# Урок 17.01. Рекурския.

Темой сегодняшнего занятия является рекурсия. Вообще оно подразумевает вызов функции самой себя. И в качестве примера возьмем задачу «Факториал». Вообще факториалом числа называется такое число, которое получается умножением всех чисел от 1 до N. Дополнительным условием будет запрет на использование цикла.

Для решения этой задачи нам понадобятся наши знания по математике, и рассмотрим такой пример.

Допустим нам нужно посчитать факториал числа 5, тогда нам нужно найти произведение чисел от 1 до 5: 5! = 1\*2\*3\*4\*5 или наоборот поставить числа 5\*4\*3\*2\*1. Теперь взглянув на эту запись мы можем увидеть, что 5! = 5\*4!, а в свою очередь 4! = 4\*3! и т.д. Т.е. получается для того чтобы найти факториал числа нужно умножить данное число на факториал числа меньше на 1, или N! = N\*N!. И когда мы дойдем до 1! = 1. Такую логику мы сейчас и попробуем сделать с помощью функции.

def fact(num):

fact\_num\_minus\_1 = fact(num - 1)

return num \* fact\_num\_minus\_1

num\_fact = fact(5)

print(num\_fact)

Запустим программу, и увидим ошибку.

[Previous line repeated 996 more times].

RecursionError: maximum recursion depth exceeded

Эта ошибка говорит нам, о том, что достигнута максимальная глубина (depth) рекурсии (996), т. е. функция вызвала себя 996 раз и python говорит, что больше погружаться не будет. Это конечно же произошло из-за нашей ошибки. А именно, как и с бесконечными циклами у функций рекурсии должно быть условие остановки или выхода. Так же мы это может увидеть если запустим программу в режиме отладки. А раз нам нужно условие остановки функции, то добавим его.

def fact(num):

if num == 1:

return 1

fact\_num\_minus\_1 = fact(num - 1)

return num \* fact\_num\_minus\_1

num\_fact = fact(5)

print(num\_fact)

Вот теперь наша программа работает верно, и через рекурсию. Тут стоит сделать один важный вывод: при написании функции рекурсии нужно начинать с условия остановки рекурсии.

Рассмотрим ещё одну задачу «Код сайта».

У нас есть словарь, который представляет структуру какого-нибудь сайта. Структура сайта в виде словаря:

site = {

'html': {

'head': {

'title': 'Мой первый сайт',

},

'body': {

'h2': 'Здесь должен быть заголовок',

'div': ' Тут должен быть рекламный блок',

'p': 'А это для маленького текста',

}

}

}

Вообще, с такой структурой постоянно сталкиваются программисты, которые занимаются обработкой сайтов. Нам нужно по заданному пользователем ключу найти данные в этой структуре. Давайте приступим.

site = {

'html': {

'head': {

'title': 'Мой первый сайт',

},

'body': {

'h2': 'Здесь должен быть заголовок',

'div': 'Тут должен быть рекламный блок',

'p': 'А это для маленького текста',

}

}

}

def find\_key(struct, key):

if key in struct:

return struct[key]

for sub\_struct in struct.values():

if isinstance(sub\_struct, dict):

result = find\_key(sub\_struct, key)

if result:

break

else:

result = None

return result

user\_key = input('Какой ключ ищем:')

value = find\_key(site, user\_key)

if value:

print(value)

else:

print('Такого ключа в структуре сайта нет.')

Вот так мы и решили нашу задачу. Однако нужно понимать, что с использованием рекурсий связано множество ошибок, так же рекурсии не всегда эффективны. Но есть задачи, где без рекурсии невозможно обойтись, как в нашей задаче.