29/8/25

Task : 5(A) → Implement various searching and Sorting operations in python programming.

a) Student Record search

Aim: To write a python program that stores roll number in a list and binary search to check whether a roll number exists.

Algorithm :-

```
1. Start
2. Store roll number in a list
3. ~~Linsearch~~ Linear search :
    • Traverse the list sequentially.
    • If the target is found, return Success.
4. Binary search :
    • Set low = 0, high = n-1.
    • Repeat until low <= high :
      → find mid = (low + high) / 2
      → If element at mid equals the target → found
5. Display appropriate message
6. Stop
```

program :-

```
id_number = (105, 101, 108, 107, 120)
 print ("roll number:", id_number)
 target = 108
 print ("\n searching (Linear search) for roll number:", target)
 found = false
  for i in range (len (roll_number)) :
    if roll_number [i] == target :
     print ("roll number found at position :", i)
     found = True
     break

   if not found :
        Print (" roll number not found.")
  roll_number. sort ()
  printf ("\n sorted roll number :", roll_number)
```

Output:

```
roll number : [105, 101, 108, 107, 120]
Searching (Linear search) for roll number : 107
roll_number found at position : 3
Sorted roll number : [101, 105, 107, 108, 120]
Searching (Binary search) for roll number : 120
roll_number found at position : 4
```

```
target = 120
print ("Searching (Binary Search) for roll number:", target)
low=0
high= len (roll_number)-1
found = False
While low<= high :
    mid = (Low + high)
    if roll_number [mid] = = target :
        print ("roll_number found at position:", mid)
        found = True
        break
    elif roll_number [mid]< target :
        low = mid + 1
    else :
        high = mid - 1
    if not found :
        print ("roll number not found")
```

<u>Result</u> : Thus, a python program to perform linear search and binary search on student roll number was successfully implemented.

Task 5(b) : product price Sorting 29/8/25

Aim : To write a python program to sort product prices using bubble sort, selection sort and insertion sort.

Algorithm :-

1. Start
2. Store product price in a list.
3. Implement sorting technique :
   - Bubble sort (Ascending)
   - Selection sort (Descending)
   - Insertion sort (ascending)
4. Display sorted list after each sorting method.
5. Stop

program :-

```
 # product price sorting
 # Bubble sort (Ascending)
 def bubble_sort(arr):
        n = len(arr)
    for i in range(n):
        max_idx = i
     for j in range (i+1, n):
     if arr[j] > arr[max_idx]:
          Max_idx = j
     arr[i], arr[Max_idx] = arr[Max_idx],
                                  arr[i]
     return arr

# insertion sort (Ascending)
 def insertion_sort(arr):
 for i in range (1, len(arr)):
     Key = arr[i]
       j = i-1
 While j>=0 and arr[j]>Key:
     arr[j+1] = arr[j]
```

<u>output</u> :-

```
original prices = [250, 120, 300, 90, 150 ]
Bubble sort (Ascending ): [90, 120, 150, 250, 300]
Selection sort (Descending ): [300, 250, 150, 120, 90]
Insertion sort (Ascending ): [90, 120, 150, 250, 300]
```

```
    j -= 1
  arr[j+1] = key
  return arr
# Main program
  prices = [250, 120, 300, 90, 150]
print ("original prices:", prices)
print ("Bubble sort (Ascending):", bubble_sort
                                    (prices. copy()))
print ("Selection sort (Descending):", selection_sort
                                    (prices. copy()))
print ("Insertion sort (Ascending):", insertion_sort
                                    (prices. copy()))
```

VELTECH	
EX No.	
PERFORMANCE (5)	[illegible]
RESULT AND ANALYSIS (3)	[illegible]
VIVA VOCE (2)	[illegible]
RECORD (?)	
DATE	

Result: Thus, a python program was successfully written to sort product using bubble sort, selection sort and insertion sort.