**Рекурсия**

Мы уже сталкивались с тем, что функция может вызывать другую функцию. Но функция также может вызывать и саму себя! В этом случае она будет называться рекурсивной.

Рассмотрим в качестве примера функцию, которая вычисляет факториал числа n, то есть произведение чисел от 1 до n. Факториал числа n обозначается как n!, n!=1∗2∗…∗n. Сначала напишем нерекурсивную реализацию этой функции:

**def** factorial(n):

res = 1

**for** i **in** range(2, n + 1):

res \*= i

**return** res

Эта функция будет работать верно для всех неотрицательных значений n. При n, равном 0 или 1, цикл не выполнится ни разу, и функция, как и положено, будет возвращать число 1..

Теперь напишем рекурсивную реализацию функции factorial(). Нам известно, что 0!=1, 1!=1. А как вычислить величину n! для большого n? Если бы мы могли вычислить величину (n−1)!, то тогда легко вычислили бы n!, поскольку n!=n∗(n−1)!. Но как вычислить (n−1)!? Если бы мы вычислили (n−2)!, то смогли бы вычислить (n−1)!=(n−1)∗(n−2)!. А как вычислить (n−2)!? Если бы... В конце концов, мы дойдём до величины 0!, которая равна 1. Таким образом, для вычисления факториала мы можем использовать значение факториала для меньшего числа, а если n=0, то сразу выводить ответ, равный 1:

**def** factorial(n):

**if** n == 0:

**return** 1

**else**:

**return** factorial(n - 1) \* n

Давайте напишем рекурсивную функцию print\_n(), которая печатает все числа от 1 до n для заданного числа n. Такая функция не будет ничего возвращать, а результатом её действий будут напечатанные ею числа. Сначала приведём нерекурсивную реализацию с использованием цикла **for**:

**def** print\_n(n):

**for** i **in** range(1, n + 1):

**print**(i)

Теперь напишем рекурсивную реализацию. Для этого обратим внимание на то, что если n=1, то нужно напечатать число 1. Если же n>1, то нужно напечатать числа от 1 до n−1 (а это умеет делать наша функция, если её вызвать с параметром n−1), а затем напечатать число n. Получается, что при рекурсивном способе написания мы можем обойтись без цикла:

**def** print\_n(n):

**if** n == 1:

**print**(1)

**else**:

print\_n(n - 1)

**print**(n)

Давайте попробуем в этой функции переставить две последние строки местами:

**def** print\_n(n):

**if** n == 1:

**print**(1)

**else**:

**print**(n)

print\_n(n - 1)

Что будет делать такая функция? Так как сначала мы будем печатать число n, а потом вызывать функцию с параметром n−1, то такая функция будет печатать числа от n до 1, то есть печатать то же самое, но в обратном порядке. Обратим внимание на то, что момент остановки рекурсии можно сделать другим. Мы переставали делать рекурсивные вызовы при условии n=1. Но можно делать остановку рекурсии при n=0. Тогда при n=0 нам не нужно вообще ничего печатать. Тогда получим вот такую реализацию нашей функции:

**def** print\_n(n):

**if** n > 0:

print\_n(n - 1)

**print**(n)

Важно помнить, что при разработке рекурсивной функции необходимо прежде всего оформлять условие завершения рекурсии и думать, почему рекурсия когда-либо завершит работу. Если же у нас нет случая, при котором рекурсия прекращается, или этот случай не всегда достигается, то мы получим бесконечную рекурсию.

[Вопросы-ответы по видеолекции](https://edu.sirius.online/#/qa/324_3384_122565)