ASSIGNMENT-6.4

2303A51546

Batch-10

TASK-1:

Prompt:

Create a method to display the student's name, roll number, and marks in a neat format
Write a method that takes class average as input

Use if-else to compare student's marks with the class average

Return a message saying whether the student performed above or below average

CODE:

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

    def display_info(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")

    def compare_with_average(self, class_average):
        if self.marks > class_average:
            return f"{self.name} performed above average."
        elif self.marks < class_average:
            return f"{self.name} performed below average."
        else:
            return f"{self.name} performed exactly at the average."
```

Example usage

student1 = Student("Alice", 1, 85)

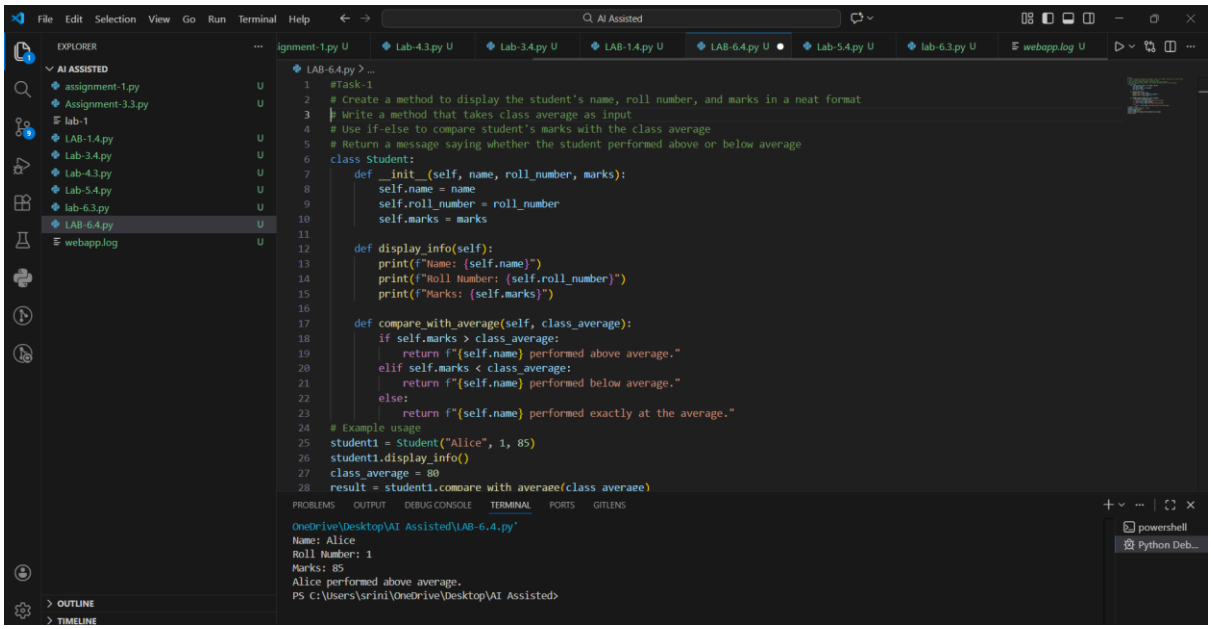
student1.display_info()

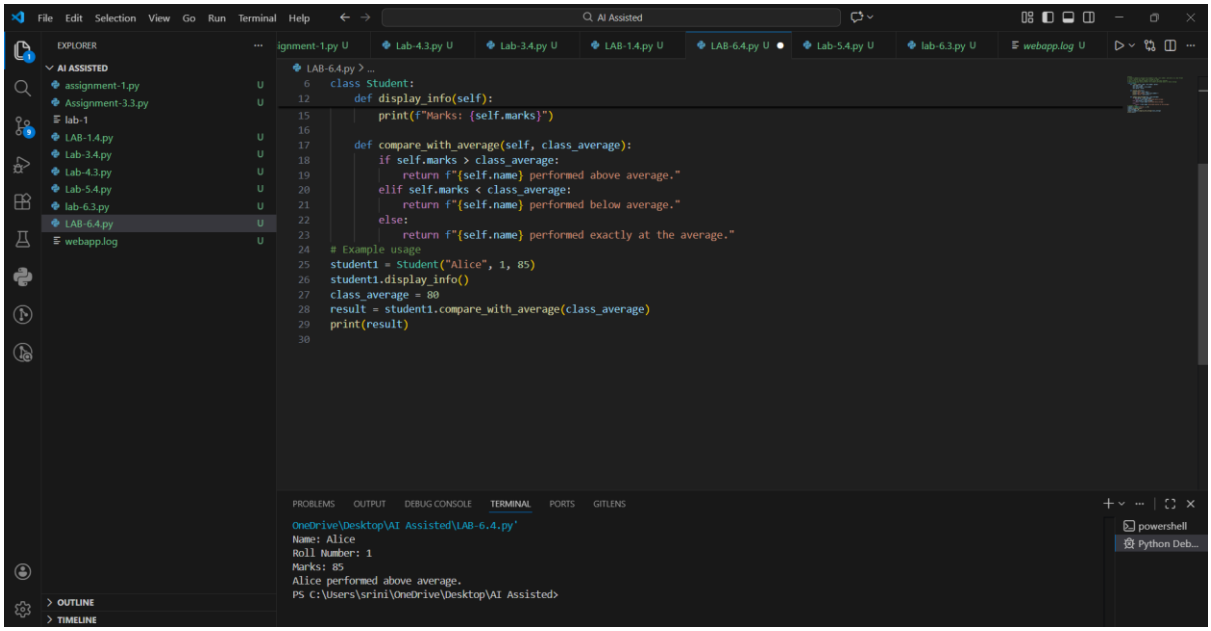
class_average = 80

result = student1.compare_with_average(class_average)

print(result)

Output:





Analysis:

In this task, AI helped to complete the methods inside the Student class.
The display_info() method prints the student's name, roll number, and marks using self, which shows correct use of class attributes.

The compare_with_average() method uses if-else conditions to check whether the student scored above or below the class average. The messages returned are clear and meaningful. The AI-generated code is correct and easy to understand.

TASK-2

Prompt:

Check if the current reading is an even number using modulus operator

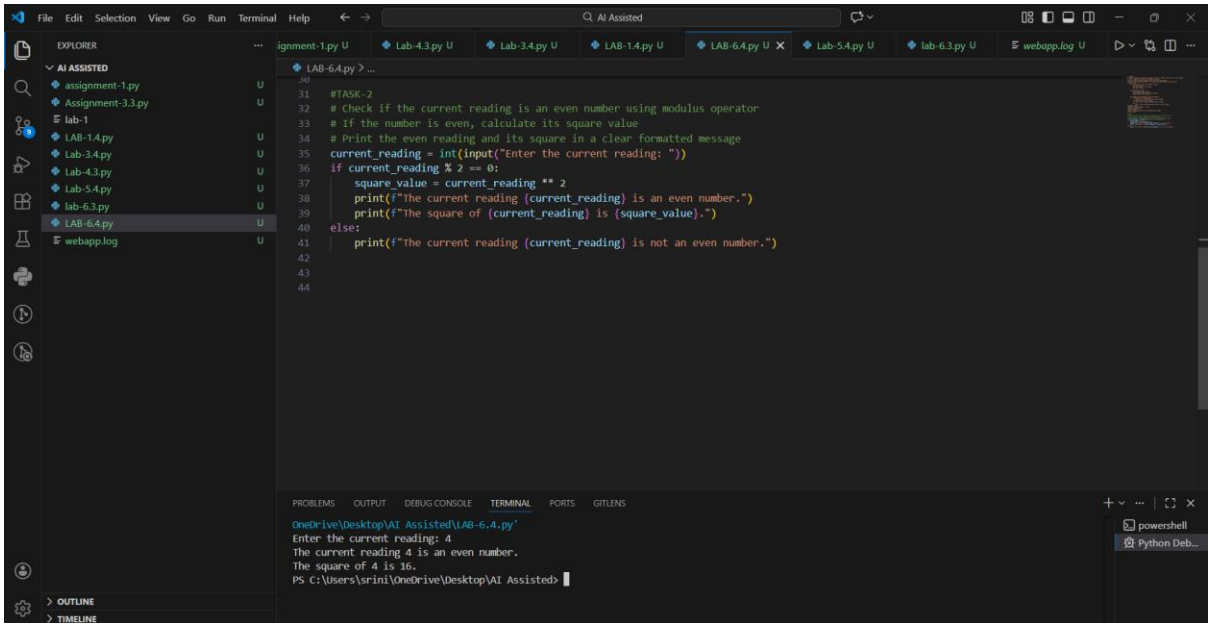
If the number is even, calculate its square value

Print the even reading and its square in a clear formatted message

CODE:

```
current_reading = int(input("Enter the current reading: "))

if current_reading % 2 == 0:

    square_value = current_reading ** 2

    print(f"The current reading {current_reading} is an even number.")

    print(f"The square of {current_reading} is {square_value}.")

else:

    print(f"The current reading {current_reading} is not an even number.")
```

Output:



Analysis:

In this task, AI helped write logic to check whether a number is even. It used the modulus operator (%) correctly. If the number is even, the program calculates the square and prints it.

The condition works properly for both even and odd numbers. The output format is also clear. This shows that AI can correctly generate basic loop/conditional logic.

TASK-3:

Prompt:

Create a method to deposit money into the account

Add the amount to balance and print the updated balance

Create a method to withdraw money from the account

Use if-else to check if there is enough balance before withdrawing

If balance is insufficient, show a proper warning message

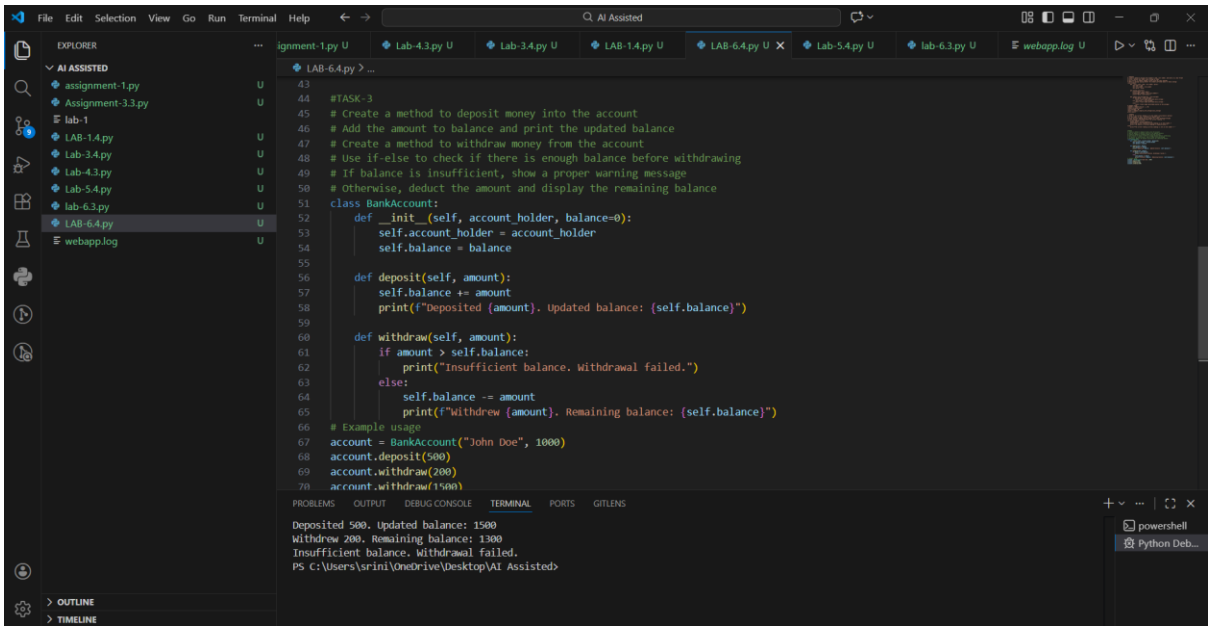
Otherwise, deduct the amount and display the remaining balance

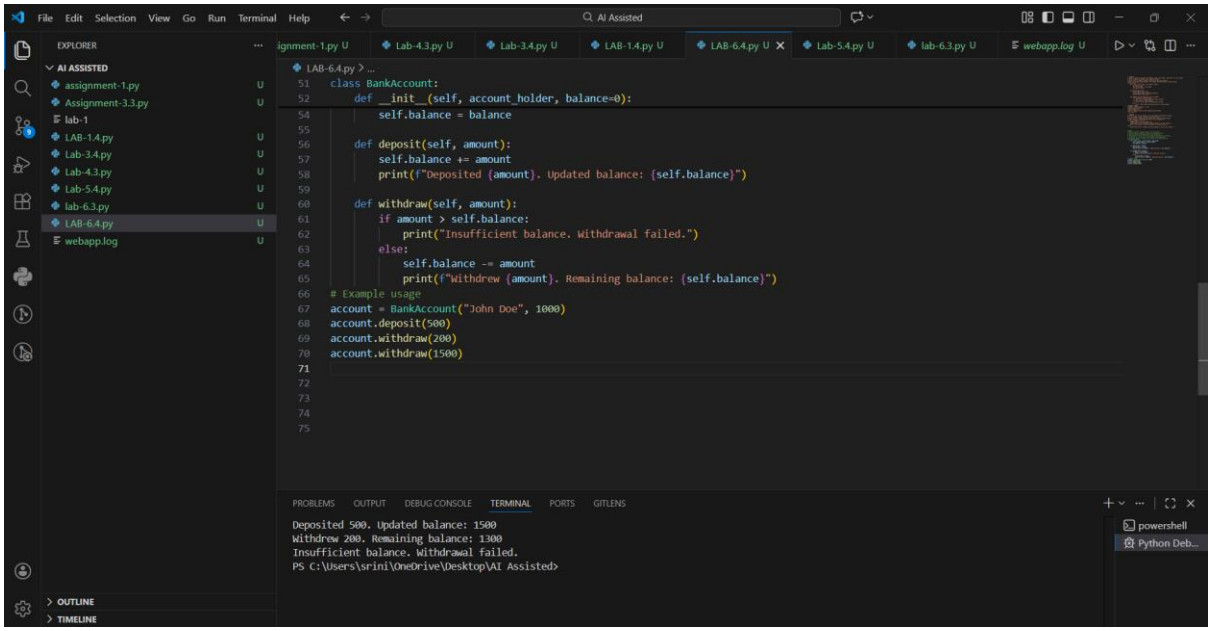
CODE:

```
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance
```

```
    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited {amount}. Updated balance: {self.balance}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance. Withdrawal failed.")
        else:
            self.balance -= amount
            print(f"Withdrew {amount}. Remaining balance: {self.balance}")
# Example usage
account = BankAccount("John Doe", 1000)
account.deposit(500)
account.withdraw(200)
account.withdraw(1500)
```

Output:





Analysis:

Here, AI completed the BankAccount class by adding deposit and withdraw methods. The deposit method adds money to the balance, and the withdraw method checks if enough balance is available before subtracting money.

If the balance is low, it shows an error message. This shows correct use of if-else statements and class variables. The AI-generated solution is practical and works correctly.

TASK-4:

Prompt:

Use a while loop to iterate through each student in the list

Check if the student's score is greater than 75

If eligible, print the student's name with a scholarship eligibility message

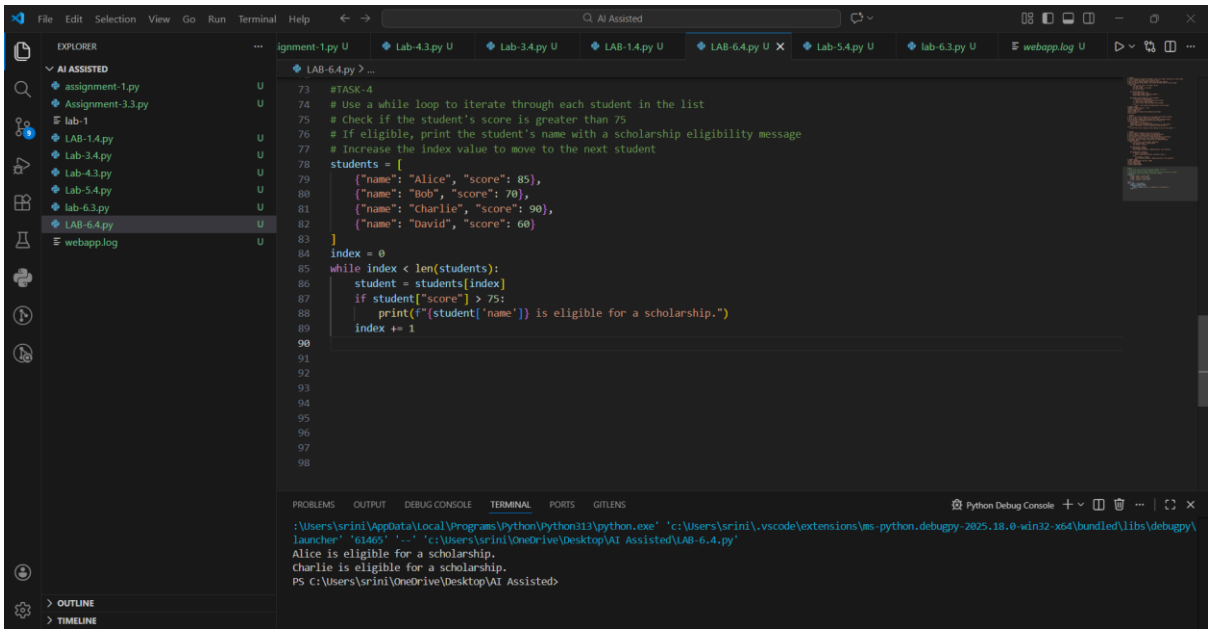
Increase the index value to move to the next student

CODE:

```
students = [
    {"name": "Alice", "score": 85},
    {"name": "Bob", "score": 70},
    {"name": "Charlie", "score": 90},
    {"name": "David", "score": 60}
```

```
]
index = 0
while index < len(students):
    student = students[index]
    if student["score"] > 75:
        print(f"{student['name']} is eligible for a scholarship.")
    index += 1
```

Output:



Analysis:

In this task, AI helped generate a while loop that goes through a list of students. It checks each student's score and prints the names of students who scored above 75.

The loop control using index and the condition check are correct. The output is clear and easy to read. This shows AI can generate loop logic with conditions properly.

TASK-5:

Prompt:

Write a method to add an item to the cart with name, price, and quantity

Write a method to remove an item from the cart by matching the item name

Create a method to calculate the total bill by looping through all items

Multiply price and quantity for each item and add to total amount

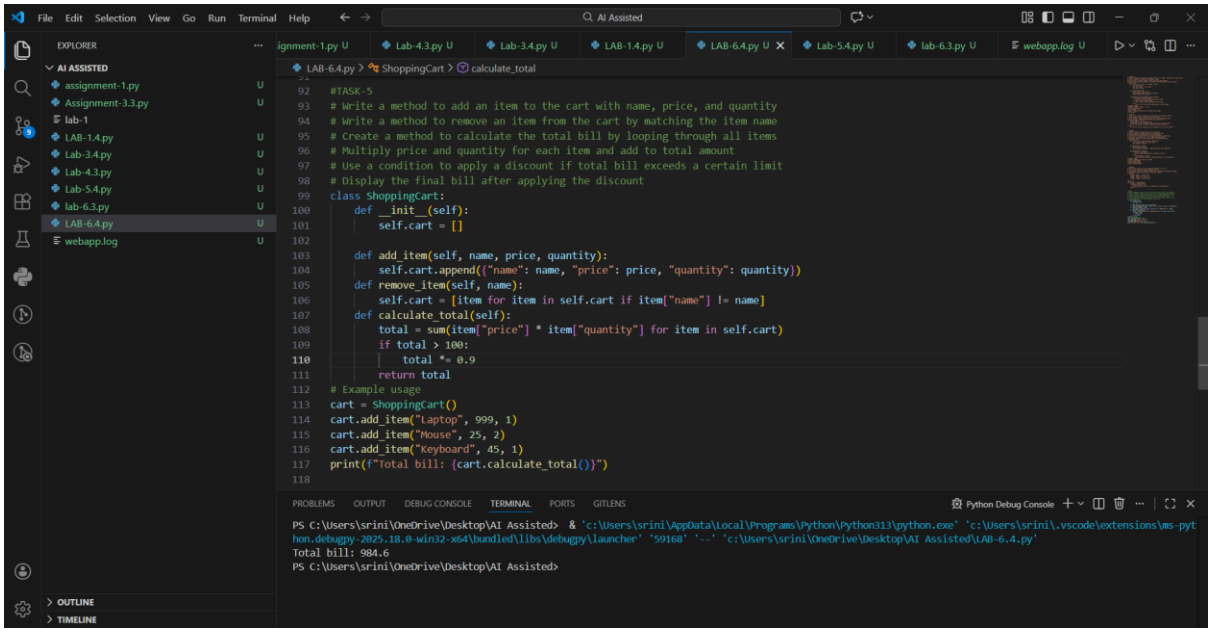
Use a condition to apply a discount if total bill exceeds a certain limit

Display the final bill after applying the discount

CODE:

```
class ShoppingCart:
    def __init__(self):
        self.cart = []

    def add_item(self, name, price, quantity):
        self.cart.append({"name": name, "price": price, "quantity": quantity})
    def remove_item(self, name):
        self.cart = [item for item in self.cart if item["name"] != name]
    def calculate_total(self):
        total = sum(item["price"] * item["quantity"] for item in self.cart)
        if total > 100:
            total *= 0.9
        return total
# Example usage
cart = ShoppingCart()
cart.add_item("Laptop", 999, 1)
cart.add_item("Mouse", 25, 2)
cart.add_item("Keyboard", 45, 1)
print(f"Total bill: {cart.calculate_total()}")
```

Output:



Analysis:

For this task, AI generated methods for adding items, removing items, and calculating the total bill. It used a loop to calculate the total price based on item price and quantity.

AI also added a discount condition when the total exceeds a limit. This shows correct use of lists, loops, and if conditions. The shopping cart system works correctly and simulates a real example.