1 Week 1: Data Types and Operators

Python is one of the most widely used programming languages in the world. It is a very powerful, yet general, language that has applications ranging from automation to data science.

1.1 A brief word on the print() function

Before we start looking at python properly it is useful to first define the print function. The print function takes the argument between the brackets and writes it as a string to the screen. We will make use of this function **a lot** over the next 6 weeks.

```
#Example 1.0
print('Hello, World!')
----------
Hello, World!
```

1.2 Arithmetic Operators

We begin our journey with Python by looking at arithmetic operators which, while they may seem mundane, are incredibly useful. We will make use of the following operators in Python:

$$+ \text{ Addition} \quad - \text{ Subtraction} \quad * \text{ Multiplication} \quad / \text{ Division} \quad ** \text{ Exponential}$$

An important note is that Python follows mathematical convention for order of operations, as you can see in the following example.

```
#Example 1.1
5+2*3
----------
11
```

1.3 Variables and Assignment

By understanding how to use variable we turn Python from being a simple calculator into something much more powerful. Assigning variables in Python is very simple.

$$\text{variable\_name} \quad = \quad \text{value}$$

It is important to note that this is not an equality like in maths! Instead we should really think of '=' as the assignment operator that assigns the variable name on the left to the value on the right,

the order is important. We can also assign multiple variables at once using commas to separate the values.

$$\text{variable\_1},\quad \text{variable\_2} \quad = \quad \text{value\_1},\quad \text{value\_2}$$

Variable names can use letters, numbers and underscores with some exceptions for forbidden words. You can find a list of forbidden words here.

```
#Example 1.2
x = 1
y = 2
z = 3
a, b, c = 10, 11, 12
print(x,y,z,a,b,c)
----------
1 2 3 10 11 12
```

1.4 Booleans & Comparison and Logical Operators

The bool data type holds one of two values, True or False. As well as assigning a variable the type bool we can generate it when we implement a comparison operator. The 6 common comparisons are:

$$< \text{ Less Than} \quad > \text{ Greater Than} \quad <= \text{ Less Than or Equal}$$
$$>= \text{ Greater Than or Equal} \quad == \text{ Equal} \quad != \text{ Not Equal}$$

We illustrate a simple using this data type in a simple example.

```
#Example 1.3
a = 1
b = 2
less_than = a<b
print(less_than)
----------
True
```

We also have two logical operators at our disposal. Imagine we have two statements,

$$\text{Statement\_A} \quad \text{(Logical Operator)} \quad \text{Statement\_B}$$

if we place the logical operator in the middle of these two statements they have the following outcomes:

- *and*: If statement A **and** statement B are True then the value is True, otherwise it is False
- *or*: If at least one of the statements A **or** B is True then the value is True.

1.5 Integers and Floats

When we are storing numbers in Python we will consider two data types, integers and floats, although others exist. Integers, as the name suggests, is for storing integers and takes less memory to store. Floats, standing for floating-point number, can be used for storing very large or very small numbers with high precision. We can use the Python functions *int()* and *float()* to cast values to either data type when storing it in a variable.

```
#Example 1.4
integer_pi = int(3.14159265359)
float_pi = float(3.14159265359)
print(integer_pi)
print(float_pi)
----------
3
3.14159265359
```

1.6 Strings

A string is a data type for immutable[1] ordered sequence of characters. We can define a string using either " " or ' ' quotation marks. Just like numbers (int() and float()) strings can be assigned to variables. Strings are also very literal, for example, when adding two strings together if you want a space to appear you must add one in.

```
#Example 1.5
user_first_name = 'Kyle'
user_last_name = 'Fogarty'
print(user_first_name +' ' +  user_last_name)
----------
Kyle Fogarty
```

[1] cannot be modified after it is created