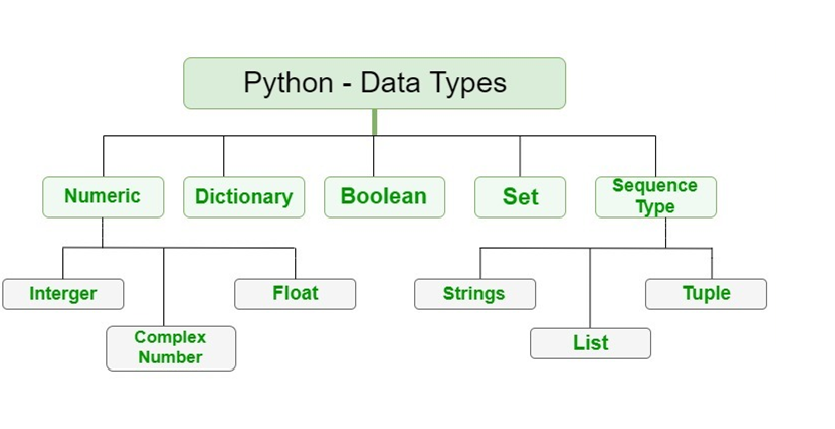
# Session 2

* Python data types
  + Basic data types (int, float, and bool)
* type() function
* Strings
* Type casting
* Lists
* String and list indexing
* Indexing and Slicing

## 1) Python data types





**We will be learning integer, float, Boolean, Complex, and string data types in this session**

* **integers** are whole numbers (python denotes integers with the keyword int )
* **floats** are decimal point numbers (python denotes floats with the keyword float )
* **Boolean** is a data type that can hold only 2 values either True or False (python denotes Booleans with the keyword bool )
* **Complex Number** is a specific data type for complex numbers. Complex numbers have both a real and an imaginary part.
* **strings** are a collection of individual characters(python denotes strings with the keyword str )

### Basic data types (int, float, and bool)

a = 10

In the above cell, we are assigning the integer value to the variable a. So the datatype of variable a is int **The variable name can be any combination of characters as long as it follows the rules for defining variable names we discussed earlier.** Usually, when we define a variable we name it depending on the value that it is going to hold. This helps us recognize and remember the variable name later in the code.

my\_varibale = 20

no\_of\_lines = 4

**Since Python is dynamically typed, the variable data type is determined by the value we assign to it on the right-hand side of the assignment operator (=)**



my\_float = 12.12

**In the above line, the variable my\_float is of the type float since we are assigning it a decimal point number**



my\_bool = True  
bool\_2 = False

complex data type

x = 9 + 8j # both value are int type

y = 10 + 4.5j # one int and one float

print(x) # (9+8j)

print(y) # (10+4.5j)

**When we assign the value of True or False to the variable my\_bool or bool\_2 they are considered as bool datatype by Python** The T and F in True and False should be capital

## 2) type() function

* The type function is used to check the datatype of the variable declared.
* This function will show the keyword used by the python for that particular data type
* the type() function returns a string which should be printed out to see the output

a = 500  
print(type(a))

<class 'int'>

b = 500.12  
print(type(b))

<class 'float'>

a = True  
print(type(a))

<class 'bool'>

mystring = "hello world"  
print(mystring)  
print(type(mystring))

hello world  
<class 'str'>

## 3) String datatype

**A string is a collection of characters** **In other languages we have a separate data type called char which stores individual characters** **In Python we can get the same behavior by defining a string of length 1**



In Python, a string can be defined by enclosing the characters in single double or triple quotes

my\_string1 = 'hello world'

my\_string2 = "hello world"

my\_string3 = ''' hello world'''

The intention behind providing this kind of flexibility is considering the cases where we will have to have an actual single or double quote in our sentence Suppose we want to print the following line This is Sam's house

my\_string1 = 'This is Sam's house'

File "<ipython-input-11-6f004d05cb8a>", line 1  
 my\_string1 = 'This is Sam's house'  
 ^  
SyntaxError: invalid syntax

In such cases, we use the double quotes

my\_string1 = "This is Sam's house"

Whenever we need double quotes inside a string we declare the string with triple quotes

my\_string1 = ''' Sam says "Hi everyone"'''

## 4) Type Casting

**Type casting is the process of converting one datatype to another datatype.**

Let's try to build an application that takes input from the user and returns the number entered + 10

a = input("enter the number : ")  
print(a+10)

enter the number: 20

---------------------------------------------------------------------------  
TypeError Traceback (most recent call last)  
<ipython-input-8-f16ddfb0e55c> in <module>  
 1 a = input("enter the number : ")  
----> 2 print(a+10)  
  
TypeError: can only concatenate str (not "int") to str

We get a type Error. Which says we cannot add an Integer data type with a string datatype. Remember when we learned the input() function we saw how all things entered by a keyboard including numbers and symbols are strings by default. So we need to Type cast the string representation of the number back to an integer to do an arithmetic operation.

a = input("enter the number : ")  
a\_int = int(a)  
print(a\_int+10)

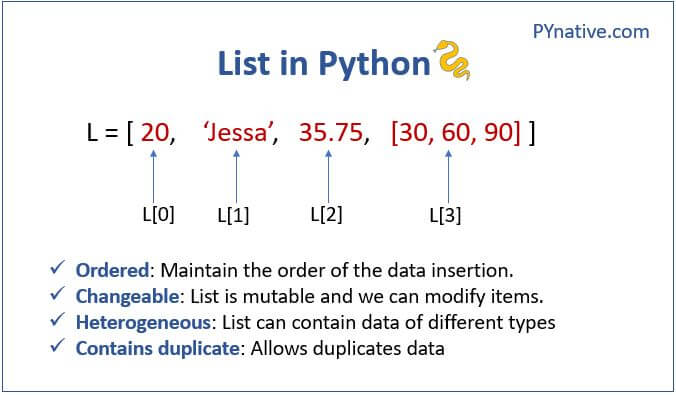
enter the number: 20  
30

We can convert any datatype to any other data type provided that it justifies this operation. When we convert a float to an int something called truncation happens. This is different from rounding off as the numbers after the decimal point are simply removed.

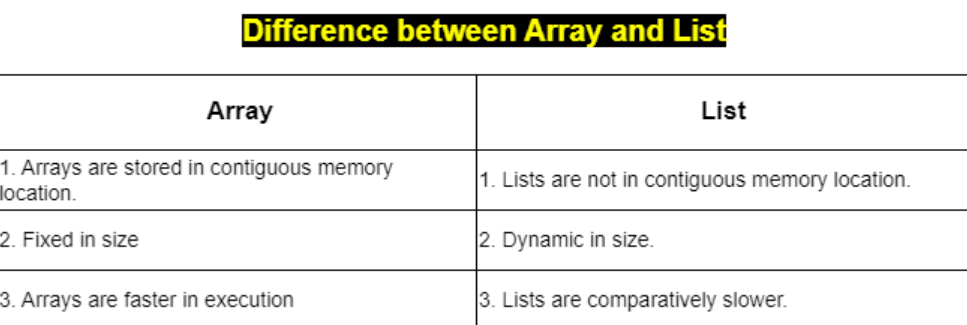
a = 12.12  
print(type(a))  
b = int(a)  
print(type(b))  
print(b)

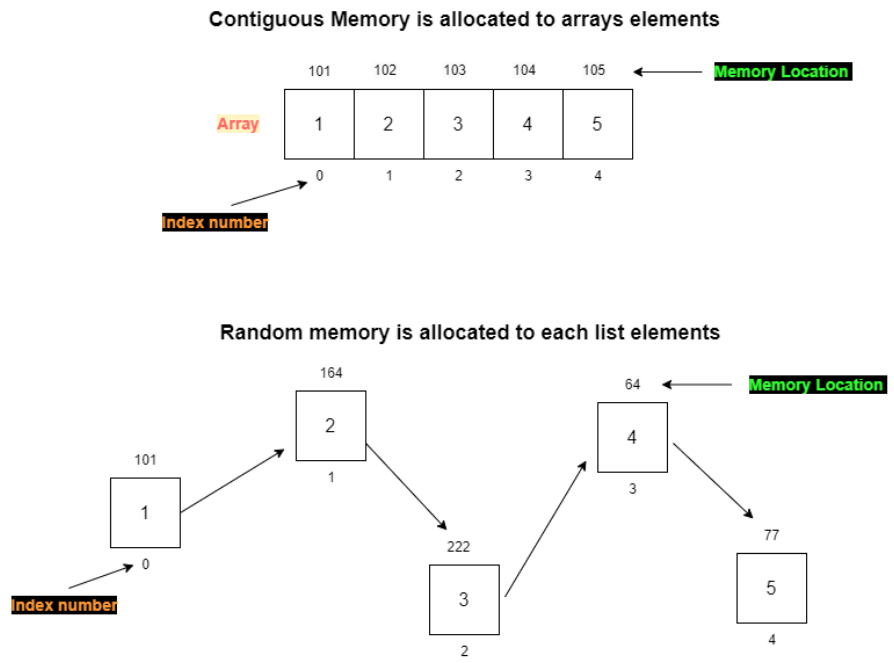
<class 'float'>  
<class 'int'>  
12

## 5) Lists



### Difference between list and array





Lists are constructed with brackets [ ] and commas separating every element in the list.

*# Assign a list to a variable named my\_list*  
my\_list = [1,2,3]  
print(my\_list)

[1, 2, 3]

We just created a list of integers, but lists can hold different object types. For example:

my\_list = ['A string',23,100.232,'o']  
print(my\_list)

['A string', 23, 100.232, 'o']

Lists can hold any object in Python. Including another list. Such a list can also be called a multi-dimensional list

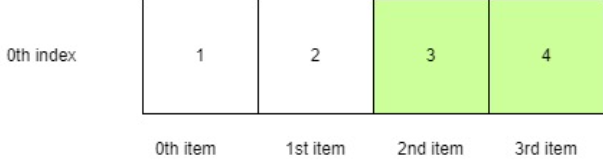
my\_list = [[1,2,3],[4,5,6],[7,8,9]]  
print(my\_list)

[[1, 2, 3], [4, 5, 6], [7, 8, 9]]

## 6) Indexing and Slicing

* Indexing refers to accessing individual elements on a collection data type
* Slicing refers to accessing a group of elements from the collection data type

### Indexing



my\_list = [11,22,33,44,55,66]  
my\_list[3]

44

**The Index number of elements starts from 0**

list elements: 11 22 33 44 55 66  
index numbers : 0 1 2 3 4 5

**The Indexing can also be applied to a string**

x = "hello"  
x[4]

'o'

The examples we saw were of single-dimensional indexing. We can also do indexing on multi-dimensional objects to access elements

my\_list = [[1,2,3],[4,5,6],[7,8,9]]  
my\_list[1]

[4, 5, 6]

As we can see indexing a multi-dimensional object with a single index returns the entire list itself. We can use an additional index parameter to access individual element

my\_list = [[1,2,3],[4,5,6],[7,8,9]]  
my\_list[1][1]

5

my\_list = [[1,2,3],'hello',[7,8,9]]  
my\_list[1][3]

'l'

#### After indexing we can replace the selected item with the item of our choice.

x = [10,20,30,40,50,60,70]  
print(x)  
x[1] = 2000  
print(x)

[10, 20, 30, 40, 50, 60, 70]  
[10, 2000, 30, 40, 50, 60, 70]

In the above example, we selected the 2nd item in the list using indexing and replaced that item with a 2000

**NOTE: A string inside a list also acts like a muti-dimensional object**

## Indexing continued

we can use negative indexing to get the elements from the last of a list or collection datatype

list elements: 11 22 33 44 55 66  
index numbers: -6 -5 -4 -3 -2 -1

my\_list = [11,22,33,44,55,66]  
my\_list[-2]

55

## Slicing

**syntax**

my\_list[start index : stop index : step size]

step size is 1 by default

my\_list = [11,22,33,44,55,66,77,88,99]  
my\_list[2:5:1]

[33, 44, 55]

my\_list = [11,22,33,44,55,66,77,88,99]  
my\_list[2:7]

[33, 44, 55, 66, 77]

**Note**: the last element is always the (index-1)

my\_list = [11,22,33,44,55,66,77,88,99]  
my\_list[2:7:2]

[33, 55, 77]

We can skip the first, last element, or both.

my\_list = [11,22,33,44,55,66,77,88,99]  
my\_list[:7:1] *##skipping the first element means we want all the elements*   
 *##from the start to the specified last index number*

[11, 22, 33, 44, 55, 66, 77]

x = "hello world"  
print(x[5:])

world

### Home Work

1. create a basic calculator
   * Take 2 numbers from the user (using input statement)
   * Print out the addition and multiplication of those 2 numbers
2. Try to grab the number 400 using a list

my\_list = [[1,2,3],'hello',[7,8,[100,200,300,400]]]

1. Try to grab the word 'own' from the below string

my\_str = "ON my own technology"

1. Replace the 3rd item in the list with the string "OMOTEC"

my\_lst = ["welcome", "to", "on my own technology"]

1. Try to grab the word 'OMOTEC' from the list

my\_list = [[1,2,3],'hello',[7,8,[100,200,"OMOTEC",400]]]

1. Try to grab the word 'C' from the list below

my\_list = ['O','M','O','T','E','C',]

6. Find the position of 55 from the list using index fruits = [22,33,42,55,61]

7. Find the position of bat list2 = ['cat', 'bat', 'mat', 'cat', 'pet']

8. Use the step parameter to return every third item using the slice:

a = ("a", "b", "c", "d", "e", "f", "g", "h")

9. Start the slice object at position 3, and slice to position 5, and return the result:

a = ("s", "e", "n", "t", "e", "n", "c", "e")