

# **AI Assisted Coding lab Assignment:6.4**

Hall no: 2303A52343

**Name: E.srivarsha**

Batch : 44

## Task 1: Student Performance Evaluation System

Prompt: Complete the Student class by adding methods to display student details

The screenshot shows the PyCharm IDE interface. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a back/forward button. The title bar says "Q\_Ai Assistant Lab". The left sidebar has sections for EXPLORER, AI ASSISTANT LAB (with files like AI lab 5.4.py, AI lab 7.1.py, etc.), and application.log. The main editor area displays the content of "AI lab6.4.py". The code is a Python script for a Student Performance Evaluation System:

```
ai lab6.4.py ...
1 #Task 1: Student Performance Evaluation System
2 class Student:
3     def __init__(self, name, roll_number, marks):
4         self.name = name
5         self.roll_number = roll_number
6         self.marks = marks
7     def display_details(self):
8         print("Name:", self.name)
9         print("Roll Number:", self.roll_number)
10        print("Marks:", self.marks)
11
12    def check_performance(self, class_average):
13        if self.marks > class_average:
14            return "Performance: Above Class Average"
15        else:
16            return "Performance: Below Class Average"
17
18 student1 = Student("Meghana", 101, 82)
19 student1.display_details()
20 print(student1.check_performance(75))
21
```

Below the editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, AZURE, and GITLens. The TERMINAL tab is active, showing command-line output:

```
PS C:\Users\MEGHANA\OneDrive\Desktop\AI Assistant Lab & C:/Users/MEGHANA/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/MEGHANA/OneDrive/Desktop/AI Assistant Lab\AI lab6.4.py"
Name: Meghana
Roll Number: 101
Marks: 82
○ Performance: Above Class Average
PS C:\Users\MEGHANA\OneDrive\Desktop\AI Assistant Lab>
```

The bottom left corner shows navigation icons for OUTLINE and TIMELINE.

## Explanation:

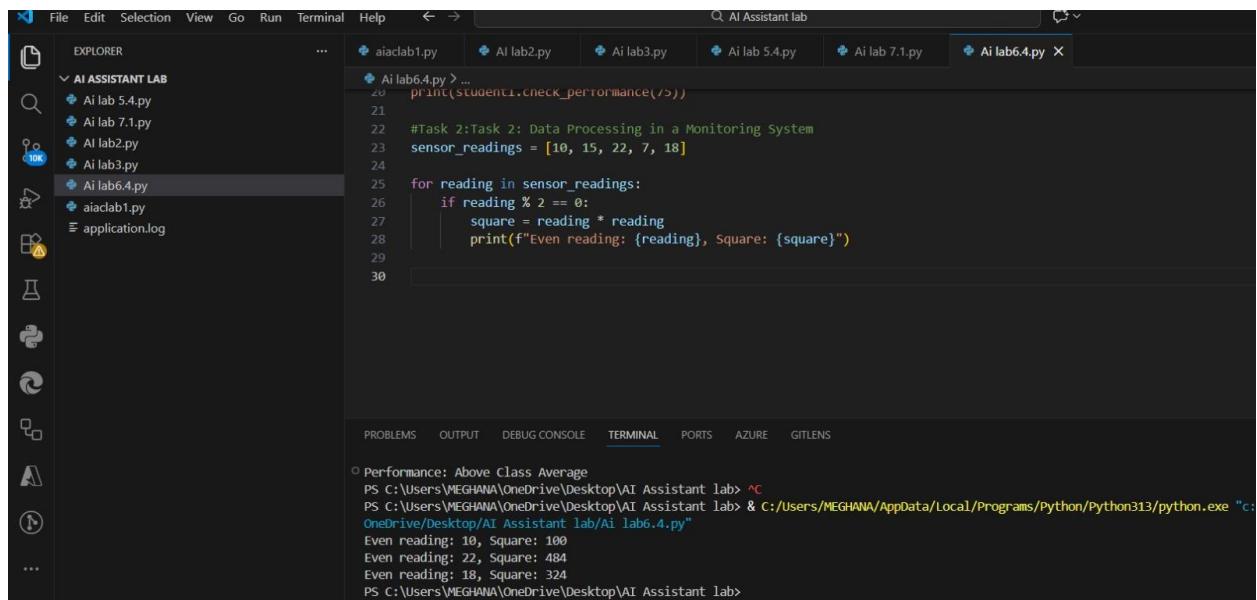
1. A Student class is created with name, roll number, and marks.
  2. The constructor initializes these attributes using self.

3. GitHub Copilot is guided using comments to generate methods.
4. One method displays student details.
5. Another method checks performance using if-else conditions.

## Task 2: Data Processing in a Monitoring System

Prompt:

Generate the loop logic to identify even sensor readings, calculate square, and print the result using if-else.



```

File Edit Selection View Go Run Terminal Help ⏮ ⏶ Q AI Assistant lab

EXPLORER
AI ASSISTANT LAB
Ai lab 5.4.py
Ai lab 7.1.py
Ai lab2.py
Ai lab3.py
Ai lab6.4.py
aiaclab1.py
application.log

Ai lab6.4.py > ...
20  print(student1.check_performance(>))
21
22 #Task 2: Task 2: Data Processing in a Monitoring System
23 sensor_readings = [10, 15, 22, 7, 18]
24
25 for reading in sensor_readings:
26     if reading % 2 == 0:
27         square = reading * reading
28         print(f"Even reading: {reading}, Square: {square}")
29
30
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE GITLENS
Performance: Above Class Average
PS C:\Users\MEGHANA\OneDrive\Desktop\AI Assistant lab> ^C
PS C:\Users\MEGHANA\OneDrive\Desktop\AI Assistant lab> & C:/Users/MEGHANA/AppData/Local/Programs/Python/Python313/python.exe "c:/OneDrive/Desktop/AI Assistant lab/Ai lab6.4.py"
Even reading: 10, Square: 100
Even reading: 22, Square: 484
Even reading: 18, Square: 324
PS C:\Users\MEGHANA\OneDrive\Desktop\AI Assistant lab>

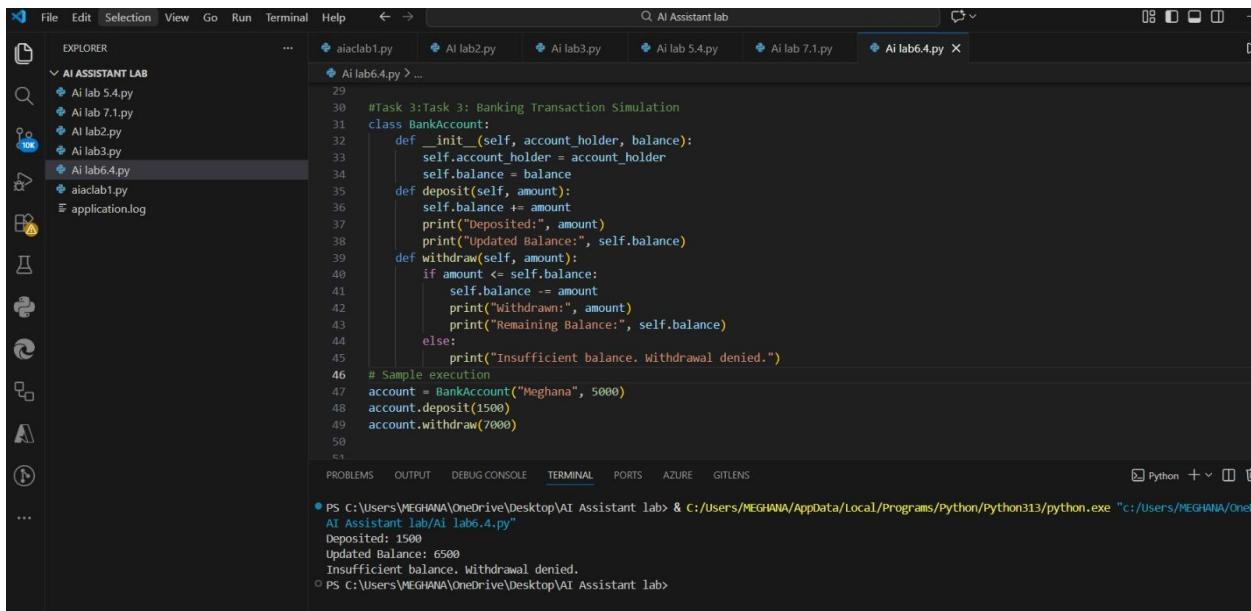
```

Explanation:

- A for loop iterates through sensor readings.
- Copilot identifies even numbers using the modulus operator.
- Squares of even numbers are calculated and printed.
- Conditional statements control the processing logic.

## Task 3: Banking Transaction Simulation

Prompt: Generate Complete the BankAccount class by adding deposit and withdrawal methods with balance checks using if-else.



```
File Edit Selection View Go Run Terminal Help < > Q: AI Assistant lab
EXPLORER ... aiaclab1.py Ai lab2.py Ai lab3.py Ai lab 5.4.py Ai lab 7.1.py Ai lab6.4.py
AI ASSISTANT LAB
Ai lab 5.4.py
Ai lab 7.1.py
Ai lab2.py
Ai lab3.py
Ai lab6.4.py
aiaclab1.py
application.log
29
30 #Task 3:Task 3: Banking Transaction Simulation
31 class BankAccount:
32     def __init__(self, account_holder, balance):
33         self.account_holder = account_holder
34         self.balance = balance
35     def deposit(self, amount):
36         self.balance += amount
37         print("Deposited:", amount)
38         print("Updated Balance:", self.balance)
39     def withdraw(self, amount):
40         if amount <= self.balance:
41             self.balance -= amount
42             print("Withdrawn:", amount)
43             print("Remaining Balance:", self.balance)
44         else:
45             print("Insufficient balance. Withdrawal denied.")
46 # Sample execution
47 account = BankAccount("Meghana", 5000)
48 account.deposit(1500)
49 account.withdraw(7000)
50
51
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE GITLENS
Python + ▾ ▾
PS C:\Users\MEGHANA\OneDrive\Desktop\AI Assistant lab> & c:/Users/MEGHANA/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/MEGHANA/OneDrive/Desktop/AI Assistant lab/Ai lab6.4.py"
AI Assistant lab/Ai lab6.4.py"
Deposited: 1500
Updated Balance: 6500
Insufficient balance. Withdrawal denied.
PS C:\Users\MEGHANA\OneDrive\Desktop\AI Assistant lab>
```

## Explanation

- A BankAccount class stores account holder and balance.
- Copilot generates deposit and withdraw methods.
- If-else conditions prevent invalid withdrawals.
- User-friendly messages handle insufficient balance cases.

## Task 4: Student Scholarship Eligibility Check

### Prompt

Generate a while loop to iterate through the student list and print names of students scoring above 75.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar says "AI Assistant lab". The left sidebar has an "EXPLORER" section with files like "AI ASSISTANT LAB", "ai lab 5.4.py", "Ai lab 7.1.py", "Ai lab2.py", "Ai lab3.py", "Ai lab6.4.py", "aiaclab1.py", and "application.log". The main editor area contains the following Python code:

```
aiaclab6.4.py > ...
47 account = BankAccount("regional", 5000)
48 account.deposit(1500)
49 account.withdraw(7000)"""
50
51 #Task 4Task 4: Student Scholarship Eligibility Check
52 students = [
53     {"name": "Asha", "score": 78},
54     {"name": "Ravi", "score": 65},
55     {"name": "Meghana", "score": 88},
56     {"name": "Kiran", "score": 72}
57 ]
58
59 index = 0
60 while index < len(students):
61     if students[index]["score"] > 75:
62         print(students[index]["name"], "is eligible for scholarship")
63     index += 1
64
65
66
```

The bottom status bar shows the path "C:\Users\MEGHANA\OneDrive\Desktop\AI Assistant lab\ai lab6.4.py" and the command "python.exe". The terminal tab shows the output of the script running in PowerShell, which prints "Asha is eligible for scholarship" and "Meghana is eligible for scholarship".

## ❖ Explanation

- Student data is stored as a list of dictionaries.
- Copilot creates a while loop with index handling.
- Scores are checked using conditional statements.
- Eligible student names are printed clearly.

## Task 5: Online Shopping Cart Module

### Prompt :

Complete the shopping cart class by adding methods to add items, remove items, and calculate total price using conditions.

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows a tree view of files under "AI ASSISTANT LAB". Files listed include: aiaclab1.py, Al lab2.py, Al lab3.py, Al lab4.py, Al lab5.4.py, Al lab 7.1.py, and Al lab6.4.py (which is currently selected).
- Code Editor:** The main area displays Python code for a shopping cart module. The code defines a class `ShoppingCart` with methods for initializing the cart, adding items, removing items, and calculating the total price. It includes a discount logic for purchases over 2000.

```
64
65 #Task 5:Task 5: Online Shopping Cart Module
66 class ShoppingCart:
67     def __init__(self):
68         self.items = []
69     def add_item(self, name, price, quantity):
70         self.items.append({"name": name, "price": price, "quantity": quantity})
71         print(name, "added to cart")
72     def remove_item(self, name):
73         for item in self.items:
74             if item["name"] == name:
75                 self.items.remove(item)
76                 print(name, "removed from cart")
77                 return
78     def calculate_total(self):
79         total = 0
80         for item in self.items:
81             total += item["price"] * item["quantity"]
82
83         if total > 2000:
84             discount = total * 0.1
85             total -= discount
86             print("Discount applied:", discount)
87
88         print("Total Amount:", total)
89 # Sample execution
90 cart = ShoppingCart()
91 cart.add_item("Shoes", 1500, 1)
92 cart.add_item("Bag", 800, 1)
93 cart.calculate_total()
```

## Explanation

- A shopping cart structure manages items and prices.
- Copilot generates methods for cart operations.
- Conditional logic ensures valid item handling.
- The total cost is calculated and displayed correctly.