

Assignment – 6

AI Assistance in coding

Name:K.Raghavendra, Ht.no:2303a51897 , Batch:09

Task 1 – Student Performance Evaluation

Create a method to display student details using self attributes

Create a method that checks whether the student's marks

are above the class average using if-else

Return an appropriate performance message

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar has 'EXPLORER' and 'TEMP' expanded, showing a file named '2303a51897_6_4.py'. The main editor area contains the following Python code:

```
class Student:
    def __init__(self, name, marks, class_average):
        self.name = name
        self.marks = marks
        self.class_average = class_average

    def display_details(self):
        """Display student details using self attributes"""
        print(f"Student Name: {self.name}")
        print(f"Marks: {self.marks}")
        print(f"Class Average: {self.class_average}")

    def check_performance(self):
        """Check if student's marks are above class average and return performance message"""
        if self.marks > self.class_average:
            return f"{self.name} is performing above average. Marks: {self.marks}"
        else:
            return f"{self.name} is performing at or below average. Marks: {self.marks}"
```

The terminal at the bottom shows the output of running the script:

```
PS C:\Users\HP\Desktop\Temp> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/HP/Desktop/Temp/2303a51897_6_4.py
Student Name: John
Marks: 85
Class Average: 75
John is performing above average. Marks: 85 > Class Average: 75

Student Name: Alice
Marks: 70
Class Average: 75
Alice is performing at or below average. Marks: 70 <= Class Average: 75
PS C:\Users\HP\Desktop\Temp>
```

A floating AI assistance panel on the right says "Ask about your code" and "Generate Agent Instructions to onboard AI onto your codebase".

Task 2 – Data Processing in Monitoring System

Inside the loop, identify even numbers using modulus operator

Calculate the square of even readings

Print the reading and its square in a readable format

```
class Student:
    def __init__(self, name, marks, class_average):
        self.name = name
        self.marks = marks
        self.class_average = class_average

    def display_details(self):
        """Display student details using self attributes"""
        print(f"Student Name: {self.name}")
        print(f"Marks: {self.marks}")
        print(f"Class Average: {self.class_average}")

    def check_performance(self):
        """Check if student's marks are above class average and return performance message"""
        if self.marks > self.class_average:
            return f"{self.name} is performing above average. Marks: {self.marks}"
        else:
            return f"{self.name} is performing at or below average. Marks: {self.marks}"

Marks: 85
Class Average: 75
John is performing above average. Marks: 85 > Class Average: 75

Student Name: Alice
Marks: 70
Class Average: 75
Alice is performing at or below average. Marks: 70 <= Class Average: 75
Even Reading: 12, Square: 144
Even Reading: 22, Square: 484
Even Reading: 8, Square: 64
PS C:\Users\HP\Desktop\Temp>
```

Task 3 – Banking Transaction Simulation

Create a deposit method that adds money to balance

Create a withdraw method

Use if-else to prevent withdrawal when balance is insufficient

Display user-friendly messages

Task 4 – Student Scholarship Eligibility

Use a while loop to iterate through the student list

Check if the score is greater than 75

Print the name of eligible students

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes File, Edit, Selection, View, Go, Run, and Temp tabs. The left sidebar has sections for Explorer, TEMP, and Chat. The main area displays a Python file named '2303a51897_6_4.py' with the following code:

```
2303a51897_6_4.py
1 # Welcome
2
3 # students = [{"name": "John", "marks": 75}, {"name": "Jane", "marks": 82}, {"name": "Bob", "marks": 65}, {"name": "Alice", "marks": 90}]
4
5 for student in students:
6     if student["marks"] > 75:
7         print(f"Eligible Student: {student['name']}")
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
```

The code includes two examples: one for finding eligible students based on marks and another for a BankAccount class. The terminal below shows the output of the first example:

```
Even Reading: 12, Square: 144
Even Reading: 22, Square: 484
Even Reading: 8, Square: 64
Eligible Student: John
```

The terminal also shows the output of the BankAccount example:

```
--- Bank Account Demo ---
Deposited 50. New balance: 150
Withdraw 30. New balance: 120
Insufficient balance. Available: 120, Requested: 150
Deposit amount must be positive.
Withdrawal amount must be positive.
```

The status bar at the bottom indicates the file is saved, the current line is 61, column 36, and the version is 3.13.9 (Microsoft Store). A floating AI interface on the right says "Ask about your code" with a "Generate Agent Instructions" button.

Task 5 – Online Shopping Cart Module

Create a method to add items to the shopping cart

Create a method to remove items from the cart

Create a method to calculate total bill using a loop

Apply discount if total exceeds a certain amount

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar has 'EXPLORER' and 'TEMP' sections. The main area displays a Python file named '2303a51897_6_4.py'. The code implements a bank account demo and a shopping cart example. The terminal below shows the execution of the script, including withdrawal errors and shopping cart operations like adding items and calculating totals. A GitHub Copilot AI assistant is visible on the right, suggesting code improvements.

```
print("n--- Bank Account Demo ---")
account = BankAccount("Bob", 100)
account.deposit(50)
account.withdraw(30)
account.withdraw(150)
account.deposit(-10)
account.withdraw(0)

# ShoppingCart example
print("n--- Shopping Cart Demo ---")
cart = ShoppingCart()
cart.add_item("Book", 40)
cart.add_item("Pen", 10)
cart.add_item("Bag", 80)
cart.remove_item("Pen")
cart.calculate_total()

--- Shopping Cart Demo ---
Added Book for 40.
Added Pen for 10.
Added Bag for 80.
Removed Pen from cart.
Total bill: 108.0 (Discount applied: 12.0)
PS C:\Users\HP\Desktop\Temp>
```

Analysis:

This experiment helped the practical use of Python concepts such as classes, loops, and conditional statements. By working on real-world scenarios like student evaluation, banking, and shopping systems, improved problem-solving skills and learned how GitHub Copilot can assist in faster and more efficient code development.