

Chapter 12  
List  
&  
Sorting Techniques

Computer Science  
Class XI ( As per CBSE Board)

Visit : [python.mykvs.in](http://python.mykvs.in) for regular updates

# LIST



It is a collections of items and each item has its own index value.

Index of first item is 0 and the last item is n-1. Here n is number of items in a list.

## Indexing of list

0	1	2	3	4	index
80	60	70	85	75	value
-5	-4	-3	-2	-1	Negative index

# LIST



## Creating a list

Lists are enclosed in square brackets [ ] and each item is separated by a comma.

## Initializing a list

Passing value in list while declaring list is initializing of a list

e.g.

```
list1 = ['English', 'Hindi', 1997, 2000]
```

```
list2 = [11, 22, 33, 44, 55 ]
```

```
list3 = ["a", "b", "c", "d"]
```

## Blank list creation

A list can be created without element

```
List4=[ ]
```



## Access Items From A List

List items can be accessed using its index position.

e.g.

```
list =[3,5,9]
```

```
print(list[0])
```

```
print(list[1])
```

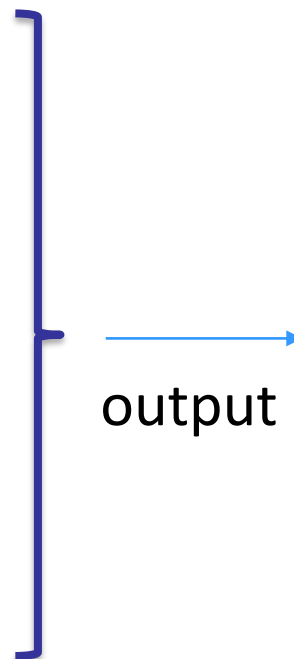
```
print(list[2])
```

```
print('Negative indexing')
```

```
print(list[-1])
```

```
print(list[-2])
```

```
print(list[-3])
```



3

5

9

Negative indexing

9

5

3



# LIST

## Iterating/Traversing Through A List

List elements can be accessed using looping statement.

e.g.

```
list = [3,5,9]
for i in range(0, len(list)):
    print(list[i])
```

Output

3

5

9



## Slicing of A List

List elements can be accessed in subparts.

e.g.

```
list = ['I', 'N', 'D', 'I', 'A']
```

```
print(list[0:3])
```

```
print(list[3:])
```

```
print(list[:])
```

Output

```
['I', 'N', 'D']
```

```
['I', 'A']
```

```
['I', 'N', 'D', 'I', 'A']
```

## Updating / Manipulating Lists

We can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator.

e.g.

```
list = ['English', 'Hindi', 1997, 2000]
print ("Value available at index 2 : ", list[2])
list[2:3] = 2001,2002 #list[2]=2001 for single item update
print ("New value available at index 2 : ", list[2])
print ("New value available at index 3 : ", list[3])
```

Output

```
('Value available at index 2 : ', 1997)
('New value available at index 2 : ', 2001)
('New value available at index 3 : ', 2002)
```



## Add Item to A List

append() method is used to add an Item to a List.

e.g.

```
list=[1,2]
```

```
print('list before append', list)
```

```
list.append(3)
```

```
print('list after append', list)
```

Output

```
('list before append', [1, 2])
```

```
('list after append', [1, 2, 3])
```

**NOTE :- extend() method can be used to add multiple item at a time in list.eg - list.extend([3,4])**



# LIST



## Add Item to A List

`append()` method is used to add an Item to a List.

e.g.

```
list=[1,2]
```

```
print('list before append', list)
```

```
list.append(3)
```

```
print('list after append', list)
```

Output

```
('list before append', [1, 2])
```

```
('list after append', [1, 2, 3])
```

**NOTE :-** `extend()` method can be used to add multiple item at a time in list.eg - `list.extend([3,4])`

# LIST



## Add Two Lists

e.g.

```
list = [1,2]
```

```
list2 = [3,4]
```

```
list3 = list + list2
```

```
print(list3)
```

OUTPUT

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



## Delete Item From A List

e.g.

```
list=[1,2,3]
print('list before delete', list)
del list [1]
print('list after delete', list)
```

## Output

```
('list before delete', [1, 2, 3])
('list after delete', [1, 3])
```

e.g.

```
del list[0:2] # delete first two items
del list # delete entire list
```



# LIST

Python Expression	Results	Description
<code>len([4, 2, 3])</code>	3	<b>Length</b>
<code>[4, 2, 3] + [1, 5, 6]</code>	<code>[4, 2, 3, 1, 5, 6]</code>	<b>Concatenation</b>
<code>['cs!'] * 4</code>	<code>['cs!', 'cs!', 'cs!', 'cs!']</code>	<b>Repetition</b>
<code>3 in [4, 2, 3]</code>	True	<b>Membership</b>
<code>for x in [4,2,3]: print (x,end = ' ')</code>	4 2 3	<b>Iteration</b>

## Important methods and functions of List

Function	Description
<code>list.append()</code>	Add an Item at end of a list
<code>list.extend()</code>	Add multiple Items at end of a list
<code>list.insert()</code>	insert an Item at a defined index
<code>list.remove()</code>	remove an Item from a list
<code>del list[index]</code>	Delete an Item from a list
<code>list.clear()</code>	empty all the list
<code>list.pop()</code>	Remove an Item at a defined index
<code>list.index()</code>	Return index of first matched item
<code>list.sort()</code>	Sort the items of a list in ascending or descending order
<code>list.reverse()</code>	Reverse the items of a list
<code>len(list)</code>	Return total length of the list.
<code>max(list)</code>	Return item with maximum value in the list.
<code>min(list)</code>	Return item with min value in the list.
<code>list(seq)</code>	Converts a tuple, string, set, dictionary into list.
<code>Count(element)</code>	Counts number of times an element/object in the list

# LIST

## Some Programs on List

\* find the largest/max number in a list #Using sort

```
a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=int(input("Enter element:"))
    a.append(b)
a.sort()
print("Largest element is:",a[n-1])
```

### #using function definition

```
def max_num_in_list( list ):
    max = list[ 0 ]
    for a in list:
        if a > max:
            max = a
    return max
print(max_num_in_list([1, 2, -8, 0]))
```

```
list1, list2 = [123, 'xyz', 'zara', 'abc'], [456, 700, 200]
print "Max value element : ", max(list1)
print "Max value element : ", max(list2)
Output
Max value element : zara
Max value element : 700
```



## Some Programs on List

\* find the mean of a list

```
def Average(lst): #finding mean of a number
    return sum(lst) / len(lst)
```

# Driver Code

```
lst = [15, 9, 55, 41, 35, 20, 62, 49]
average = Average(lst)
```

# Printing average of the list

```
print("Average of the list =", round(average, 2))
```

Output

Average of the list = 35.75

Note : The inbuilt function `mean()` can be used to calculate the mean( average ) of the list.e.g. `mean(list)`



## Some Programs on List

\* Linear Search

```
list_of_elements = [4, 2, 8, 9, 3, 7]
```

```
x = int(input("Enter number to search: "))
```

```
found = False
```

```
for i in range(len(list_of_elements)):
    if(list_of_elements[i] == x):
        found = True
        print("%d found at %dth position"%(x,i))
        break
if(found == False):
    print("%d is not in list"%x)
```





## Some Programs on List

\* Frequency of an element in list

```
import collections
```

```
my_list = [101,101,101,101,201,201,201,201]
```

```
print("Original List : ",my_list)
```

```
ctr = collections.Counter(my_list)
```

```
print("Frequency of the elements in the List : ",ctr)
```

## OUTPUT

```
Original List : [101, 101,101, 101, 201, 201, 201, 201]
```

```
Frequency of the elements in the List : Counter({101: 4, 201:4})
```

NOTE :SAME CAN BE DONE USING COUNT FUNCTION.E.G. lst.count(x)



# ~~SORTING~~ ~~DEPRECETED FROM~~ ~~SYLLABUS~~

Sorting is process of arranging items systematically ,according to a comparison operator applied on the elements.

## SORTING (ASCENDING ORDER)



## SORTING (DESCENDING ORDER)



# SORTING



There are various softing algorithms .Two of them are-

1. Bubble Sort

2. Insertion Sort

# SORTING



## 1. Bubble Sort-

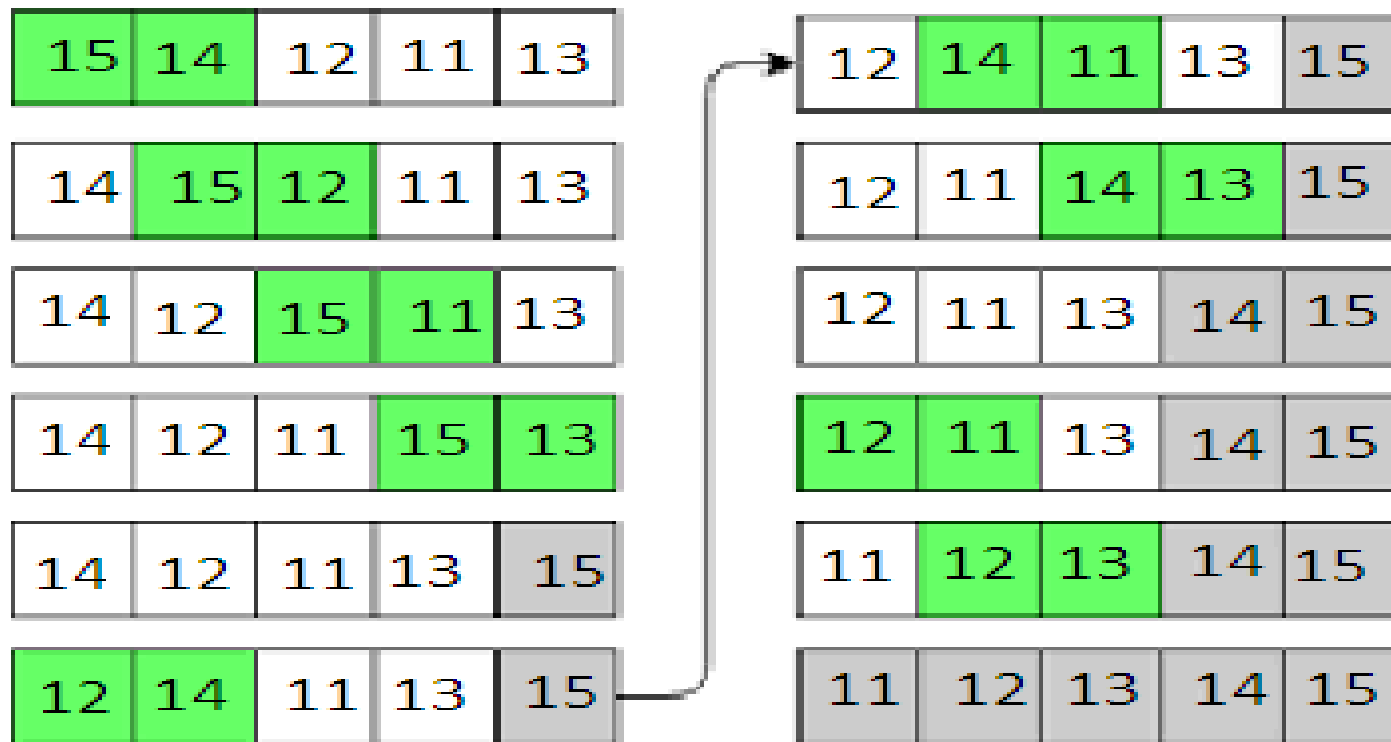
It is one of the simplest sorting algorithms. The two adjacent elements of a list are checked and swapped if they are in wrong order and this process is repeated until the whole list elements are sorted. The steps of performing a bubble sort are:

1. Compare the first and the second element of the list and swap them if they are in wrong order.
2. Compare the second and the third element of the list and swap them if they are in wrong order.
3. Proceed till the last element of the list in a similar fashion.
4. Repeat all of the above steps until the list is sorted.

# SORTING

## 1. Bubble Sort-

### BUBBLE SORT (ALGORITHM)



# SORTING

## 1. Bubble Sort-Python Program

```
a = [6, 19, 1, 15, 11, 12, 14]
```

```
#repeating loop len(a)(number of elements) number of times
```

```
for j in range(len(a)):
```

```
    #initially swapped is false
```

```
    swapped = False
```

```
    i = 0
```

```
    while i<len(a)-1:
```

```
        #comparing the adjacent elements
```

```
        if a[i]>a[i+1]:
```

```
            #swapping
```

```
            a[i],a[i+1] = a[i+1],a[i]
```

```
            #Changing the value of swapped
```

```
            swapped = True
```

```
            i = i+1
```

```
        #if swapped is false then the list is sorted
```

```
        #we can stop the loop
```

```
        if swapped == False:
```

```
            break
```

```
print (a)
```

## 1. Bubble Sort- No of Operation in sorting

In Bubble Sort,  $n-1$  comparisons will be done in the 1st pass,  $n-2$  in 2nd pass,  $n-3$  in 3rd pass and so on. So the total number of comparisons will be as follows-

$$(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$$
$$\text{Sum} = n(n-1)/2$$
$$\text{i.e } O(n^2)$$

Hence time complexity of Bubble Sort is  $O(n^2)$ .

The main advantage of Bubble Sort is the simplicity of the algorithm.

The space complexity for Bubble Sort is  $O(1)$ , because only a single additional memory space is required .

Also, the best case time complexity will be  $O(n)$ , only when the list is already sorted.

Following are the Time and Space complexity for the Bubble Sort algorithm.

Worst Case Time Complexity [ Big-O ]:  $O(n^2)$

Best Case Time Complexity [Big-omega]:  $O(n)$

Average Time Complexity [Big-theta]:  $O(n^2)$

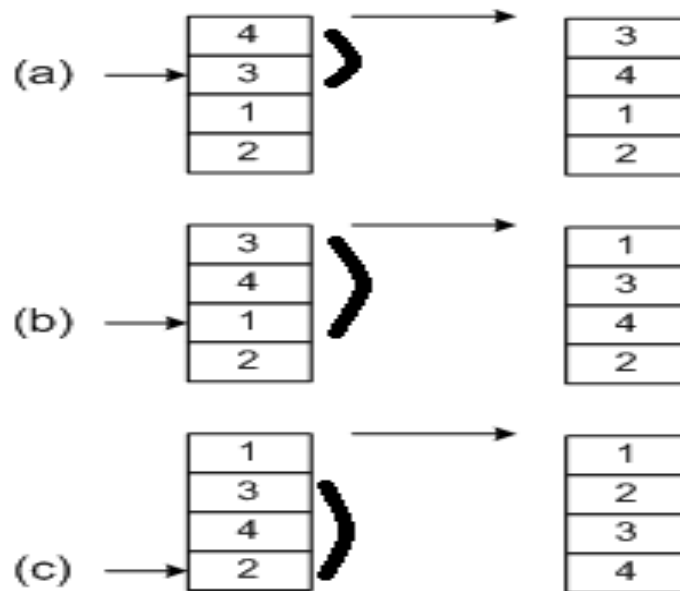
Space Complexity:  $O(1)$

# SORTING

## 2. Insertion Sort –

Insertion sort is a simple sorting algorithm .It is just similar the way we sort playing cards in our hands.

### INSERTION SORTING (ALGORITHM)





## 2. Insertion Sort – Python Program

```
list = [19, 12, 13, 15, 6]
for i in range(1, len(list)):
    key = list[i]
    # Move elements of list[0..i-1], that are
    # greater than key, to one position next
    # of their current position
    j = i-1
    while j >=0 and key < list[j] :
        list[j+1] = list[j]
        j -= 1
    list[j+1] = key
print ("Sorted listay is:")
for i in range(len(list)):
    print ("%d" %list[i])
```

## 2. Insertion Sort – No of Operation in sorting

In insertion sort ,to insert the last element at most  $n-1$  comparisons and  $n-1$  movements needed.

To insert the  $n-1$ st element  $n-2$  comparisons and  $n-2$  movements needed.

....

To insert the 2nd element 1 comparison and one movement needed.

Its sum up is given below:

$$2 * (1 + 2 + 3 + \dots + N - 1) = 2 * (N - 1) * N / 2 = (N-1) * N = \Theta (N^2)$$

If the greater part of the array is sorted, the complexity is almost  $O(N)$

The average complexity is proved to be  $= \Theta (N^2)$