

Low-Level Design (LLD) for Student Exam Dictionary

Advanced Operations

Difficulty Level: Medium | Total Marks: 20

Standards Followed: 3 Functions | 3 Visible Test Cases

Concepts Tested

- Nested dictionary operations
 - Iteration using items(), keys(), values()
 - Aggregation functions (sum, len)
 - Average calculations
 - Conditional logic and comparisons
 - Data processing and transformation
-

Problem Statement

Design a system that manages a dictionary of student exam results and performs operations such as:

- Calculating the average score for each student across all subjects
- Identifying the student with the highest average score
- Calculating the average score for each subject across all students

Given Input:

```
students = {  
    "S001": {"name": "Arjun", "math": 85, "science": 92, "english": 78},  
    "S002": {"name": "Nisha", "math": 95, "science": 88, "english": 90},  
    "S003": {"name": "Rohan", "math": 72, "science": 79, "english": 85},  
    "S004": {"name": "Divya", "math": 88, "science": 95, "english": 92},  
    "S005": {"name": "Karan", "math": 80, "science": 82, "english": 88}  
}
```

Operations

1. Calculate Average Score for Each Student

Function Prototype:

```
def calculate_student_averages(self):
```

Expected Output:

Student Averages: {'S001': 85.0, 'S002': 91.0, 'S003': 78.7, 'S004': 91.7, 'S005': 83.3}

2. Find Student with Highest Average Score

Function Prototype:

```
def find_highest_average(self):
```

Expected Output:

Highest Average: Divya (S004) - 91.7

3. Calculate Average Score for Each Subject

Function Prototype:

```
def calculate_subject_averages(self):
```

Expected Output:

```
Subject Averages: {'math': 84.0, 'science': 87.2, 'english': 86.6}
```

Implementation Code

```
class StudentResultManager:
```

```
    def __init__(self):
        """Initialize student results dictionary."""
        self.students = {
            "S001": {"name": "Arjun", "math": 85, "science": 92, "english": 78},
            "S002": {"name": "Nisha", "math": 95, "science": 88, "english": 90},
            "S003": {"name": "Rohan", "math": 72, "science": 79, "english": 85},
            "S004": {"name": "Divya", "math": 88, "science": 95, "english": 92},
            "S005": {"name": "Karan", "math": 80, "science": 82, "english": 88}
        }
```

```
    def calculate_student_averages(self):
        """Calculate and print average score for each student."""
```

```
averages = {}
for sid, data in self.students.items():
    total = data["math"] + data["science"] + data["english"]
    avg = round(total / 3, 1)
    averages[sid] = avg
print("Student Averages:", averages)
```

```
def find_highest_average(self):
    """Find and print student with highest average score."""
    highest_id = None
    highest_name = None
    highest_avg = 0
    for sid, data in self.students.items():
        avg = round((data["math"] + data["science"] + data["english"]) / 3, 1)
        if avg > highest_avg:
            highest_avg = avg
            highest_id = sid
            highest_name = data["name"]
    print(f"Highest Average: {highest_name} ({highest_id}) - {highest_avg}")
```

```
def calculate_subject_averages(self):
    """Calculate and print subject-wise averages."""
    math_total = 0
    science_total = 0
    english_total = 0
    count = len(self.students)
```

```
for data in self.students.values():
```

```

math_total += data["math"]
science_total += data["science"]
english_total += data["english"]

subject_avg = {
    "math": round(math_total / count, 1),
    "science": round(science_total / count, 1),
    "english": round(english_total / count, 1)
}

print("Subject Averages:", subject_avg)

```

Test Case Table

Test Case ID	Test Case Description	Associated Function(s)	Marks
TC1	Calculate average score for each student	calculate_student_averages()	7 Marks
TC2	Find student with highest average score	find_highest_average()	7 Marks
TC3	Calculate subject-wise averages	calculate_subject_averages()	6 Marks
TOTAL	All test cases passed	-	20 Marks

Visible Test Cases

TC1 Input:

1

student_avg

Output:

Student Averages: {'S001': 85.0, 'S002': 91.0, 'S003': 78.7, 'S004': 91.7, 'S005': 83.3}

TC2 Input:

1

highest

Output:

Highest Average: Divya (S004) - 91.7

TC3 Input:

1

subject_avg

Output:

Subject Averages: {'math': 84.0, 'science': 87.2, 'english': 86.6}