        if kf == -1:
            return False
        if l == ")]}>"[kf]:
            return check_par(stri[1:len(stri) - 1])
        else:
            return False


def task(stri):
    return clear_str(clear_str(stri))

if __name__ == '__main__':

    print(check_par("( [ ) ]"))
```

Рисунок 11.5 – Код индивидуального задания

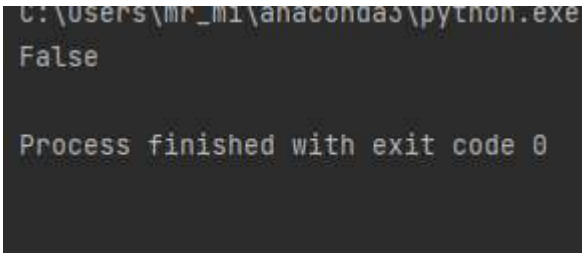

Рисунок 11.6 – Результат выполнения кода

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(limit).

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.

Оптимизация хвостовой рекурсии путём преобразования её в плоскую итерацию реализована во многих оптимизирующих компиляторах.

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