Final Practical Worksheet

Student Name: Sahil Kaundal UID: 21BCS8197

Branch: CSE (Lateral Entry) Section/Group:20BCS-807\_B Semester: 4th Date of Performance: 21/05/2022

Subject Name: Programming in Python Lab Subject Code: 20CSP-259

1. Aim/Overview of the practical:

Write a Python program for insertion sort.

1. Task to be done/ Which logistics used:

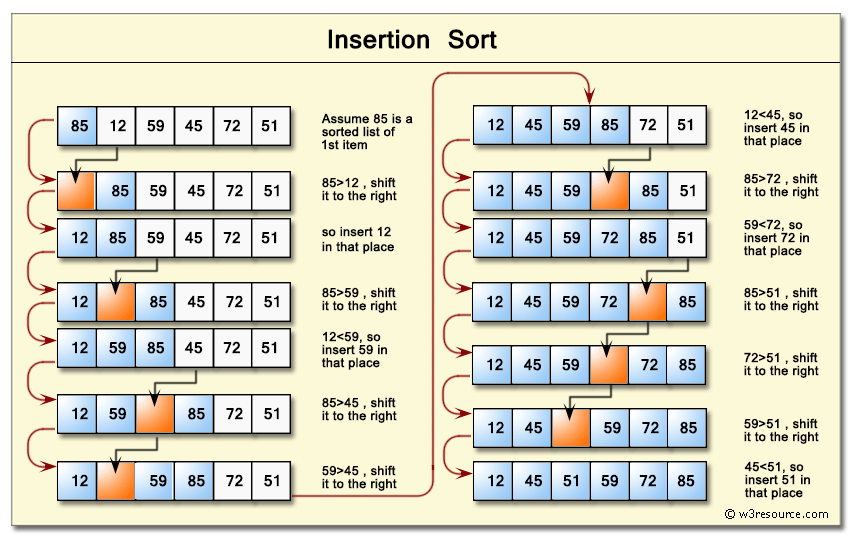


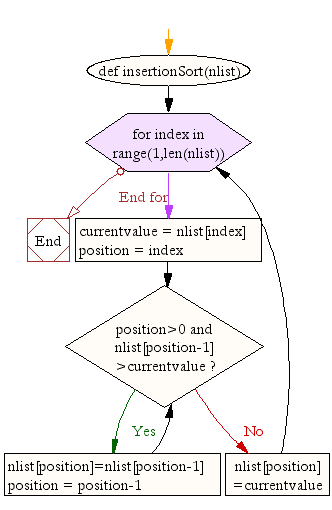
Write a Python program for insertion sort.

The Insertion sort is a straightforward and more efficient algorithm than the previous bubble sort algorithm. The insertion sort algorithm concept is based on the deck of the card where we sort the playing card according to a particular card. It has many advantages, but there are many efficient algorithms available in the data structure.

While the card-playing, we compare the hands of cards with each other. Most of the player likes to sort the card in the ascending order so they can quickly see which combinations they have at their disposal.

The insertion sort implementation is easy and simple because it's generally taught in the beginning programming lesson. It is an in-place and stable algorithm that is more beneficial for nearly-sorted or fewer elements.





Time Complexities

Worst Case Complexity: O(n2)

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 nth element, (n-1) number of comparisons are made.

Thus, the total number of comparisons = n\*(n-1) ~ n2

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(n2)

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

Space Complexity

Space complexity is O(1) because an extra variable key is used.

1. 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

1. Steps for experiment/practical/Code:

* Write a Python program for insertion sort.

def insertion\_sort(list1):

for i in range(1, len(list1)):

value = list1[i]

j = i - 1

while j >= 0 and value < list1[j]:

list1[j + 1] = list1[j]

j -= 1

list1[j + 1] = value

return list1

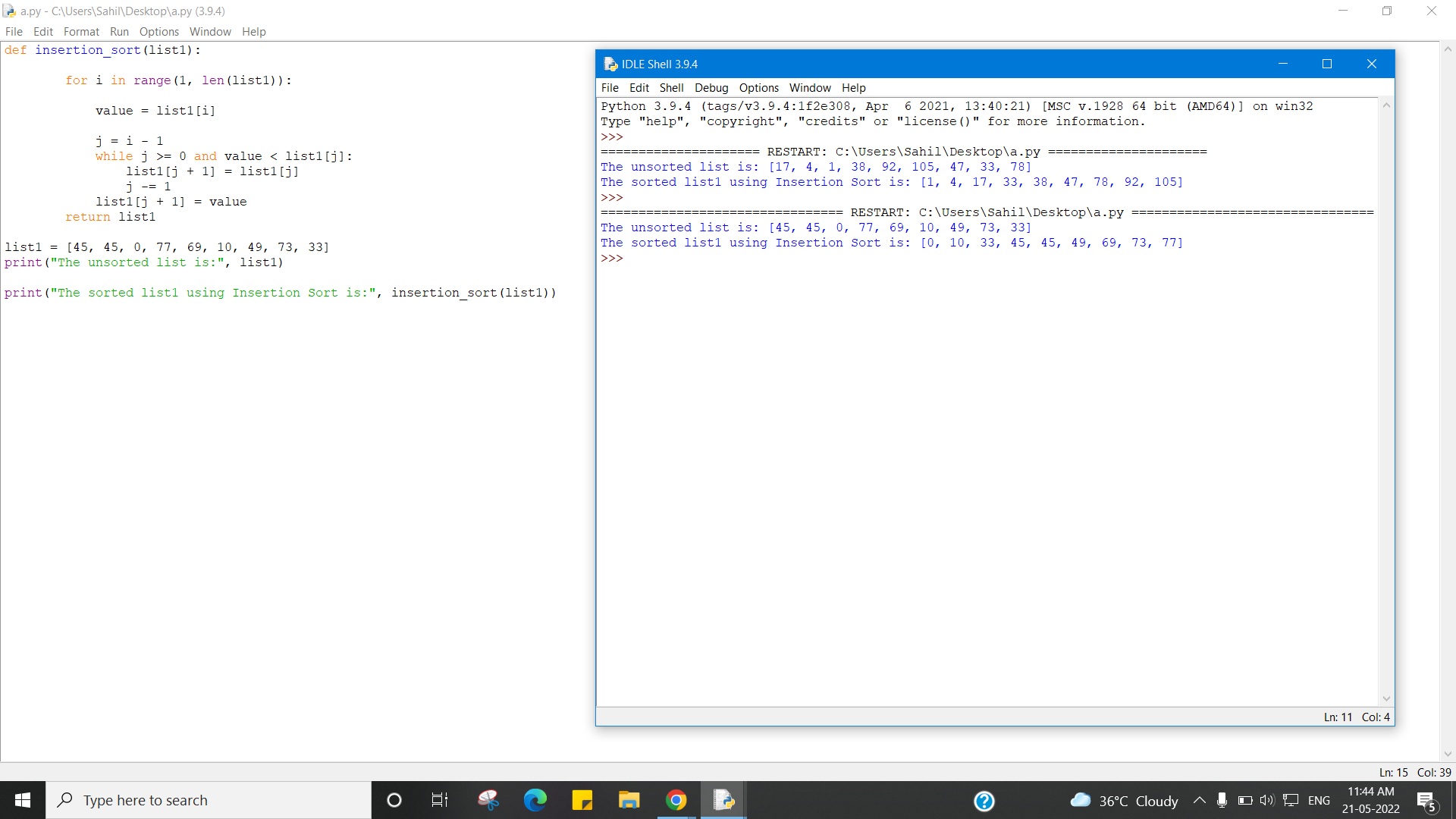
list1 = [45, 45, 0, 77, 69, 10, 49, 73, 33]

print("The unsorted list is:", list1)

print("The sorted list1 using Insertion Sort is:", insertion\_sort(list1))

1. Result/Output/Writing Summary:

* Write a Python program for insertion sort.



Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):



|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |