

## Different data types we will encounter in Python

- Numeric - Numeric variables take values which are numbers like 9, 3.14, 0
- String - String variables are used to store textual information
- Boolean - Boolean variables have two modes either True or False.

### ✓ Integers and Floats

#### ✓ Basic Arithmetic

```
# Addition  
2+1
```

 3

```
# Subtraction  
2-5
```

 -3

```
# Multiplication  
2*2
```

 4

```
# Division  
3/2
```

 1.5

```
# Floor Division  
7//3
```

 2

```
# Exponentiation  
2**5
```

 32

```
# Modulus  
15%6
```

 3

```
# Order of Operations followed in Python  
2 + 10 * 10 + 3
```

 105

```
2+ 10* (10+3)
```

 132

```
# Scientific Notations  
4E3
```

 4000.0

## ✓ Let's talk about numbers!

- We will use integer 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. 4E2 (4 times 10 to the power of 2) is also an example of a floating point number in Python.
- In computing, floating-point arithmetic is arithmetic using formulaic representation of real numbers

The table below summarises the two numeric data types, Integers and Floats:

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

## ✓ What is a Variable?

✓ VARIABLES are entities which help us store information and retrieve it later.

- A variable with a fixed name can store information of nature like numeric, textual, boolean etc.
- A Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.
- The type of data contained in a variable can be changed at user's will.

```
# You can store numbers in variables.
# The standard rule is you write the variable name followed by = sign and the value it will take
```

```
x=5
```

```
x
```

```
↔ 5
```

```
y=6.4
```

```
y
```

```
↔ 6.4
```

```
print(y)
```

```
↔ 6.4
```

✓ Basic Arithmetic operations we can do on x and y. Later we will be doing operations on thousands of such numbers in one go!

```
# Addition
```

```
z = x+y
```

```
print(z)
```

```
↔ 11.4
```

- ✓ A variable can be assigned different values and data types and it will store the last value assigned

```
# Subtraction
```

```
z = x-y
```

```
# Use the in-built print function to print the variable
print(z)
```

```
↵ -1.4000000000000004
```

```
# Find out the data type of variable z
type(y)
```

```
↵ float
```

```
# Multiplication
z = x*y
```

```
print(z) # Print the variable z
type(z) # Get the data type of variable z
```

```
↵ 32.0
float
```

```
# Division
z = x/y
```

```
print(z) # Print the variable z
type(z) # Get the data type of variable z
```

```
↵ 0.78125
float
```

```
# Floor division
z = x//y # Remember x=5, y=6.4
print(z)
```

```
↵ 0.0
```

## ✓ Rules for naming a variable in Python

- Variables names must start with a letter or an underscore like `_product` , `product_`
- The remainder of your variable name may consist of letters, numbers and underscores
- `spacy1`, `pyThon`, `machine_learning` are some valid variable names
- Names are case sensitive.
- `case_sensitive`, `CASE_SENSITIVE`, and `Case_Sensitive` are each a different variable.

```
1oNone = 4
```

```
↵ File "<ipython-input-41-5af1d05f4ba0>", line 1
    1oNone = 4
      ^
SyntaxError: invalid decimal literal
```

Next steps: [Fix error](#)

```
onone_abc_1 = 5
```

```
list = 5
```

```
list
```

- Names cannot begin with a number. Python will throw an error when you try to do so
- Names can not contain spaces, use `_` instead
- Names can not contain any of these symbols:

```
: ' " , < > / ? | \ ! @ # % ^ & * ~ - +
```

- It is considered best practice that names are lowercase with underscores
- Avoid using Python built-in keywords like `list`, `str`, `def` etc. We will talk more about such conventions later on

## ✓ Boolean Variables

- A Boolean variable only takes two values either `True` or `False`. It is used for comparisons

## ✓ Comparison Operators

- These operators will allow us to compare variables and output a Boolean value (`True` or `False`).
- If you have any sort of background in Math, these operators should be very straight forward.
- First we'll present a table of the comparison operators and then work through some examples:
- In the table below, `a=3` and `b=4`.

Operator	Description	Example
<code>==</code>	If the values of two operands are equal, then the condition becomes true.	<code>(a == b)</code> is not true.
<code>!=</code>	If values of two operands are not equal, then condition becomes true.	<code>(a != b)</code> is true
<code>&gt;</code>	If the value of left operand is greater than the value of right operand, then condition becomes true.	<code>(a &gt; b)</code> is not true.
<code>&lt;</code>	If the value of left operand is less than the value of right operand, then condition becomes true.	<code>(a &lt; b)</code> is true.
<code>&gt;=</code>	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	<code>(a &gt;= b)</code> is not true.
<code>&lt;=</code>	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	<code>(a &lt;= b)</code> is true.

- Python comes with Booleans (with predefined `True` and `False` displays that are basically just the integers 1 and 0). It also has a placeholder object called `None`. Let's walk through a few quick examples of Booleans (we will dive deeper into them later in this course).

```
# Set object to be a boolean
boolean_variable = False
type(boolean_variable)
```

```
bool
```

```
#Show
boolean_variable
```

```
False
```

## ✓ Equal

```
2 == 3
```

```
False
```

```
2==0
```

```
False
```

- Note that `==` is a comparison operator, while `=` is an assignment operator.

## ✓ Not equal

```
2!=0
```

```
True
```

```
2!=2
```

```
False
```

### ✓ Greater than

```
a=3  
b=2  
a> b
```

 True

```
a == 3
```

 True

```
b > 4
```

 False

### ✓ Less than

```
10 < 45
```

 True

```
4 < 2
```

 False

### ✓ Greater than or equal to

```
3 >=2
```

 True

```
4 >= 4
```

 True

### ✓ Less than or equal to

```
3 <= 0
```

 False

```
1 <= 2
```

 True

Start coding or [generate](#) with AI.

