

## 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 take 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'},  
]
```

Point(`find-employee-by-id(employees, 2)`) // output `{'id': 2, 'name': 'Bob', 'department': 'Engineering'}`

creation f32 #

(opgelds - editors) mifasstni. radtloob - id . Höd

(opgelds - editors) mifasstni. radtloob - id = id - ulro

(radtloob - id) mifasstni. opgelds - editors - editors - ulro

output (opgelds - editors) mifasstni. radtloob - id = 110 - eupim

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



fall geht stob mifasstni d seu Forte sunt bluest  
mifasstni und richtig mi gemitziib . slqut

base class's methods are inherited by derived class

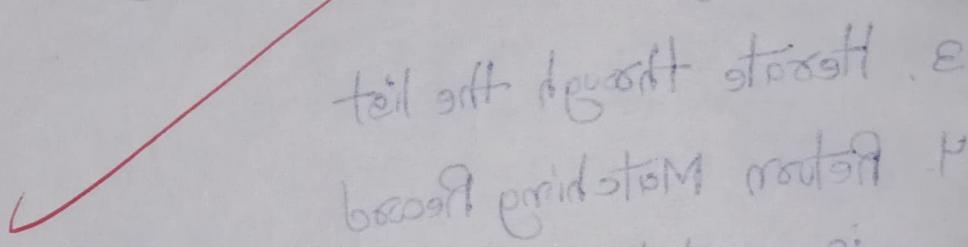
Output:   
 Alice has scored 88  
 Bob has scored 95  
 Charlie has scored 75  
 Diana has scored 85

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

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

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

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



## Task 5.2

Aim: To implement a feature that sorts the student records 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[i]['score'] > students[i+1]['score']:
                # Swap if the score of the current student is greater
                # than the next
                students[i], students[i+1] = students[i+1], students[i]
                swapped = True
        break
```

students = [  
 {'name': 'Alice', 'score': 88},  
 {'name': 'Bob', 'score': 95},  
 {'name': 'Charlie', 'score': 75},  
 {'name': 'Diana', 'score': 85}]

Point ("Before sorting:")  
for student in students:

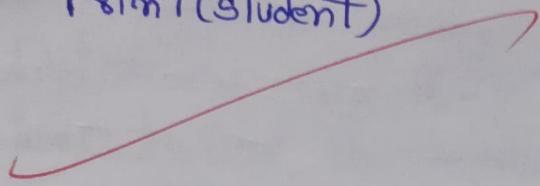
Point (student)

'bubble - sort - scores (student)'

Point ("In After sorting:")

for student in students:

Point (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	