

## 3-9-25 Task-5 Implement Various Searching and Sorting operations in Python Programming.

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

### Algorithm:

1. Input definition
2. Define the function find-employee-by-id that takes two parameters
  - a) An integer (target-id) representing the employee ID to be searched.
3. Iterate through the list
4. Return Matching Record  
if a match is found, return the current dictionary
5. Handle No Match:  
if the loop completes without finding a match, return None

### Program 5.1

```
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'}
```

# set operations

both: all: both: intersection (both: all: both:)

only: all: both: difference (both: all: both:)

only: both: all: difference (both: all: both:)

output unique: all: both: union (both: all: both:)

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

Result: Thus that use of various data type list  
table. Dictionary in Python can successfully

Task 2: Implement various searching and sorting algorithms in Python

Aim: To implement various searching and sorting algorithms in Python

Output:

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

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

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

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

3. Iterate through the list  
4. Return Matching Record

5. Handle No Match

6. If the loop completes without finding a match, return None

get\_friend - employee - id (employee, target - id)

for employee in employees:  
if employee['id'] == target - id:  
return employee

return None  
# End of the function  
employees = [

{ 'id': 1, 'name': 'Alice', 'department': 'HR' },  
{ 'id': 2, 'name': 'Bob', 'department': 'Finance' },

## Task 5.2

Aim: To implement a feature that sorts the student record by their scores using the Bubble sort.

### Algorithm:

1. Initialization
2. outer loop
3. Track Swaps
4. Inner loop
5. Compare and swap
6. Early Termination
7. Completion

### Program 5.2

```
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-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
```

```
            break
```

```
students = [
```

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

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

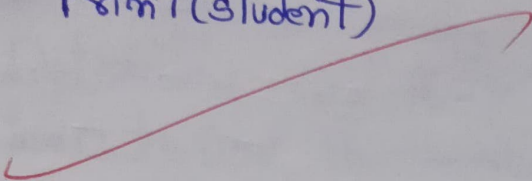
```
    {'name': 'Charlie', 'score': 75},
```

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

```

Print("Before sorting:")
for student in students:
    Print(student)
bubble-sort-scores(students)
Print("After 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.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	