Условный оператор (инструкция). Логический тип bool

§1. Синтаксис условного оператора

Все ранее рассматриваемые программы имели линейную структуру: все инструкции выполнялись последовательно, одна за одной, при этом каждая записанная инструкция обязательно выполняется. Теперь поставим себе другую задачу. Допустим, мы хотим по данному числу *x* определить его абсолютную величину (модуль). Программа должна напечатать значение переменной *x*, если *x* >= 0 (так в программировании обозначается операция сравнения ≥, см. **§3**) или же величину –*x* в противном случае. Линейная структура программы нарушается: в зависимости от справедливости условия *x* >= 0 должна быть выведена одна или другая величина. Соответствующий фрагмент программы на Python имеет вид:

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

В этой программе используется условная инструкция `if` (если). После слова `if` указывается проверяемое условие (x >= 0), завершающееся двоеточием. После этого идет блок (последовательность) инструкций, который будет выполнен, если условие истинно, в нашем примере это вывод на экран величины x. Затем идет слово `else` (иначе), также завершающееся двоеточием, и блок инструкций, который будет выполнен, если проверяемое условие неверно, в данном случае будет выведено значение –x.

Итак, условныйоператор имеет следующий синтаксис (правила записи):

```
if Условие:
    Блок инструкций 1
else:
    Блок инструкций 2
```

Блок инструкций 1 будет выполнен, если *Условие* истинно. Если *Условие* ложно, будет выполнен *Блок инструкций 2*.

В условном операторе может отсутствовать слово `else` и последующий блок. Такая инструкция называется неполным ветвлением. Например, если дано число *x* и мы хотим заменить его на абсолютную величину *x*, то это можно сделать следующим образом:

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

В этом примере переменной *x* будет присвоено значение –*x*, но только в том случае, когда *x* < 0. А вот инструкция `print(x)` будет выполнена всегда, независимо от проверяемого условия.

Для выделения блока инструкций, относящихся к инструкции `if` или `else`, в языке Python используются *отступы*. Все инструкции, которые относятся к одному блоку, должны иметь равную величину отступа, то есть одинаковое число пробелов в начале строки. Рекомендуется использовать отступ в 4 пробела и не рекомендуется использовать в качестве отступа символ табуляции. Это одно из существенных отличий синтаксиса языка Python от синтаксиса большинства языков, в которых блоки выделяются специальными словами, например, нц... кц в Кумире, begin ... end в Паскале или фигурными скобками в Си.

§2. Вложенные условные инструкции. Комментарии

Внутри условных инструкций можно использовать любые инструкции языка Python, в том числе и условную инструкцию. Получаем вложенное ветвление – после одной развилки в ходе исполнения программы появляется другая развилка. При этом вложенные блоки имеют отступ уже относительно "своей" условной инструкции. Покажем это на примере программы, которая по данным ненулевым числам *x* и *y* определяет, в какой из четвертей координатной плоскости находится точка (*x*, *y*):

```
x = int(input())
y = int(input())
# Определим, в какой четверти находится точка с координатами (x,y)
if x > 0:
    if y > 0:                # x>0, y>0
        print("Первая четверть")
    else:                    # x>0, y<0
        print("Четвертая четверть")
else:
    if y > 0:                # x<0, y>0
        print("Вторая четверть")
    else:                    # x<0, y<0
        print("Третья четверть")
```

В этом примере мы также использовали *комментарии* – дополнительный текст, который записывается, чтобы человеку было проще прочесть и разобраться в программе, и который интерпретатор игнорирует (не исполняет). Комментариями в Питоне является символ # и весь текст после этого символа до конца строки.

§3.Операции сравнения. Тип данных bool

Как правило, в качестве проверяемого условия используется результат вычисления одной из следующих операций сравнения:

<	Меньше — условие верно, если первый операнд меньше второго.
>	Больше — условие верно, если первый операнд больше второго.
<=	Меньше или равно.
>=	Больше или равно.
==	Равенство. Условие верно, если два операнда равны.
!=	Неравенство. Условие верно, если два операнда неравны.

Обратите внимание, что знак = операцией сравнения не является!!!

Например, условие `(x * x <= 1000)` означает "значение $x*x$ меньше или равно 1000", а условие `(2 * x != y)` означает "удвоенное значение переменной *x* не равно значению переменной *y*". Операторы сравнения в языке Python можно объединять в цепочки (в отличие от большинства других языков программирования, где для этого нужно использовать логические связки!!!), например, `a == b == c` или `1 <= x <= 10`.

Операции сравнения возвращают значения специального логического типа `bool`. Значения логического типа могут принимать одно из двух значений: `True` (истина) или `False` (ложь). Если преобразовать логическое `True` к типу `int`, то получится 1, а преобразование `False` даст `0`. При обратном преобразовании число `0` преобразуется в `False`, а любое ненулевое число в `True`. При преобразовании `str` в bool пустая строка преобразовывается в `False`, а любая непустая строка в `True`. О преобразованиях типов вообще рассказывалось в уроке 1.

§4. Логические операции

Иногда нужно проверить одновременно не одно, а несколько условий. Например, проверить, является ли данное число чётным, можно при помощи условия `(n % 2 == 0)` (остаток от деления `n` на `2` равен `0`). А если необходимо проверить, что два данных целых числа `n` и `m` являются четными, необходимо проверить справедливость обоих условий: `n % 2 == 0` и `m % 2 == 0`, для чего их необходимо объединить при помощи оператора `and` (логическое И): `n % 2 == 0 and m % 2 == 0`.

В языке Python существуют стандартные логические операции: логическое И, логическое ИЛИ, логическое отрицание.

Логическое И является бинарной операцией (то есть операцией с двумя операндами: левым и правым) и имеет вид `and`. Операция `and` возвращает `True` тогда и только тогда, когда оба ее операнда имеют значение `True`.

Логическое ИЛИ является бинарной операцией и возвращает `True` тогда и только тогда, когда хотя бы один операнд равен `True`. Операция "логическое ИЛИ" имеет вид `or`.

Логическое НЕ (отрицание) является унарной (то есть с одним операндом) операцией и имеет вид `not`, за которым следует единственный операнд. Логическое НЕ возвращает `True`, если операнд равен `False` и наоборот.

Пример. Проверим, что хотя бы одно из чисел a или b оканчивается на 0:

```
if a % 10 == 0 or b % 10 == 0:
```

Проверим, что число a — положительное, а b — неотрицательное:

```
if a > 0 and not (b < 0):
```

Или можно вместо `not (b < 0)` записать `(b >= 0)`.

§5. Каскадные условные инструкции

Пример программы из §2, определяющий четверть координатной плоскости, можно переписать, используя "каскадную" последовательность операцией `if... elif... else`:

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
    print("Первая четверть")
elif x > 0 and y < 0:
    print("Четвертая четверть")
elif y > 0:
    print("Вторая четверть")
else:
    print("Третья четверть")
```

В такой конструкции условия `if`, ..., `elif` проверяются по очереди, выполняется блок, соответствующий первому из истинных условий. Если все проверяемые условия ложны, то выполняется блок `else`, если он присутствует.

Задачи для самостоятельного решения

1) A – 3527. При решении задачи можно использовать упрощенный обмен значений двух переменных, имеющийся в языке Python: `a, b = b, a`

2) B – 3506

3) C – 3507

4) D – 3510. При решении этой и некоторых следующих задач удобно использовать стандартную функцию abs(), которая предназначена для вычисления модуля выражения, стоящего в скобках.

5) E – 3512

6) F – 3514

7) G – 3515

8) H – 3516. Сначала решите уравнение на бумаге в общем случае. Определите когда такое решение возможно и является целочисленным. Выпишите, во что вырождается уравнение, не подходящее под общий случай. Определите, когда полученное равенство имеет смысл. Составьте программу, в которой бы анализ каждого из параметров задачи производился не более одного раза.

9) I – 3517. В этой программе обязательно использовать конструкцию `elif` в случае решения задачи на языке Python.

10) J – 3508. В этой задаче главное – правильно понять ее условие, что в олимпиадных задачах по информатике является очень важным умением.