Task-5:

Implement various Searching and Sorting Operations in python programming.

Aim:-

To Implement various Searching and Sorting Operations in python programming.

5.1 A Company stories 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.

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 record with keys id, name, and department

b) An Integer (target_id) representing the employee ID to be searched.

3) Iterate Through the list

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:
If a match is found. return the current dictionary.

6) Handle No Match.
If the loop completes without finding a match return None.

Program :- 5.1

```
def find employee_by_id (employees, target_id):
    for employee in employees:
        if employee : ('id') == target_id:
            return employee.
    return None
# Test the function
employee = (
{'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'}

O/P

5.2 you are developing a grade management 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 generate a report that displays students scores.

Algorithm:-

1) Initialization:
   -> Get the length of the students list and store it in n

2) Outer loop:-
   -> Iteration a boolean variable i=0 to n-1. (inclusive): This loop represents the number 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-i-2 (inclusive) This loop compares adjacent elements in the list and performs swap 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)('score'): 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

Program :- 5.2

```
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], studen[j+1] = student[j+1], studen[0]
    swapped = True
    # if no two elements were swapped the list is already
sorted
    if not swapped:
        break
```

Output:-

Before sorting:-

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

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

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

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

After sorting:-

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

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

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

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

```
# 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)
bubble_sort_scores(students)
print("\nAfter sorting:")
for student in students:
  print(student)
```

Result :-

Thus, the program for various Searching and Sorting Operations is executed and Verified Successfully