Numbers in Python!

We'll learn about the following topics…

1) Types of Numbers in Python
2) Basic Arithmetic
3) Differences between classic division and floor division
4) Object Assignment in Python

Types of Numbers

Python has various "types" of numbers (numeric literals). We'll mainly focus on integers and floating-point numbers.

Integers are just whole numbers, positive or negative. For example: 2 and -2 are examples of integers.

Floating point numbers in Python are notable because they have a decimal point in them, or use an exponential (e) to define the number. For example, 2.0 and -2.1 are examples of floating-point numbers.

Here is a table of the two main types we will spend most of our time working with some examples:

Examples	Number "Type"
1,2,-5,1000	Integers
1.2,-0.5,2e2,3E2	Floating-point numbers

We'll be working with integers and float number types.

Basic Arithmetic

Here are some arithmetic operations and their symbols used in python.

Name	Symbols
Addition	+
Subtraction	-
Multiplication	*
Division (float)	/
Division (floor)	//
Modulus	%
Power	**

How is floor division different from normal float division?

The // operator (two forward slashes) or the floor division truncates the decimal without rounding, and returns an integer result.

Basically, floor division (//) returns the result in the form type of integer unlike float division (/) which returns the result in the form of float.

Variable Assignments

Now that we've seen how to use numbers in Python as a calculator let's see how we can assign names and create variables.

We use a single equal’s sign to assign labels to variables. Let's see a few examples of how we can do this.

Note: The following code(s) were run and executed using Jupytr Notebook.

In [11]:
```
# Let's create an object called "a" and assign it the number 5
a = 5
```

Now if I call *a* in my Python script, Python will treat it as the number 5.

In [12]:
```
# Adding the objects
a+a
```

Out[12]:
```
10
```

What happens on reassignment? Will Python let us write it over?

In [13]:
```
# Reassignment
a = 10
```

In [14]:
```
# Check
a
```

Out[14]:
```
10
```

Yes! Python allows you to write over assigned variable names. We can also use the variables themselves when doing the reassignment. Here is an example of what I mean:

In [15]:
```
# Check
a
```

Out[15]:
```
10
```

```
In [16]: # Use A to redefine A
         a = a + a

In [17]: # Check
         a

Out[17]: 20
```

Some important rules to follow while naming variables:

1) Names cannot start with a number.

2) There can be no spaces in the name, use _ instead.

3) Can't use any of these symbols:'",<>/?|\()!@#$%^&*~
-+

4) Avoid using words that have special meaning in Pyth
on like "list" and "str".

So, we learned some of the basics of numbers in Python. We then wrapped it up with learning about Variable Assignment in Python.