M: Modules

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

Python provides a *huge* number of useful functions and other such things. (These worksheets describe, perhaps, about 1% of what there is. Maybe 2%.) So many different functions might be confusing, so they're parcelled up into things called "modules".

Importing modules

You can think of a module as a collection of useful things. (The actual definition is a bit complicated, and you really don't need it.) To make the Useful Things in a module available to you, you have to "import" the module. So:

```
import stuff
```

makes the things in the module `stuff` available for your program to use. Those things will have names that look like `stuff.thingy` so that there's no danger of two things in different modules "colliding" by having the same name. It's a bit like with telephone numbers: there are probably lots of people whose phone number is 321987; that's why there are "dialling codes" like 01223 (for Cambridge). The module name is like a dialling code.

Living dangerously

It can get very tiresome, having to type a module name in over and over again. So Python gives you a way to get at the things in a module more directly. If you say

```
from stuff import blarp,weeble
```

then the things that would have been called `stuff.blarp` and `stuff.weeble` are available under the shorter names `blarp` and `weeble`. (This doesn't import anything else from the module, under any name.)

It might be worth doing this if you find yourself using some bits of a module a lot.

Living even more dangerously

What if you use *lots* of things from a module a lot? Then you can say

```
from stuff import *
```

which has the effect of making *everything* (well, almost everything) in module `stuff` available without needing to type `stuff.` at the start of any of the names. Doing this is usually a Bad Idea; there might be two modules containing different things with the same name, and then all kinds of bad things could happen.

There's just one case in which we recomend that you do this: the special LiveWires module we've written to make your life easier. Any program that uses anything from it should begin with `from livewires import *`. (The main reason for that is that we wanted you to be able to use the things in the module without needing to understand what modules are and why they result in names like `stuff.thingy` ...)

Finding out what's in a module

After doing `import stuff`, you can get a list of all the names of things in `stuff` by typing `dir(stuff)`. This probably won't usually be very helpful, but if you're feeling like exploring ...

Some useful modules

As I said at the start, there are a lot of modules out there. There are more than 170 modules that come as standard with Python! When you're feeling brave, you might like to find out about some of them. Here's a very brief list of things it's worth knowing about. Look them up in the official Python documentation, or investigate for yourself. (I recommend looking them up.) To understand some of the documentation, you may need to know about "classes" and "objects": see Sheet O (*Objects and Classes*).

`sys`	Mostly some slightly useful variables.
`string`	Things for strings.
`re`	"Regular expressions": a complicated but powerful way of doing more advanced things with strings.
`math`	Mathematical functions and constants.
`random`	Random numbers.
`time`	Time and date.