Task-5 Implement various searching and sorting operations in python programming

Aim: To implement various searching & 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

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 represent 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 employee list
4. check for matching ID
   within the loop check if the id 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

```
def find_employee_by_id(employee, target_id):
for employee in employees:
if employee[id] == target_id:
return employee
return none
#test the function
  employee =[
   {'id': 1'name': 'Alice','department': 'HR'},
   {'id': 2'name': "Bob",' department': 'engineering'},
   {'id': 3,'name': 'Charlie','department': 'sales'},
  ]
  printf(find_employee_by_id(employee,2))
  # output : {'id': 2'name': 'Bob','department':
  'engineering'}
```

output

====== RESTART: C:/Users/41979/Desktop
/point 11/t3.py

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

5.2 you are developing a grade managment system for a school the system maintains a list of student records is represented as a dictionary containing a students name and score. the school needs to generate a report that displays students scores in ascending order your task is to implement a feature that sorts the students records by their scores using the Bubble sort algorithm

Algorithm:

1. initialization:-
   Get the length of the students list & sort it inn.
2. outer loop
   . iterate from i=0 to n-1 (inclusive) this loop represents the number of passes through the list
3. Track swap
   . 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 adjacement elements in the list and perform swap if necessary
5. Compare and swap:
   . for each pair of adjucent element (ie students [j] and students[j+1]),
   . compare their score value

- if students[j]['score'] > students[j+1]['score'] swap the two elements
- set swapped to True to indicate that a swap was made
- 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 and you can break out of the outer loop early

7. completion:
.The function modifies the students list in place, sorting it by score

Program:

```
def bubble_sort_score(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 than the next student[j],
students[j+1], students[j]
                swapped = True
# if no two elements were swapped
the list is already sorted
        if not swapped:
            break
# Example usage
```

output:

Before storing

{'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}

```
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 students in students:
print(students
```

VEL TECH	
EX No	
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	20
SIGN WITH DATE	25/10

Result:

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