Selection Sort

A : 50	25	38	44	99	16	11	21
1	2	3	4	5	6	7	8

i = 1

min_index = 7

swap(A[i],A[min_index]) => swap(50,11)

11	25	38	44	99	16	50	21
1	2	3	4	5	6	7	8

i = 2

min_index = 6

swap(A[i],A[min_index]) => swap(25,16)

11	16	38	44	99	25	50	21
1	2	3	4	5	6	7	8

i = 3

min_index = 8

swap(A[i],A[min_index]) => swap(38,21)

11	16	21	44	99	25	50	38
1	2	3	4	5	6	7	8

i = 4

min_index = 6

swap(A[i],A[min_index]) => swap(44,25)

11	16	21	25	99	44	50	38
1	2	3	4	5	6	7	8

i = 5

min_index = 8

swap(A[i],A[min_index]) => swap(99,38)

11	16	21	25	38	44	50	99
1	2	3	4	5	6	7	8

i = 6

min_index = 6

swap(A[i],A[min_index]) => swap(44,44)

11	16	21	25	38	44	50	99
1	2	3	4	5	6	7	8

i = 7

min_index = 7

swap(A[i],A[min_index]) => swap(50,50)

11	16	21	25	38	44	50	99

1 2 3 4 5 6 7 8

Few Points to Remember about which sorting algorithm to use :

When we want minimum number of swaps - Selection Sort

When we want to sort an already almost sorted array - Insertion Sort

When you actually want to perform sort in an unsorted array - QuickSort

Implementation :

```
import sys

A = [50,25,38,44,99,16,11,21]

for i in range(len(A)):

    # Find the minimum element from the

    # remaining unsorted array

    min_index = i

    for j in range(i+1, len(A)):

        if A[min_index] > A[j]:

            min_index = j

    # Swap the minimum element with the first element

    # in the given array

    A[i], A[min_index] = A[min_index], A[i]

print ("Sorted array below :")

for i in range(len(A)):
    print("%d" %A[i]),
```

Time Complexity Analysis : O(n^2)

Inplace Sorting Algorithm

Space Complexity : O(1)