Insertion Sort:

General Description: Insertion Sort, like bubble sort and selection sort splits an array into an unsorted and a sorted section. Like selection sort, in ascending order, the left side of the array becomes the sorted part while the right side is the unsorted part. With each pass of the array, we pick an element immediately right of the sorted section, which initially contains zero elements, and then shift it leftward until it is in its correct position. In other words, we shift the chosen element left until the element immediately towards its left is no longer larger than it. In actual practice, we actually shift the elements compared to this chosen element until the hole left is the correct position of the chosen element. By the end of the pass, the sorted section of the array has been incremented and the unsorted section has been decremented. Like with selection sort and bubble sort, there are two loops, one outer and one inner, so the time complexity is O(n^2). There are no auxiliary arrays required, so space complexity is O(1) and sorting is done in place.

```
def insertionSort(arr):

    # Traverse from index 1 to the end of the array
    for i in range(1, len(arr)):

        key = arr[i] # our comparison value for each pass

        # Shift elements of arr[0..i-1], that are
        # greater than key rightward
        j = i-1
        while j >= 0 and key < arr[j] :
                arr[j + 1] = arr[j]
                j -= 1
        arr[j + 1] = key # place key in the hole left at end of pass
```