Логический тип данных

Кроме уже известных нам целочисленных и строковых типов данных в Питоне существует также логический тип данных, который может принимать значения "истина" (True) или "ложь" (False).

По аналогии с арифметическими выражениями существуют логические выражения, которые могут быть истинными или ложными. Простое логическое выражение имеет вид <арифметическое выражение> <знак сравнения> <арифметическое выражение>. Например, если у нас есть переменные x и y с какими-то значениями, то логическое выражение x + y < 3 * y в качестве первого арифметического выражения имеет x + y, в качестве знака сравнения < (меньше), а второе арифметическое выражение в нём 3 * y.

В логических выражениях допустимы следующие знаки сравнений:

Знак сравнения	Описание
<	меньше
>	больше
<=	меньше либо равно
>=	больше либо равно
==	**равно**
!=	**не равно**

В Питоне допустимы и логические выражения, содержащие несколько знаков сравнения, например x < y < z. При этом все сравнения обладают одинаковым приоритетом, который меньше, чем у любой арифметической операции.

Результат вычисления логического выражения можно сохранять в переменную, которая будет иметь тип bool. Переменные такого типа, как и числа и строки, являются неизменяемыми объектами.

Строки также могут сравниваться между собой. При этом сравнение происходит в лексикографическом порядке (как упорядочены слова в словаре).

Логические операции

Чтобы записать сложное логическое выражение, часто бывает необходимо воспользоваться логическими связками "и", "или" и "не". В Питоне они обозначаются как and, or и not соответственно. Операции and и or являются бинарными, т.е. должны быть записаны между операндами, например x < 3 or y > 2. Операция not - унарная и должна быть записана перед единственным своим операндом.

Все логические операции имеют приоритет ниже, чем операции сравнения (а значит, и ниже чем арифметические операции). Среди логических операций наивысший приоритет имеет операция not, затем идет and и наименьший приоритет имеет операция or. На порядок выполнения операций можно влиять с помощью скобок, как и в арифметических выражениях.

Условный оператор

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**.

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.

> *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.

Вложенный условный оператор

Внутри блока команд могут находиться другие условные операторы. Посмотрим сразу на примере. По заданному количеству глаз и ног нужно научиться отличать кошку, паука, морского гребешка и жучка. У морского гребешка бывает более сотни глаз, а у пауков их

восемь. Также у пауков восемь ног, а у морского гребешка их нет совсем. У кошки четыре ноги, а у жучка - шесть ног, но глаз у обоих по два. Решение:

```
eyes = int(input())
legs = int(input())
if eyes >= 8:
    if legs == 8:
        print("spider")
    else:
        print("scallop")
else:
    if legs == 6:
        print("bug")
    else:
        print("cat")
```

Если вложенных условных операторов несколько, то, к какому из них относится else, можно понять по отступу. Отступ у else должен быть такой же, как у if, к которому он относится.

Конструкция "иначе-если"

В некоторых ситуациях необходимо осуществить выбор больше чем из двух вариантов, которые могут быть обработаны с помощью if-else. Рассмотрим пример: необходимо вывести словом название числа 1 или 2 или сообщить, что это другое число:

```
number = int(input())
if number == 1:
    print('One')
elif number == 2:
    print('Two')
else:
    print('Other')
```

Здесь используется специальная конструкция **elif**, обозначающая "иначе, если", после которой записывается условие. Такая конструкция введена в язык Питон, потому что запись if-else приведет к увеличению отступа и ухудшению читаемости.

Конструкций elif может быть несколько, условия проверяются последовательно. Как только условие выполнено, запускается соответствующий этому условию блок команд и дальнейшая проверка не выполняется. Блок else является необязательным, как и в обычном if.