Наиболее частое применение логические выражения находят в условных операторах.

Условный оператор позволяет выполнять действия в зависимости от того, выполнено условие или нет. Записывается условный оператор как "if <логическое выражение>:", далее следует блок команд, который будет выполнен только если логическое выражение приняло значение True. Блок команд, который будет выполняться, выделяется отступами в 4 пробела (в IDE можно нажимать клавишу tab).

Рассмотрим, например, задачу о нахождении модуля числа. Если число отрицательное, то необходимо заменить его на минус это число. Решение выглядит так:

```
x = int(input())
if x < 0:
    x = -x
print(x)
```

В этой программе с отступом записана только одна строка, x = -x. При необходимости выполнить несколько команд все они должны быть записаны с тем же отступом. Команда print записана без отступа, поэтому она будет выполняться в любом случае, независимо от того, было ли условие в if'е истинным или нет.

В дополнение к if можно использовать оператор else: (иначе). Блок команд, который следует после него, будет выполняться если условие было ложным. Например, ту же задачу о выводе модуля числа можно было решить, не меняя значения переменной x:

```
x = int(input())
if x > 0:
    print(x)
else:
    print(-x)
```

Все команды, которые выполняются в блоке else, должны быть также записаны с отступом. Else должен следовать сразу за блоком команд if, без промежуточных команд, выполняемых безусловно. Else без соответствующего if'а не имеет смысла.

Если после if записано не логическое выражение, то оно будет приведено к логическому, как если бы от него была вызвана функция bool. Однако, злоупотреблять этим не следует, т.к. это ухудшает читаемость кода.

Для подсчета модуля числа в Питоне существует функция abs, которая избавляет от необходимости каждый раз писать подсчет модуля вручную.

В Питоне, как и во многих других языках программирования, если результат вычисления выражения однозначно понятен по уже вычисленной части, то оставшаяся часть выражения даже не считается. Например, выражение True or 5 // 0 == 42, не будет вызывать ошибки деления на ноль, т.к. по левой части выражения (True) уже понятно, что результат его вычисления также будет True и арифметическое выражение в правой части даже не будет вычисляться.

Пометить как выполненное