Datatypes

In [5]:

```
list=[1,2,3,4]
print(type(list))
list[0]=45
print(list)
```

```
<class 'list'>
[45, 2, 3, 4]
```

In [15]:

```
t=(1,3,4,5)
print(type(t))
print(t)
```

```
<class 'tuple'>
(1, 3, 4, 5)
```

In [10]:

```
t=(1)
print(type(t))
print(t)
```

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

In [8]:

```
t=(1,)
print(type(t))
print(t)
```

```
<class 'tuple'>
(1,)
```

In [14]:

```
s={"a","c","a","b","c"} # It does not follow insertion order,the elements will be randomly orderded
print(type(s))
print(s)
```

```
<class 'set'>
{'a', 'b', 'c'}
```

operators

In [19]:

```
sum_of_numbers=10+2
print(sum_of_numbers)
mulOfNumbers=10*10
print(mulOfNumbers)
```

```
12
100
```

In [18]:

```
d1=5/2 #float division
d2=5//2 #integer div
print(d1)
print(d2)
```

```
2.5
2
```

In [20]:

```
n1=10%2
n2=10/2
print(n1)
print(n2)
```

```
0
5.0
```

In [22]:

```
#logical operators
print(1 and 1)
print(0 and 1)
print(1 and 0)
```

```
1
0
0
```

In [30]:

```
23 and 7
```

Out[30]:

```
7
```

In [33]:

```
7 and 23
```

Out[33]:

```
23
```

In [31]:

```
23 or 7
```

Out[31]:

```
23
```

In [28]:

```
1 and 2
```

Out[28]:

```
2
```

In [29]:

```
2 and 1
```

Out[29]:

```
1
```

In [32]:

```
74 and 31
```

Out[32]:

```
31
```

In [34]:

```
bin(74)
```

Out[34]:

```
'0b1001010'
```

In [35]:

```
bin(31)
```

Out[35]:

```
'0b11111'
```

In [38]:

```
print(bin(74))
type(bin(74))
```

```
0b1001010
```

Out[38]:

```
str
```

In [41]:

```
#relational operators <,>,<=,>=,==,!=
1>2
```

Out[41]:

```
False
```

In [42]:

```
10<20
```

Out[42]:

```
True
```

In [1]:

```
#assignment operators
```

In [2]:

```
0 and 1
```

Out[2]:

```
0
```

In [3]:

```
7 & 1
```

Out[3]:

```
1
```

In [4]:

```
7 and 1
```

Out[4]:

```
1
```

In [5]:

```
1 and 7
```

Out[5]:

```
7
```

In [7]:

```
7 and 0
```

Out[7]:

```
0
```

In [8]:

```
0 and 7
```

Out[8]:

```
0
```

In [6]:

```
1 & 64
```

Out[6]:

```
0
```

In [9]:

```
1 | 64
```

Out[9]:

```
65
```

In [11]:

```
#membership operator  in  not in-->It checks the given element is present in list or not
l=[1,2,3,4,5]
print(1 in l)
print(3 not in l)
```

```
True
False
```

In [16]:

```
#identity operator
a=10
print(a is None)
```

```
False
```

In [13]:

```
a=10
print(a is not None)
```

```
True
```

In [19]:

```
#ternary operator-->?:--> It has 3 parts
#syntax: (condition) ? True Part : False Part in c
#if (condition) true part else false part in python
```

In [26]:

```
l=[]
l.append(10)
l.append(23.4) #append accepts only 1 argument
l.append(2+5j)
l.append([1,2,3])
print(l)
print(len(l))
```

```
[10, 23.4, (2+5j), [1, 2, 3]]
4
```

In [35]:

```
ls=[1,2,3,4,5,6]
print(ls.pop())  #pop() operates on index value,if index not given last value will be removed,it is index based,it returns a valu
ls
```

```
6
```

Out[35]:

```
[1, 2, 3, 4, 5]
```

In [36]:

```
ls=[1,2,3,4,5,6]
ls.pop(1)
ls
```

Out[36]:

```
[1, 3, 4, 5, 6]
```

In [33]:

```
ls=[1,2,3,4,5,6]
ls.remove(3)   #remove() value based,it returns None
ls
```

Out[33]:

```
[1, 2, 4, 5, 6]
```

In [38]:

```
a=[1,2,3,4,5]
a.insert(1,45)  #insert() accepts 2 parameters 1.index 2.object
a
```

Out[38]:

```
[1, 45, 2, 3, 4, 5]
```

In [41]:

```
a=[1,2,3,4]
b=[10,20,30]
a.extend(b)
print(a)
print(b)
```

```
[1, 2, 3, 4, 10, 20, 30]
[10, 20, 30]
```

In [43]:

```
a=[10,10,20,30,20,30,20,10,40]
res=a.count(10)
res
```

Out[43]:

```
3
```

In [45]:

```
a=[10,10,20,30,20,30,20,10,40]
res=a.count(11)
res
```

Out[45]:

```
0
```

In [49]:

```
a=[10,10,20,30,20,30,20,10,40]
print(len(a))
b=a.copy()
print(b)
```

```
9
[10, 10, 20, 30, 20, 30, 20, 10, 40]
```

In [56]:

```
a=[10,20,30]
c=[1]
a=c.copy()
print(a)
```

```
[1]
```

In [57]:

```
a=[11,12,13]
b=a
b[0]=100
print(a)
print(b)
```

```
[100, 12, 13]
[100, 12, 13]
```

In [58]:

```
a=[11,12,13]
b=a.copy()
b[0]=100
print(a)
print(b)
```

```
[11, 12, 13]
[100, 12, 13]
```

In [61]:

```
a=[11,12,13,14]
a.clear()
a
```

Out[61]:

```
[]
```

In [60]:

```
a=[1,2,3,4]
a.reverse()
a
```

Out[60]:

```
[4, 3, 2, 1]
```

In [62]:

```
a=[3,1,2]
a.sort()  #ascending order
a
```

Out[62]:

```
[1, 2, 3]
```

In [63]:

```
a=[3,1,2]
b=a.sort()  # it sorts only in existing list
print(a)
print(b)
```

```
[1, 2, 3]
None
```

In [64]:

```
a=[3,1,2]
a.sort(reverse=True)  #descending order
a
```

Out[64]:

```
[3, 2, 1]
```

In [66]:

```
a=[3,1,2]
b=sorted(a)  # sorted() will not sort in existing list,it will sort in newly assigned list
print("a=",a)
print("b=",b)
```

```
a= [3, 1, 2]
b= [1, 2, 3]
```

In [67]:

```
a=[3,1,2]
b=sorted(a,reverse=True)
print("a=",a)
print("b=",b)
```

```
a= [3, 1, 2]
b= [3, 2, 1]
```

Type Conversion

In [71]:

```
a='1'
b=3
c=int(a)+b
c  #or
```

Out[71]:

```
4
```

In [72]:

```
d=int('5')  #implicit
e=4
print(d+e)
```

```
9
```

In [73]:

```
a=float('2')  #explicit
b=5
c=a+b
print(c)
```

```
7.0
```

In [75]:

```
a="Mr. x is "
b=str(36)
c=" years old"
s=a+b+c
print(s)
```

```
Mr. x is 36 years old
```

In [84]:

```
a=list("12345")   #string to list
print(a)
b=map(int,a)   #map() is used to convert a list to certain datatype
print(b)
```

```
['1', '2', '3', '4', '5']
<map object at 0x0000016BC4367580>
```

In [85]:

```
a=list("12345")
print(a)
b=list(map(int,a))
print(b)
```

```
['1', '2', '3', '4', '5']
[1, 2, 3, 4, 5]
```

In [86]:

```
a=list("12345a")
print(a)
b=list(map(int,a)) #because a is not an integer
print(b)
```

```
['1', '2', '3', '4', '5', 'a']

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\Users\PRAVAL~1\AppData\Local\Temp/ipykernel_21380/1708125546.py in <module>
      1 a=list("12345a")
      2 print(a)
----> 3 b=list(map(int,a))
      4 print(b)

ValueError: invalid literal for int() with base 10: 'a'
```

In [88]:

```
a=input("enter a number1:")
b=input("enter a number2:")
c=a+b
c
```

```
enter a number1:1
enter a number2:2
```

Out[88]:

```
'12'
```

In [89]:

```
a=int(input("enter a number1:"))
b=int(input("enter a number2:"))
c=a+b
c
```

```
enter a number1:1
enter a number2:2
```

Out[89]:

```
3
```

In [ ]: