DATA TYPES IN PYTHON

To allocate sufficient amount of memory space in main memory of computer for storing input/data/literals .

In python we have 14 Data types:1.int 2.float 3.bool 4.complex 5.str 6.bytes 7.bytearray 8.range 9.list 10.tuple #till is ordered 11.set 12.frozenset 13.Dice 14.NoneType

1.int

Store integer data /Whole numbers/values withot decimal

In [4]:
```
a=78
print(a,type(a))
```

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

In [6]:
```
b=897
b
```

Out[6]: 897

In [8]:
```
c=808
print(c,type(c))
```

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

In [10]:
```
c=889
c
```

Out[10]: 889

In [12]:
```
c=89
print(c,type(c))
```

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

In [14]:
```
d=89
d
```

Out[14]: 89

2.float

Store floating point value,allow only decimal value

In [17]:
```
a=90.0
print(a,type(a))
```

```
90.0 <class 'float'>
```

In [19]:
```
b=0.89
print(b,type(b))
```

```
0.89 <class 'float'>
```

In [23]:
```
c=899.98
print(c,type(c))
```

```
899.98 <class 'float'>
```

In [25]:
```
d=89.89
print(d,type(d))
```

```
89.89 <class 'float'>
```

In [27]:
```
c=89.89
print(c,type(c))
```

```
89.89 <class 'float'>
```

In [29]:
```
a=989.89
print(a,type(a))
```

```
989.89 <class 'float'>
```

3.bool

Allow only boolean value that is 'True' and 'False' here True=1 and False=0 by default.

In [33]:
```
a=True
```

```
print(a,type(a))
```

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

In [35]:
```
a=True+False
print(a,type(a))
```

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

In [39]:
```
type(True)
```

Out[39]:
```
bool
```

In [41]:
```
a=(True+False-True/False)-True     #If ans is 0 in bool itss always gives ZeroDivisonError
print(a)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[41], line 1
----> 1 a=(True+False-True/False)-True
      2 print(a)

ZeroDivisionError: division by zero
```

In [45]:
```
a=True/False
a
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[45], line 1
----> 1 a=True/False
      2 a

ZeroDivisionError: division by zero
```

In [47]:
```
a=True//False
a
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[47], line 1
----> 1 a=True//False
      2 a

ZeroDivisionError: integer division or modulo by zero
```

In [49]:
```
a=True%False
a
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[49], line 1
----> 1 a=True%False
      2 a

ZeroDivisionError: integer modulo by zero
```

In [51]:
```
a=False/True
print(a)
```

```
0.0
```

In [53]:
```
a=True*False
a
```

Out[53]:
```
0
```

In [57]:
```
a=False//True
a
```

Out[57]:
```
0
```

4.complex

Allow imagenary and complex value in the form of 'a+bj' here a=real and b=imag ,j defines complex type ,complex can not work on floordiv and modulo itzz gives TypeError.

In [63]:
```
a=4+7j
print(a,type(a))
```

```
(4+7j) <class 'complex'>
```

In [65]:
```
a=8+5j
print(a,type(a))
```

```
(8+5j) <class 'complex'>
```

In [67]:
```
a=88-89j
print(a,type(a))
```

```
(88-89j) <class 'complex'>
```

In [69]:
```
a=7*7j
print(a,type(a))
```

```
49j <class 'complex'>
```

In [71]:
```
a=7/8j
print(a,type(a))
```

```
-0.875j <class 'complex'>
```

In [73]:
```
a=8//89j
print(a,type(a))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[73], line 1
----> 1 a=8//89j
      2 print(a,type(a))

TypeError: unsupported operand type(s) for //: 'int' and 'complex'
```

In [75]:
```
a=9%8j
print(a,type(a))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[75], line 1
----> 1 a=9%8j
      2 print(a,type(a))

TypeError: unsupported operand type(s) for %: 'int' and 'complex'
```

In [77]:
```
a=89e78+88j
print(a,type(a))
```

```
(8.9e+79+88j) <class 'complex'>
```

In [79]:
```
a=89-8j
print(a,type(a))
```

```
(89-8j) <class 'complex'>
```

In [81]:
```
___=8
print(__)
```

```
{"dataframes": [], "user": "shaba"}
```

5.str

string is used to store collection of any value,char and symblos in single quotes or double quotes otherwise it gives NameError,or str only allow either single quote or double else gives SyntaxError. Python has two types string 1.Single line:allow single quotes or double quotes 2.Multiline string:allow trible single quotes or trible double quotes string allow Indexing and Slicing

In [ ]:
```
Single line string
```

In [84]:
```
a=hello
print(a,type(a))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[84], line 1
----> 1 a=hello
      2 print(a,type(a))

NameError: name 'hello' is not defined
```

In [86]:
```
a="hello"
print(a,type(a))
```

```
hello <class 'str'>
```

In [88]:
```
a='hello world'
print(a,type(a))
```

```
hello world <class 'str'>
```

In [90]:
```
a='65,89,pyhon'
print(a,type(a))
```

```
65,89,pyhon <class 'str'>
```

In [92]:
```
a="788,7977,hello,77"
print(a,type(a))
```

```
788,7977,hello,77 <class 'str'>
```

In [94]:
```
a='hell"
print(a,type(a))
```

```
  Cell In[94], line 1
    a='hell"
      ^
SyntaxError: unterminated string literal (detected at line 1)
```

In [96]:
```
a='78&.88*,hello'
print(a,type(a))
```

```
78&.88*,hello <class 'str'>
```

In [98]:
```
a='*,&,^^,(),('
print(a,type(a))
```

```
*,&,^^,(),( <class 'str'>
```

Multiline string

In [104…
```
a='''hello
how are you
'''
print(a,type(a))
```

```
hello
how are you
 <class 'str'>
```

In [106…
```
a="""Ittzz me alina
from hyd
"""
print(a,type(a))
```

```
Ittzz me alina
from hyd
 <class 'str'>
```

In [108…
```
a='''append,remove,index,count,pop,clear,copy,sort,reverse,extend'''
print(a,type(a))
```

```
append,remove,index,count,pop,clear,copy,sort,reverse,extend <class 'str'>
```

In [114…
```
a="""append,remove,count"""
print(a,type(a))
```

```
append,remove,count <class 'str'>
```

In [124…
```
for i in a:
    print(i)
```

```
  Cell In[124], line 2
    print(i,end=,)
            ^
SyntaxError: expected argument value expression
```

In [128…
```
a='''itxx ,hello,workd'''
for i in a:
    print(i,end=',[::5]')
```

```
i,[::5]t,[::5]x,[::5]x,[::5] ,[::5],,[::5]h,[::5]e,[::5]l,[::5]l,[::5]o,[::5],,[::5]w,[::5]o,[::5]r,[::5]k,[::5]
d,[::5]
```

In [7]:
```
str='hello,1,0,9,0,-1,088'
print(str,type(str))
```

```
hello,1,0,9,0,-1,088 <class 'str'>
```

6.bytes

Used in network program to transmit the data between two ends Provides sequence of values in the range of 0-256-1 that is 0-255 bytes is human-unreadable format allow Indexing and Slicing bytes object is immutable.

In [179…
```
a=9,89,99
print(a)
```

```
(9, 89, 99)
```

In [187…
```
b=bytes(a)
print(b)
```

```
b'\tYc'
```

In [171…
```
for i in b:
    print(i)
```

```
99
99
77
78
68
```

In [201…
```
k=9
p=bytes(k)
for i in range(k):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
```

In [203…
```
a=88,89,88
b=bytes(a)
for i in b:
    print(i)
```

```
88
89
88
```

In [205…
```
b[1]
```

Out[205… 89

In [211…
```
b[0]
```

Out[211… 88

In [207…
```
p=bytes(k)
for i in p:
    print(i)
```

```
0
0
0
0
0
0
0
0
0
```

In [163…
```
for i in p:
    print(i)
```

```
0
0
0
0
0
0
0
0
0
```

In [189…
```
a=(99,99,77,78,68)
print(a,type(a))
```

```
(99, 99, 77, 78, 68) <class 'tuple'>
```

In [161…
```
b=bytes(a)
for i in b:
    print(i)
```

```
99
99
77
78
68
```

In [215…
```
a=[88,89,9,89]
```

```
b=bytes(a)
for i in b:
    print(i)
```

```
88
89
9
89
```

In [217…
```
b[2] ,type(b)
```

Out[217… (9, bytes)

In [3]:
```
l=[79,65,54,78,89,88]
print(l,type(l))
```

```
[79, 65, 54, 78, 89, 88] <class 'list'>
```

In [24]:
```
a=[88,89,98]
print(a,type(a))
```

```
[88, 89, 98] <class 'list'>
```

In [26]:
```
b=bytes(a)
for i in b:
    print(i)
```

```
88
89
98
```

In [16]:
```
id(b)
```

Out[16]: 2072776275984

In [32]:
```
a.append(9) #here 9 is int arg
print(a)
```

```
[88, 89, 98, 9, 9, 9]
```

In [34]:
```
b=bytes(a)
for i in b:
    print(i)
```

```
88
89
98
9
9
9
```

In [36]:
```
id(b)
```

Out[36]: 2072777920304

In [5]:
```
k=bytes(l)
for i in k:   #i means iteration means value after value
    print(i)
```

```
79
65
54
78
89
88
```

In [18]:
```
b.append(9)
print(b)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[18], line 1
----> 1 b.append(9)
      2 print(b)

AttributeError: 'bytes' object has no attribute 'append'
```

whenever you write the code with colun by default your cell(prompt) automatic understand user enter loop and i have to give space called indentation

7.bytearray

Used in network program to transmit the data between two ends Provides sequence of values in the range of 0-256-1 that is 0-255 ,it gives ValueError bytes is human-unreadable format also known as cipher text allow Indexing and Slicing bytes object is mutable.

In [15]:
```
a=88,78,87,88,88,878
b=bytearray(a)
for i in b:
    print(i)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[15], line 2
      1 a=88,78,87,88,88,878
----> 2 b=bytearray(a)
      3 for i in b:
      4     print(i)

ValueError: byte must be in range(0, 256)
```

In [9]:
```
a=[99,88,77,66,55,44,33,22]
b=bytearray(a)
print(b,type(b),id(b))
```

```
bytearray(b'cXMB7,!\x16') <class 'bytearray'> 2705315034480
```

In [11]:
```
for i in b:
    print(i)
```

```
99
88
77
66
55
44
33
22
```

In [19]:
```
a=[99,88,77,66,55,44,33,22,5,77]
b=bytearray(a)
for i in b:
    print(i,id(b))
```

```
99 2705274504752
88 2705274504752
77 2705274504752
66 2705274504752
55 2705274504752
44 2705274504752
33 2705274504752
22 2705274504752
5 2705274504752
77 2705274504752
```

In [23]:
```
b[0]=14
print(b,id(b))
for i in b:
    print(i)
```

```
bytearray(b'\x0eXMB7,!\x16\x05M') 2705274504752
14
88
77
66
55
44
33
22
5
77
```

In [27]:
```
lst=[99,88,77,7,55,44,33,22,99]
b=bytearray(a)
print(b,type(b))
```

```
bytearray(b'cXMB7,!\x16\x05M') <class 'bytearray'>
```

In [35]:
```
for i in b[::-1]:
    print(i)
```

```
77
5
22
33
44
55
66
77
88
99
```

In [45]:
```
a={89,99,88}
b=bytearray(a)
print(b,type(b))
```

```
bytearray(b'XYc') <class 'bytearray'>
```

8.range

To store sequence of numerical int values by maintainig equal interval of value(step) it can be treated in 3 ways 1.var=range(value) # 0 to -1 2.var=range(begin,end) 3.var=range(begin,end,-1)

In [52]:
```
r=range(10)
for i in r:
    print(i)
print(r,type(r))
```

```
0
1
2
3
4
5
6
7
8
9
range(0, 10) <class 'range'>
```

In [54]:
```
r=range(10,21)
print(r,type(r))
```

```
range(10, 21) <class 'range'>
```

In [56]:
```
for i in r:
    print(i)
```

```
10
11
12
13
14
15
16
17
18
19
20
```

In [58]:
```
r=range(10,21,2)
print(r,type(r))
```

```
range(10, 21, 2) <class 'range'>
```

In [72]:
```
for i in r[::-1]:  #reverse of obj
    print(i)
```

```
20
18
16
14
12
10
```

In [74]:
```
for i in range(100,80,2):
    print(i)
```

In [76]:
```
for val in range(1000,499,100):
    print(val)
```

In [100…
```
print(range(100,111)[-1]):
```

```
  Cell In[100], line 1
    print(range(100,111)[-1]):
                              ^
SyntaxError: invalid syntax
```

In [104…
```
r=range(100,111)
print(r[0])
```

```
100
```

In [108…
```
n=9
for i in range(1,11):
```

```
    print(n*i)
```

```
9
18
27
36
45
54
63
72
81
90
```

n=99

for i in range(1,11): print(n,"",*i*,"=",*n*i)

In [ ]:
```
n=14
for i in range(1,11):

```

9.list

The purpose of list is that to store multiple values either same or different or both and separated by comma ,value can be unique and duplicates. list is mutable

In [4]:
```
l1=[10,20,30,40,50,60,20,10]
print(l1,type(l1))
```

```
[10, 20, 30, 40, 50, 60, 20, 10] <class 'list'>
```

In [28]:
```
l2=[10,"rossum",2.4,7+7j]
print(l2,type(l2))
```

```
(10, 'rossum', 2.4, (7+7j)) <class 'tuple'>
```

In [10]:
```
len(l2)
```

Out[10]:
```
4
```

In [12]:
```
len(l3)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 len(l3)

NameError: name 'l3' is not defined
```

In [14]:
```
l3=[]
len(l3)
```

Out[14]:
```
0
```

l4=list()

print(l4,type(l4))

In [18]:
```
len(l4)
```

Out[18]:
```
0
```

In [20]:
```
l1=[11,-33,88,99]
print(l1,type(l1))
```

```
[11, -33, 88, 99] <class 'list'>
```

In [5]:
```
l2=[12,13,14,15,16,17,18]
print(l2,type(l2))
```

```
[12, 13, 14, 15, 16, 17, 18] <class 'list'>
```

In [13]:
```
l4=l1+l2                #concatenation:The process of joining two or more obj together and forming one single 
print(l4,type(l4))       #You can use the + operator, or use a built-in function like str.join() to obj.
```

```
[11, 33, 88, 99, 12, 13, 14, 15, 16, 17, 18] <class 'list'>
```

10.tuple

The purpose of tuple is to store values either same or different or both values can be unique and duplicate. tuple belongs immutable because obj does not support item assignment.

In [16]:
```
tpobj=tuple()           #empty tuple
print(tpobj,type(tpobj))
```

```
() <class 'tuple'>
```

In [18]:
```
tp=()           #empty tuple
print(tp,type(tp))
```

```
() <class 'tuple'>
```

In [22]:
```
tp=(77,-8,99)      #non-empty tuple
print(tp,type(tp))
```

```
(77, -8, 99) <class 'tuple'>
```

In [24]:
```
tp=(10,20,30,40,50,60,110,10)
print(tp,type(tp))
```

```
(10, 20, 30, 40, 50, 60, 110, 10) <class 'tuple'>
```

In [26]:
```
t2=("rossum","hyd","Telangana","Alina","nareshit")
print(t2,type(t2))
```

```
('rossum', 'hyd', 'Telangana', 'Alina', 'nareshit') <class 'tuple'>
```

In [32]:
```
t3='hello','hyd','world',"prakash sir",100
print(t3,type(t3))
```

```
('hello', 'hyd', 'world', 'prakash sir', 100) <class 'tuple'>
```

11.set

Purpose of set is store multiple values eihther same different or both,does not maintain insertation order,print unique values from obj, obj of set is both mutable(in case of adding elements) and immutable(in case of item assignment). Does not allow IS it print TypeError.

In [41]:
```
s=set() #empty set is one which does not contain any element note set={} is dict not set
print(s,type(s))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[41], line 1
----> 1 s=set()
      2 print(s,type(s))

TypeError: 'dict' object is not callable
```

In [47]:
```
set={1,3,4,5,'hello','print',98}
print(set,type(set))
```

```
{1, 98, 3, 4, 5, 'print', 'hello'} <class 'set'>
```

In [49]:
```
se={1,2,3,4,5,6,7,8,9,0,1,0,1}
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9} <class 'set'>
```

In [51]:
```
se[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[51], line 1
----> 1 se[0]

TypeError: 'set' object is not subscriptable
```

In [55]:
```
 s1={'hell','print',1.3,5,5,6,7,8,1,2}
print(s1,id(s1))
```

```
{1.3, 2, 'print', 1, 5, 6, 7, 8, 'hell'} 1849049822624
```

In [57]:
```
s1.add(9)
print(s1,id(s1))
```

```
{1.3, 2, 'print', 1, 5, 6, 7, 8, 9, 'hell'} 1849049822624
```

12.frozenset

Purpose of frozenset is store multiple values eihther same different or both,does not maintain insertation order,print unique values from obj, obj of frozenset is immutable. Does not allow IS it print TypeError. elements of frozenset must be obtain from different obj like list and tuple,fs maintain value order according to position.

In [65]:
```
fros=frozenset()
print(fros,type(fros))
```

```
frozenset() <class 'frozenset'>
```

In [67]:
```
rf='hello',1,2,{'hell','print',3},98
print(rf,type(rf))
```

```
('hello', 1, 2, {'hell', 3, 'print'}, 98) <class 'tuple'>
```

In [71]:
```
rf=frozenset({'hell','print',3})
print(rf,type(rf))
```

```
frozenset({'hell', 3, 'print'}) <class 'frozenset'>
```

In [75]:
```
hel=[{2,3,4,5,6,7,910,99,9,8}]
print(hel,type(hel),id(len))
```

```
[{2, 3, 4, 5, 6, 7, 99, 9, 8, 910}] <class 'list'> 1851059364832
```

In [89]:
```
s={2,3,4,5,9}
ss=frozenset(s)
print(ss,type(ss))
```

```
frozenset({2, 3, 4, 5, 9}) <class 'frozenset'>
```

In [87]:
```
s=[1,2,3,4,4,5,6,7,8,8,88]
se=frozenset(s)
print(se,type(se))
```

```
frozenset({1, 2, 3, 4, 5, 6, 7, 8, 88}) <class 'frozenset'>
```

In [93]:
```
se.add(89)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[93], line 1
----> 1 se.add(89)

AttributeError: 'frozenset' object has no attribute 'add'
```

In [95]:
```
se=add(20)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[95], line 1
----> 1 se=add(20)

NameError: name 'add' is not defined
```

In [97]:
```
se[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[97], line 1
----> 1 se[0]

TypeError: 'frozenset' object is not subscriptable
```

13.dict

The purpose of dict is to stor(key,value) in single var,key is unique and value may or may not unique, value of key is treated as immutable and values of vale is mutable.Origanally dict is mutable

In [2]:
```
dict={}
print(dict,type(dict))
```

```
{} <class 'dict'>
```

In [6]:
```
d=dict()
print(d,type(d))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 d=dict()
      2 print(d,type(d))

TypeError: 'dict' object is not callable
```

In [8]:
```
d={10:'python',20:"rossum",30:'listentome',40:'hello'}
print(d,type(d))
```

```
{10: 'python', 20: 'rossum', 30: 'listentome', 40: 'hello'} <class 'dict'>
```

In [10]:
```
d1={10:2.3,20:4.5,8:9.0,55:76}
print(d1,type(d1))
```

```
{10: 2.3, 20: 4.5, 8: 9.0, 55: 76} <class 'dict'>
```

In [12]:
```
d1[0]=9.0  #indices
print(d1)
```

```
{10: 2.3, 20: 4.5, 8: 9.0, 55: 76, 0: 9.0}
```

In [33]:
```
d3={}
print(d3,type(d3))
```

```
{} <class 'dict'>
```

In [37]:
```
d3['python']='rossum'
print(d3,type(d3))
```

```
{'python': 'rossum'} <class 'dict'>
```

In [41]:
```
d3['numpy']='travis'
print(d3)
```

```
{'python': 'rossum', 'numpy': 'travis'}
```

In [59]:
```
for v in d3.items():
    print(v)
```

```
('python', 'rossum')
('numpy', 'travis')
```

In [61]:
```
for k,v in d3.items():
    print(v)
```

```
rossum
travis
```

In [63]:
```
for k,v in d3.items():
    print(k,"---",v)
```

```
python --- rossum
numpy --- travis
```

In [49]:
```
d4={}
print(d4)
```

```
{}
```

In [53]:
```
d4[10]="pyhton"
d4[20]="travis"
d4[30]="head"
d4[40]="numpy"
print(d4,type(d4))
```

```
{10: 'pyhton', 20: 'travis', 30: 'head', 40: 'numpy'} <class 'dict'>
```

In [69]:
```
d=(3,4,{10:'numpy',20:'pandas',30:'travis',40:'python'},8,9)
print(d,type(d))
```

```
(3, 4, {10: 'numpy', 20: 'pandas', 30: 'travis', 40: 'python'}, 8, 9) <class 'tuple'>
```

In [71]:
```
for k,v in d.items():
    print(k)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[71], line 1
----> 1 for k,v in d.items():
      2     print(k)

AttributeError: 'tuple' object has no attribute 'items'
```

In [73]:
```
for k,v in d[2].items():
    print(k,v)
```

```
10 numpy
20 pandas
30 travis
40 python
```

In [89]:
```
for k,v in d[-3].items():
    print(k,v)
```

```
10 numpy
20 pandas
30 travis
40 python
```

In [113…
```
l1={10,20,30}
l2={'print','hello','how are you'}
h=zip(l1,l2)
print(h,type(h))
```

```
<zip object at 0x000001F4EE836740> <class 'zip'>
```

In [119…
```
d=dict(h)
print(d)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[119], line 1
----> 1 d=dict(h)
      2 print(d)

TypeError: 'dict' object is not callable
```

12.NoneType

nonetype is a keyword and act like an value in

In [19]:
```
a=None
print(a,type(a))
```

```
None <class 'NoneType'>
```

In [27]:
```
a=[]
print(a.clear())
```

```
None
```

In [29]:
```
a=set()
print(a.clear())
```

```
None
```

In [31]:
```
d1={10:20.9,200:2.3}
print(d1.get(100))
```

```
None
```

In [2]:
```
a={1:'one',2:'two',3:'three'}
for i in a:
    print(a ,':',a[i])
```

```
{1: 'one', 2: 'two', 3: 'three'} : one
{1: 'one', 2: 'two', 3: 'three'} : two
{1: 'one', 2: 'two', 3: 'three'} : three
```

In [4]:
```
for i in a:
    print(i,'*',a[i])
```

```
1 * one
2 * two
3 * three
```

In [ ]: