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 stores employee records in a list of dictionaries, where each dictionary contain 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 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:-
   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.

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': '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'}
```

✓

5.2) you are developing a grade Management system for a school. The system maintains a list of stude-nt's ~~name and score~~ The records, where each record 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 imple-ment 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 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 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]['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.

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}
```

```
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 student[j]['score']>student[j+1]['score'];
          #swap if the score of current student is gre
          ter 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)
bubble_sort_scores(students)
Print("\n After sorting:")
for Student in students:
   Print(student)
```
✓

```
e, 65]
]
Print ("Before sorting:")
for student in students:
    Print(Student)
bubble_sort_scores(stu
Print ("\n After sorting:")
for student in students:
    Print(Student)
```

VELTECH	
EX No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	17/9.

Result:- Thus, the Program for various searchin and sorting operations is executed and verifie successfully