Task :- 5 Date:- 28/08/25

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 contains id, name, and department. write a function. field find - employee - by - id that takes. this list and a target employee ID are arguments and returns the dictionary of the. employee with the matching ID, or none if no. such employee is found.

Alogrithm:

1. initialization.
 . Get the length of the students list and store it in n.
2. outer loops.
 . 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

Algorithm:-

1. Input Definition.
2. Define the function find_employee_by id. that takes two parameters. where each
 a. A list of dictionaries (employees) record with. dictionary represents an employee keys id, name and department.
 b. An integer (target_id) representing the employee ID to be searched.

output

```
{'id':2, 'name': 'Bob', 'department': 'Engineerin
```

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.

Program:-

```
def find-employee-by-id (employee, target-id):
  for employees 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'},
]
print(find-employee-by-id (employee, 2)
# output: {'id': 2, 'name': 'Bob', 'department':
'Engineering'}.
```

Result:- Thus the program for various searching and sorting operations is excuted and verified successfully.

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

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

5.2. you are developing a grade management. System for a School. The system maintain a list of student records, where each record is repre-sented as a dictionary containing a student's name and score. the school needs to generate a report that displays students' score in ascending order. your task is to implement 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 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

Output.

```
Before sorting'
{'name':'Alice','score':88}.
{'name':'Bob','score':95}.
{'name':'charlie','score':75}.
{'name':'Diana','scrore':85}.

After sorting.
{'name':'charlie','score':75}.
{'name':'Diana','score':85}.
{'name':'Alice','score':88}.
{'name':'Bob','score':95}.
```

where made during the pass, the list is already sorted, 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 -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  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
 sorted if not swapped:
        break.
# example usage.
     {'name': 'Alice', 'score':88},
     {'name': 'Bob', 'score': 95},
     {'name': 'Charlie', 'score': 75},
     {'name': 'Diana', 'score': 85}.
 ]  print ("Before sorting:")  for students 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)	4
TOTAL (20)	19
SIGN WITH DATE	[illegible] 28/8/25

Results:- Thus the program for various searching and sorting operations is executed and verified successfuly.