

Федеральное государственное автономное образовательное учреждение
высшего образования «Северо-Кавказский федеральный университет»
Институт цифрового развития

ОТЧЕТ ПО ДИСЦИПЛИНЕ
ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ
Лабораторная работа №12

Выполнила:

Мирзаева Камилла Мирзаевна

2 курс, группа ПИЖ-б-о-20-1

Принял:

Воронкин Роман Александрович

Ставрополь, 2021 г.

Ход работы:

```
def rec(n):  
    if n > 0:  
        rec(n - 1)  
  
    print(n)
```

Рисунок 1 - Пример рекурсивной функции

```
def recursion(n):  
    if n == 1:  
        return 1  
  
    return n + recursion(n - 1)
```

Рисунок 2 - Выполнение примера №1

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

Рисунок 3 - Использование факториала

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

Рисунок 4 - Выполнение базового случая

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

Рисунок 5 - «Параллельный» рекурсивный вызов функции

```
def factorial(n):
    product = 1
    while n > 1:
        product *= n
        n -= 1
    return product

def fib(n):
    a, b = 0, 1
    while n > 0:
        a, b = b, a + b
        n -= 1
    return a
```

Рисунок 6 - Вариант обхода ограничения ТСО

```

from functools import lru_cache
...
@lru_cache
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 2) + fib(n - 1)

```

Рисунок 7 - Использование декоратора lru_cache

```

def fib(n):
    if n <= 1:
        return (n, 0)
    else:
        (a, b) = fib(n - 1)
        return (a + b, a)

```

Рисунок 8 - Пример линейной рекурсии

```

def cursing(depth):
    try:
        cursing(depth + 1)
    except RuntimeError as RE:
        print('I recursed {} times!'.format(depth))

if __name__ == '__main__':
    cursing(0)

```

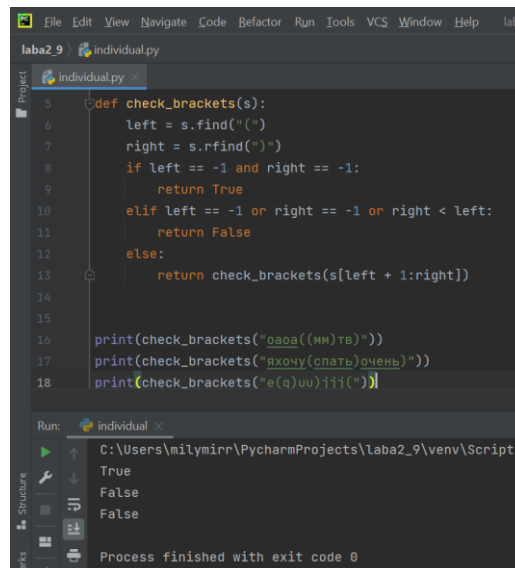
modul x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/12/modul.py
I recursed 998 times!

Рисунок 9 - Увеличение максимальной глубины рекурсии

Индивидуальное задание №1 (15)

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.



```
def check_brackets(s):
    left = s.find("(")
    right = s.rfind(")")
    if left == -1 and right == -1:
        return True
    elif left == -1 or right == -1 or right < left:
        return False
    else:
        return check_brackets(s[left + 1:right])

print(check_brackets("oaoa((mn)ta)"))
print(check_brackets("яxочу(спать)очень"))
print(check_brackets("e(q)uu{{{}}})")
```

Run: individual

C:\Users\milymirr\PycharmProjects\laba2_9\venv\Scripts\python.exe

True
False
False

Process finished with exit code 0

Контрольные вопросы:

1. Для чего нужна рекурсия? Рекурсия подразумевает более компактный вид записи выражения. Обычно это зависимость процедур (функций, членов прогресс и т.д.) соседних порядковых номеров. Некоторые зависимости очень сложно выразить какой-либо формулой, кроме как рекурсивной. Рекурсия незаменима в ряде случаев при программировании замкнутых 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 вызовов – в теории вычислительных систем, LIFO-стек, хранящий информацию для возврата управления из подпрограмм в программу и для возврата в программу из обработчика прерывания.

При вызове подпрограммы или возникновении прерываний, в стек заносится адрес возврата – адрес в памяти следующей инструкции приостановленной программы и управление передаётся 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? Программа выдаст ошибку: `RuntimeError: Maximum Recursion Depth Exceeded`

6. Как изменить максимальную глубину рекурсии в языке Python? Изменить максимальную глубины рекурсии можно с помощью `sys.setrecursionlimit(limit)`. Чтобы проверить параметры лимита, нужно запустить `sys.getrecursionlimit()`.

7. Каково назначение декоратора `lru_cache` ? Декоратор можно использовать для уменьшения количества 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.