Searching and Sorting in Python

- Linear or Sequential Search
- Binary Search

Linear Search

- Start from the left most element of list and one by one compare key with each element of the list
- If key matches with an element return True
- else return False

In [7]:

```
def LinearSearch(lst,key):
    for i in range(len(lst)):
        if key == lst[i]:
            print("Key is found at index:",i)
            break

    else:
        print("Key is not found")

lst=[45,56,1,10,89,75,46,82,11,9,444,89,1,555]
# size=int(input("Enter size of List:"))
# lst=[int(input()) for i in range(size)]
key=int(input("Enter the element you want to search:"))
print(lst)
LinearSearch(lst,key)
```

```
Enter the element you want to search:10
[45, 56, 1, 10, 89, 75, 46, 82, 11, 9, 444, 89, 1, 555]
Key is found at index: 3
```

In [1]:

```python
def LinearSearch(lst,key):
    lst2=[]
    b=False
    for i in range(len(lst)):
        if key == lst[i]:
            b=True
            lst2.append(i)
    if b == True:
        print("Key is found at index:")
        for i in lst2:
            print(i)
    else:
        print("Key is not found")

# lst=[45,56,1,10,89,75,46,82,11,9,444,10,89,1,555,10]
size=int(input("Enter size of List:"))
lst=[int(input("Enter element:")) for i in range(size)]
key=int(input("Enter the element you want to search:"))
print(lst)
LinearSearch(lst,key)
```

```
Enter size of List:5
Enter element:10
Enter element:15
Enter element:10
Enter element:16
Enter element:10
Enter the element you want to search:10
[10, 15, 10, 16, 10]
Key is found at index:
0
2
4
```

Binary Search

- Sort the list of elements
- Find out the middle element in the sorted list
- 1) Compare Key with the middle element
- 2) if key matches with middle element print message key is found and Stop the execution
- 3) else if key is greater than the middle element then key is searched in right sublist start with the step1
- 4) else if key is smaller than the middle element key searched in left sublist and start with step1
- 5) else print key is Not Found..

In [2]:

```
def BinarySearch(lst,key):
    low=0
    high=len(lst)-1
    flag=False
    while low<=high and not flag:
        mid=(low+high)//2
        if key==lst[mid]:
            flag=True
        elif key>lst[mid]:
            low=mid+1
        else:
            high=mid-1
    if flag == True:
        print("Key is Found")
    else:
        print("Key is Not Found")
lst=[45,63,89,55,78,44,1,10,9,5,555]
print(lst)
lst.sort()
print(lst)
print("After Sorting")
key=int(input("Enter the element you want search:"))
BinarySearch(lst,key)
```

```
[45, 63, 89, 55, 78, 44, 1, 10, 9, 5, 555]
After Sorting
[1, 5, 9, 10, 44, 45, 55, 63, 78, 89, 555]
Enter the element you want search:1000
Key is Not Found
```

Sorting Technics

- sort()---> built in list method
- sorted()---->Built in function

- Selection Sort
- Insertion Sort
- Bubble Sort
- Quick Sort
- Heap Sort

Selection Sort

- 1) Search the list and find out the minimum value
- 2) Swap the smallest number to 0th index
- 3) start searching from remaing sublist(1-remaining) and repeat step1 and step 2 untill all the elements are sorted

In [23]:

```
l=[6,90,15,4,78,0,3]
print(l)
print("After Sorting:")
for i in range(len(l)):
    m_val=min(l[i:])
    m_ind=l.index(m_val)
    l[i],l[m_ind]=l[m_ind],l[i] # l[0]=0,l[5]=6
print(l)
```

```
[6, 90, 15, 4, 78, 0, 3]
After Sorting:
[0, 3, 4, 6, 15, 78, 90]
```

In [24]:

```
l=[6,90,15,4,78,0,3]
print(l)
print("After Sorting:")
for i in range(len(l)):
    m_val=min(l[i:])
    m_ind=l.index(m_val)
    l[i],l[m_ind]=l[m_ind],l[i] # l[0]=0,l[5]=6
    print(l)
```

```
[6, 90, 15, 4, 78, 0, 3]
After Sorting:
[0, 90, 15, 4, 78, 6, 3]
[0, 3, 15, 4, 78, 6, 90]
[0, 3, 4, 15, 78, 6, 90]
[0, 3, 4, 6, 78, 15, 90]
[0, 3, 4, 6, 15, 78, 90]
[0, 3, 4, 6, 15, 78, 90]
[0, 3, 4, 6, 15, 78, 90]
```

In [25]:

```
l=[6,90,15,4,78,0,3]
print(l)
print("After Sorting:")
for i in range(len(l)-1):
    m_val=min(l[i:])
    m_ind=l.index(m_val)
    l[i],l[m_ind]=l[m_ind],l[i] # l[0]=0,l[5]=6
    print(l)
```

```
[6, 90, 15, 4, 78, 0, 3]
After Sorting:
[0, 90, 15, 4, 78, 6, 3]
[0, 3, 15, 4, 78, 6, 90]
[0, 3, 4, 15, 78, 6, 90]
[0, 3, 4, 6, 78, 15, 90]
[0, 3, 4, 6, 15, 78, 90]
[0, 3, 4, 6, 15, 78, 90]
```

In [26]:

```
l=[6,90,15,4,78,0,3]
print(l)
print("After Sorting:")
for i in range(len(l)):
    m_val=max(l[i:])
    m_ind=l.index(m_val)
    l[i],l[m_ind]=l[m_ind],l[i] # l[0]=0,l[5]=6
    print(l)
```

```
[6, 90, 15, 4, 78, 0, 3]
After Sorting:
[90, 6, 15, 4, 78, 0, 3]
[90, 78, 15, 4, 6, 0, 3]
[90, 78, 15, 4, 6, 0, 3]
[90, 78, 15, 6, 4, 0, 3]
[90, 78, 15, 6, 4, 0, 3]
[90, 78, 15, 6, 4, 3, 0]
[90, 78, 15, 6, 4, 3, 0]
```

without using built in function(min and max)

In [28]:

```
lst=[55,87,23,59,44,33,10,89]
print("Before Sorting")
print(lst)
print("After Sorting:")
for i in range(len(lst)):
    m_val=lst[i] # m_val=55
    for j in range(i+1,len(lst)):
        if lst[j]<m_val:
            m_val=lst[j] # m_val=23, m_val=10
    m_ind=lst.index(m_val)
    lst[i],lst[m_ind]=lst[m_ind],lst[i]
    print(lst)
```

```
Before Sorting
[55, 87, 23, 59, 44, 33, 10, 89]
After Sorting:
[10, 87, 23, 59, 44, 33, 55, 89]
[10, 23, 87, 59, 44, 33, 55, 89]
[10, 23, 33, 59, 44, 87, 55, 89]
[10, 23, 33, 44, 59, 87, 55, 89]
[10, 23, 33, 44, 55, 87, 59, 89]
[10, 23, 33, 44, 55, 59, 87, 89]
[10, 23, 33, 44, 55, 59, 87, 89]
[10, 23, 33, 44, 55, 59, 87, 89]
```

In [29]:

```
# Sort elements in Descending Order
lst=[55,87,23,59,44,33,10,89]
print("Before Sorting")
print(lst)
print("After Sorting:")
for i in range(len(lst)):
    m_val=lst[i] # m_val=55
    for j in range(i+1,len(lst)):
        if lst[j]>m_val:
            m_val=lst[j] # m_val=23, m_val=10
    m_ind=lst.index(m_val)
    lst[i],lst[m_ind]=lst[m_ind],lst[i]
    print(lst)
```

```
Before Sorting
[55, 87, 23, 59, 44, 33, 10, 89]
After Sorting:
[89, 87, 23, 59, 44, 33, 10, 55]
[89, 87, 23, 59, 44, 33, 10, 55]
[89, 87, 59, 23, 44, 33, 10, 55]
[89, 87, 59, 55, 44, 33, 10, 23]
[89, 87, 59, 55, 44, 33, 10, 23]
[89, 87, 59, 55, 44, 33, 10, 23]
[89, 87, 59, 55, 44, 33, 23, 10]
[89, 87, 59, 55, 44, 33, 23, 10]
```

Insertion Sort

-- It is a simple sorting algorithm that builds the final sorted list one item at a time

Algorithm

- 1) Consider the first element to be sorted and rest to be unsorted
- 2) Take the first element in the unsorted part(u1) and compare it with sorted part elements(s1)
- 3) If u1<s1 then insert u1 in the correct index, else leave it as it is.
- 4) Take next element in the unsorted part and compare with sorted elements.
- 5) Repeat 3 and 4 untill all the elements are sorted

eg:-

In [5]:

```
def InsertionSort(lst):
    for index in range(1,len(lst)):
        c_element=lst[index]
        pos=index
        while c_element<lst[pos-1] and pos>0:
            lst[pos]=lst[pos-1]
            pos-=1
        lst[pos]=c_element
        print(lst)


lst=[10,4,25,1,6]
print("Before Sorting :",lst)
InsertionSort(lst)
```

```
Before Sorting : [10, 4, 25, 1, 6]
[4, 10, 25, 1, 6]
[4, 10, 25, 1, 6]
[1, 4, 10, 25, 6]
[1, 4, 6, 10, 25]
```

Bubble Sort

--It is also refered as Sinking Sort. which sorts n number of elements in the list by comparing the each pair of adjacent items and swaps them if they are in Unsorted order.

Algorithm

- 1) Starting with the first element(index=0) and compare the current element with the next element of the list.
- 2) If the current element is greater than the next element of the list swap them.
- 3) If the current element is less than the next element move to the next element and Repeat step1.

In [14]:

```
lst1=[10,4,15,25,0]
print("Before Sorting:",lst1)
for j in range(len(lst1)-1):
    for i in range(1,len(lst1)):
        if lst1[i-1]>lst1[i]:
            lst1[i-1],lst1[i]=lst1[i],lst1[i-1]
            print(lst1)
        else:
            print(lst1)
    print()

print("After Sorting:",lst1)
```

```
Before Sorting: [10, 4, 15, 25, 0]
[4, 10, 15, 25, 0]
[4, 10, 15, 25, 0]
[4, 10, 15, 25, 0]
[4, 10, 15, 0, 25]

[4, 10, 15, 0, 25]
[4, 10, 15, 0, 25]
[4, 10, 0, 15, 25]
[4, 10, 0, 15, 25]

[4, 10, 0, 15, 25]
[4, 0, 10, 15, 25]
[4, 0, 10, 15, 25]
[4, 0, 10, 15, 25]

[0, 4, 10, 15, 25]
[0, 4, 10, 15, 25]
[0, 4, 10, 15, 25]
[0, 4, 10, 15, 25]

After Sorting: [0, 4, 10, 15, 25]
```

In [ ]:

```
```