Task 5: implement various searching and sorting operations in python progvaming.

Aim: To implement various searching and sorting operations in python programming.

5.1 A company stores employee record 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 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:
   with in the loop, check if the id field of the current dictionary matches the target-id.

output

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

5. Return matching Record:

if a match is found, return the current dicitonary.

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 empbyees:
if employee ['id] == target_id:
 return employee
return None.

# Test the function
 employess= [
{'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 ~~syst~~ student records, where each record is represented as a dictionary containing a student's scores in ascending order. Your task is to implements 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 lis

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 y=0 to n-i-2 this loop compare adjacent elements in the list and performs swaps if necessary.

5. compare and swap:-

- For each pair of adyacent elements and students[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'] > student [j+1] [score], swap them 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, soorting it by score.

Program 5.2:-

```
let bubble - sort - scores (students);
    n = len (students)
    for i in ranse (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},
{'name': 'charlie', 'score': 75}
{'name': 'Diana', 'score': 85}
]
print ("Before Sorting":)
for student students
print (student)
bubble_sort_scores (students)
print ("\n After Sorting":)
for student in students:
Point (student)
```

VEL TECH	
Ex. No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	18
SIGN WITH DATE	3/9

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

Result:-

Thus, the program for various seaching and sorting operation is executed and verified successfully.