LAB-06

G.Sai teja

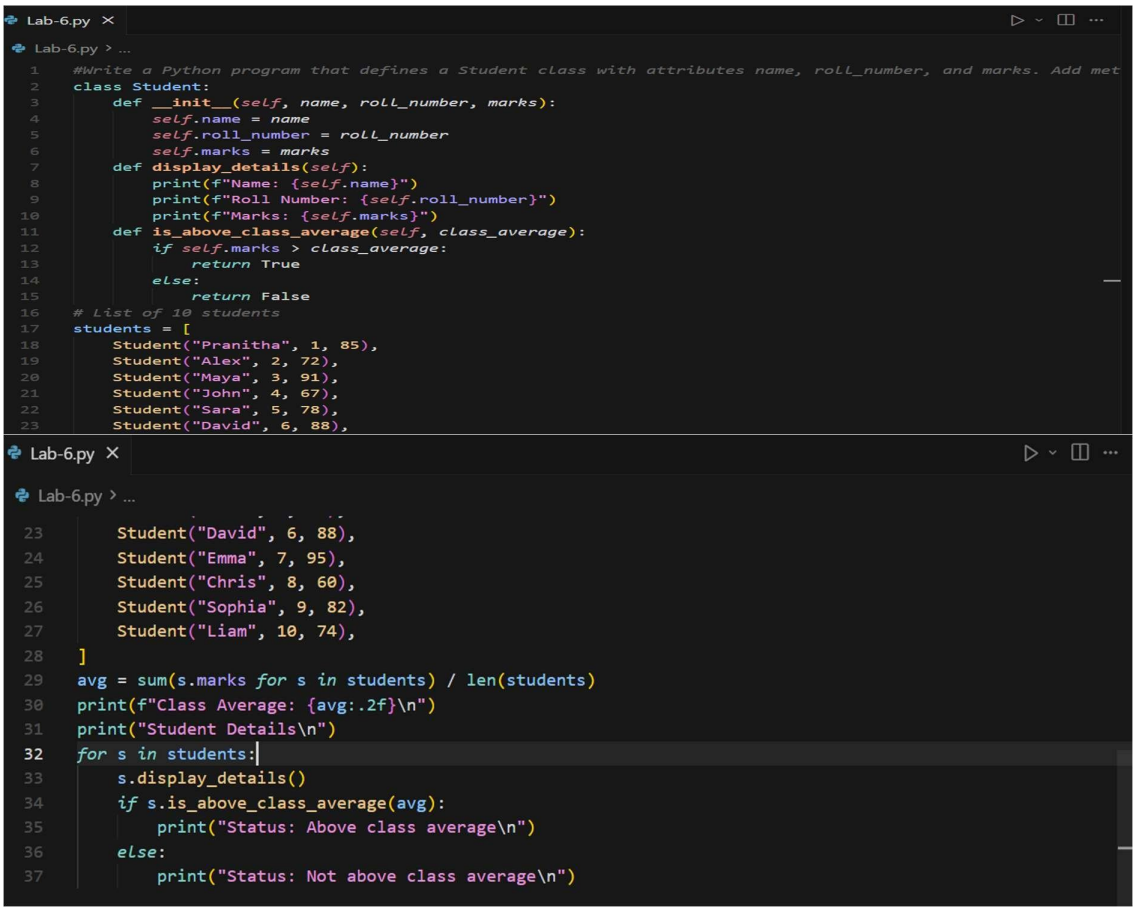
2303A52135

Batch 41

Task 1: Student Performance Evaluation System

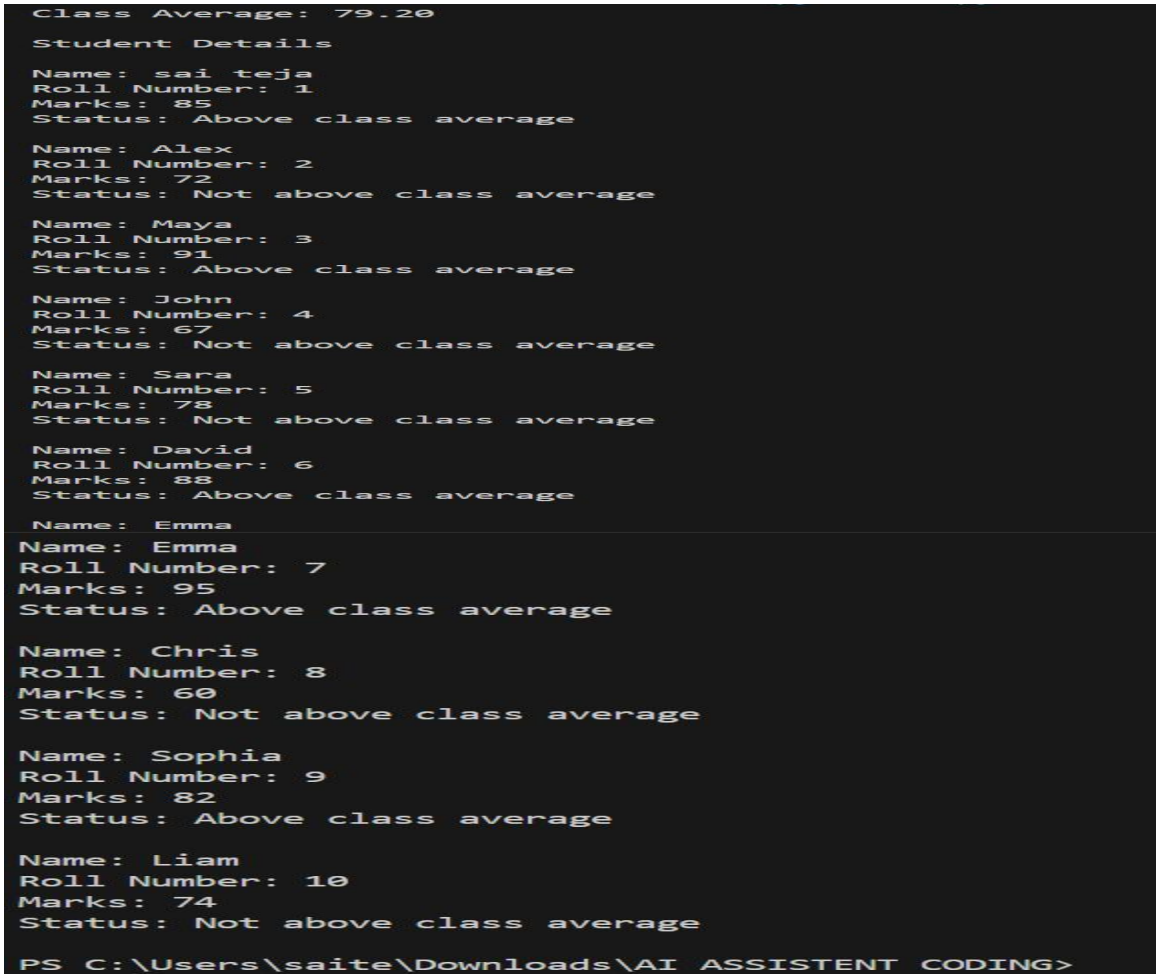
Prompt: Write a Python program with a Student class having attributes name, roll_number, and marks. Include methods to display student details and check if a student's marks are above the class average. Create a list of students, calculate the class average, and print each student's details along with their performance status.

CODE:



```python
#Write a Python program that defines a Student class with attributes name, roll_number, and marks. Add met
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks
    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")
    def is_above_class_average(self, class_average):
        if self.marks > class_average:
            return True
        else:
            return False
# List of 10 students
students = [
    Student("Pranitha", 1, 85),
    Student("Alex", 2, 72),
    Student("Maya", 3, 91),
    Student("John", 4, 67),
    Student("Sara", 5, 78),
    Student("David", 6, 88),
    Student("David", 6, 88),
    Student("Emma", 7, 95),
    Student("Chris", 8, 60),
    Student("Sophia", 9, 82),
    Student("Liam", 10, 74),
]
avg = sum(s.marks for s in students) / len(students)
print(f"Class Average: {avg:.2f}\n")
print("Student Details\n")
for s in students:
    s.display_details()
    if s.is_above_class_average(avg):
        print("Status: Above class average\n")
    else:
        print("Status: Not above class average\n")
```

OUTPUT:



JUSTIFICATION: This Python program defines a Student class with attributes for name, roll number, and marks. It includes methods to display each student's details and to check whether their marks are above the class average. A list of ten students is created, and the program calculates the overall class average by summing all marks and dividing by the number of students. It then prints the class average once, followed by a heading "Student Details," and iterates through each student to show their information along with a status message indicating if they are above or below the class average.

Task 2: Data Processing in a Monitoring System

PROMPT: Write a Python for loop to process sensor readings. Iterate over a list of integers, identify even numbers using the modulus operator, calculate their square, and print the result in a readable format.

CODE:

```python
#Write a Python for loop to process sensor readings. Iterate over a list of integers, identify even number
# List of sensor readings (integers)
sensor_readings = [12, 7, 20, 3, 16, 9, 4]
# Process readings: find even numbers, square them, and print
for reading in sensor_readings:
    if reading % 2 == 0:  # identify even numbers using modulus
        squared = reading ** 2
        print(f"Sensor reading {reading} is even; its square is {squared}.")
```

OUTPUT:

```
PS C:\Users\saite\Downloads\AI ASSISTENT CODING> & C:/Users/saite
/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/sai
te/Downloads/AI ASSISTENT CODING/As6.py/task2.py"
Sensor reading 12 is even; its square is 144.
Sensor reading 20 is even; its square is 400.
Sensor reading 16 is even; its square is 256.
Sensor reading 4 is even; its square is 16.
PS C:\Users\saite\Downloads\AI ASSISTENT CODING>
```

JUSTIFICATION: This code loops through sensor readings, checks if each number is even using the modulus operator, then calculates its square and prints a clear message showing the reading and its squared value, while ignoring odd numbers.

Task 3: Banking Transaction Simulation

PROMPT: This Python program defines a BankAccount class with attributes for the account holder and balance. It includes methods to deposit money, withdraw money with checks for positive amounts and sufficient funds, and display account information. Example transactions show deposits, successful withdrawals, and handling of insufficient balance with clear messages.

CODE:

Lab-6.py

```python
class BankAccount:
    def __init__(self, account_holder, balance=0.0):
        self.account_holder = account_holder
        self.balance = balance
    def deposit(self, amount):
        if amount <= 0:
            print("Deposit amount must be positive.")
        else:
            self.balance += amount
            print(f"₹{amount} deposited successfully. New balance: ₹{self.balance}")
    def withdraw(self, amount):
        if amount <= 0:
            print("Withdrawal amount must be positive.")
        elif amount > self.balance:
            print("Insufficient funds. Withdrawal denied.")
            print(f"Your current balance is: ₹{self.balance}")
        else:
            self.balance -= amount
            print(f"₹{amount} withdrawn successfully. New balance: ₹{self.balance}")
    def display_info(self):
        print(f"Account Holder: {self.account_holder}")
        print(f"Current Balance: ₹{self.balance}")
# Example usage:
account = BankAccount("ANJALI", 1000)
account.display_info()
account.deposit(500)
account.withdraw(300)
account.withdraw(2000)
```

OUTPUT:

```
/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/sai
te/Downloads/AI ASSISTENT CODING/As6.py/task3.py"
Account Holder: ANJALI
Current Balance: ₹1000
₹500 deposited successfully. New balance: ₹1500
₹300 withdrawn successfully. New balance: ₹1200
Insufficient funds. Withdrawal denied.
Your current balance is: ₹1200
PS C:\Users\saite\Downloads\AI ASSISTENT CODING>
```

JUSTIFICATION: This code defines a BankAccount class with attributes for the account holder and balance. It includes methods to deposit money, withdraw money with checks for positive amounts and sufficient funds, and display account details. The example shows deposits, successful withdrawals, and handling of insufficient balance with clear messages.

Task 4: Student Scholarship Eligibility Check

PROMPT: Write a Python program with a list of student dictionaries containing name and score. Use a while loop to iterate through the list, check if each student's score is greater than 75, and print the names of eligible students for a scholarship.

CODE:

```python
#Write a Python program with a list of student dictionaries containing name and score. Use a while loop to
# List of student dictionaries with name and score
students = [
    {"name": "Alice", "score": 82},
    {"name": "Bob", "score": 70},
    {"name": "Charlie", "score": 90},
    {"name": "Diana", "score": 76},
    {"name": "Evan", "score": 65}
]
i = 0
print("Students eligible for scholarship (score > 75):")
while i < len(students):
    student = students[i]
    # Check if score is greater than 75
    if student["score"] > 75:
        print(student["name"])
    i += 1
```

OUTPUT:

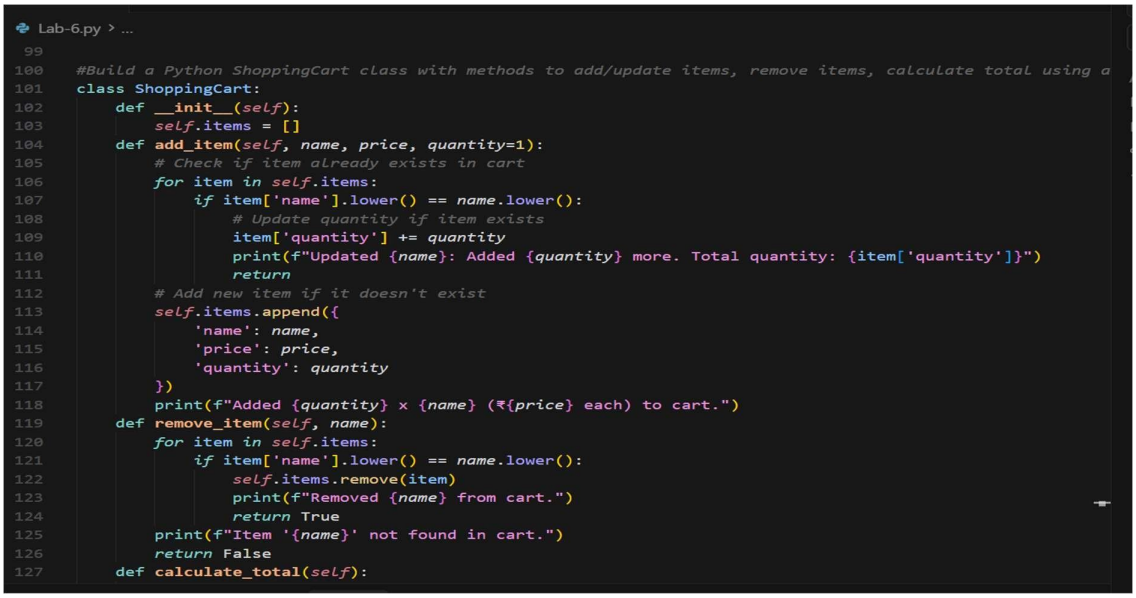
```
PS C:\Users\saite\Downloads\AI ASSISTENT CODING> & C:/Users/saite
/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/sai
te/Downloads/AI ASSISTENT CODING/As6.py/task4.py"
Students eligible for scholarship (score > 75):
Alice
Charlie
Diana
PS C:\Users\saite\Downloads\AI ASSISTENT CODING>
```

JUSTIFICATION: This code stores students in a list of dictionaries with names and scores. Using a while loop, it checks each student's score, and if it is greater than 75, prints the student's name as eligible for a scholarship.

Task 5: Online Shopping Cart Module

PROMPT: Build a Python ShoppingCart class with methods to add/update items, remove items, calculate total using a loop, apply 10% discount if total exceeds ₹1000, display cart in a table, and run a checkout demo with sample input/output.

CODE:



```
Lab-6.py > ...
 99
100  #Build a Python ShoppingCart class with methods to add/update items, remove items, calculate total using a
101  class ShoppingCart:
102      def __init__(self):
103          self.items = []
104      def add_item(self, name, price, quantity=1):
105          # Check if item already exists in cart
106          for item in self.items:
107              if item['name'].lower() == name.lower():
108                  # Update quantity if item exists
109                  item['quantity'] += quantity
110                  print(f"Updated {name}: Added {quantity} more. Total quantity: {item['quantity']}")
111                  return
112          # Add new item if it doesn't exist
113          self.items.append({
114              'name': name,
115              'price': price,
116              'quantity': quantity
117          })
118          print(f"Added {quantity} x {name} (₹{price} each) to cart.")
119      def remove_item(self, name):
120          for item in self.items:
121              if item['name'].lower() == name.lower():
122                  self.items.remove(item)
123                  print(f"Removed {name} from cart.")
124                  return True
125          print(f"Item '{name}' not found in cart.")
126          return False
127      def calculate_total(self):
```

```
Lab-6.py > ...
101  class ShoppingCart:
127      def calculate_total(self):
128          total = 0.0
129          # Loop through all items to calculate total
130          for item in self.items:
131              item_total = item['price'] * item['quantity']
132              total += item_total
133          return total
134      def apply_discount(self, total):
135          if total > 1000:
136              discount = total * 0.10  # 10% discount
137              discounted_total = total - discount
138              print(f"\n🎉 Discount Applied! (10% off on orders above ₹1000)")
139              print(f"Original Total: ₹{total:.2f}")
140              print(f"Discount Amount: ₹{discount:.2f}")
141              print(f"Final Total: ₹{discounted_total:.2f}")
142              return discounted_total
143          else:
144              print(f"\nTotal Amount: ₹{total:.2f}")
145              print("(No discount applied. Spend ₹1000+ to get 10% off!)")
146              return total
147      def display_cart(self):
148          if not self.items:
149              print("\nYour cart is empty!")
150              return
151          print("\n" + "="*50)
152          print("SHOPPING CART")
153          print("="*50)
154          print(f"{'Item':<20} {'Price':<15} {'Quantity':<10} {'Subtotal':<10}")
```

Lab-6.py > ...

```
101  class ShoppingCart:
147      def display_cart(self):
155          print("-"*50)
156          for item in self.items:
157              subtotal = item['price'] * item['quantity']
158              print(f"{item['name']:<20} ₹{item['price']:<14.2f} {item['quantity']:<10} ₹{subtotal:<10.2f}")
159          print("-"*50)
160      def checkout(self):
161          self.display_cart()
162          total = self.calculate_total()
163          final_total = self.apply_discount(total)
164          return final_total
165  # Demonstration with sample input and output
166  if __name__ == "__main__":
167      print("="*50)
168      print("SHOPPING CART DEMONSTRATION")
169      print("="*50)
170      # Create a new shopping cart
171      cart = ShoppingCart()
172      # Add items to the cart
173      print("\n--- Adding Items to Cart ---")
174      cart.add_item("Laptop", 45000, 1)
175      cart.add_item("Mouse", 500, 2)
176      cart.add_item("Keyboard", 1500, 1)
177      cart.add_item("Monitor", 12000, 1)
178      cart.add_item("Mouse", 500, 1)  # Adding more of existing item
179      # Display cart contents
180      print("\n--- Cart Contents ---")
181      cart.display_cart()
```

Lab-6.py > ...

```
182      # Calculate and apply discount
183      print("\n--- Checkout ---")
184      final_amount = cart.checkout()
185      # Remove an item and recalculate
186      print("\n--- Removing Item ---")
187      cart.remove_item("Mouse")
188      print("\n--- Updated Cart ---")
189      cart.display_cart()
190      print("\n--- Updated Checkout ---")
191      final_amount = cart.checkout()
192      # Demonstrate with smaller cart (no discount)
193      print("\n\n" + "="*50)
194      print("SMALL CART DEMONSTRATION (No Discount)")
195      print("="*50)
196      small_cart = ShoppingCart()
197      small_cart.add_item("Pen", 10, 5)
198      small_cart.add_item("Notebook", 50, 2)
199      small_cart.display_cart()
200      small_cart.checkout()
```

OUTPUT:

```
==================================================
SHOPPING CART DEMONSTRATION
==================================================

--- Adding Items to Cart ---
Added 1 x Laptop (₹45000 each) to cart.
Added 2 x Mouse (₹500 each) to cart.
Added 1 x Keyboard (₹1500 each) to cart.
Added 1 x Monitor (₹12000 each) to cart.
Updated Mouse: Added 1 more. Total quantity: 3

--- Cart Contents ---

==================================================
SHOPPING CART
==================================================
Item                 Price            Quantity   Subtotal
--------------------------------------------------
Laptop               ₹45000.00        1          ₹45000.00
Mouse                ₹500.00          3          ₹1500.00
```

```
Monitor                 ₹12000.00       1           ₹12000.00
--------------------------------------------------

--- Checkout ---

==================================================
SHOPPING CART
==================================================
Item                    Price           Quantity    Subtotal
--------------------------------------------------
Laptop                  ₹45000.00       1           ₹45000.00
Mouse                   ₹500.00         3           ₹1500.00
Keyboard                ₹1500.00        1           ₹1500.00
Monitor                 ₹12000.00       1           ₹12000.00
--------------------------------------------------

🎉 Discount Applied! (10% off on orders above ₹1000)
Original Total: ₹60000.00
Discount Amount: ₹6000.00
Final Total: ₹54000.00

--- Removing Item ---
```

```
Removed Mouse from cart.

--- Updated Cart ---

==================================================
SHOPPING CART
==================================================
Item                    Price           Quantity    Subtotal
--------------------------------------------------
Laptop                  ₹45000.00       1           ₹45000.00
Keyboard                ₹1500.00        1           ₹1500.00
Monitor                 ₹12000.00       1           ₹12000.00
--------------------------------------------------

--- Updated Checkout ---

==================================================
SHOPPING CART
==================================================
Item                    Price           Quantity    Subtotal
--------------------------------------------------
Laptop                  ₹45000.00       1           ₹45000.00
Keyboard                ₹1500.00        1           ₹1500.00
Monitor                 ₹12000.00       1           ₹12000.00
--------------------------------------------------

🎉 Discount Applied! (10% off on orders above ₹1000)
Original Total: ₹58500.00
Discount Amount: ₹5850.00
Final Total: ₹52650.00


==================================================
SMALL CART DEMONSTRATION (No Discount)
==================================================
Added 5 x Pen (₹10 each) to cart.
Added 2 x Notebook (₹50 each) to cart.

==================================================
SHOPPING CART
==================================================
Item                    Price           Quantity    Subtotal
--------------------------------------------------
```

```
==================================================
Item                 Price          Quantity   Subtotal
--------------------------------------------------
Pen                  ₹10.00         5          ₹50.00
Notebook             ₹50.00         2          ₹100.00
--------------------------------------------------

==================================================
SHOPPING CART
==================================================
Item                 Price          Quantity   Subtotal
--------------------------------------------------
Pen                  ₹10.00         5          ₹50.00
Notebook             ₹50.00         2          ₹100.00
--------------------------------------------------

Total Amount: ₹150.00
(No discount applied. Spend ₹1000+ to get 10% off!)
PS C:\Users\saite\Downloads\AI ASSISTENT CODING>
```

JUSTIFICATION: This code defines a ShoppingCart class that lets you add or update items, remove them, calculate the total, apply a 10% discount if the bill exceeds ₹1000, display the cart in a table, and checkout. The demo shows adding items, updating quantities, removing items, and checking out with and without discounts.