Low-Level Design (LLD) – Student Score Analyzer

Difficulty Level: Easy
Total Marks: 10

Standards Followed:

- 4 Functions
- 5 Visible Test Cases
- No Hidden Test Cases

Concepts Tested

Python dictionaries
List filtering and sorting
Conditional logic (if/else)
Aggregation operations (max, average)
Class-based implementation

Problem Statement

Create a **Student Score Analyzer** system that processes a list of student scores and performs multiple operations.

The system should:

- Assign Pass/Fail status to each score
- Identify the highest score
- Calculate the average score
- Filter passing scores and sort them in descending order

All logic must be implemented using a class-based structure.

Operations

---

1. Create Score Status Dictionary

Create a dictionary where:

- Key = score
- Value = status

Rules:

- score >= 80 → "Pass"
- score < 80 → "Fail"

Function Prototype:

```
def create_score_status(self, score_list: list)
```

Example Input:

```
[75, 85, 90, 60]
```

Expected Output:

```
{75: 'Fail', 85: 'Pass', 90: 'Pass', 60: 'Fail'}
```

---

2. Find Highest Score

Return the highest score from the list.

Function Prototype:

```
def find_highest_score(self, score_list: list)
```

Expected Output:

```
90
```

---

3. Calculate Average Score

Calculate the average value of all scores.

Function Prototype:

```
def calculate_average_score(self, score_list: list)
```

Expected Output:

```
77.5
```

---

4. Get Sorted Passing Scores

Create a new list:

- Include only passing scores (>= 80)
- Sort in descending order

Function Prototype:

```
def get_sorted_passing_scores(self, score_list: list)
```

Expected Output:

```
[90, 85]
```

Implementation Code (Student Scaffold)

Students must complete the following class:

```
class StudentScoreAnalyzer:

    def __init__(self):
        pass

    def create_score_status(self, score_list):
        # create dictionary with pass/fail status
        pass

    def find_highest_score(self, score_list):
        # return highest score
        pass

    def calculate_average_score(self, score_list):
        # calculate average
        pass

    def get_sorted_passing_scores(self, score_list):
        # filter and sort passing scores
        pass
```

Test Cases and Marks Allocation (All Visible)

Test Case ID	Description	Function	Marks
TC1	Create pass/fail dictionary correctly	create_score_status()	2
TC2	Identify highest score correctly	find_highest_score()	2
TC3	Calculate average score	calculate_average_score()	2
TC4	Filter passing scores correctly	get_sorted_passing_scores()	2
TC5	Sort passing scores descending	get_sorted_passing_scores()	2

TOTAL MARKS: 10