Commands

```
In [1]: x="Hello"
        print(x)
        print(type(x))
```

```
Hello
<class 'str'>
```

DataTypes in Python

DataTypes in Python is categorized as:
1) Primitive Type:- Numeric Type, String, Boolean.
2) Non-primitive Type:- List, Arrays, Tuples, Sets.

Built in Data types in python are of five types:

1. None Type
2. Numeric Type
3. Sequences
4. Sets
5. Mappings

None Type

The None data type represents an object that does not contain any value. In language like Java, it is called ‘null’ object. But in python, it is called ‘None’ object.

In [2]:
```
a = None
print(type(a))
```

```
<class 'NoneType'>
```

Numeric Type

1. int:int data type represents an integer numbers. An integer number is a number without any decimal point or fraction part. a= 15; b=-30; c=0
2. float:float data type represents an floating point numbers. An floating number is a number withdecimal point or fraction part. a= 55.679995 ; b= 22.55e3
3. Complex:complex number is a number that is written in the form of “a+b j” or “a+b J”. where a is a real part and b is a imaginary part. a= - 1 – 5.5 j

In [3]:
```
a = 55.6789995
b= 30.55e3

print(type(a))
print(b)
```

```
<class 'float'>
30550.0
```

Boolean Type

The Bool data type represents Boolean types. There are two Boolean values “True” and “False”. Python internally represents True as 1 and false as 0.

In [4]:
```
d = False
print(type(d))
```

```
<class 'bool'>
```

Sequences Type

1. Python String
2. Bytes
3. Bytearray
4. List
5. Tuple
6. Range

A sequence represents a group of elements or items:

1. **Python String**: str represents string data type. A string is a group of characters. Strings are enclosed in single quotes or double quotes.

In [5]:
```
s1 = "Priyanka"
s2 = "Garach"
s1 * 2


a = "5"
b = int(a)
print(type(b))
```

```
<class 'int'>
```

1. **Bytes** :bytes data type represents a group of byte numbers just like an array does. The byte number is any positive integer from 0 to 255.

In [6]:
```
elements= [10, 20, 30,40]  #list
x = bytes(elements)
print(type(elements))
```

```
#to print single element
x[1]
```

```
<class 'list'>
```

Out[6]:
```
20
```

1. **Bytearray** :bytearray data type represents a group of byte numbers just like an bytes does. But the main difference is bytes type array can not be modified but bytearray type array can be modified.

In [7]:
```
elements= [10, 20, 30, 40]
x = bytearray(elements)
x[1] = 80
print(type(x))
```

```
<class 'bytearray'>
```

1. **List** :List is an ordered sequence of values, where each value is identified by an index.In Python programming, a list is created by placing all the items (elements) inside a square bracket [ ], separated by commas.It can have any number of items and they may be of different types (integer, float, string etc.).

In [8]:
```
mylist = [1, 5.5 , "Hi"]
print(type(mylist))
print(mylist[2])

mylist[2] = 89
print(mylist)
```

```
<class 'list'>
Hi
[1, 5.5, 89]
```

1. **Tuple**: A tuple is similar to the list in many ways. Like lists, tuples also contain the collection

of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

In [9]:
```
mytuple = (1, 5.5 , "Hi")
print(type(mytuple))

mytuple[2] = 89
```

```
<class 'tuple'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-726c5cd247c2> in <module>
      2 print(type(mytuple))
      3 
----> 4 mytuple[2] = 89

TypeError: 'tuple' object does not support item assignment
```

1. **Range**: range represents the sequence of numbers. The numbers in range are not modified.

In [10]:
```
r = range(10)
print(r)
for i in r:
    print(i)  #it will print 0 to 9
```

```
range(0, 10)
0
1
2
3
4
5
6
7
```

```
8
9
```

Sets

A set is a unordered collection of elements. It means that the elements may not appear in the same order as they are entered into set. Set does not accept duplicate elements. {} are used for Set in python.

Set: set are written in this form: s = {10, 20, 30, 40}
print(s)
It may display output: {40, 20, 10, 30}

Frozenset:same as set but we cannot modify values. s = {10, 20, 30, 40}
fs = frozenset (s)
print(fs)

In [11]:
```
s = {10, 20, 30, 40}
print(s)
```

```
{40, 10, 20, 30}
```

Mapping

A map represents a group of elements in the form of key and value pairs so that when the key is given, we can retrieve the value associated with it. The **dict (dictionary) data type** is a example of map.

Dictionary data type

In [12]:
```
my_dict = {'apple':'hi all', 'ball':2,'ball':3}
print(my_dict)
```

```
{'apple': 'hi all', 'ball': 3}
```

In [13]:
```
my_dict['apple']
```

Out[13]:
```
'hi all'
```

Summary

Here are four collection data types in the Python programming language:

- List is a collection which is ordered and changeable. Allows duplicate members.
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
- Set is a collection which is unordered and unindexed. No duplicate members.
- Dictionary is a collection which is unordered, changeable and indexed. No duplicate members.

Branching in Python

In [14]:
```
num=0
if num>0:
    print(num, "is positive no.")
elif num==0:
    print(num, "is zero no.")
else:
    print(num,"is -ve no.")

print("Always printed")
```

```
0 is zero no.
Always printed
```

In [16]:
```
# nested if
```

```
num = float(input("Enter a number: "))
if num>= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

```
Enter a number: -5
Negative number
```

In [17]:
```
x = 10
y = 12
print('x > y is',x>y)
```

```
x > y is False
```

In [18]:
```
x = True
y = False
print('x and y is', x or y)
```

```
x and y is True
```

In [19]:
```
x=10
print(x<<2)
```

```
40
```

In [20]:
```
#operators
x1 = 'Hello world'
print('Z' not in x1)
```

```
True
```