

TASK-5.1

Implement various searching and sorting operations in Python Programming

Aim: To implement various searching and sorting operations in Python Programming.

Algorithm:

1. Input Definition
2. Define the function `find-employee-by-id` takes two parameters
 - a. A list of dictionaries (`employee`) where each dictionary represents an employee record with keys `id`, `name`, and `department`
 - b. An integer (`target-id`) representing the employee id to searching
3. Iterate through the list:
use a for loop to iterate through each dictionary in the `employee` list.
4. check for matching id.
which in the loop, check if the `id` field of current dictionary matches the `target-id`.
5. Return Matching Record:
if a match is found, return the current dictionary
6. ~~if~~ Handle No match.
if the loop completes without finding a match return

output
{ id: 2, name: 'Bob', department: 'engineering' }

output:
Enter the first number

output
{ 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
```

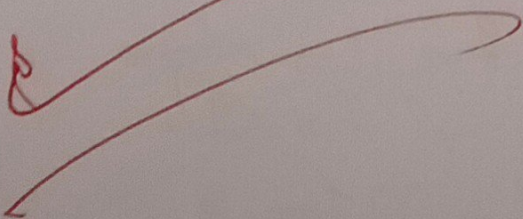
Test the function.

```
employees = [  
    {'id': 1, 'name': 'Alica', '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' }

Result: Thus to implement various searching
and sorting operation in Python
Programming.



TASK - 5.2

Aim: To develop a Python Program that store student records by score in ascending order 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

3. Track swaps:

- Initialize a boolean variable swapped to false
This variable will track if any swap are made current Pass

4. Inner loop

- Iterate from $j=0$ to $n-i-2$ (inclusive). This loop compares adjacent elements in the list and perform swaps if necessary

5. compare and swap

- for each pair of adjacent element (ie student $[j]$ and students $[j+1]$);

output

Before sorting

```
{ 'name': 'Alice', 'score': 88 }
```

```
{ 'name': 'Bob', 'score': 95 }
```

```
{ 'name': 'Charlie', 'score': 75 }
```

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

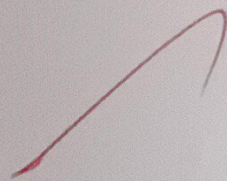
After sorting

```
{ 'name': 'Charlie', 'score': 75 }
```

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

```
{ 'name': 'Alice', 'score': 88 }
```

```
{ 'name': 'Bob', 'score': 95 }
```



- 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 swap were 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 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
```


$students[j] > students[j+1] \Rightarrow students[j], students[j+1]$
 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': 85},
 {'name': 'Diana', 'score': 85}.
]

print("Before sorting")
 for student in students:
 print(student)
 bubble_sort_scores(students)
 for student in students:
 print(student)

VELTECH	
EX No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	2
RECORD (4)	9
TOTAL (15)	
SIGN WITH DATE	

Result: Thus the Program for various searching and
 sorting operating is executed and verify successfully