1 Variable types

Python is a dynamic language where variable type (e.g. whether a variable is a string or an integer) is not fixed. Sometime though you might like to force a particular type, or convert between types (most common where numbers may be contained within a string.

The most common types of variables in python are integers, float (nearly all types of numbers that are not integers) and strings.

Within python code the function *type* will show what variable type a string is.

```
x = 11 # This is an integer
print (type(x))

OUT:
<class 'int'>
```

Outside of the code, within the interpreter requesting help on a variable name will give the help for its variable type.

```
help (x)

OUT:
Help on int object:

class int(object)
 |  int(x=0) -> integer
 |  int(x, base=10) -> integer
 |
 |  Convert a number or string to an integer, or return 0 if no arguments
 |  are given.  If x is a number, return x.__int__().  For floating point
 |  numbers, this truncates towards zero.
 |
 |  If x is not a number or if base is given, then x must be a string,
 |  bytes, or bytearray instance representing an integer literal in the
 |  given base.  The literal can be preceded by '+' or '-' and be surrounded
 |  by whitespace.  The base defaults to 10.  Valid bases are 0 and 2-36.
 |  Base 0 means to interpret the base from the string as an integer literal.
 |  >>> int('0b100', base=0)
 |  4
 |
 |  Methods defined here:

...
```

Though x is an integer, if we divide it by 2, Python will dynamically change its type to float. This may be common sense to some, but may be irritating to people who are used to coding languages where variables have 'static' types (this particular behaviour is also different to Python2 where the normal behaviours is for integers to remain as integers).

```
x = x/2
print (x)
print (type(x))

OUT:
5.5
<class 'float'>
```

Python is trying to be helpful, but this dynamic behaviour of variables can sometimes need a little debugging. In our example here, to keep our answer as an integer (if that is what we needed to do) we need to specify that:

```
x = 11
x = int(x/2)
print (x)
print (type(x))

OUT:
5
<class 'int'>
```

We can convert between types, such as in the follow examples. Numbers may exist as text, particularly in some file imports.

```
x = '10'
print ('Look what happens when we try to multiply a string')
print (x *3)
x = float (x)
print ('Now let us multiply the converted variable')
print (x*3)

OUT:
Look what happens when we try to multiply a string
101010
Now let us multiply the converted variable
30.0
```