17/9/25

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 find-employee-by-id that takes list and employee.

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:
   within the loop, check if the id field of the current dictionary matches the target-id.

```
Restart: c/ users/ 41970/ Desktop/ Print/ t3.py.
('id': 2, 'name': 'bob', 'department' 'Engineering')
```

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'}
]
Print (find_employee_by_id(employees 2))
```

Output:-

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

5.2 You are developing a grade management system for a school. The system maintains 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. 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. This loop compares adjacent elements in the list and performs swaps if necessary.
5. 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 great-
er 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.
 students = [.
   {'name': 'Alice', 'score': 88},
   {'name': 'Bob', 'score': 95},
   {'name': 'charlie', 'score': 75}.
 ]
  Print ("Before Sorting:")
  for student in students:
    Print (student).
  bubble_sort_scores(students)
  Print( "\nAfter Sorting:")
  for student in students:
    Print (student).
```

Result:-

Thus the Program for various searching and sorting operations is executed and verified successfully.

VEL TECH	
EX NO.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	[illegible] 17/9