

Low-Level Design (LLD) for Student Score List Operations

Difficulty Level: Easy | Total Marks: 10

Standards Followed: 4 Functions | 3 Visible Test Cases

Concepts Tested

- Python list operations
- Dictionary creation and mapping
- Conditional statements (if-else)
- Aggregation functions (sum, len, max)
- Sorting and filtering lists
- Basic algorithmic logic

=====

Problem Statement

Design a system that manages a list of student scores and performs operations such as:

- Creating a dictionary mapping each score to its status ("Pass"/"Fail")
- Finding the highest score
- Calculating the average score
- Filtering and sorting passing scores in descending order

Given Input:

scores = [45, 78, 92, 55, 88, 67, 95, 52]

Rules:

- If score $\geq 80 \rightarrow$ status = "Pass"

- If score < 80 → status = "Fail"

=====

Operations

1. Create Score Status Dictionary

Function Prototype:

```
def create_status_dict(self):
```

Expected Output:

```
{45: 'Fail', 78: 'Fail', 92: 'Pass', 55: 'Fail', 88: 'Pass', 67: 'Fail', 95: 'Pass', 52: 'Fail'}
```

2. Find Highest Score

Function Prototype:

```
def find_highest_score(self):
```

Expected Output:

```
95
```

3. Calculate Average Score

Function Prototype:

```
def calculate_average(self):
```

Expected Output:

```
71.5
```

4. Get Passing Scores (Descending Order)

Function Prototype:

```
def get_passing_scores(self):
```

Expected Output:

```
[95, 92, 88]
```

Implementation Code

```
class ScoreManager:
```

```
    def __init__(self):
```

```
        """Initialize score list."""
```

```
        self.scores = [45, 78, 92, 55, 88, 67, 95, 52]
```

```
    def create_status_dict(self):
```

```
        """Create dictionary mapping score to Pass/Fail."""
```

```
        """Your code here"""
```

```
        print(status_dict)
```

```
    def find_highest_score(self):
```

```
        """Print highest score."""
```

```
        """Your code here"""
```

```
    def calculate_average(self):
```

```
        """Print average score."""
```

```
        """Your code here"""
```

```
        print(round(avg, 1))
```

```
def get_passing_scores(self):
    """Print passing scores sorted in descending order."""
    """Your code here"""
    print(passing)
```

Test Case Table:

Test Case ID Test Case Description		Associated Function(s)	Marks
TC1	Creating score status dictionary	create_status_dict()	2.5 Marks
TC2	Finding highest score	find_highest_score()	2.5 Marks
TC3	Calculating average score	calculate_average()	2.5 Marks
TC4	Getting passing scores in descending order	get_passing_scores()	2.5 Marks
TOTAL	All test cases passed	-	10 Marks

Visible Test Cases

TC1 Input:

1
status

Output:

{45: 'Fail', 78: 'Fail', 92: 'Pass', 55: 'Fail', 88: 'Pass', 67: 'Fail', 95: 'Pass', 52: 'Fail'}

TC2 Input:

1
highest

Output:

95

TC3 Input:

1

average

Output:

71.5

TC4 Input:

1

passing

Output:

[95, 92, 88]