Task-5 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 of employee is found.

Algorithm:

1 Input Defination:

2 Define the function find_employee_by_id that takes the 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 Though the list:
Use a for loop to iterate though each dictionary in the employee list.

4. Check for Matching ID:
Within the loop, check if the id field of the current dictionary mactches 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 the None.

Output:

{'id': 2, 'name': 'Bob', 'department': 'Engineering'}

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':'Bob','department':'HR'},{'id':2,'name':'Bob','department':'Engineering'},{'id':3,'name':'Charlie','department':'Sales'}]
Print(find_employee_by_id(employees, 2))
```

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

Algorithm:

1. Initialization: Get length of the students list and store it in n.
2. Outer loop: Iterate from i=0 to n-1. 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.

Program

```
def bubble_sort_scores(students):
    n=len(students)
    for i in range(n):
        swapped=False
        for j in range(0,n-i-1):
            if students[j]['score']>students[j+1]['score']:
                students[j],students[j+1]=students[j+1],students[j]
                swapped=True
        if not swapped:
            break
students=[{'name':'Alice','score':88},{'name':'Bob','score':95},
          {'name':'Charlie','score':75},{'name':'Diana','score':85}]
```

4. Inner loop: Iterate from j=0 to n-i-2. This loop compares adjacent elements in the list and performs swaps if it is necessary.

5. Compare and swap:

- ~~Swap~~ for each pair of adjacent elements (i.e., student[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: Check if swapped is False

7. Completion:

- The function modifies the students list in place, sorting it by score.

Program

```
def bubble_sort_scores(students):
n=len(students)
for i in range(n):
swapped =False
for j in range(0,n-i-1):
if students[j]['score']
students[j+1]['score']:
students[j], students[j+1] = students[j+1], students[j]
swapped = True
if not swapped:
break
students = [{'name':'Alice','score':88},{'name':'Bob','score':95},
```

output:

Before Sorting:

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

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

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

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

After Sorting:

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

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

{'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)
~~Print(student)~~
bubble_sort_scores(students)
Print("\nAfter Sorting:")
for student in students:
~~Print(student)~~
    Print(student)
```

VEL TECH	
EX No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	28/8/25

Result:

Thus, the program for various Searching and Sorting Operations is excuted and verified successfully.