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 «программирование на Python»

Выполнил студент группы ИВТ-б-о-21-1
Харченко Б.Р. « »_____20__г.
Подпись студента_____
Работа защищена « »_____20__г.

Проверил Воронкин Р.А._____
(подпись)

Ставрополь 2022

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

Ход работы

```
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 рекурсивной функции числа Фибоначи: ",
          timeit(setup=fib1, number=1000))
    print("Время выполнения итеративной функции числа Фибоначи: ",
          timeit(setup=fib2, number=1000))
    print("Время выполнения рекурсивной функции числа Фибоначи с"
          " использованием декоратора lru_cache: ",
          timeit(setup=fib3, number=1000))
    print("Время выполнения рекурсивной функции факториала: ",
          timeit(setup=factorial1, number=1000))
    print("Время выполнения итеративной функции факториала: ",
          timeit(setup=factorial2, number=10000))
    print("Время выполнения рекурсивной функции факториала с"
          " использованием декоратора lru_cache: ",
          timeit(setup=factorial3, number=10000))
```

Рисунок 1 – Задание 1

```
@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 2 – Код второго общего задания

Рисунок 3 – Код индивидуального задания и результат выполнения кода

Контрольные вопросы:

1. Для чего нужна рекурсия?

В программировании рекурсия — вызов функции (процедуры) из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия). Рекурсивная программа позволяет описать повторяющееся или даже потенциально бесконечное вычисление, причём без явных повторений частей программы и использования циклов.

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 — это структура данных, в которой элементы хранятся в порядке поступления.

Стек хранит последовательность данных. Связаны данные так: каждый элемент указывает на тот, который нужно использовать следующим. Это линейная связь — данные идут друг за другом и нужно брать их по очереди. Из середины стека брать нельзя.

Главный принцип работы стека — данные, которые попали в стек недавно, используются первыми. Чем раньше попал — тем позже используется. После использования элемент стека исчезает, и верхним становится следующий 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. Этот предел предотвращает бесконечную рекурсию от переполнения стека языка C и сбоя Python. Это значение может быть установлено с помощью sys.

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 RunTime.

6. Как изменить максимальную глубину рекурсии в языке Python?

С помощью sys.setrecursionlimit(число).

7. Каково назначение декоратора lru_cache?

Функция lru_cache предназначается для мемоизации (предотвращения повторных вычислений), т. е. кэширует результат в памяти.

Полезный инструмент, который уменьшает количество лишних вычислений.

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.

Типовой механизм реализации вызова функции основан на сохранении адреса возврата, параметров и локальных переменных функции в стеке и выглядит следующим образом:

1. В точке вызова в стек помещаются параметры, передаваемые функции, и адрес возврата.

2. Вызываемая функция в ходе работы размещает в стеке собственные локальные переменные.

3. По завершении вычислений функция очищает стек от своих локальных переменных, записывает результат (обычно — в один из регистров процессора).

4. Команда возврата из функции считывает из стека адрес возврата и выполняет переход поэтому адресу. Либо непосредственно перед, либо сразу после возврата из функции стек очищается от параметров.

Вывод: были приобретены навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.