

Task 5 - Implement various searching and sorting operations in Python Programming.

01/09

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 this list and a target employee ID as arguments and returns the dictionary of 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, where each dictionary represents an employee record with keys id, name, and department.
 - b) An integer representing 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.
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 (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' },

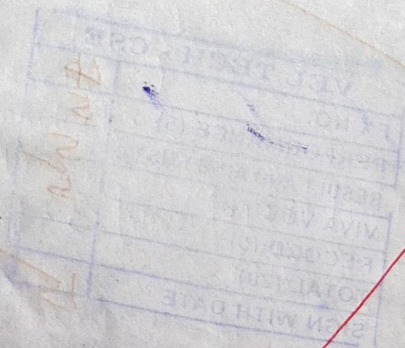
]

Print (find_employee_by_id (employees, 2)) # output: { 'id': 2,
'name': 'Bob', 'department': 'Engineering' }

output:

===== Restart : c: /users / 91979 / Desktop / Print 1 / ts. Py

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



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 current student is greater than the  
                next students [j], students[j] = students[j+1], students[j+1]
```

```
                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},
```

```
        {'name': 'Diana', 'score': 85}
```

```
    ]
```

```
    print("Before sorting:")
```

```
    for student in students:
```

```
        print(student)
```

```
    bubble_sort_scores(students)
```

```
    print("\nAfter sorting:")
```

```
    for student in students:
```

```
        print(student)
```

Output:

== Restart : c:\users\191979\Desktop\Print\1\3.py

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}
```


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. The school needs to generate a report that displays students' scores in ascending order. Your task is to implement a feature that sorts the student records by their scores using 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 $n-1$. 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 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. Compare and swap:
 - * For each pair of adjacent elements:
 - compare their score values
 - if $\text{students}[j][\text{'score'}] > \text{students}[j+1][\text{'score'}]$, 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 pass.
7. Completion.

VEL TECH - CSE	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	25
SIGN WITH DATE	

Result:

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