C: Conditions and conditionals

Gareth McCaughan
Revision 1.8, May 14, 2001

Credits

Introduction

“Conditions” are things that can be true or false. For instance, the expression `x<3` which is true if `x` is the name of a number less than 3 and false otherwise.

“Conditionals” are things that depend on conditions. The most important kind of conditional in Python is the `if` statement, which lets you do one thing or another depending on whether a condition is true.

This sheet tells you about conditions and conditionals.

Conditions

Try typing these things in. I haven’t shown the answers, because you’ll learn better if you try them.

```
>>> 1 < 2                         1 is less then 2, so this condition is true
[CENSORED]
>>> 1 > 2                         1 is not greater than 2, so this condition is false
[CENSORED]
```

As you’ll have seen, Python likes to use *numbers* to represent the “truth value” of conditions. 0 means “false”, and 1 means “true”. In fact, any non-zero number means “true”.

You can actually use other things as truth values. I don’t recommend this, though; it’s just likely to be confusing.

Comparisons

Most conditions are comparisons of one object with another. Here’s a brief list of ways to compare things in Python.

`a < b`	True if `a` is less than `b`
`a <= b`	True if `a` is less than or equal to `b`
`a > b`	True if `a` is greater than `b`
`a >= b`	True if `a` is greater than or equal to`b`
`a == b`	True if `a` is equal to `b`
`a <> b`	True if `a` is not equal to `b`

It's pretty obvious what these things mean when `a` and `b` are numbers. But they make sense for other sorts of objects, too. For instance, strings are compared in something rather like alphabetical order. In fact, you can compare *any* two objects, though many of the possible comparisons are Very Silly. For instance, it turns out that Python thinks that 3 is less than `'silly'`.

When you're testing whether two things are unequal, you can use `!=` instead of `<>` if you prefer.

Be careful, by the way, to notice the difference between `=` and `==`. You use `=` for setting a variable (i.e., giving a name to an object), and `==` for testing whether two things are equal.

Combining comparisons

You can say things like `1 < x < 2`, meaning "1 is less than `x`, and `x` is less than 2".

Other conditions

Here are some other useful conditions.

`0`	Always false
`1`	Always true
`x in y`	True if `x` is equal to some element of `y`
`x not in y`	True if `x` is not equal to any element of `y`

For `in` and `not in`, `y` should be a sequence: that is, a list or a tuple or a string,

Combining conditions

You can join conditions together using the words `and`, `or` and `not`. So, for instance, `x<3 and y>6` is a condition.

The 'if' statement

So, now we know about conditions. One very important thing we can do with them is to use them in an `if` statement. This is pretty straightforward:

```
if x < 3:
  print 'x is less than 3. I'm setting it to 3.'
  x = 3
```

Often, you want to do one thing if a condition is true and another thing if the condition is false. To do this, use the magic word `else`:

```
if x<3:
  print 'x is less than 3.'
else:
  print 'x is not less than 3.'
```

And, less often, you want to test a whole bunch of conditions and do something according to the first one that comes out true. For this, you need the strange word `elif`, which is short for "else if":

```
if x<3:
  print 'x is less than 3'
elif x<4:
  print 'x is not less than 3, but it's less than 4.'
else:
  print "x isn't even less than 4."
```

Other uses of conditions

Conditions are also used in `while` loops: to learn about those, see Sheet L (*Loops*).