20/8/25

5. Implement various searching and sorting operations in python programming

a. Library book search.

Aim:- To search for a book in a library catalog using Linear search for unsorted lists and binary search for sorted lists.

Algorithm

1. **Linear search:**
   - Check each book in the list one by one.
   - If found, return its position.
   - If not found after checking all books, return -1.
2. **Binary search:**
   - Requires the list to be sorted.
   - Compare the target with middle element
   - If equal, return the position
   - If target is smaller, search the left half
   - If target is larger, search the righ half
   - Repeat until found or search space exhausted.

output

```
Library Books: ['Python', 'Java', 'C++', 'Javascript', 'HTML']
Lists is sorted : false
Enter book to search: C++
Book found at position 3 using Linear search.
```

Result:-

The program successfully implemented both search algorithms. It detected that the list was not sorted and used Linar search to find "C++" at position 3.

10/9

Program

```
# Library Book search
def linear_search (books, target):
    for i in range (len (books)):
        if books [i] == target:
            return i
    return -1
def binary_search (books, target):
    low, high = 0, len (books) -1
    while low <= high:
        mid = (low + high) // 2
        if books [mid] == target:
            return mid
        elif books [mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1
books. = ["Python", "Java", "C++", "Javascript", "HTML"]
Print ("Library Books:", books)

is_sorted = books == sorted (books)
Print ("List is sorted:", is_sorted)
target = input ("Enter book to search:")
if is_sorted:
    result = binary_search (books, target)
    method = "Binary search"
else:
    result = linear_search (books, target)
    method = "Linear search"
if result != -1:
    Print (f"Book found at position {result+1} using {method}")
else:
    Print ("Book not found").
```

Program

```
def bubble_sort_asc(grades):
    n = len(grades)
    for i in range(n):
        for j in range(0, n-i-1):
            if grades[j] > grades[j+1]:
                grades[j], grades[j+1] = grades[
    return grades.
def selection_sort_desc(grades):
    n = len(grades)
    for i in range(n):
        max_idx = i
        for j in range(i+1, n):
            if grades[j] > grades[MAX_idx]:
                max_idx = j
        grades[i], grades[max_idx] = grades[max_i
grades = [85, 92, 78, 90, 65, 88, 72]
print("original Grades:", grades)
asc_sorted = bubble_sort_asc(grades.copy())
desc_sorted = selection_sort_desc(grades.copy())

Print("Ascending order (Bubble sort):", asc_sorted)
Print("Desending order (selection sort):", desc_sorted
Print("Top 3 scores:", desc_sorted[:3])
```

output

```
Original Grades: [85, 92, 78, 90, 65, 88, 72]
Ascending order (Bubble sort): [65, 72, 78, 85, 88, 90, 92]
Descending order (selection sort): [92, 90, 88, 85, 78, 72, 65]
Top 3 scores: [92, 90, 88].
```

6. Student Grade organizer

Aim: To sort Student grades using different algorithms and display the top 3 scores.

Algorithm:-

1. Bubble sort (Ascending):
   - Compare adjacent elements & swap if in wrong order
   - Repeat until no more swaps are needed.
2. Selection sort (Descending):
   - find the max element and swap it first position
   - Repeat for remaining elements.
3. Top 3 scores:-
   - After sorting in descending order, the first three elements are the top scores.

VEL TECH - CSE	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	10
TOTAL (20)	
SIGN WITH DATE	

Result: Hence the Student grade organizer in both ascending & descending order using different algorithms are executed successfully.