Рекурсия

Если просто, то рекурсия - это вызов функции самой в себе.

Звучит просто, но для понимания уже не так просто.

Давайте рассмотрим следующий пример:

```
def tmp(a, b):
    if a > b:
        return
    tmp(a + 2, b)
    print(a, end=" ")

tmp(0, 6)
```

Результатом будет:

6 4 2 0

Что тут вообще происходит и почему так?

Начнем с условия завершения рекурсии:

```
if a > b:
        return
```

Мы будем выходить из функции, когда a > b. Если это условие не поставить, то функция будет вызывать себя бесконечно, так как на каждом вызове tmp(a, b) мы будем делать вызов tmp(a + 2, b). Про это важно помнить и всегда создавать условие для остановки рекурсии.

Теперь давайте рассмотрим последовательность запусков:

tmp(0, 6) → tmp(2, 6) → tmp(4, 6) → tmp(6, 6) → tmp(8, 6)

Но почему тогда вывод идет в обратном порядке?

Потому что функция после завершения возвращается в то место, где ее вызвали. То есть tmp(8, 6) вернется в тот момент tmp(6, 6), где ее вызвали, а tmp(6, 6) в tmp(4, 6) и тд.

Первой завершится вызов tmp(8, 6), но там ничего не выводится, поскольку выполнится условие завершения рекурсии и до print() в этом вызове не дойдет. И программа вернется в то место, откуда вызывали tmp(8, 6) - то есть в tmp(6, 6), где следующей же строчкой идет вывод a, которая в этот момент равна 6. И так далее.

То есть последовательность действий будет следующей:

tmp(0, 6)

tmp(2, 6)

tmp(4, 6)

tmp(6, 6)

tmp(8, 6)

print(6) внутри tmp(6, 6)

print(4) внутри tmp(4, 6)

print(2) внутри tmp(2, 6)

print(0) внутри tmp(0, 6)

завершение программы

И, наконец, стоит упомянуть про ограничение глубины рекурсии.

По дефолту оно выставлено как 1000, то есть запуск tmp(0, 2004) уже не смог бы выполнится полностью, поскольку этот лимит был бы превышен

Чтобы его поменять на X можно использовать метод setrecursionlimit

```
import sys
sys.setrecursionlimit(X) # X - новое значение для ограничения глубины
рекурсии
```
