МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №2.9

Рекурсия в языке Python

по дисциплине «Технологии программирования и алгоритмизации»

Выполнил студент группы ИВТ-б-о-20-11

Погорелов Д.Н « »_____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

Ход работы: Задание 1

https://github.com/Dmitry4311/lab2.9/

Самостоятельно изучите работу со стандартным пакетом Python timeit . Оцените с помощью этого модуля скорость работы итеративной и рекурсивной версий функций factorial и fib .

1. Поставил по умолчанию проверку скорости 100 раз для более точного вычисления, программа просто выводит на экран время выполнения функции числа Фибоначи:

```
Время выполнения итеративной функции числа Фибоначи:  5.6999999999973736e-08
Время выполнения рекурсивной функции числа Фибоначи:  6.899999999997186e-08
Время выполнения декоративной функции числа Фибоначи:  5.574099999999999e-05
```

Рисунок 1 – Время выполнения числа Фибоначи

2. И время выполнения функции факториала

```
Время выполнения итеративной функции факториала:  4.900000000002125e-08
Время выполнения рекурсивной функции факториала:  2.2950000000000056e-06
Время выполнения декоративной функции факториала:  7.000000000000061e-08
```

Рисунок 2 – Время выполнения факториала

Задание 2

Самостоятельно проработайте пример с оптимизацией хвостовых вызовов в Python. С помощью пакета timeit оцените скорость работы функций factorial и fib с использованием интроспекции стека и без использования интроспекции стека..

1. Организовал вывод времени выполнения функций факториала и числа фибоначи обычных и с оптимизацией хвостовой рекурсии:

```
Время выполнения функции factorial:  5.990000000000162e-05
Время выполнения функции factorial с оптимизацией хвостовой рекурсии:  0.004690000000000003
```

Рисунок 4 – Нахождение времени для факториалов

2. Программа также выводит время вополнения числа фибоначи:

```
Время выполнения функции fib:  5.980000000000221e-05
Время выполнения функции fib с оптимизацией хвостовой рекурсии:  0.005078000000000003
```

Рисунок 5 – Нахождение времени для чисел Фибоначи

Индивидуальное задание

Составить программу с использованием рекурсивных функций для решения задачи. Номер

варианта определяется по согласованию с преподавателем.

1. Напишите рекурсивную функцию, проверяющую правильность расстановки скобок в строке.

При правильной расстановке выполняются условия:

количество открывающих и закрывающих скобок равно.

внутри любой пары открывающая – соответствующая закрывающая скобка, скобки

расставлены правильно.

Примеры неправильной расстановки:)(, ())(, ())(() и т. п.

```
Введите список аргументов: l;kasdfj(lkasdfj(
Результат:
False
```

Рисунок 6. Выполнение неправильной расстановки()

```
Введите список аргументов: kldjfas(la;ksdfj)
Результат:
True
```

Рисунок 7 – Выполнение правильной расстановки ()

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