Функции - части программы, которые можно повторно вызывать с разными параметрами, чтобы не писать много раз одно и то же. Функции в программировании немного отличаются от математических функций. В математике функции могут только получить параметры и дать ответ, в программировании же функции умеют делать что-нибудь полезное, например, ничего не возвращать, но что-то печатать.

Функции чрезвычайно полезны, если одни и те же действия нужно выполнять несколько раз. Но некоторые логические блоки работы с программой иногда тоже удобно оформлять в виде функции. Это связано с тем, что человек может одновременно держать в голове ограниченное количество вещей. Когда программа разрастается, отследить все уже очень сложно. В пределах одной небольшой функции запутаться гораздо сложнее - известно, что она получает на вход, что должна выдать, а об остальной программе в это время можно не думать.

В программировании также считается хорошим стилем писать функции, умещающиеся на один экран. Тогда можно одновременно окинуть взглядом всю функцию и не нужно крутить текст туда-сюда. Поэтому, если получилось что-то очень длинное, то нужно нарезать его на кусочки так, чтобы каждый из них был логичным (делал какое-то определенное действие, которое можно назвать) и не превышал при этом 10-15 строк.

Мы уже использовали готовые функции, такие как print, len и некоторые другие. Эти функции описаны в стандартной библиотеке или других подключаемых библиотеках. Сегодня мы научимся создавать свои функции.

Рассмотрим, как создать свою функцию, на примере вычисления факториала. Текст программы без функции выглядит так:

```
n = int(input())
fact = 1
i = 2
while i <= n:
    fact *= i
    i += 1
print(fact)
```

Вычисление факториала можно вынести в функцию, тогда эта же программа будет выглядеть так:

```
def factorial(num):
    fact = 1
    i = 2
    while i <= num:
        fact *= i
        i += 1
    return fact
n = int(input())
print(factorial(n))
```

Описание функции должно идти в начале программы. На самом деле, оно может быть в любом месте, до первого вызова функции factorial.

Определение функции должно начинаться со слова def (сокращение от define, определить). Дальше идет имя функции, после которого в скобках через запятую перечисляются параметры (у нашей функции всего один параметр). После закрытия скобки должно стоять двоеточие.

Команды, выполняемые в функции должны записываться с отступом, как в блоках команд if или while.

В нашей функции num - это параметр, на его место подставляется то значение, с которым функция была вызвана. Действия внутри функции точно такие же, как в обычной программе, кроме дополнительной команды return. Команда return возвращает значение функции (оно должно быть записано через пробел после слова return) и прекращает её работу. Возвращенное значение подставляется на то место, где осуществлялся вызов функции.

Команда return может встречаться в любом месте функции. После того как она выполнится, работа функции будет прекращена. Здесь есть некоторая аналогия с командой break, применяемой для выхода из цикла.

Вызовы функций из функции

Функцию подсчета факториала можно использовать для подсчета биномиальных коэффициентов (числа сочетаний). Формула для подсчета числа сочетаний выглядит так:

n! / (k! * (n - k)!).

Если бы мы не пользовались функциями, то нам потребовалось бы три раза записать почти одно и то же. С помощью функций вычисление выглядит намного проще:

```
print(factorial(n) // (factorial(k) * factorial(n - k)))
```

Подсчет биномиальных коэффициентов можно также оформить в виде функции с двумя параметрами:

```
def binomial(n, k):
    return factorial(n) // (factorial(k) * factorial(n - k))
```