AI Assisted Coding LAB:-6.4

NAME:- K.SAI RAHUL

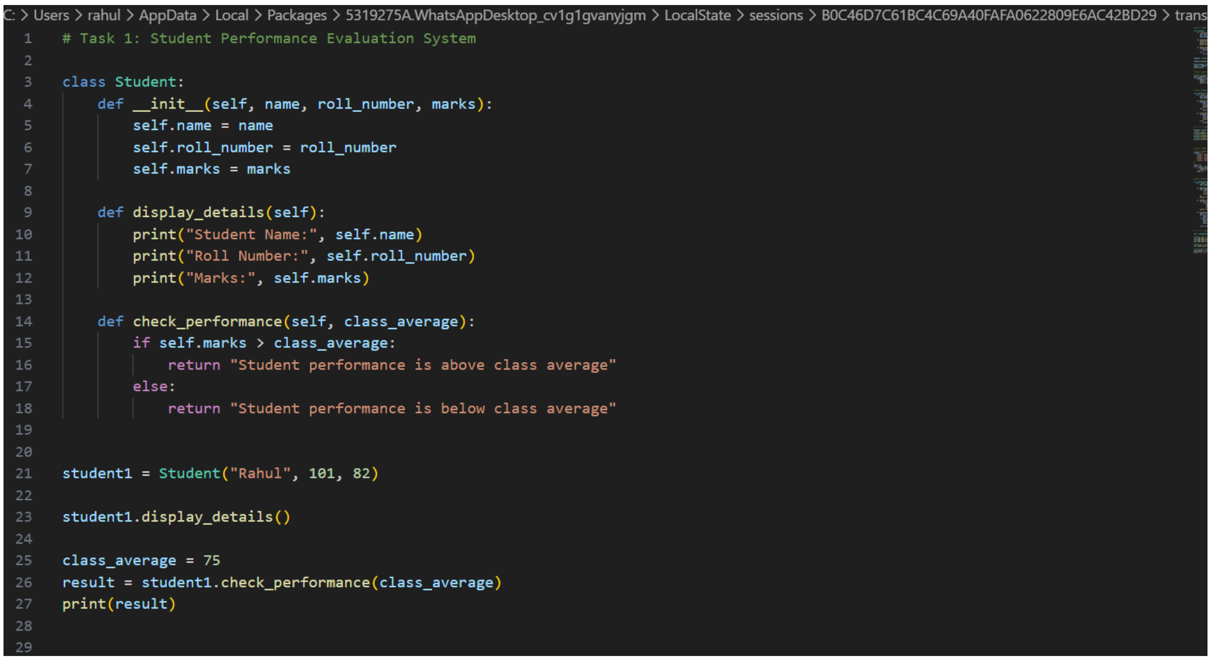
H.T.NO:- 2303A52392

BATCH – 43

==================================

⇨ TASK 1:-

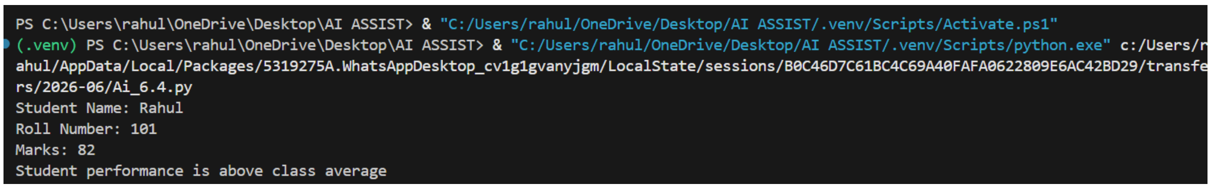
==================================



C: > Users > rahul > AppData > Local > Packages > 5319275A.WhatsAppDesktop_cv1g1gvanyjgm > LocalState > sessions > B0C46D7C61BC4C69A40FAFA0622809E6AC42BD29 > trans

```python
# Task 1: Student Performance Evaluation System

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("Student Name:", self.name)
        print("Roll Number:", self.roll_number)
        print("Marks:", self.marks)

    def check_performance(self, class_average):
        if self.marks > class_average:
            return "Student performance is above class average"
        else:
            return "Student performance is below class average"


student1 = Student("Rahul", 101, 82)

student1.display_details()

class_average = 75
result = student1.check_performance(class_average)
print(result)

```

OUTPUT:-



```
PS C:\Users\rahul\OneDrive\Desktop\AI ASSIST> & "C:/Users/rahul/OneDrive/Desktop/AI ASSIST/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\rahul\OneDrive\Desktop\AI ASSIST> & "C:/Users/rahul/OneDrive/Desktop/AI ASSIST/.venv/Scripts/python.exe" c:/Users/r
ahul/AppData/Local/Packages/5319275A.WhatsAppDesktop_cv1g1gvanyjgm/LocalState/sessions/B0C46D7C61BC4C69A40FAFA0622809E6AC42BD29/transfe
rs/2026-06/Ai_6.4.py
Student Name: Rahul
Roll Number: 101
Marks: 82
Student performance is above class average
```

Summary :-

This task focused on building a basic academic evaluation system using a Python class.
A Student class was created with attributes name, roll_number, and marks.
AI-based code completion was used to generate methods for:

- Displaying student details
- Evaluating student performance using conditional logic

The AI successfully generated:

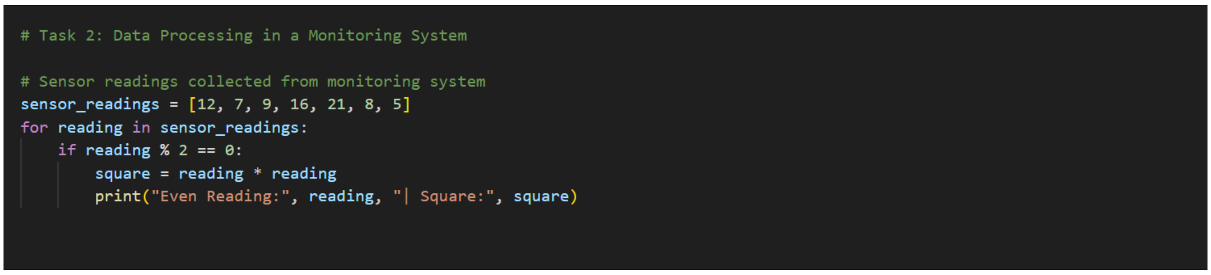
- Proper use of self for class attributes
- if-else conditions for performance checking
- Clean and readable method structures

This task demonstrated how AI can assist in building structured class definitions and logic-based methods efficiently.

==================================

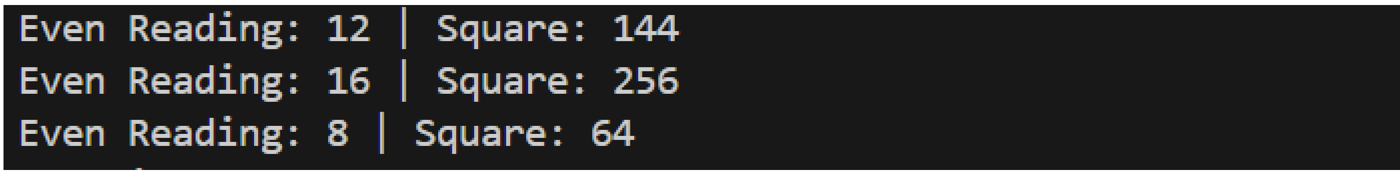
⇨ TASK 2:-

==================================



```python
# Task 2: Data Processing in a Monitoring System

# Sensor readings collected from monitoring system
sensor_readings = [12, 7, 9, 16, 21, 8, 5]
for reading in sensor_readings:
    if reading % 2 == 0:
        square = reading * reading
        print("Even Reading:", reading, "| Square:", square)
```

OUTPUT:-



```
Even Reading: 12 | Square: 144
Even Reading: 16 | Square: 256
Even Reading: 8 | Square: 64
```

Summary:-

This task involved processing numerical sensor data using loops and conditions.
A for loop was initiated manually, and AI-based completion was guided using comments.

The AI generated logic to:

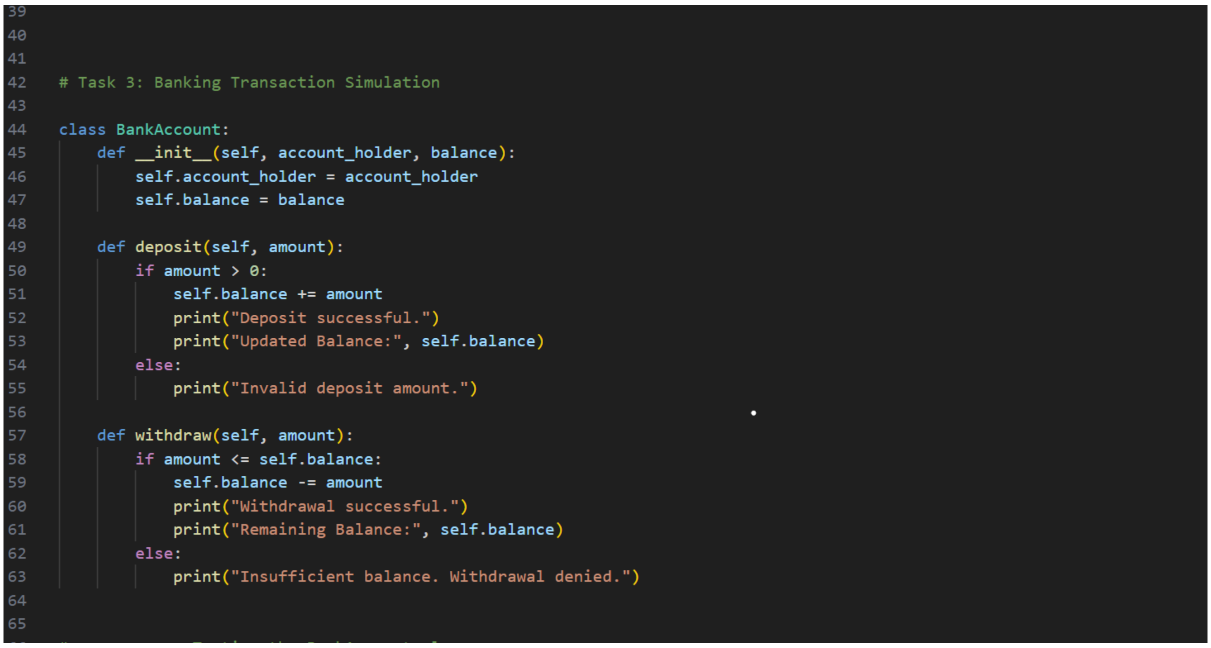
- Identify even numbers using the modulus (%) operator
- Calculate the square of even values
- Print results in a readable format

This task demonstrated AI's ability to understand loop structures, conditional logic, and mathematical operations through prompt-driven guidance.

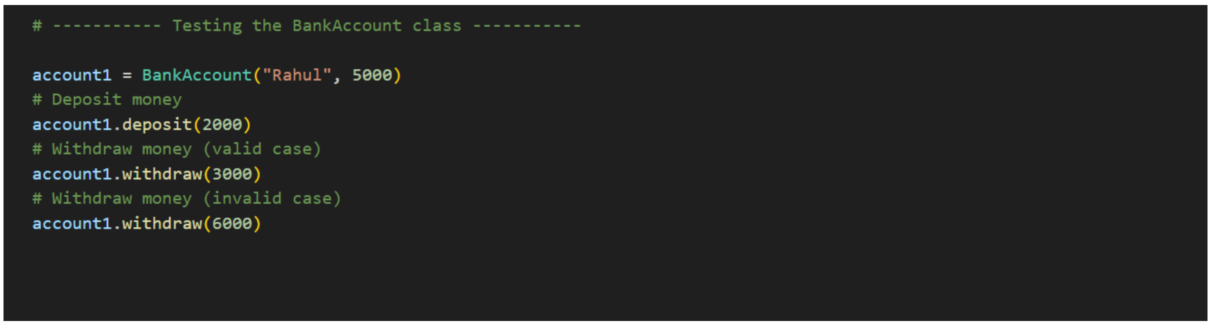
==

⇨ TASK 3:-

==

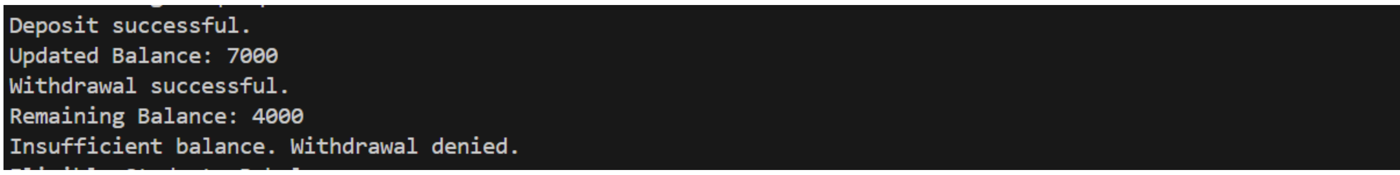


```python


# Task 3: Banking Transaction Simulation

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

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print("Deposit successful.")
            print("Updated Balance:", self.balance)
        else:
            print("Invalid deposit amount.")

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawal successful.")
            print("Remaining Balance:", self.balance)
        else:
            print("Insufficient balance. Withdrawal denied.")


```



```python
# ---------- Testing the BankAccount class ----------

account1 = BankAccount("Rahul", 5000)
# Deposit money
account1.deposit(2000)
# Withdraw money (valid case)
account1.withdraw(3000)
# Withdraw money (invalid case)
account1.withdraw(6000)
```

OUTPUT:-



```
Deposit successful.
Updated Balance: 7000
Withdrawal successful.
Remaining Balance: 4000
Insufficient balance. Withdrawal denied.
```

Summary:-

In this task, a BankAccount class was created to simulate real-world banking operations. The class included attributes for account_holder and balance.

AI-assisted code completion generated methods for:

- Depositing money
- Withdrawing money
- Preventing withdrawals when balance is insufficient

The AI used:

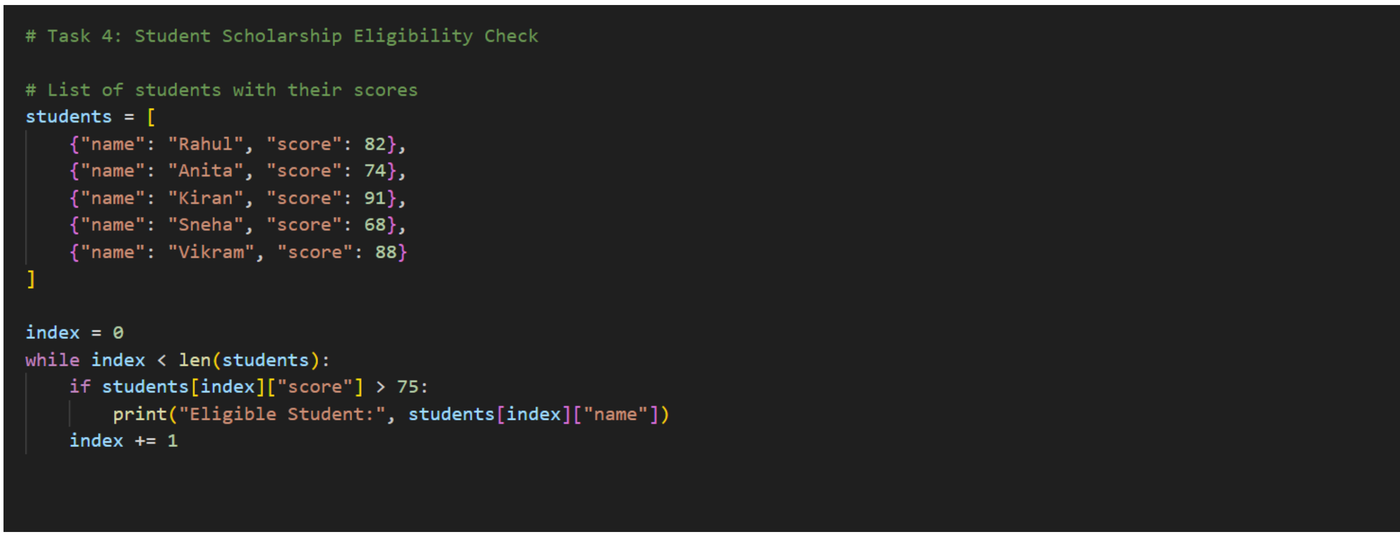
- Conditional statements (if-else)
- Class attributes via self
- User-friendly output messages

This task showed how AI can help implement real-life business logic safely and logically.

================================

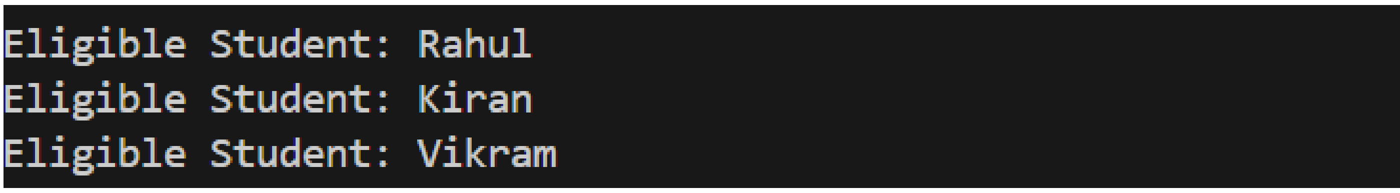
⇨ TASK 4:-

================================



```python
# Task 4: Student Scholarship Eligibility Check

# List of students with their scores
students = [
    {"name": "Rahul", "score": 82},
    {"name": "Anita", "score": 74},
    {"name": "Kiran", "score": 91},
    {"name": "Sneha", "score": 68},
    {"name": "Vikram", "score": 88}
]

index = 0
while index < len(students):
    if students[index]["score"] > 75:
        print("Eligible Student:", students[index]["name"])
    index += 1
```

Output:-



```
Eligible Student: Rahul
Eligible Student: Kiran
Eligible Student: Vikram
```

Summary:-

This task focused on using loops and conditionals with structured data.
A list of dictionaries was created to store student names and scores.

AI-based completion was used to generate a while loop that:

- Iterates through the list
- Checks eligibility using conditions
- Prints names of students scoring more than 75

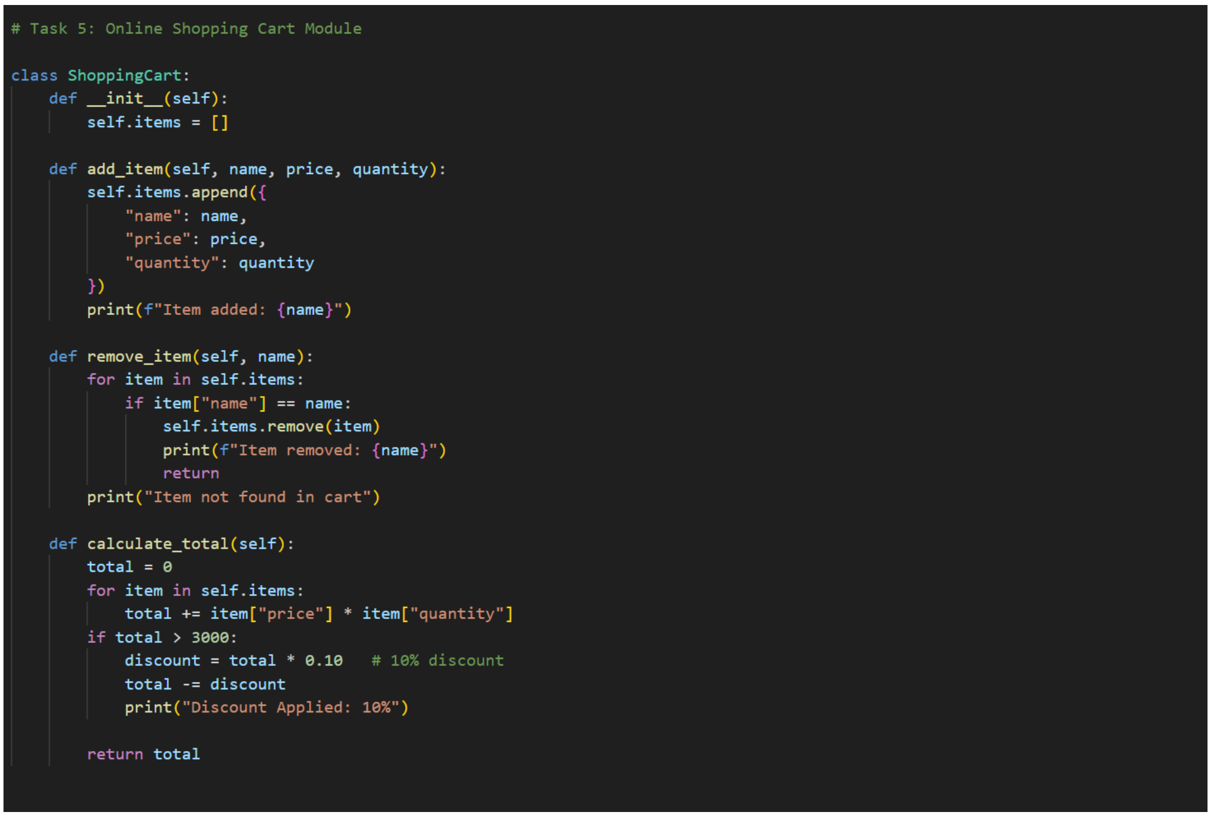
The task demonstrated AI's ability to manage:

- Index-based iteration
- Data structure access
- Logical filtering using conditions

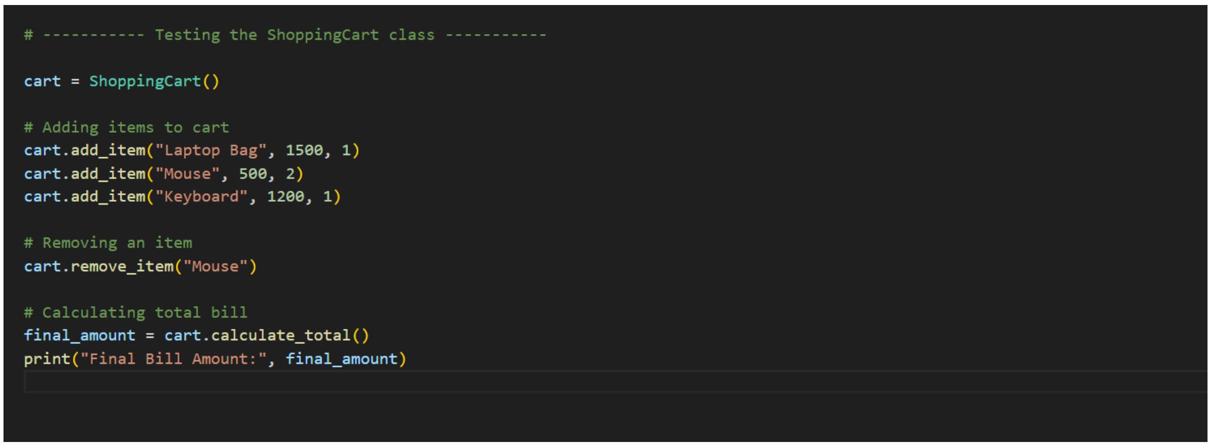
=================================

⇨ TASK 5:-

=================================

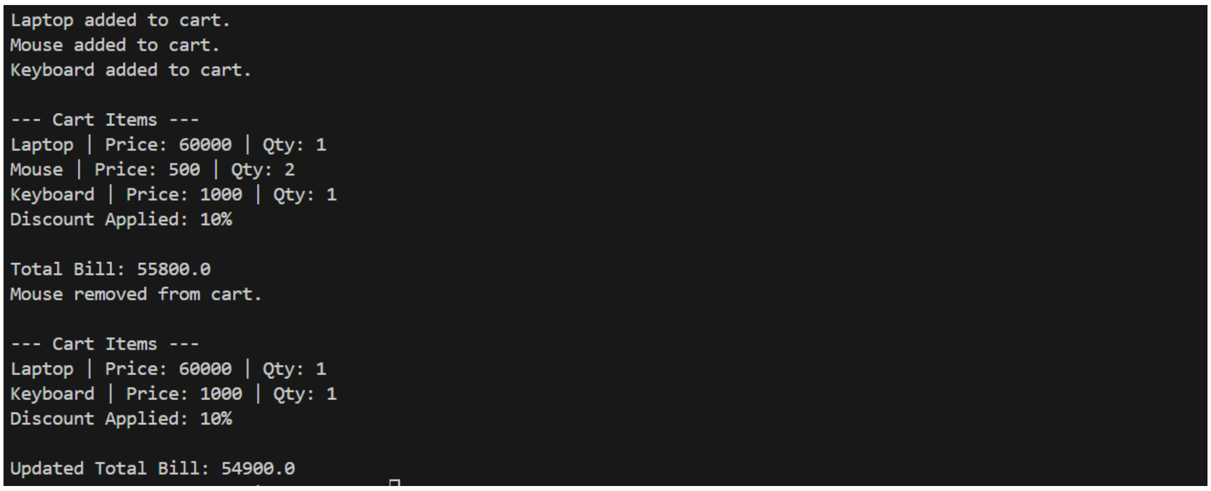


```python
# Task 5: Online Shopping Cart Module

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

    def add_item(self, name, price, quantity):
        self.items.append({
            "name": name,
            "price": price,
            "quantity": quantity
        })
        print(f"Item added: {name}")

    def remove_item(self, name):
        for item in self.items:
            if item["name"] == name:
                self.items.remove(item)
                print(f"Item removed: {name}")
                return
        print("Item not found in cart")

    def calculate_total(self):
        total = 0
        for item in self.items:
            total += item["price"] * item["quantity"]
        if total > 3000:
            discount = total * 0.10   # 10% discount
            total -= discount
            print("Discount Applied: 10%")

        return total
```



```python
# ----------- Testing the ShoppingCart class -----------

cart = ShoppingCart()

# Adding items to cart
cart.add_item("Laptop Bag", 1500, 1)
cart.add_item("Mouse", 500, 2)
cart.add_item("Keyboard", 1200, 1)

# Removing an item
cart.remove_item("Mouse")

# Calculating total bill
final_amount = cart.calculate_total()
print("Final Bill Amount:", final_amount)
```

Output:-



```
Laptop added to cart.
Mouse added to cart.
Keyboard added to cart.

--- Cart Items ---
Laptop | Price: 60000 | Qty: 1
Mouse | Price: 500 | Qty: 2
Keyboard | Price: 1000 | Qty: 1
Discount Applied: 10%

Total Bill: 55800.0
Mouse removed from cart.

--- Cart Items ---
Laptop | Price: 60000 | Qty: 1
Keyboard | Price: 1000 | Qty: 1
Discount Applied: 10%

Updated Total Bill: 54900.0
```

Summary:-

This task involved building a simplified e-commerce shopping cart system.
A ShoppingCart class was created with an empty list to store cart items.

AI-assisted completion generated methods for:

- Adding items to the cart
- Removing items from the cart
- Calculating total bill using loops
- Applying conditional discounts

The AI correctly implemented:

- Loop-based total calculation
- Conditional discount logic
- Data structure handling
- Object-oriented design principles

This task demonstrated AI’s effectiveness in building scalable, real-world application logic.