Objective: Implementation of Bubble Sort

Bubble Sort

Bubble sort is an algorithm that compares the adjacent elements and swaps their positions if they are not in the intended order. The order can be ascending or descending.

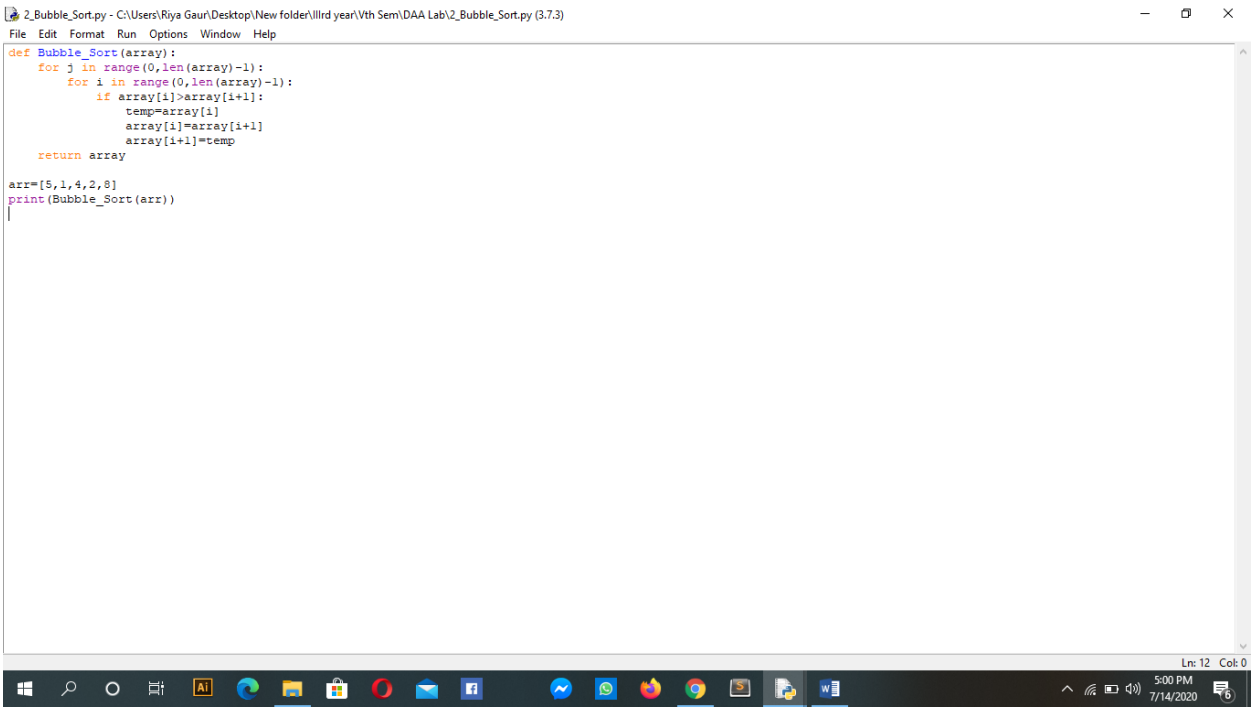
Bubble Sort Algorithm:

```
bubbleSort(array)

  for i <- 1 to indexOfLastUnsortedElement-1

    if leftElement > rightElement

      swap leftElement and rightElement

end bubbleSort
```

Code:

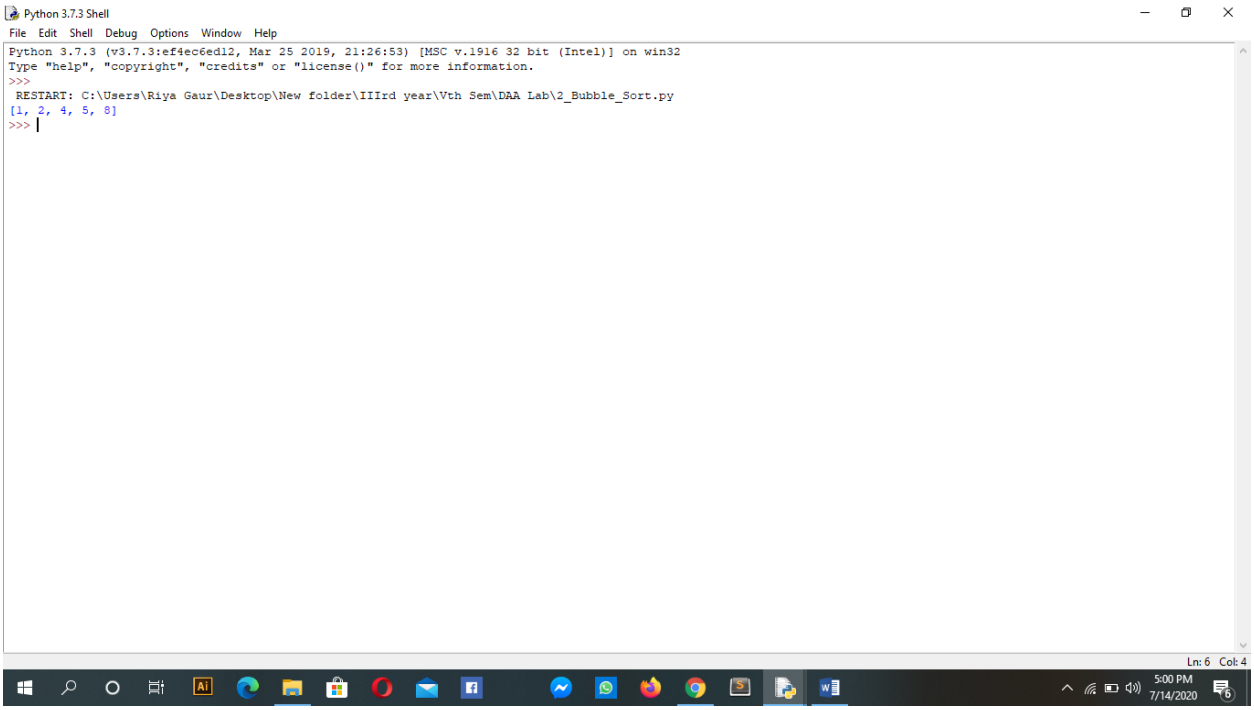
```
def Bubble_Sort(array):
   for j in range(0,len(array)-1):
      for i in range(0,len(array)-1):
         if array[i]>array[i+1]:
            temp=array[i]
            array[i]=array[i+1]
            array[i+1]=temp
   return array

arr=[5,1,4,2,8]
```

print(Bubble_Sort(arr))



Output:

[1, 2, 4, 5, 8]



Complexity:

Bubble Sort is one of the simplest sorting algorithms. Two loops are implemented in the algorithm.

Number of comparisons: `(n - 1) + (n - 2) + (n - 3) +.....+ 1 = n(n - 1) / 2` nearly equals to n^2

Also, we can analyze the complexity by simply observing the number of loops. There are 2 loops so the complexity is `n*n = n`^{2}

Time Complexities:

- **Worst Case Complexity:** `O(n`^{2}`)`

  If we want to sort in ascending order and the array is in descending order then, the worst case occurs.

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

  If the array is already sorted, then there is no need for sorting.

- **Average Case Complexity:** `O(n`^{2}`)`

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

Bubble Sort Applications

Bubble sort is used in the following cases where

- the complexity of the code does not matter.
- a short code is preferred.