Python Crash Course

Arithmetic Operators

In [1]:
```
90 + 45
```
Out[1]: 135

In [2]:
```
90 + 45 - 32
```
Out[2]: 103

In [3]:
```
3 * 33
```
Out[3]: 99

In [4]:
```
100 / 4
```
Out[4]: 25.0

In [5]:
```
2 ** 4
```
Out[5]: 16

In [6]:
```
4 % 2   # give me the remainder after I divide 4 by 2
```
Out[6]: 0

In [7]:
```
5 % 2
```
Out[7]: 1

In [8]:
```
(2 + 3) * (5 + 5)   # you can try to make any complex equations
```
Out[8]: 50

In [9]:
```
4 + 8 / 2  * 4
```
Out[9]: 20.0

In [10]:
```
4 + 8 / (2 * 4)
```
Out[10]: 5.0

In [11]:
```
(4 + ( 8 / 2 ) )  * 4
```
Out[11]: 32.0

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

In [12]:
```
# Can not start with number or special characters . you can check if you want
name_of_var = 2
```

In [13]:
```
x = 2
y = 3
```

In [14]:
```
z = x + y
```

In [15]:
```
z
```

Out[15]: 5

In [16]:
```
z = z + 3
z
```

Out[16]: 8

In [17]:
```
z += 3
z
```

Out[17]: 11

Assignment 1: Solve Maths

Construct the following equation and calculate the value the value of `y` using Python

$$y = \frac{2x^2 + 3x - 1}{1 - x}$$

In [ ]:
```
## ADD CODE BELOW
# ~ 1 line of code
```

Strings

A string is usually a bit of text you want to display to someone, or "export" out of the program you are writing. Python knows you want something to be a string when you put either " (double-quotes) or ' (single-quotes) around the text

In [30]:
```
'single quotes'
```

Out[30]:
```
'single quotes'
```

In [31]:
```
"double quotes"
```

Out[31]:
```
'double quotes'
```

And just like Integers you can save strings in varaibles and display them

In [32]:
```
my_string = 'hello world'
my_string
```

Out[32]:
```
'hello world'
```

You can even add Strings

In [18]:
```
part_1 = 'Hello'
part_2 = ' World'

part_1 + part_2
```

Out[18]:
```
'Hello World'
```

But Don't Try to add two differnt Data Types Together

In [19]:
```
4 + '4'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-ae36418677da> in <module>
----> 1 4 + '4'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

When you are going to be using quotations as part of the string then you can use escape characters like `\` to ignore them.

In [20]:
```
'wrap lot\'s of other quotes'
```

Out[20]:
```
"wrap lot's of other quotes"
```

Or you can wrap single quotes in double quotes

In [21]:
```
"wrap lot's of other quotes"
```

Out[21]:
```
"wrap lot's of other quotes"
```

Printing

In [22]:
```
my_string = "Hello world"
print("Hello world")
```

```
Hello world
```

In [23]:
```
my_string = "Hello world"
my_string
```

Out[23]:
```
'Hello world'
```

In [24]:
```
my_string = "Hello world"
my_string
x = 5
```

In [25]:
```
my_string = "Hello world"
print("Hello world")
x = 5
```

```
Hello world
```

4 Different Methods to Print combining Variables and text

In [26]:
```
num = 12
name = 'Sam'
```

Method 1

In [27]:
```
print('My age is: {one}, and my name is: {two}'.format(one=num,two=name))
```

```
My age is: 12, and my name is: Sam
```

Method 2

In [28]:
```
print('My age is: {}, and my name is: {}'.format(num,name))
```

```
My age is: 12, and my name is: Sam
```

Method 3

In [29]:
```
print('My age is: ' + str(num) + ', and my name is: ' + name)
```

```
My age is: 12, and my name is: Sam
```

Method 4

In [30]:
```
and18 = '%s how are you %s %d' % ('hello', 'Android', 18)  # sprintf style string
print(and18)
```

```
hello how are you Android 18
```