```
# Mutable Data structures
```

1.List

Creation of list

In [1]:
```
l=[]
print(l)
print(type(l))
```

```
[]
<class 'list'>
```

In [3]:
```
l=[10]
print(l)
print(type(l))
```

```
[10]
<class 'list'>
```

In [4]:
```
l=[1,2,3,4]
print(l)
print(type(l))
```

```
[1, 2, 3, 4]
<class 'list'>
```

Creation list with dynamic input

In [5]:
```
l=eval(input("Enter list:"))
print(l)
print(type(l))
```

```
Enter list:10,20,30
(10, 20, 30)
<class 'tuple'>
```

In [9]:
```
l=eval(input("Enter list:"))
print(l)
print(type(l))
```

```
Enter list:'abc'
abc
<class 'str'>
```

In [10]:
```
l=eval(input("Enter list:"))
print(l)
print(type(l))
```

```
Enter list:10
10
<class 'int'>
```

In [11]:
```
l=eval(input("Enter list:"))
print(l)
print(type(l))
```

```
Enter list:[10,20,30]
[10, 20, 30]
<class 'list'>
```

using list()

In [12]:
```
l=list(range(10,20,2))
print(l)
```

```
[10, 12, 14, 16, 18]
```

In [13]:
```
l="Arman"
x=list(l)
print(x)
```

```
['A', 'r', 'm', 'a', 'n']
```

using split() function

In [14]:
```
s="Learning Python is very easy"
l=s.split()
print(l)
```

```
['Learning', 'Python', 'is', 'very', 'easy']
```

List Mutability

In [15]:
```
l=[1,2,3,4,5]
l[2]=10
print(l)
```

```
[1, 2, 10, 4, 5]
```

In [16]:
```
l=(1,2,3,4,5)
l[2]=10
print(l)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-0de4cc17f410> in <module>
      1 l=(1,2,3,4,5)
----> 2 l[2]=10
      3 print(l)

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

Accessing the elements of list

- 1.By using index

In [17]:
```
l=[10,20,30,40,50,60]
print(l[0])
print(l[3])
print(l[-2])
print(l[10])
```

```
10
40
50

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-17-98d4d1f82ab1> in <module>
      3 print(l[3])
      4 print(l[-2])
----> 5 print(l[10])

IndexError: list index out of range
```

- 2.By using slicing operator

In [24]:
```
l=[1,2,3,4,5,6,7,8,9,10]
print(l[2:7:2])
print(l[4::2])
print(l[3:7])
print(l[8:2:-2])
print(l[4:100])
print(l[-5:-2])
print(l[-5:-2:-1])
```

```
[3, 5, 7]
[5, 7, 9]
[4, 5, 6, 7]
[9, 7, 5]
[5, 6, 7, 8, 9, 10]
[6, 7, 8]
[]
```

Mathematical operators

- 1.concatenation(+)

In [25]:
```
a=[1,2,3]
b=[4,5,6]
c=a+b
print(c)
```

```
[1, 2, 3, 4, 5, 6]
```

In [26]:
```
d=a+[4]
print(d)
```

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

- 2.repetition(*)

In [27]:
```
l=[1,2,3]
l1=l*3
print(l1)
```

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

Membership operator

In [28]:
```
l=[10,20,30,40,50]
print(10 in l)
print(10 not in l)
print(50 in l)
print(60 not in l)
```

```
True
False
True
True
```

Comparison operator for list

In [1]:
```
x=["Dog","Cat","Rat"]
y=["Dog","Cat","Rat"]
z=["DOG","CAT","RAT"]
print(x==y)
print(x==z)
print(x!=z)
```

```
True
False
True
```

In [2]:
```
x=[50,20,30]
y=[40,50,60,100,200]
print(x>y)
print(x>=y)
print(x<y)
print(x<=y)
```

```
True
True
False
False
```

In [3]:
```python
x=["Dog","Cat","Rat"]
y=["Rat","Cat","Dog"]
print(x>y)
print(x>=y)
print(x<y)
print(x<=y)
```

```
False
False
True
True
```

Aliasing and cloning of list

In [4]:
```python
l=[1,2,3,4,5]
x=l
print(x)
x[2]=10
print(x)
print(l)
print(id(x))
print(id(l))
```

```
[1, 2, 3, 4, 5]
[1, 2, 10, 4, 5]
[1, 2, 10, 4, 5]
1936226937152
1936226937152
```

In [5]:
```python
l=[1,2,3,4,5]
x=l.copy()
print(x)
x[2]=10
print(x)
print(l)
print(id(x))
print(id(l))
```

```
[1, 2, 3, 4, 5]
[1, 2, 10, 4, 5]
[1, 2, 3, 4, 5]
1936226784512
1936227111168
```

In [6]:
```python
l=[1,2,3,4,5]
x=l[:]
print(x)
x[2]=10
print(x)
print(l)
print(id(x))
print(id(l))
```

```
[1, 2, 3, 4, 5]
[1, 2, 10, 4, 5]
[1, 2, 3, 4, 5]
1936226937152
1936226978752
```

Nested list

In [12]:
```
n=[10,20,[30,40]]
print(n)
print(n[2])
print(n[2][1])
n[2].index(40)
```

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

Out[12]:
```
1
```

In [11]:
```
n=[[10,20,30],[40,50,60],[70,80,90]]
print(n[1][2])
```

```
60
```

Important functions of list

- 1.len()
- 2.count()

In [13]:
```
l=[1,2,2,3,3,4,4,5,6]
print(len(l))
print(l.count(2))
print(l.count(7))
```

```
9
2
0
```

- 3.index()

In [15]:
```
n=[1,2,2,2,3,3]
print(n.index(1))
print(n.index(2))
#print(n.index(4))
print(n.index(2,2,5))
```

```
0
1
2
```

- 4.append()

```
        ---> used to add item at the end of the list
```

In [16]:
```
l=["A","B","C"]
l.append("D")
l.append("E")
l.append([1,2,3])
print(l)
```

```
['A', 'B', 'C', 'D', 'E', [1, 2, 3]]
```

- 5.insert()

```
    ---> to insert item at the specific index position
    Syntax---> insert(index,value)
```

In [21]:
```
n=[1,2,3,4,5]
n.insert(2,10)
print(n)
n.insert(10,50)
print(n)
n.insert(-10,0)
print(n)
n.insert(-1,40)
print(n)
```

```
[1, 2, 10, 3, 4, 5]
[1, 2, 10, 3, 4, 5, 50]
[0, 1, 2, 10, 3, 4, 5, 50]
[0, 1, 2, 10, 3, 4, 5, 40, 50]
```

- 6.extend() ---> To add all items of one list to another list

```
    l1.extend(l2)---> all items present in l2 will be added to l1
```

In [22]:
```
l1=["Apple","Banana"]
l2=["Orange","Mango"]
l1.extend(l2)
print(l1)
print(l2)
```

```
['Apple', 'Banana', 'Orange', 'Mango']
['Orange', 'Mango']
```

In [23]:
```
l1=["Apple","Banana"]
l2=["Orange","Mango"]
l2.extend(l1)
print(l1)
print(l2)
```

```
['Apple', 'Banana']
['Orange', 'Mango', 'Apple', 'Banana']
```

In [24]:
```
l1=["amit","sumit"]
l1.extend("kumar")
print(l1)
```

```
['amit', 'sumit', 'k', 'u', 'm', 'a', 'r']
```

- 7.remove()

```
    ---> remove specified item from the list
    ---> if item is multiple times then only first occurence will
be removed
```

In [27]:

```
l=[1,2,1,3,2,3]
l.remove(1)
print(l)
l.remove(1)
print(l)
l.remove(1)
print(l)
```

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

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-27-bc0a31d196ba> in <module>
      4 l.remove(1)
      5 print(l)
----> 6 l.remove(1)
      7 print(l)

ValueError: list.remove(x): x not in list
```

- 8.pop()

```
  ---> it removes and returns the last element of the list
  ---> This is the only function which manipulates list and return
s some element
```

In [29]:

```
n=[10,20,30,40]
print(n.pop())
print(n.pop())
print(n.pop())
print(n.pop())
print(n.pop())
```

```
40
30
20
10

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-29-9ebe82cf1276> in <module>
      4 print(n.pop())
      5 print(n.pop())
----> 6 print(n.pop())

IndexError: pop from empty list
```

In [32]:
```
n=[1,2,3,4,5]
print(n.pop(1))
```
```
2
```

- 9.clear()

```
    ---> to remove all the elements of list
```

In [33]:
```
n=[1,2,3,4]
n.clear()
print(n)
```
```
[]
```

- 10.reverse()

In [34]:
```
n=[1,2,3,4,5]
n.reverse()
print(n)
```
```
[5, 4, 3, 2, 1]
```

sort()

In [36]:
```
n=[2,5,15,1,0]
n.sort()
print(n)
n.sort(reverse=True)
print(n)
```
```
[0, 1, 2, 5, 15]
[15, 5, 2, 1, 0]
```

In [37]:
```
l=["D","B","C","A"]
l.sort()
print(l)
```
```
['A', 'B', 'C', 'D']
```

In [38]:
```
n=[10,20,"A","B"]
n.sort()
print(n)
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-38-96b6e3a2e401> in <module>
      1 n=[10,20,"A","B"]
----> 2 n.sort()
      3 print(n)

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [41]:
```python
words=["Python","is","very","easy"]
words.sort(key=len)
print(words)
```

```
['is', 'very', 'easy', 'Python']
```

In [42]:
```python
words=["Python","is","very","easy"]
words.sort(key=len,reverse=True)
print(words)
```

```
['Python', 'very', 'easy', 'is']
```

In [43]:
```python
def myfunc(e):
    return len(e)
cars=['Ford','Mitsubishi','BMW']
cars.sort(key=myfunc)
print(cars)
```

```
['BMW', 'Ford', 'Mitsubishi']
```

Nested list as matrix

In [55]:
```python
n=[[10,20,30],[40,50,60],[70,80,90]]
for i in range(len(n)):
    print(n[i])
for l in range(len(n)):
    for m in range(len(n)):
        print(n[l][m]," ",end="")
    print()
```

```
[10, 20, 30]
[40, 50, 60]
[70, 80, 90]
10  20  30
40  50  60
70  80  90
```

List comprehension

- Syntax ---> list=[expression for item in list if condition]

In [56]:
```python
s=[x*x for x in range(1,11)]
print(s)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [57]:
```python
a=[1,2,3,4,5,6,7,8,9,10]
x=[num for num in a if num%2==0]
print(x)
```

```
[2, 4, 6, 8, 10]
```

In [58]:
```
y=[x+2 for x in range(10)]
print(y)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [61]:
```
y=(x+2 for x in range(10))
print(y)
print(tuple(y))
```

```
<generator object <genexpr> at 0x000001C2D02D5040>
(2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
```

WAP to perform a circular shift on a list to the right direction

In [76]:
```
l=[1,2,3,4,5,6,7]
shift=32
x=shift%len(l)
n=[]
for i in l:
    n=l[-x:]+l[:-x]
print(n)
```

```
[4, 5, 6, 7, 1, 2, 3]
```

WAP to print elements with frequency greater than a given value k

In [84]:
```
n=[1,1,1,1,2,2,2,3,4,4,5,5,5,6,6]
k=int(input("Enter k:"))
l=[]
for i in n:
    if(n.count(i)>k) and i not in l:
        l.append(i)
print(l)
```

```
Enter k:2
[1, 2, 5]
```

In [ ]:
```

```