LIST

1] List item acess

```
l=[10,20,30,40]
print(l)
print(l[1])
print(l[3])
print(l[-1])
print(l[-2])
```

OUTPUT

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

20

40

40

30
```

2] Insert and search

```
l=[10,20,30,40,50]

#for appending element at last
l.append(100)
print(l)

#for insert element at specific position
l.insert(2,130)
print(l)

#for checking element is present in list or not
print(15 in l)
print( 100 in l)

#for count of specific element in list
print(l.count(100))
l.insert(3,100)
print(l.count(100))
```

```
#for finding first index of element if its is not present then it will show an error
print(l.index(100))
```

OUTPUT:

[10, 20, 30, 40, 50, 100]

[10, 20, 130, 30, 40, 50, 100]

False

True

1

2

3

3] removal of item

```
l=[10,20,30,40,50,60]
print(l)

#for removing speficied element, and if specified element is not present in list then it
will show an error
l.remove(20)
print(l)

# for removing a last element of list
print(l.pop())




#for removing the element at specified position using pop
print(l)
print(l.pop(2))
print(l)
```

```
# del element using del
del l[1]
print(l)


l.append(130)
l.append(140)
l.append(160)
print(l)


# del element using slicing
del l[1:3]
print(l)


# if list is already empty and if you want to remove an element then it will show an
error
li=[]
print(li.pop())     #show error
```

OUTPUT :

```
[10, 20, 30, 40, 50, 60]

[10, 30, 40, 50, 60]

60

[10, 30, 40, 50]

40

[10, 30, 50]

[10, 50]

[10, 50, 130, 140, 160]

[10, 140, 160]
```

4] some general purpose

```
#min max sort function not work if you give some element as string but reverse will
work
#if all the element in list are string then max function will give you lexigraphically
largest string and min and sort
# function work similary, but sum function will not work


l=[10,20,30,40,5,50,120,60,70,80]

#for printing maximum elemnt in list
print(max(l))

#for printing minimum element in the list
print(min(l))

d=[10,20]
#for printing sum of element in list
print(sum(d))

print(l)
# for reversing the list
l.reverse()
print(l)

#for sorting elemnt in list
l.sort()
print(l)
```

Output :

```
120

5

30

[10, 20, 30, 40, 5, 50, 120, 60, 70, 80]

[80, 70, 60, 120, 50, 5, 40, 30, 20, 10]

[5, 10, 20, 30, 40, 50, 60, 70, 80, 120]
```

5] List advantages and disadvatages

list

advantages
1) Random acess
2) Cache Friendly

Disadvantages
1)insertion and deletion are costly operation
2)search is also costly when you don't have sorted data

but pop and append are constant time operation

**The primary difference between the list sort() function and the sorted() function is that the sort() function will modify the list it is called on. The sorted() function will create a new list containing a sorted version of the list it is given. The sorted() function will not modify the list passed as a parameter. If you want to sort a list but still have the original unsorted version, then you would use the sorted() function. If maintaining the original order of the list is unimportant, then you can call the sort() function on the list.

**The reverse() method edits the list to be in a reversed order. However, reversed() method takes a list and returns an iterator of it in reverse order.

6] Separate Even Odd:

```
def getEvenOdd(l):
    even=[]
    odd=[]

    for x in l:
        if x%2==0:
            even.append(x)
        else:
            odd.append(x)

    return even,odd

l=[10,15,26,31,40,50,60,87,97]
even, odd=getEvenOdd(l)

print("even value are ", even,)
print("odd value are ", odd)
```

OUTPUT :

```
even value are  [10, 26, 40, 50, 60]

odd value are  [15, 31, 87, 97]
```

7] average or mean of list

```
def average(l):
    sum=0
    for i in l:
        sum+=i
    n=len(l)
    return sum/n


l=[10,15,30,20,25]
print(average(l))


#using build in function
def averagge(l):
    return sum(l)/len(l)

print(average(l))
```

OUTPUT :

```
20.0

20.0
```

8] get smaller elements

```
def getSmaller(l,n):
    res=[]

    for x in l:
        if x<n:
            res.append(x)

    return  res

l=[10,60,80,12,30,80,99,41,2]
n=50
print(getSmaller(l,n))
```

OUTPUT :

```
[10, 12, 30, 41, 2]
```

9] list slicing :

```
l=[10,20,30,40,50]

print("l[0:5:2]->",l[0:5:2])

print("l[:4]->",l[:4])

print("l[2:]->",l[2:])

print("l[1:4]->",l[1:4])

print("l[4:1:-1]->",l[4:1:-1])

print("l[-1:-6:-1]->",l[-1:-6:-1])

print("l[::-1]->",l[::-1])

print("l[0:5]->",l[0:5])

print("l[:]->",l[:])
```

OUTPUT :

```
l[0:5:2]-> [10, 30, 50]

l[:4]-> [10, 20, 30, 40]

l[2:]-> [30, 40, 50]

l[1:4]-> [20, 30, 40]

l[4:1:-1]-> [50, 40, 30]

l[-1:-6:-1]-> [50, 40, 30, 20, 10]

l[::-1]-> [50, 40, 30, 20, 10]

l[0:5]-> [10, 20, 30, 40, 50]

l[:]-> [10, 20, 30, 40, 50]
```

10] difference slicing between list tuple string :

```
l1=[10,20,30]
l2=l1[:]

t1=(10,20,30)
t2=t1[:]

s1="ganesh"
s2=s1[:]

print("list having same element but doesn't have same id->",l1 is l2)
print("tuple having same elment has same id->", t1 is t2)
print("string of same value have same id->",s1 is s2)
```

OUTPUT :

list having same element but doesn't have same id-> False

tuple having same elment has same id-> True

string of same value have same id-> True

11] List comprehension :

```
l1=[x for x in range(11) if x%2==0]
print(l1)

l2=[x for x in range(11) if x%2!=0]
print(l2)

#code for smaller
def getsmaller(l,n):
    return [e for e in l if e<n]

l=[1,30,20,2,7,8,45,54]
n=22

print(getsmaller(l,n))


"""
```

```
OUTPUT
[0, 2, 4, 6, 8, 10]
[1, 3, 5, 7, 9]
[1, 20, 2, 7, 8]
"""
```

12] list comprehension def

```
def getEevenOdd(l):
    even=[x for x in l if x%2==0]
    odd=[x for x in l if x%2!=0]
    return  even, odd

l=[10,20,23,54,57,32,14,21,51]

even,odd=getEevenOdd(l)
print("even number are",even)
print("odd number are",odd)



"""
OUTPUT
even number are [10, 20, 54, 32, 14]
odd number are [23, 57, 21, 51]
"""
```

13] comprehension in string :

```
s="geekforgeeks"
l1=[x for x in s if x in "aeiou"]
print(l1)

l2=["geeks","ide","courses","gfg"]
l3=[x for x in l2 if x.startswith("g")]
print(l3)

l4=[x*2 for x in range(6)]
print(l4)


print("-------------------------")
l1=["geeks","fear","geeks","gfg","ide"]
l2=[x.upper() for x in l1 if x.startswith("g")]
print(l2)
```

OUTPUT :

```
['e', 'e', 'o', 'e', 'e']

['geeks', 'gfg']

[0, 2, 4, 6, 8, 10]

-------------------------

['GEEKS', 'GEEKS', 'GFG']
```

14] set dict comprehension :

```
l={10,20,3,4,10,20,7,3}

s1={x for x in l if x%2==0}
s2={x for x in l if x%2!=0}
print(s1)
print(s2)

l1=[1,3,4,2,5]
```

```
d1={x:x**3 for x in l1}
print(d1)

"""The f means Formatted string literals and it's new in Python 3.6 . A formatted
string literal
 or f-string is a string literal that is prefixed with 'f' or 'F' . These strings may contain
 replacement fields, which are expressions delimited by curly braces {}"""
d2={x:f"ID{x}" for x in range(5)}
print(d2)

#named indexes:
txt1 = "My name is {fname}, I'm {age}".format(fname = "ganesh", age = 20)
#numbered indexes:
txt2 = "My name is {0}, I'm {1}".format("ganesh",20)
#empty placeholders:
txt3 = "My name is {}, I'm {}".format("ganesh",20)
print(txt1)
print(txt2)
print(txt3)


l2=[101.103,102]
l3=["gfg","ide","course"]

d3={l2[i]:l3[i] for i in range(len(l2))}
print(d3)

"""To create a dictionary from two sequences, use the dict() and zip() method.
 In Python 3, the zip() method now returns a lazy iterator, which is now the most used
approach.
 The dict(zip(keys, values)) need the one-time global lookup each for dict and zip."""
d4=dict(zip(l2,l3))
print(d4)
```

OUTPUT :

```
{10, 4, 20}

{3, 7}

{1: 1, 3: 27, 4: 64, 2: 8, 5: 125}

{0: 'ID0', 1: 'ID1', 2: 'ID2', 3: 'ID3', 4: 'ID4'}

My name is ganesh, I'm 20

My name is ganesh, I'm 20
```

My name is ganesh, I'm 20

{101.103: 'gfg', 102: 'ide'}

{101.103: 'gfg', 102: 'ide'}

15] inverting dictionary

```
# items() method is used to return the list with all dictionary keys with values.
d1={101:"gfg",103:"practice",102:"ide"}
d2={v:k for(k,v) in d1.items()}

print(d2)
```

OUTPUT :

{'gfg': 101, 'practice': 103, 'ide': 102}

16] largest element in the list naïve solution :

```
def getMax(l):
    for x in l:
        for y in l:
            if y>x:
                break
        else:         #if loop is existed naturally,no longer element found
            return x
    return  None   # this excute when list is empty


"""The developer often wants a user to enter multiple values or inputs in one line.
In C++/C user can take multiple inputs in one line using scanf but in Python user can
 take multiple values or inputs in one line by two methods.

Using split() method
Using List comprehension
This function helps in getting multiple inputs from users. It breaks the given input
by the specified separator. If a separator is not provided then any white space is a
separator.
Generally, users use a split() method to split a Python string but one can use it in
taking multiple inputs."""



l = [int(x) for x in input().split()]
print(getMax(l))
print(l)




x, y = [int(x) for x in input("Enter two values: ").split()]
print("First Number is: ", x)
print("Second Number is: ", y)
```

OUTPUT :

```
2

2

[2]

Enter two values: 2 3
```

First Number is: 2

Second Number is: 3

17] Largest element in the list efficient :

```
def getMax(l):
    if not l:
        return  None
    else:

        res=l[0]
        for i in range(1,len(l)):
            if l[i]>res:
                res=l[i]
        return res


""""The developer often wants a user to enter multiple values or inputs in one line.
In C++/C user can take multiple inputs in one line using scanf but in Python user can
 take multiple values or inputs in one line by two methods.

Using split() method
Using List comprehension
This function helps in getting multiple inputs from users. It breaks the given input
by the specified separator. If a separator is not provided then any white space is a
separator.
Generally, users use a split() method to split a Python string but one can use it in
taking multiple inputs.""""



l = [int(x) for x in input().split()]
print(getMax(l))
print(l)

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

18] second largest element using two traversal :

```
"""None is used to define a null value. It is not the same as an empty string,
False, or a zero. It is a data type of the class NoneType object. Assigning a value of
 None to a variable is one way to reset it to its original, empty state"""
def getMax(l):

    if not l:
        return None

    else:
        res = l[0]
        for i in range(1,len(l)):
            if l[i]>res:
                res=l[i]
        return res


def getSecMax(l):
    if len(l)<=1:
        return None

    lar=getMax(l)
    slar=None

    for x in l:
        if x!=lar:
            if slar==None:
                slar=x
            else:
                slar=max(slar,x)
    return  slar

l=[int(x) for x in input().split()]
print(getSecMax(l))
```

OUTPUT :

```
10 20 50 3 60 80 40

60
```

19] Second largest element efficient :

```
def getSecMax(l):
    if len(l)<=1:
        return None
    lar=l[0]; slar=None

    for x in l[1:]:
        if x> lar:
            slar=lar
            lar=x
        elif x!=lar:
            if slar==None or slar<x:
                slar=x
    return  slar

l=[int(x) for x in input("Enter element in list: ").split()]
print(getSecMax(l))

"""
OUTPUT
Enter element in list: 10 20 6 50 88 4 6
50
"""
```

20] Check list is sorted or not :

```
def isSorted(l):
    i=1
    while i<len(l):
        if l[i]<l[i-1]:
            return  False
        i+=1

    return True

l=[int(x) for x in input("Enter Element for list: ").split(",")]

if isSorted(l):
    print("Yes, Given Element in sorted Order")
else:
    print("No, Given Element is not in sorted order")
```

OUTPUT :

Enter Element for list: 10,20,30,40

Yes, Given Element in sorted Order

21] Check element is sorted or not :

```
def isSorted(l):
    l2=sorted(l)  # build in function for sorting list

    if l==l2:
        return True
    else:
        return False

l=[int(x) for x in input("Enter Element for list: ").split()]

if isSorted(l):
    print("Yes, Given Element is in Sorted Order")
else:
    print("No, Given Element is not in Sorted Order")


"""
OUTPUT
1)Enter Element for list: 10 20 30 40 50 60
Yes, Given Element is in Sorted Order

2)Enter Element for list: 10 20 30 60 50 40
No, Given Element is not in Sorted Order
"""
```

22] find odd only using count :

```
def findodd(l):
    res=None
    for x in l:
def findodd(l):
    res=None
    for x in l:
        count=l.count(x)

        if count%2!=0:
            res=x
            break
    return res

l=[int(x) for x in input("Enter element for list: ").split()]
print(findodd(l))

"""
OUTPUT
Enter element for list: 10 20 30 30 20 10
None


OUTPUT :
Enter element for list: 10 20 30 20 10
30
"""
```

23] find xor using odd only:

```
def findOdd(l):
    res=0

    for x in l:
        res=res^x

    return res

l=[int(x) for x in input("Enter Elemenet for list: ").split(",")]
print(findOdd(l))

"""
OUTPUT
Enter Elemenet for list: 10,20,30,10,20
```

```
30
"""
```

24] reverse list in direct library method:

```
l=[10,20,30]
l.reverse()
print(l)

l=[10,20,30]
new_l=list(reversed(l))
print(new_l)

l=[10,20,30]
new_l=l[::-1]
print(new_l)
```

OUTPUT :

```
[30, 20, 10]

[30, 20, 10]

[30, 20, 10]
```

25] reverse a list own method:

```
def reverseList(l):
    s=0
    e=len(l)-1

    while s<e:
        l[s],l[e]=l[e],l[s]
        s+=1
        e-=1

l=[int(x) for x in input("Enter A number for a list").split(',')]
reverseList(l)
print(l)
```

```
"""
OUTPUT

Enter A number for a list 10,20,30,40,50,60,70
[70, 60, 50, 40, 30, 20, 10]
"""
```

26] reverse a list by one by using slice append pop

```

l=[10,20,30,40]
print(l)
l=l[1:]+l[0:1]
print(l)

l=[10,20,30,40]
print(l)
l.append(l.pop(0))
print(l)
```

OUTPUT :

[10, 20, 30, 40]

[20, 30, 40, 10]

[10, 20, 30, 40]

[20, 30, 40, 10]

27] left rotate list using own method:

```
def rotateByone(l):
    n=len(l)
    x=l[0]

    for i in range(1,n):
        l[i-1]=l[i]

    l[n-1]=x

l=[10,20,30,40]
rotateByone(l)
print(l)
```

OUTPUT :

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

28] left rotate by d places direct method using slicing

```
l=[10,20,30,40,50]
d=2
l=l[d:]+l[:d]
print(l)
```

OUTPUT :

```
[30, 40, 50, 10, 20]
```

29] left rotate by d places using collection deque:

```python
from collections import deque
l=[10,20,30,40,50]
d=2

dq=deque(l)
dq.rotate(-d)
l=list(dq)
print(l)
```

OUTPUT :

[30, 40, 50, 10, 20]

30] left rotate by d places own method :

```python
def leftRoatate(l,d):
    for i in range(0,d):
        l.append(l.pop(i))

l=[10,20,30,40,50]
d=3
print(l)
leftRoatate(l,d)
print(l)
```

OUTPUT :

[10, 20, 30, 40, 50]

[20, 40, 10, 30, 50]

31] left rotate d places own method 2:

```
def reverse(l,s,e):
    while s<e:
        l[s],l[e]=l[e],l[s]
        s+=1
        e-=1

def leftRoate(l,d):
    n=len(l)
    reverse(l,0,d-1)
    reverse(l,d,n-1)
    reverse(l,0,n-1)

l=[10,20,30,40,50,60]
d=3
print(l)
leftRoate(l,d)
print(l)
```

OUTPUT :

```
[10, 20, 30, 40, 50, 60]

[40, 50, 60, 10, 20, 30]
```