УСЛОВНЫЙ ОПЕРАТОР

Возможности, описанные в предыдущих параграфах, позволяют писать *линейные* программы, в которых операторы выполняются последовательно друг за другом, и порядок их выполнения не зависит от входных данных.

В большинстве реальных задач порядок действий может несколько изменяться, в зависимости от того, какие данные поступили. Например, программа для системы пожарной сигнализации должна выдавать сигнал тревоги, если данные с датчиков показывают повышение температуры или задымленность.

Для этой цели в языках программирования предусмотрены условные операторы. Например, для того, чтобы записать в переменную **M** максимальное из значений переменных **a** и **b**, можно использовать оператор:

```
if a>b:
  M=a
else:
  M=b
```

Слово **if** переводится с английского языка как *«если»,* а слово **else** – как *«иначе».* Если верно (истинно) условие, записанное после ключевого слова **if**, то затем выполняются все команды (*блок команд*), которые расположены до слова **else**. Если же условие после **if** неверно (ложно), выполняются команды, стоящие после **else**.

В Python, в отличие от других языков, важную роль играют сдвиги операторов относительно левой границы (отступы). Обратите внимание, что слова **if** и **else** начинаются на одном уровне, а все команды внутренних блоков сдвинуты относительно этого уровня вправо на одно и то же расстояние. Это позволяет не использовать особые ограничители блоков (слова **begin** и **end** в языке Паскаль, фигурные скобки в Си-подобных языках). Для сдвига используют символы табуляции (которые вставляются при нажатии на клавишу Tab) или пробелы.

Если в блоке всего один оператор, иногда бывает удобно записать блок в той же строке, что и ключевое слово **if** (**else**):

```
if a>b: M=a
else:   M=b
```

В приведенных примерах условный оператор записан в полной форме: в обоих случаях (истинно условие или ложно) нужно выполнить некоторые действия. Программа выбора максимального значения может быть написана иначе:

```
M=a
if b>a:
  M=b
```

Здесь использован условный оператор в неполной форме, потому что в случае, когда условие ложно, ничего делать не требуется (нет слова **else** и блока операторов после него).

Поскольку операция выбора максимального из двух значений нужна очень часто, в Python есть встроенная функция **max**, которая вызывается так:

```
M=max( a, b )
```

Если выбирается максимальное из двух чисел, можно использовать особую форму условного оператора в Python:

```
M=a if a>b else b
```

которая работает так же, как и приведённый выше условный оператор в полной форме: записывает в переменную **M** значение **a**, если выполняется условие **a> b**, и значение **b**, если это условие ложно. Часто при каком-то условии нужно выполнить сразу несколько действий. Например, в задаче сортировки значений переменных **a** и **b** по возрастанию нужно поменять местами значения этих переменных, если **a> b**:

```
if a>b:
  c=a
  a=b
  b=c
```

Все операторы, входящие в блок, сдвинуты на одинаковое расстояние от левого края. Заметим, что в Python, в отличие от многих других языков программирования, есть множественное присваивание, которое позволяет выполнить эту операцию значительно проще:

```
a, b = b, a
```

Кроме знаков **<** и **>**, в условиях можно использовать другие знаки отношений: **<=** (меньше или равно), **>=** (больше или равно), **==** (равно, два знака «=» без пробела, чтобы отличить от операции присваивания) и **!=** (не равно).

Внутри условного оператора могут находиться любые операторы, в том числе и другие условные операторы. Например, пусть возраст Андрея записан в переменной **a**, а возраст Бориса – в переменной **b**. Нужно определить, кто из них старше. Одним условным оператором тут не обойтись, потому что есть три возможных результата: старше Андрей, старше Борис и оба одного возраста. Решение задачи можно записать так:

```
if a > b:
  print ( "Андрей старше" )
else:
  if a == b:
    print ( "Одного возраста" )
  else:
    print ( "Борис старше" )
```

Условный оператор, проверяющий равенство, находится внутри блока **иначе** (**else**), поэтому он называется *вложенным* условным оператором. Как видно из этого примера, использование вложенных условных операторов позволяет выбрать один из *нескольких* (а не только из двух) вариантов. Если после **else** сразу следует еще один оператор **if**, можно использовать так называемое «каскадное» ветвление с ключевыми словами **elif** (сокращение от **else-if**): если очередное условие ложно, выполняется проверка следующего условия и т.д.

```
if a > b:
  print ( "Андрей старше" )
elif a == b:
  print ( "Одного возраста" )
else:
   print ( "Борис старше" )
```

Обратите внимание на отступы: слова **if**, **elif** и **else** находятся на одном уровне. Если в цепочке **if-elif-elif-…** выполняется несколько условий, то срабатывает первое из них. Например, программа

```
if cost < 1000:
  print ( "Скидок нет." )
elif cost < 2000:
  print ( "Скидка 2%." )
elif cost < 5000:
  print ( "Скидка 5%." )
else:
  print ( "Скидка 10%." )
```

при **cost = 1500** выдает «**Скидка 2%.**», хотя условие **cost < 5000** тоже выполняется.

СЛОЖНЫЕ УСЛОВИЯ

Предположим, что ООО «Рога и Копыта» набирает сотрудников, возраст которых от 25 до 40 лет включительно. Нужно написать программу, которая запрашивает возраст претендента и выдает ответ: «подходит» он или «не подходит» по этому признаку.

На качестве условия в условном операторе можно указать любое логическое выражение, в том числе сложное условие, составленное из простых отношений с помощью логических операций (связок) «И», «ИЛИ» и «НЕ». В языке Python они записываются английскими словами «**and**», «**or**» и «**not**».

Пусть в переменной **v** записан возраст сотрудника. Тогда нужный фрагмент программы будет выглядеть так:

```
if v >= 25 and v <= 40:
  print ( "подходит" )
else:
  print ( "не подходит" )
```

при вычислении сложного логического выражения сначала выполняются отношения (**<**, **<=**, **>**, **>=**, **==**, **!=**) а затем – логические операции в таком порядке: сначала все операции **not**, затем – **and**, и в самом конце – **or** (во всех случаях – слева направо). Для изменения порядка действий используют круглые скобки.

Иногда условия получаются достаточно длинными и их хочется перенести на следующую строку. Сделать это в Python можно двумя способами: использовать обратный слэш (это не рекомендуется):

```
if v >= 25 \
       and v <= 40:
   ...
```

или взять все условие в скобки (перенос внутри скобок разрешён):

```
if ( v >= 25
       and v <= 40 ):
   ...
```

В языке Python разрешены двойные неравенства, например

```
if A < B < C:
   ...
```

означает то же самое, что и

```
if A < B and B < C:
   ...
```