Task 5:- Implement various searching and sorting operations in python programing.

Aim:- To implement various searching and operations in python programing.

A company stores employee records in a list of dictionaries 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 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 loop to iterate through each dictionary in the employee list.
4. Check for matching record or ID within the loop, check if the id field of the current dictionary matches the target-id
5. Return matching record. If a matching is found, return the current dictionary
6. Handle no match.
   If the loop completely without finding an match return none.

output :-

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

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))
```

5.2 You are developing a grade management system. The system must store a list of student records. Where each record is represented as a dictionary containing a student's name and score.

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 other loop early.

7. ~~Comp~~ completion.

* The function modifies the Students 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 // made in this pass swapped
    : False
for i in range (0, n-i-1):
if students [j] ['score'] > students [j+1] ['score']
# swap if the score of the current student is
greater 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 stored.
If not swapped.
```

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}

break

Example page

```
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)
```

Result:- Thus the program from various searching and operations is ~~done~~ executed and verified successfully.

VEL TECH

ORMANCE (5)	
T AND ANALYSIS (5)	
OCE (5)	
(5)	
AL :	