Operators

Numerical Operators

Coding, in the simplest sense, is merely the action of assigning values to variables, and then 'doing things' with those variables.

In this section we will look at some of the basic operators used for integers and floats. Other variable types have different operators, which we shall see later.

An obvious starting point is basic arithmetic; addition, subtraction, multiplication and division:

[2]:
```
a = 2.0
b = 3.0
c = 10.0

print('a added to b is', a + b)
print('a multiplied by c is', a*c)
print('c divided by b is', c/b)
print('a subtracted from c', c - a)
```

```
a added to b is 5.0
a multiplied by c is 20.0
c divided by b is 3.3333333333333335
a subtracted from c 8.0
```

These operators can also be used for integers:

[3]:
```
print('2 multiplied by 3 is', 2*3)
```

```
2 multiplied by 3 is 6
```

and between integers and floats:

[4]:
```
print('1.5 multiplied by 2 is', 1.5*2)
```

```
1.5 multiplied by 2 is 3.0
```

If you are using Python version 2.xxx or below, beware of dividing by integers…

The Exponential Operator **

Another useful operator is the exponential operator **. This returns the left number to the power of the right:

[6]:
```
print(2**3)
```

```
8
```

which can be read as 2^3.

Note that this operator also works on floats and float-integer combinations:

[8]:
```
print(4**0.5) #square root of 4
```

```
2.0
```

The Modulo Operator %

The modulo operator returns the remainder of the left number divided by the right:

[9]:
```
print(16%3)
```

```
1
```

In mathematics this would be expressed as

16 mod 3.

This operator can also act on floats and integer-float combinations:

[10]:
```
print(16.3%3)
```

```
1.3000000000000007
```

The Floor Division Operator //

This returns the result of the left number divided by the right, but without the remainder:

[11]:
```
print(16//3)
```

```
5
```

Like the others this works for both integers and floats.

Special Functions and Advanced Mathematics

For more complex mathematics involving logs, trigonometry, etc. we'll rely on the scientific packages SciPy and NumPy. We'll discuss these at a later stage.

Multiple Operations in a Single Expression

Though we have only seen one operation or function used per line, you can combine as many as you'd like:

[14]:
```
print( 2**3 + 4)
```

```
12
```

If you want to group or control the order in which operations are executed use brackets. For example:

[15]:
```
print( (2 + 17//2)**5 )
```

```
100000
```

where the `//` is applied first, then the `+` and lastly the `**`. We shall discuss the order in which operations and function calls are executed later.