Task 5 : Implement Various Searching and Sorting Operations in python programming

Dt: 03/09/25

Aim :-

To implement Various Searching and Sorting operations in python Programming

5.1 A Company stores employee records in a list of dictionaries, where each dictionary contains id, name and department. write a function find-employee-by-id that takes this list and a target employee ID as arguments and returns the dictionary of the employee with the matching ID, or None if no such employee is found.

Algorithm :-

1. Input Definition
2. Define the function find_employee_by_id that takes two parameters
   a. A list of dictionaries (employees), where each dictionary represents an employee records with keys, id, name and department
   b. An integer (target_id) representing the employee ID to be Searched.
3. Iterate through the list:
   use a for loop to iterate through each dictionary matches the target_id
4. Check for Matching ID :
   within the loop, check if the id field of the current dictionary matches the target_id
5. Return Matching Record
6. Handle No Match
   If the loop completes without finding a match, return None.

Program :-

```
def find_employee_by_id(employees,target_id):
    for employee in employees:
        if employee['id'] == target_id:
            return employee
    return None
# Test the function
employees = [
    {'id': 1,'name':'Alice','department': 'HR'},
    {'id': 2,'name': 'Bob','department': 'Engineering'},
    {'id': 3,'name': 'Charlie','department':'Sales'},
]
Print(find_employee_by_id(employees,2)) # output: {'id': 2,'name':'Bob',
'department': 'Engineering'}
```

Output:

{'id': 2, 'name': 'bob', 'department': 'engineering'}

5.2 You are developing a grade managment system for a school. The System maintains a list of Student records, where each record is represented as dictionary containing a student's name and score. the school needs to genelate a report that displays students scores in ascending order. your task is to implement a feature that sorts the student records by their scores using the Bubble sort algorithm.

Algorithm :-

1. Initialization :
   - Get the length of the Students list and store it in n.
2. Outer loop :
   - Iterate from i=0 to n-1 (inclusive). This loop represents the no of Passes through the list.
3. Track Swaps :
   - Initialize a boolean variable swapped to false. this variable will track if any swaps are made in the current pass.
4. Inner Loop :
   - Iterate from j=0 to n-1-2 (inclusive). This loop compares adjacent elements in the list and performs swaps if necessary
5. Compare and swap :
   - for each pair of adjacent elements (i.e, students[j] and students [j+1]):
   - Compare their score values
   - If students[j]['score'] > students[j+1]['scoree'], swap the two elements
   - Set swapped to true to indicate that a swap was made.
6. Early Termination :
   - After each pass of the inner loop, check if swapped is false. If no swaps were made during the pass, the list is already sorted, and you can break out of the outer loop early.
7. Completion :
   - The function modifies the students list in place, sorting it by score.

Output:

Before Sorting

{'name' : 'Alice', 'Score' : 88}

{'name' : 'bob', 'score' : 95}

{'name' : 'Charlie', 'score' : 75}

{'name' : 'Diana', 'score' : 85}

After Sorting

[{'name' : 'alice', 'score' : 88}

{'name' : 'bob', 'score' : 95}

{'name' : 'charlie', 'score' : 75}

{'name' : 'diana', 'score' : 85}]

Program :-

```
def bubble_sort_scores(students):
    n = len(students)
    for i in range(n):
        # Track if any swap is made in this pass
        swapped = False
        for j in range(0, n-i-1):
            if students[j]['score'] > students[j+1]['score']:
                # Swap if the score of the current student is greater than the next
                students[j], students[j+1] = students[j+1], students[j]
                swapped = True
        # If no two elements were swapped, the list is already sorted.
        if not swapped:
            break
# Example usage
students = [
    {'name': 'Alice', 'score': 88},
    {'name': 'Bob', 'score': 95},
    {'name': 'Charlie', 'score': 75},
    {'name': 'Diana', 'score': 85},
]
print("Before sorting:")
for student in students:
    print(student)
```

EX NO.	
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (15)	
SIGN WITH DATE	15

Result:

Thus, the program for various searching and sorting operations is executed and verified successfully