3/9/25 Task 5 :- implement various Searching and Sor-
-ting operations in python poograming.

Aim:- To implement various searching and Sorti-
-ng 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 list and a target employee ID as arguments and returns the dicti-
-onary 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 record 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 in the employees list.

4. Check For matching ID:

with in the loop, check if the id field of the Current dictionary matches the target - id.

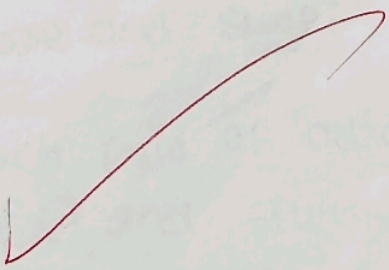
5. Return matching Record:

6. Handle No match:
if the loop completes without finding a match, return None.

Program 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
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 management system for a school. The system maintains a list of student records, where each record is represented as a dictionary containing a student's 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 current pass.

4. inner loop:-

iterate from j=0 to n-i-2. This loop compares adjacent elements in the list and performs swaps if necessary.

5. Compare and swap:-

- For each pair of adjacent elements and students[j] and students[j+1]:
- Compare their score values.
- if students[j]['score'] < student[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 outer loop early.

7. Completion:-

The function modifies the studently 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'] > student[j+1][score]
                # Swap if score of the current student is greater
                than the next.
                student[j], students[j+1] = students[j+1],
                                            students[j]
                swapped = True.
        if not swapped:
            break
student = [
    {'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)
bubble_sort_scores(students)
print("\nAfter sorting:")
for student in students:
    print(student)
```

VELTECH	
EX No.	
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	4
TOTAL (20)	19
SIGN WITH DATE	20

Result :-

Thus, the Program for various Searching and Sorting operation is executed and verified Successfully.