Data types : Data types are always represent what type of data to be allowed in the specified memory location.

In Python we have 2 categories

1.Basic Types

a.Number Types [integer | float | complex]
b.bool type
c.None Type

Number Types :

Integer type :

1.If any Variable is initialized with a number without decimal , then they are treated as integer type
2.Predefined class for int type is <class 'int'>
3.It may be +ve integer or –ve Integer

Eg: x=10 → int x=10; [in c][java] x=10; [JavaScript]
type(x) #<class 'int'>

y=-123
type(y) #<class 'int'>

```
>>>
z=12345678912345678912345678912345678912345678923468237468
263487628347682734682638476283468273487236487628347682734 6
>>> type(z)
<class 'int'>
```

2.Float

1.If any variable is assigned with a number with decimal , then they considered by python as float
2.Predefine class for Float → <class 'float'>

Eg: x=3.14 [in java or c-language x=3.14; treated as double
In java or c-Language x=3.14f;]

y=-12.232334;
type(y) #<class 'float'>

```
>>> x=1.2e2
>>> type(x)
<class 'float'>
>>> print(x)
120.0

>>> y=1.2E3
>>> type(y)
<class 'float'>
>>> print(y)
1200.0
```

3.Complex

* Complex types are mostly used for mathematics and scientific related projects
* In math's complex numbers are represented by (a+bi)
* Here a is real part, b is imaginary part
* i represent under square root of -1
* But in python we have to represent by using (a+bj)
* Here real and imag part can be either of int or float , but internally it will represents in the form of float

```
>>> x=10+20j
>>> type(x)
<class 'complex'>
>>> print(x)
```

(10+20j)

To get real and imag values from the Complex object, then we have to use “real” and “imag” attributes

```
>>> print(x.real)
10.0
>>> print(x.imag)
20.0
>>> type(x.real)
<class 'float'>
>>> type(x.imag)
<class 'float'>
```

Bool type:

1.If any variable is assigned with either True or False , Then they are considered by python as bool type
2. Predefined Class for bool type <class ‘bool’>
3. In C-Language 0-False and 1-True But in java true or false

Eg: x=True
type(x) #<class ‘bool’>
y=False
type(y) #<class ‘bool’>

Eg 2:
x=1
type(x) #<class ‘int’>

Eg 3:
X=true #NameError

None Type:

- ***If you want declare a variable without any value then , you have to declare the variable is None.***

In C-Language :

```
int x;          //x will be initialized with Garbage .Value
```

In C and Java :

```
int x=NULL;
in Java
String s=null;
```

In Python:

```
x=None
type(x) #<class 'NoneType'>
```

```
>>> x=None
>>> type(x)
<class 'NoneType'>
>>> print(x)
None
```

X=10
X=None

Note: A reference variable can refer only an Object

Eg: x=10

- A reference variable can't refer more than object
- Eg: x="Shashi" , x=30
- An Object can be referred by more than one reference
  - Eg: x=10
  - y=x #ref.copy
  - z=y #ref.copy
  - id(x)
  - id(y)
  - id(z)

del is a keyword, used to delete object | variable from python memory.

Syn: del variable
Eg: del x
Eg: del x,y,z

del vs None

1.if you don't want variable in the python memory then we have delete variable | object using del keyword

2. Whenever you want variable but I want erase the value of variable then we have to use None "Data type"

```
>>> x=10
>>> print(x)
10
>>> x=None
>>> print(x)
None
>>> del x
>>> print(x)
```

```
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    print(x)
NameError: name 'x' is not defined
```

Collection Data types :

- It can hold multiple Values.
- Collections also called Sequences | iterable

String Collections:

- In C and Java Language, we have character data type [char] and char[ ] → String
- In Python there is no “char” data type
- In Python we have string data type, but it can store a char or group of chars [A|N]
- String is Collection i.e. it will store the data in the form Array
- The Data from string object can be accessed by using indexing system
- String is immutable
- In Python String literals can be represented using by ‘shashi’ or “shashi” or ‘’’shashi’’’ or “””shashi”””
- Predefined class for String is <class ‘str’>

Eg:

```
>>> s1='sai'
>>> type(s1)
<class 'str'>
```

```
>>> print(s1)
sai

>>> s2="baba"
>>> type(s2)
<class 'str'>
>>> print(s2)
baba

>>> s3="'Ramesh'"
>>> type(s3)
<class 'str'>
>>> print(s3)
Ramesh



>>> s4="""Shashi"""
>>> type(s4)
<class 'str'>
>>> print(s4)
Shashi
```

How to Read the Data From String Object ?

1.We Reading Using Indexing

* Indexing will be starts from 0 to n-1, this is from left to right
* Indexing will be starts from -1 to –n , this is from right to left

```
>>> s="welcome"
>>> print(s[0])  #w
>>> print(s[3])  #c
>>> print(s[-2]) #m
```

```
>>> print(s[-1])  #e

>>> print(s[10])   #invalid Index
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    print(s[10])
IndexError: string index out of range
```

2.Using Slicing

Slicing is nothing But Extracting substring

Syn : [start : end : step]
start | end | step all are optional

But here default value for start : 0
value for end : till end
value for step : 1

```
print(s[0:2:1])  #Exp.out: wel but Result : we
            |- not includes  [end rep end-1 ]

print(s[0:3:1]) #wel
print(s[0:3:]) #wel
print(s[:3:1]) #wel
print(s[:3:]) #wel
print(s[:3]) #wel
```

Eg 2:

```
>>> s="welCOME"
>>> print(s[3:6:1])
```

```
COM
>>> print(s[3:7:1])
COME
>>> print(s[3:7:])
COME
>>> print(s[3:7])
COME
>>> print(s[3::])
COME
>>> print(s[3:])
COME
>>> print(s[3]) #index
C
```

Even Position

```
print(s[0:7:2]) #wloe
print(s[0::2])
print(s[::2])
```

Odd Position

```
print(s[1:7:2]) #ecm
print(s[1::2]) #ecm
```

```
>>> s="WELcOME"
>>> s[3]='C'
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    s[3]='C'
TypeError: 'str' object does not support item assignment
```

List Collection:

- Predefined class for list <class 'list'>

- It will allow heterogeneous mixer of Objects
- List can be indexed
- List can be sliced
- It is mutable
- Insertion order is maintained
- All the Object in the list must be taken in between [ ] and Each object is separated by a [,]
- Duplicate objects are also allowed

```
>>> lst=[10,3.14,10,None,"Ramesh"]
>>> type(lst)
<class 'list'>
>>> print(lst)
[10, 3.14, 10, None, 'Ramesh']

>>> print(lst[0])
10
>>> print(lst[-1])
Ramesh

>>> print(lst[0:4:1])
[10, 3.14, 10, None]

>>> lst[3]="Shashi"
>>> print(lst) #[10, 3.14, 10, 'Shashi', 'Ramesh']
```

Tuple:

- Predefined class for tuple is <class ‘tuple’>
- It will allow heterogeneous mixer of Objects
- Tuple can be indexed
- tuple can be sliced
- Insertion order is maintained

- All the Object in the tuple must be taken in between ( ) and Each object is separated by a [,]
- Duplicate objects are also allowed
- It is immutable

```
>>> t=(10,3.14,10,None,"Ramesh")
>>> type(t)
<class 'tuple'>

>>> print(t)
(10, 3.14, 10, None, 'Ramesh')

>>> print(t[1])
3.14
>>> print(t[-2])
None

>>> print(t[2:5:1])
(10, None, 'Ramesh')

>>> t[-2]="Amma"
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    t[-2]="Amma"
TypeError: 'tuple' object does not support item assignment
```

SET Collection:

- Predefined class for set is <class ‘set’>
- It will allow heterogeneous mixer of Objects
- Insertion order is not maintained

- All the Objects in the set must be taken with in the { } and each object should be sep by ,
- Duplicate objects are not allowed
- Set Object doesn't support indexing and slicing
- Set in mutable

```
>>> s={10,3.14,10,None,"Ramesh"}
>>> type(s)
<class 'set'>

>>> print(s)
{None, 10, 3.14, 'Ramesh'}

>>> print(s[0])
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    print(s[0])
TypeError: 'set' object is not subscriptable

>>> print(s[0:3:1])
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    print(s[0:3:1])
TypeError: 'set' object is not subscriptable

>>> print(s)
{None, 10, 3.14, 'Ramesh'}
>>> s.add("SssiT")  #it is mutable
>>> print(s)
{3.14, None, 10, 'SssiT', 'Ramesh'}
```

Dictionary:

- If you want store group of objects in the form of key and value pairs, then we have to use dictionary collection
- Key and value should be separated by :
- Key and value pair is called an item
- Each item should be separated by ,
- All items must be taken with in the { }
- Keys must be an immutable but values can be any thing

```
>>> stu={"sno":101,"age":34,"name":"sudha"}
>>> type(stu)
<class 'dict'>
>>> print(stu)
{'sno': 101, 'age': 34, 'name': 'sudha'}

>>> print(stu['name'])
sudha

>>> print(stu['age'])
34

>>> print(stu['scity'])
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    print(stu['scity'])
KeyError: 'scity'

>>> print(stu)
{'sno': 101, 'age': 34, 'name': 'sudha'}
>>> stu['name']='Radha'
>>>
>>> print(stu)
{'sno': 101, 'age': 34, 'name': 'Radha'}
```

```
>>> lst=[]  #Creating and empty list
>>> type(lst)
<class 'list'>

>>> t=()  #Creating an empty tuple
>>> type(t)
<class 'tuple'>

>>> d={}  #Creating an empty dictionary
>>> type(d)
<class 'dict'>

>>> s={}
>>> type(s)
<class 'dict'>

>>> s=set() #Creating an empty set Collection
>>> type(s)
<class 'set'>
```