Getting Started with Python

Syntax Basics

The print Function

The first bit of syntax you'll see is the print. Here we'll do the usual "Hello World!".

```
print 'Hello World!'
```

You can also easily concatenate strings...

```
a = 'Hello'
b = 'World!"

print a + b
```

As an aside, you can print any object which has a write() function.

Numerical Operations

I'm going to cut you short here and say that most of the numerical operators are the same as in every other language. As usual, the docs are you friend.

String Operations

Strings are defined by a group of characters wrapped in either "" or ''. They both work exactly the same, and can escape each other, so things like below are possible:

```
"S't'ring"
'S"t"ring'
```

You can replace inside a string by using the % operator, this is much like C's sprintf, e.g.:

```
'The Kights who said %s!' % 'Ni'
```

Other methods available on strings are:

```
S.capitalize()
S.replace(old, new)
```

You can specify Unicode strings by prefixing a 'u' on the "". (e.g.: u"this is unicode")

Flow Control

Flow statements allow you to decend through a block of code given different sets of criteria. The examples here cover the if, while and for statements.

if

Due to the whitespace requirements, you'll notice that we don't need to use any form of braces around the relevant code.

The if statement looks like this:

```
if test:
      // do stuff
```

The 'test' value can be anything that would usually go before an equals (=). So (1 == 2) would be valid.

If you wish to have a fallback option (else), that would look like:

```
if test:
      // do stuff
else:
      // do stuff
```

while

The while loop is similar to the if statement. It looks like:

```
while something:
      // do stuff
```

You can also include an else on a while loop.

for

```
for target in sequence:
      // do stuff
```

The target can mean anything which could usually go before an = statement, so (x, y) would be valid.

Functions

```
def name():
      // do stuff
```

Objects

Like most Object-Orientated languages, these created using the class operator.

A few points on objects:

- all methods are public
- methods are declared using the function operator (above)
- supports multiple-inheritance

Example:

```
class Example:
      def __init__:
            // do stuff

      def go:
            // do stuff
```