Task-5 :- Implement various searching and sorting 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 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 of employee is found.

1.) Input Defination.
2.) Define the function find-employee-by-id that takes the 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 though the list:
use a for loop to iterate though each dictionary in the employee 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 the None.

Output:-

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

[illegible]

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'},
    ]
```

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': 'Bob', '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 managment system for a school the system maintains a list of student records, where each records is represented as a dictionary containing a student name and score. the school needs to generate a report that displays student's scores in ascending order. your task is to implement a feature that sorts the students records by their scores using the Bubble by their scores using the Bubble sort algorithm.

Algorithm:-

1.) Initialization: Get length of the students list and store it in n.

2.) outer loop: Iterate from i=0 to n-1. This loop represents the number of passes through the list.

3.) Track swaps:- Initialize a boolen variable swapped to false this variable will track if any swaps are made in the current pass.

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}
```

O/P

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 student [j] ['score']
    students [j+1] ['score']:
    students [j], students [j+1] = students [j+1] ,students [j]
    Swapped = True
    if not swapped:
    break
```

4.) Inner loop:- Iterate from j=0 to n-i-2. This loop compares adjacent elements in the list and performs swaps if it is necessary.

5.) compare and swap:-

* for each pair of adjacent elements (i.e.., student [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:- check if swapped is False.

7) completion:

* The function modifies the students list in place, sorting it by score.

Program:-

```
def bubble-sort - scores(students):
n=len (students)
for i in range (n):
swapped = false.
for j in range(0,n-i-1):
if students [j] ['score']
students [j+1] ['score']:
students [j], students [j+1]= students [j+1], students [j]
swapped= True
if not swapped:
break
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).
```

VEL TECH	
EX No.	
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	[illegible] 3/4.

Result:- thus, the program for various searching and sorting operation is excuted and verified successfully.

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