

**Objective:**Implementation of Insertion Sort

# Insertion Sort

Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.

## Insertion Sort Algorithm:

```
insertionSort(array)

  mark first element as sorted

  for each unsorted element X
    'extract' the element X
    for j <- lastSortedIndex down to 0
      if current element j > X
        move sorted element to the right by 1
    break loop and insert X here

end insertionSort
```

## Code:

```
def Insertion_Sort(array):
    for i in range(1,len(array)):
        temp=array[i]
        temp2=i-1
        while temp2>=0 and array[temp2]>temp:
            array[temp2+1]=array[temp2]
```

```
temp2=temp2-1
```

```
array[temp2+1]=temp
```

```
return array
```

```
n= int(input("Enter no of test cases:"))
```

```
for i in range(0,n):
```

```
arr=[]
```

```
size=input("Enter the size:")
```

```
print ('Enter numbers in array: ')
```

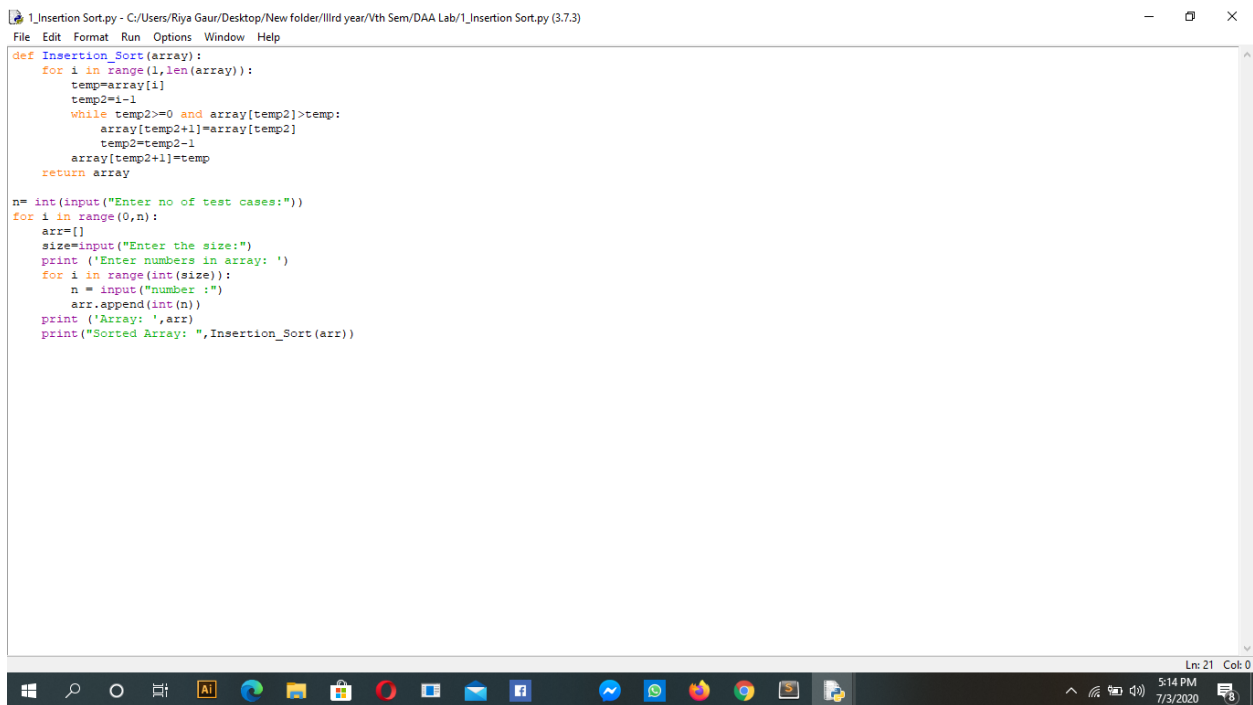
```
for i in range(int(size)):
```

```
n = input("number :")
```

```
arr.append(int(n))
```

```
print ('Array: ',arr)
```

```
print("Sorted Array: ",Insertion_Sort(arr))
```

A screenshot of a Python IDE window titled "1\_Insertion Sort.py - C:/Users/Riya Gaur/Desktop/New folder/IIIrd year/Vth Sem/DAA Lab/1\_Insertion Sort.py (3.7.3)". The window contains the following Python code:

```
def Insertion_Sort(array):  
    for i in range(1,len(array)):  
        temp=array[i]  
        temp2=i-1  
        while temp2>=0 and array[temp2]>temp:  
            array[temp2+1]=array[temp2]  
            temp2=temp2-1  
        array[temp2+1]=temp  
    return array  
  
n= int(input("Enter no of test cases:"))  
for i in range(0,n):  
    arr=[]  
    size=input("Enter the size:")  
    print ('Enter numbers in array: ')  
    for i in range(int(size)):  
        n = input("number :")  
        arr.append(int(n))  
    print ('Array: ',arr)  
    print("Sorted Array: ",Insertion_Sort(arr))
```

The IDE interface includes a menu bar (File, Edit, Format, Run, Options, Window, Help) and a status bar at the bottom showing "Ln: 21 Col: 0". The Windows taskbar is visible at the very bottom of the image.

## Output:

Enter no of test cases:1

Enter the size:5

Enter numbers in array:

number :40

number :50

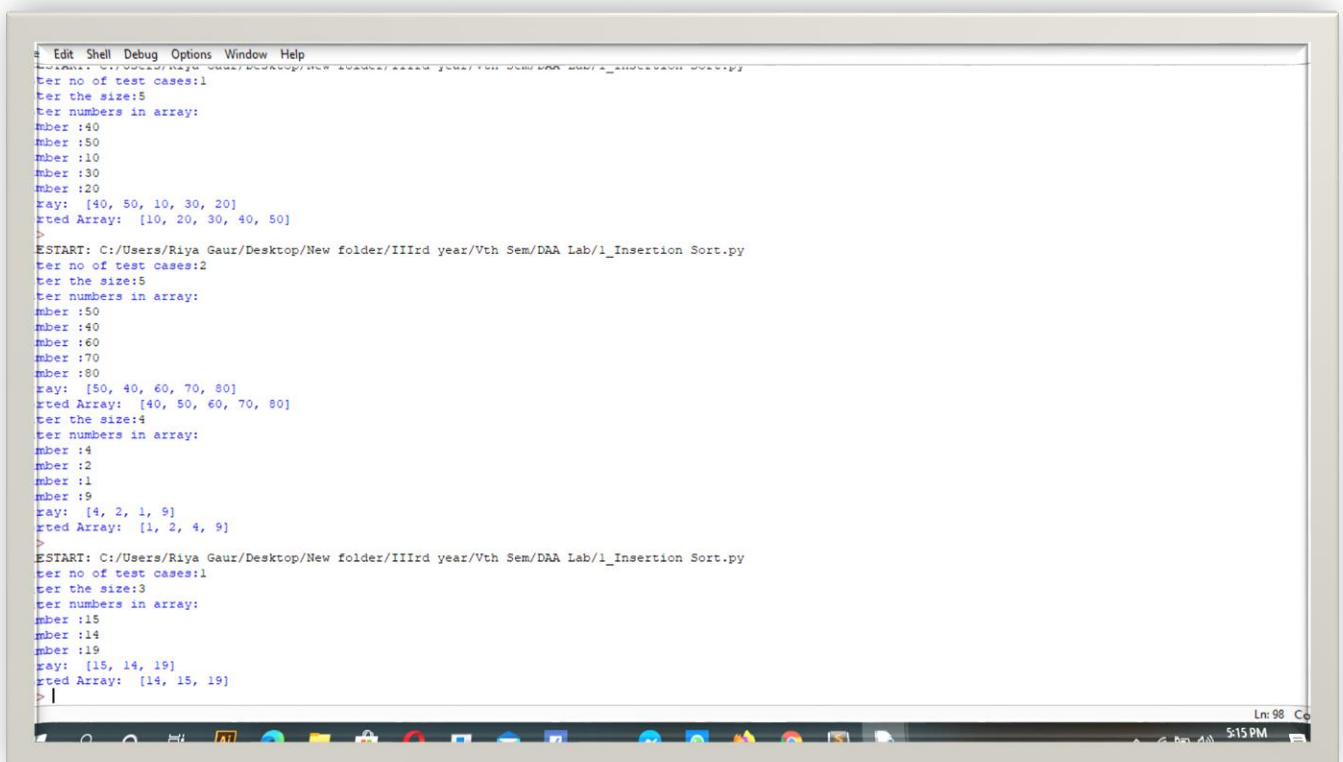
number :10

number :30

number :20

Array: [40, 50, 10, 30, 20]

Sorted Array: [10, 20, 30, 40, 50]



```
4 Edit Shell Debug Options Window Help
C:\Users\Riya Gaur\Desktop\New folder\IIIrd year\Vth Sem\DAA Lab\1_Insertion Sort.py
Enter no of test cases:1
Enter the size:5
Enter numbers in array:
number :40
number :50
number :10
number :30
number :20
Array: [40, 50, 10, 30, 20]
Sorted Array: [10, 20, 30, 40, 50]
>
ESTART: C:/Users/Riya Gaur/Desktop/New folder/IIIrd year/Vth Sem/DAA Lab/1_Insertion Sort.py
Enter no of test cases:2
Enter the size:5
Enter numbers in array:
number :50
number :40
number :60
number :70
number :80
Array: [50, 40, 60, 70, 80]
Sorted Array: [40, 50, 60, 70, 80]
Enter the size:4
Enter numbers in array:
number :4
number :2
number :1
number :9
Array: [4, 2, 1, 9]
Sorted Array: [1, 2, 4, 9]
>
ESTART: C:/Users/Riya Gaur/Desktop/New folder/IIIrd year/Vth Sem/DAA Lab/1_Insertion Sort.py
Enter no of test cases:1
Enter the size:3
Enter numbers in array:
number :15
number :14
number :19
Array: [15, 14, 19]
Sorted Array: [14, 15, 19]
>
|
```

## Time Complexities:

- **Worst Case Complexity:**  $O(n^2)$

Suppose, an array is in ascending order, and you want to sort it in descending order.

In this case, worst case complexity occurs.

Each element has to be compared with each of the other elements so, for every  $n$ th element,  $(n-1)$  number of comparisons are made.

Thus, the total number of comparisons =  $n*(n-1) \sim n^2$

- **Best Case Complexity:**  $O(n)$

When the array is already sorted, the outer loop runs for  $n$  number of times whereas the inner loop does not run at all. So, there are only  $n$  number of comparisons. Thus, complexity is linear.

- **Average Case Complexity:**  $O(n^2)$

It occurs when the elements of an array are in jumbled order (neither ascending nor descending).

## Insertion Sort Applications:

The insertion sort is used when:

- the array has a small number of elements
- there are only a few elements left to be sorted