TASK-5.1

Implement various searching and sorting operations in Python Programming

Aim:

To Implement various searching and sorting operations in Python Programming.

Algorithm:

1. Initial Definition
2. Define the function find-employee-by-Id takes two parameters
   a. A list of dictionaries (employee) where each dictionary represents and employee record with keys id, name, and department
   b. An integer (target-Id) representing the employee ID to to searching
3. Iterate through the list:
   use a for loop to iterate through each dictionary in the employee list.
4. check for matching ID.
   which in the loop, check if the id field of current dictionary matches the target-id.
5. Return matching Record:
   If a match is found, return the current dictionary
6. If Handle No match.
   If the loop completes without finding a match return

output

{'id': 2: name: Bob': 'department': 'engineering'}

output:

Enter the first number

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': 'Alica', 'department': 'HR'},
    {'id': 2, 'name': 'Bob', 'department': 'Engineering'},
    {'id': 3, 'name': 'charlie', 'department': 'sales'},
]
print(find_employee_by_id(employees,
# output {'id': 2 'name' Bob ', 'department':
'Engineering'}
```

Result: Thus to implement various searching and sorting operation in Python Programming.

TASK - 5.2

Aim: To develop a Python Program that stores student records by score in ascending order 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 number of passes
3. Track swaps:
   - Initilize a boolean variable swapped to false this variable will track if any swap are made current pass
4. Inner loop
   - Iterate from j=0 to n-i-2 (inclusive). this loop compares adject elements in the list and perform swaps if necessary
5. compare and swap
   - for each pair of adjacent element (i.e student[j] and students[j+1]);

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}

- 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 swap occur mode during the pass the list is already sorted and you can break of the outer loop early.

7. Completion
- The function modifies the students list in place sorting by score

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 that the next
```

```
student [i] students [j+1] = students [j] , students [i]
    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)
  bubble_sort_scores (students)
  for student in student:
      Print (student).
```

VEL TECH	
EX No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	
SIGN WITH DATE	15

Result: Thus the Program for various searching and sorting Operating is executed and verify successfully