I: Input and output

Gareth McCaughan Revision 1.8, May 14, 2001

Credits

For the LaTeX source of this sheet, and for more information on LiveWires and on this course, see the LiveWires web site at http://www.livewires.org.uk/python/

Introduction

Almost any interesting program has to get information from somewhere, and produce some sort of answers somewhere. These are called "input" and "output". This sheet describes some of the ways Python handles input and output.

Some of the things in here require you to have done

```
from livewires import *
```

before they'll work.

Input

Here are some useful functions that ask the user a question and wait for an answer. They all expect that the user will hit Enter after typing the answer.

`read_number()` Expects a number. If you type in something that isn't a number, you'll get told to try again.

`read_number('Enter a number: ')` The same, but prints that message before waiting for input. It's usually better to use this version of `read_number` (perhaps with a different message, like `'How old are you?'`) so that the person using your program knows what's required.

`read_string()` Expects a string. You don't need to put quotation marks around it.

`read_string("What's your name?")` Just like `read_string()`, but prints a message first.

`read_yesorno()` Expects `yes`, `no`, `y` or `n`, in either capitals or lowercase.

`read_yesorno('Would you like another game?')` Just like `read_yesorno()`, but prints a message first.

Output

The main thing you need to know about output is the `print` statement. It can print any object:

```
>>> x=[1,2,3]                                   A list,
>>> y='zog'                                     a string,
>>> z=99                                        a number,
>>> f=repr                                      a function
>>> print x,y,z,f
[1, 2, 3] zog 99 <built-in function repr>
```

Notice that it puts spaces between the things it prints.

If you write two `print` statements, one after the other, you'll see that the second starts printing on a fresh line rather than following on from the first. If that isn't what you want, put a comma at the end of the first `print`:

```
print 123,
print 456
```

This will print `123 456` on a single line.