AI-ASSISSTANT-CODING-LAB-6.4

Name: G. Kruthik Roshan

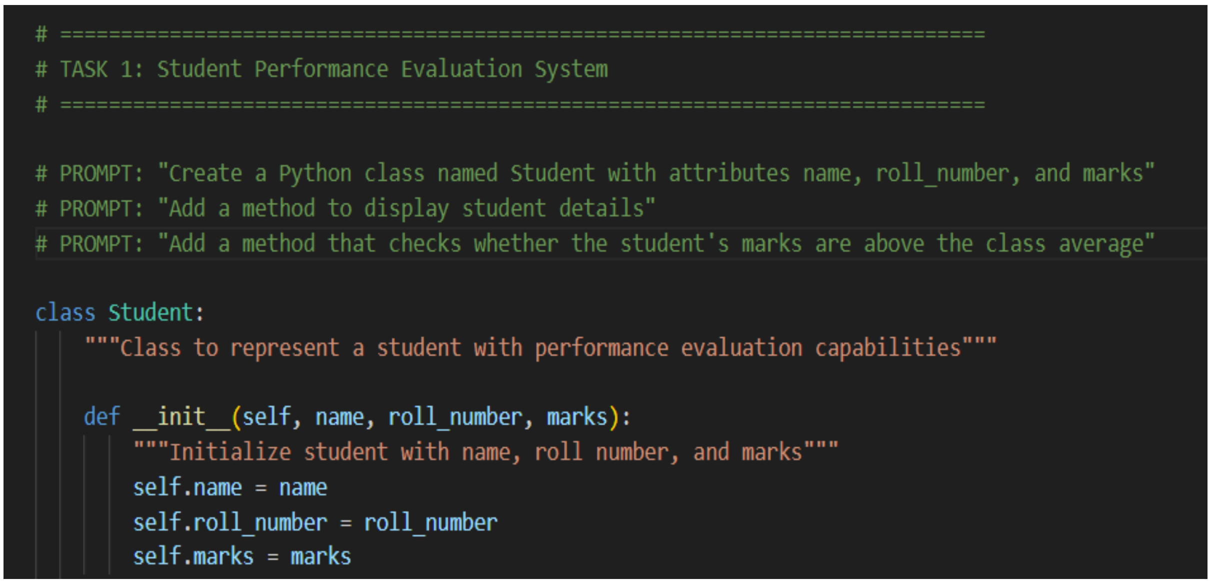
Batch:41

Roll-No:2303A52339

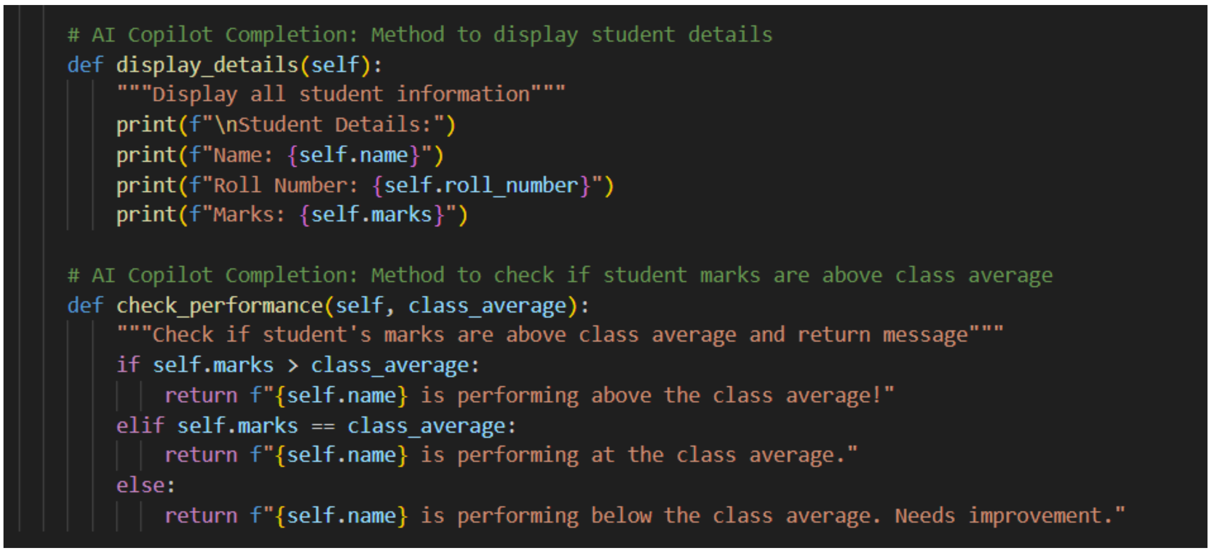
Task 1: Student Performance Evaluation System
Scenario
You are building a simple academic management module for a university system where student performance needs to be evaluated automatically.
Task Description
Create the skeleton of a Python class named Student with the attributes:
• name
• roll_number
• marks
Write only the class definition and attribute initialization.
Then, using GitHub Copilot, prompt the tool to complete:
• A method to display student details
• A method that checks whether the student's marks are above the class average and returns an appropriate message

PROMPT: Create a Python class named Student with attributes name, roll_number, and marks.Add a method to display student details.Add a method that checks whether the student's marks are above the class average

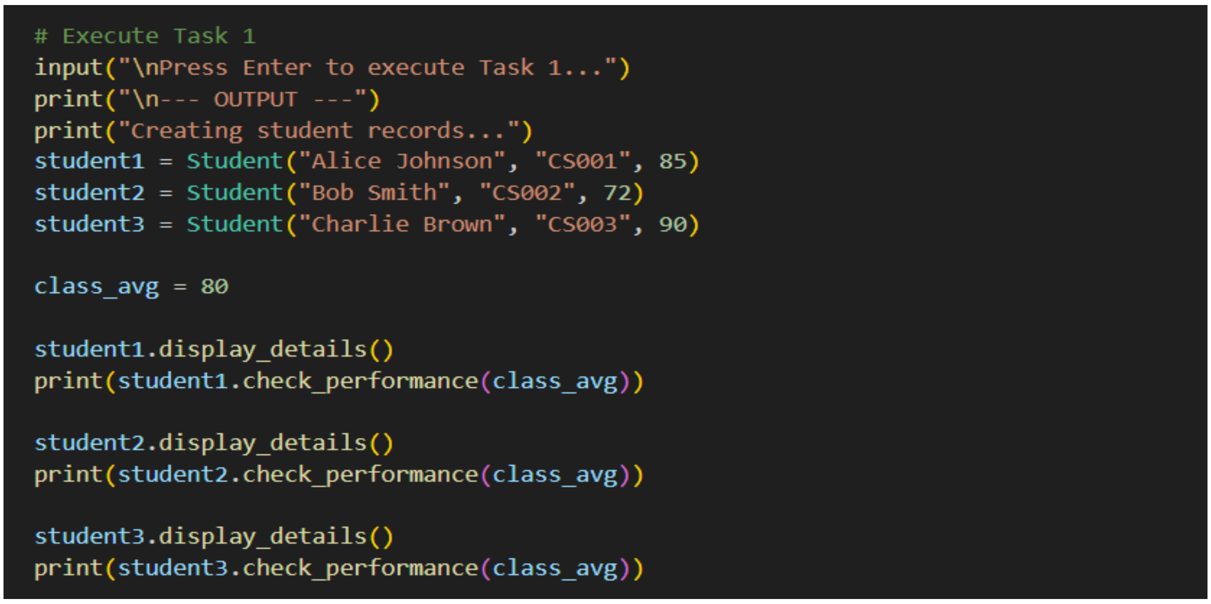
CODE:



```
# ============================================================================
# TASK 1: Student Performance Evaluation System
# ============================================================================

# PROMPT: "Create a Python class named Student with attributes name, roll_number, and marks"
# PROMPT: "Add a method to display student details"
# PROMPT: "Add a method that checks whether the student's marks are above the class average"

class Student:
    """Class to represent a student with performance evaluation capabilities"""

    def __init__(self, name, roll_number, marks):
        """Initialize student with name, roll number, and marks"""
        self.name = name
        self.roll_number = roll_number
        self.marks = marks
```

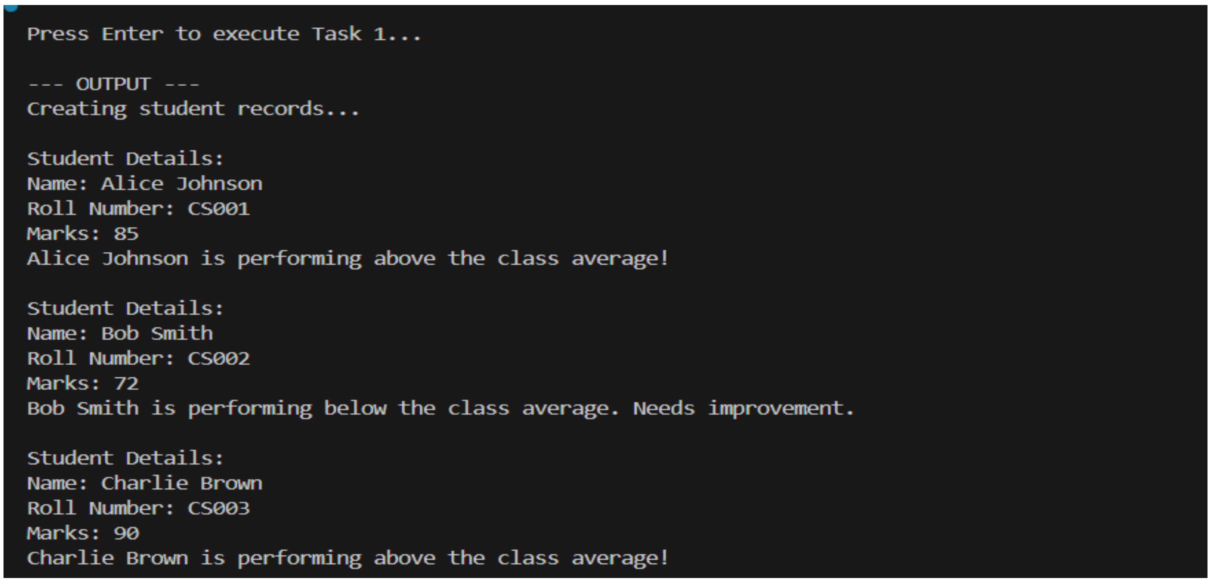


```python
    # AI Copilot Completion: Method to display student details
    def display_details(self):
        """Display all student information"""
        print(f"\nStudent Details:")
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")

    # AI Copilot Completion: Method to check if student marks are above class average
    def check_performance(self, class_average):
        """Check if student's marks are above class average and return message"""
        if self.marks > class_average:
            return f"{self.name} is performing above the class average!"
        elif self.marks == class_average:
            return f"{self.name} is performing at the class average."
        else:
            return f"{self.name} is performing below the class average. Needs improvement."
```



```python
# Execute Task 1
input("\nPress Enter to execute Task 1...")
print("\n--- OUTPUT ---")
print("Creating student records...")
student1 = Student("Alice Johnson", "CS001", 85)
student2 = Student("Bob Smith", "CS002", 72)
student3 = Student("Charlie Brown", "CS003", 90)

class_avg = 80

student1.display_details()
print(student1.check_performance(class_avg))

student2.display_details()
print(student2.check_performance(class_avg))

student3.display_details()
print(student3.check_performance(class_avg))
```

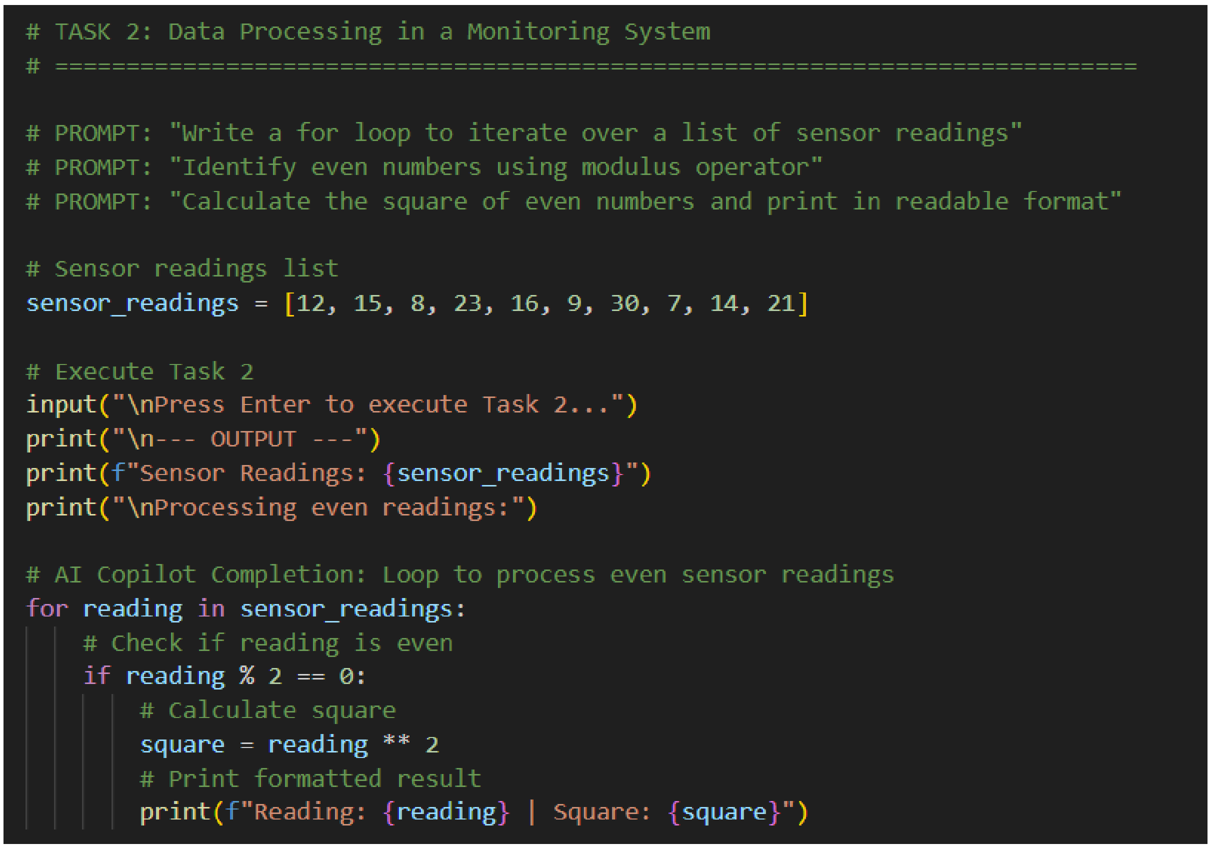
OUTPUT:



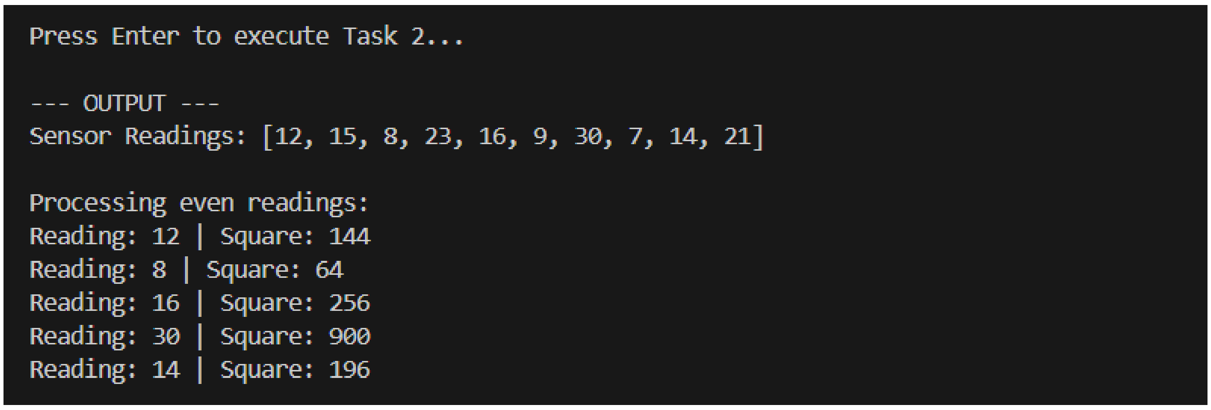
```
Press Enter to execute Task 1...

--- OUTPUT ---
Creating student records...

Student Details:
Name: Alice Johnson
Roll Number: CS001
Marks: 85
Alice Johnson is performing above the class average!

Student Details:
Name: Bob Smith
Roll Number: CS002
Marks: 72
Bob Smith is performing below the class average. Needs improvement.

Student Details:
Name: Charlie Brown
Roll Number: CS003
Marks: 90
Charlie Brown is performing above the class average!
```

Justification:
This task demonstrates how AI-assisted code completion helps in creating classes, constructors, methods, and conditional logic. It shows real-world use of object-oriented programming by evaluating student performance based on class average, improving code clarity and reusability.

Task 2: Data Processing in a Monitoring System
Scenario
You are working on a basic data monitoring script where sensor readings are collected as numbers. Only even readings need further processing.
Task Description
Write the initial part of a for loop to iterate over a list of integers representing sensor readings.
Add a comment prompt instructing GitHub Copilot to:
• Identify even numbers
• Calculate their square
• Print the result in a readable format
Allow Copilot to complete the remaining loop logic.
PROMPT: Write a for loop to iterate over a list of sensor readings.Identify even numbers using modulus operator.Calculate the square of even numbers and print in readable format.

CODE:

```python
# TASK 2: Data Processing in a Monitoring System
# ============================================================================

# PROMPT: "Write a for loop to iterate over a list of sensor readings"
# PROMPT: "Identify even numbers using modulus operator"
# PROMPT: "Calculate the square of even numbers and print in readable format"

# Sensor readings list
sensor_readings = [12, 15, 8, 23, 16, 9, 30, 7, 14, 21]

# Execute Task 2
input("\nPress Enter to execute Task 2...")
print("\n--- OUTPUT ---")
print(f"Sensor Readings: {sensor_readings}")
print("\nProcessing even readings:")

# AI Copilot Completion: Loop to process even sensor readings
for reading in sensor_readings:
    # Check if reading is even
    if reading % 2 == 0:
        # Calculate square
        square = reading ** 2
        # Print formatted result
        print(f"Reading: {reading} | Square: {square}")
```

OUTPUT:

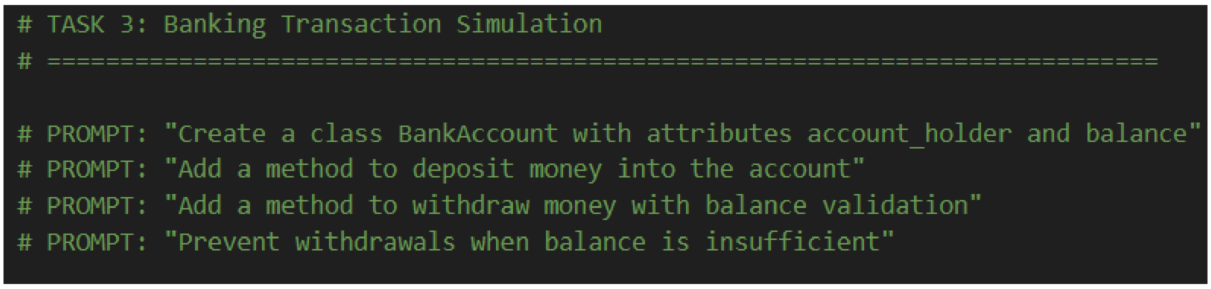


```
Press Enter to execute Task 2...

--- OUTPUT ---
Sensor Readings: [12, 15, 8, 23, 16, 9, 30, 7, 14, 21]

Processing even readings:
Reading: 12 | Square: 144
Reading: 8 | Square: 64
Reading: 16 | Square: 256
Reading: 30 | Square: 900
Reading: 14 | Square: 196
```

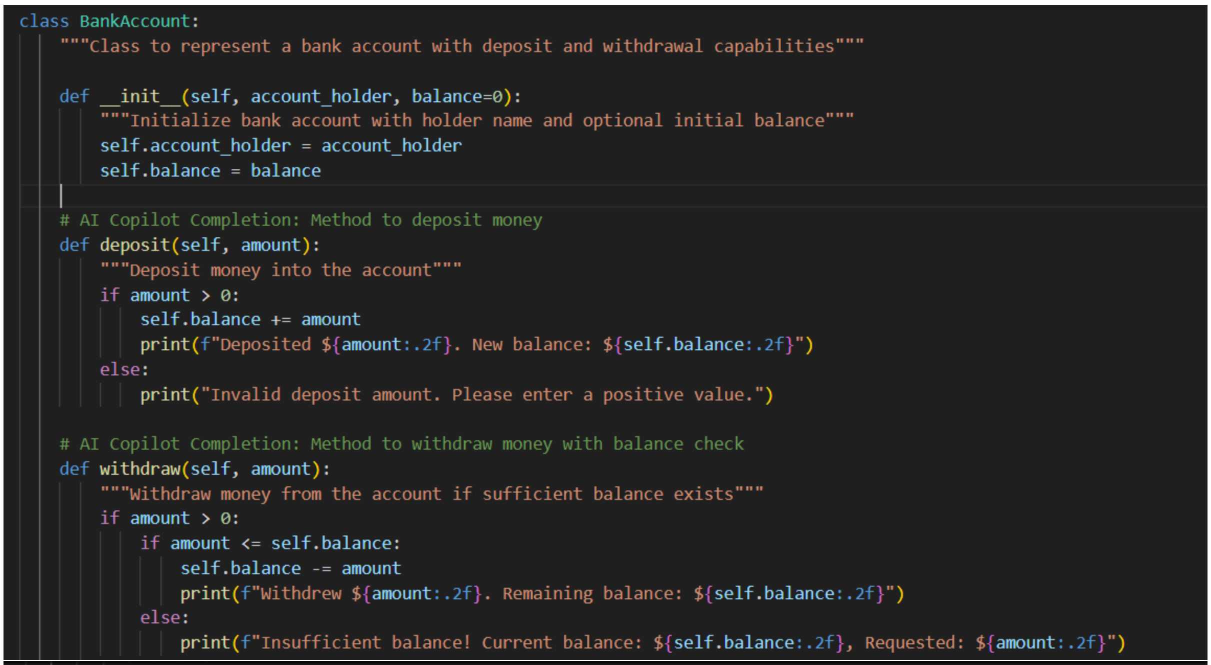
Justification:
This task highlights the use of for loops and conditional statements to process sensor data efficiently. It demonstrates how AI-generated code simplifies repetitive data analysis tasks such as filtering even values and performing mathematical computations.

Task 3: Banking Transaction Simulation
Scenario
You are developing a basic banking module that handles deposits and withdrawals for customers.
Task Description
Create the structure of a Python class named BankAccount with attributes:
• account_holder
• balance
Use GitHub Copilot to complete methods for:
• Depositing money
• Withdrawing money
• Preventing withdrawals when the balance is insufficient
Guide Copilot using method names and short comments.

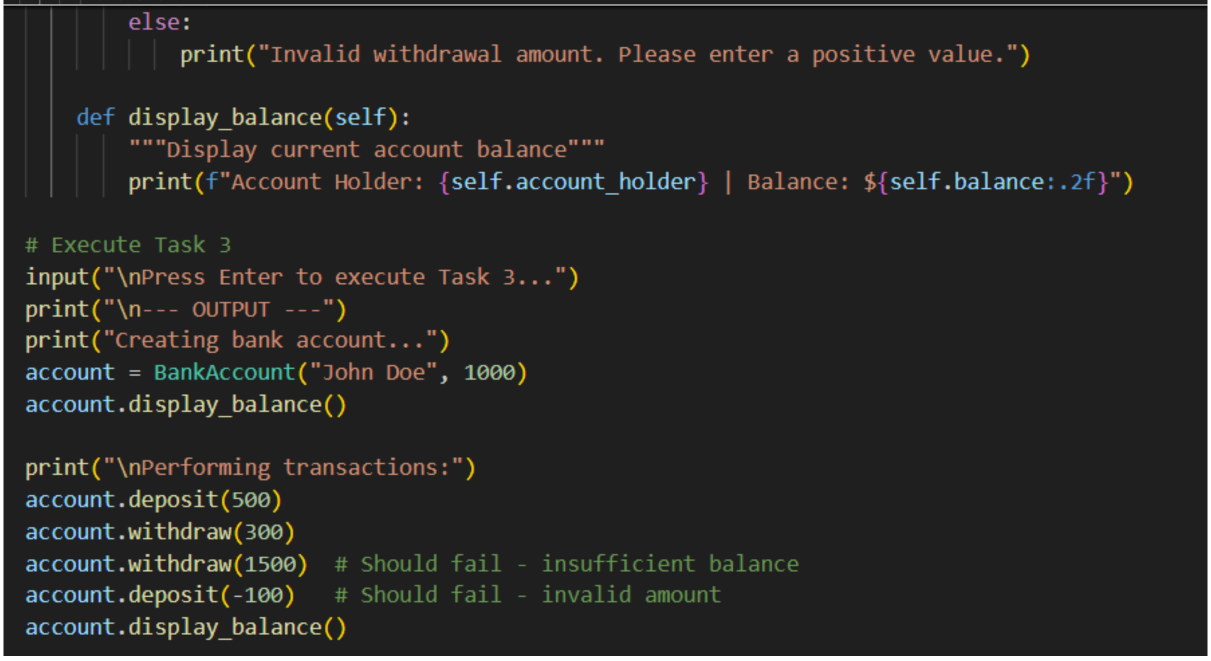
PROMPT: Create a class BankAccount with attributes account_holder and balance.Add a method to deposit money into the account.Add a method to withdraw money with balance validationPrevent withdrawals when balance is insufficient.



```
# TASK 3: Banking Transaction Simulation
# ============================================================================

# PROMPT: "Create a class BankAccount with attributes account_holder and balance"
# PROMPT: "Add a method to deposit money into the account"
# PROMPT: "Add a method to withdraw money with balance validation"
# PROMPT: "Prevent withdrawals when balance is insufficient"
```

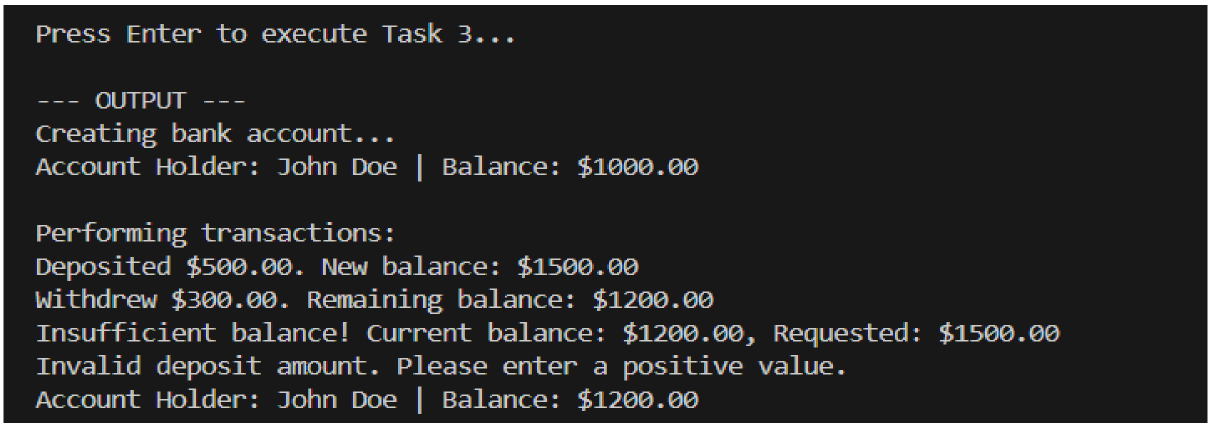


```python
class BankAccount:
    """Class to represent a bank account with deposit and withdrawal capabilities"""

    def __init__(self, account_holder, balance=0):
        """Initialize bank account with holder name and optional initial balance"""
        self.account_holder = account_holder
        self.balance = balance

    # AI Copilot Completion: Method to deposit money
    def deposit(self, amount):
        """Deposit money into the account"""
        if amount > 0:
            self.balance += amount
            print(f"Deposited ${amount:.2f}. New balance: ${self.balance:.2f}")
        else:
            print("Invalid deposit amount. Please enter a positive value.")

    # AI Copilot Completion: Method to withdraw money with balance check
    def withdraw(self, amount):
        """Withdraw money from the account if sufficient balance exists"""
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew ${amount:.2f}. Remaining balance: ${self.balance:.2f}")
            else:
                print(f"Insufficient balance! Current balance: ${self.balance:.2f}, Requested: ${amount:.2f}")
```



```python
        else:
            print("Invalid withdrawal amount. Please enter a positive value.")

    def display_balance(self):
        """Display current account balance"""
        print(f"Account Holder: {self.account_holder} | Balance: ${self.balance:.2f}")

# Execute Task 3
input("\nPress Enter to execute Task 3...")
print("\n--- OUTPUT ---")
print("Creating bank account...")
account = BankAccount("John Doe", 1000)
account.display_balance()

print("\nPerforming transactions:")
account.deposit(500)
account.withdraw(300)
account.withdraw(1500)  # Should fail - insufficient balance
account.deposit(-100)   # Should fail - invalid amount
account.display_balance()
```

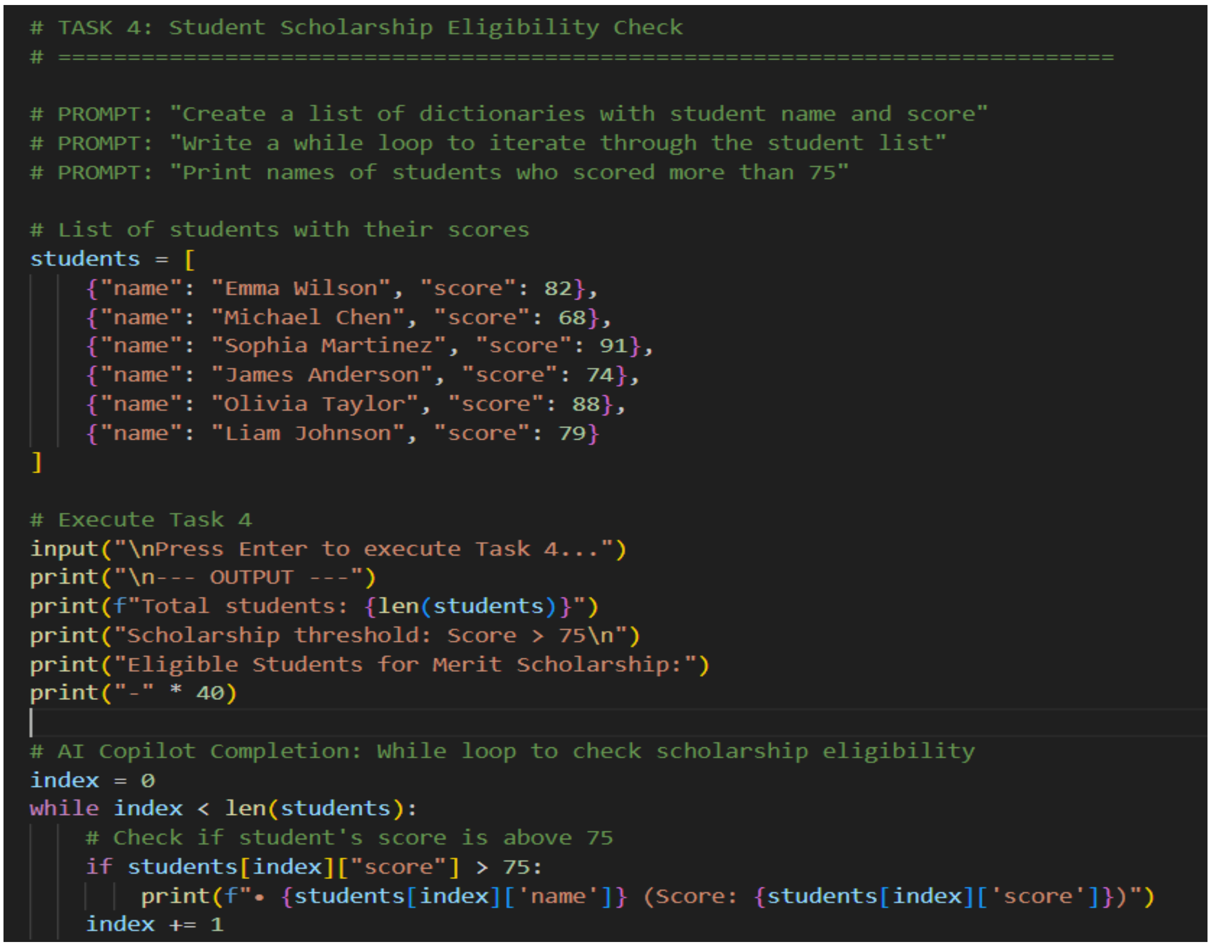
OUTPUT:



```
Press Enter to execute Task 3...

--- OUTPUT ---
Creating bank account...
Account Holder: John Doe | Balance: $1000.00

Performing transactions:
Deposited $500.00. New balance: $1500.00
Withdrew $300.00. Remaining balance: $1200.00
Insufficient balance! Current balance: $1200.00, Requested: $1500.00
Invalid deposit amount. Please enter a positive value.
Account Holder: John Doe | Balance: $1200.00
```

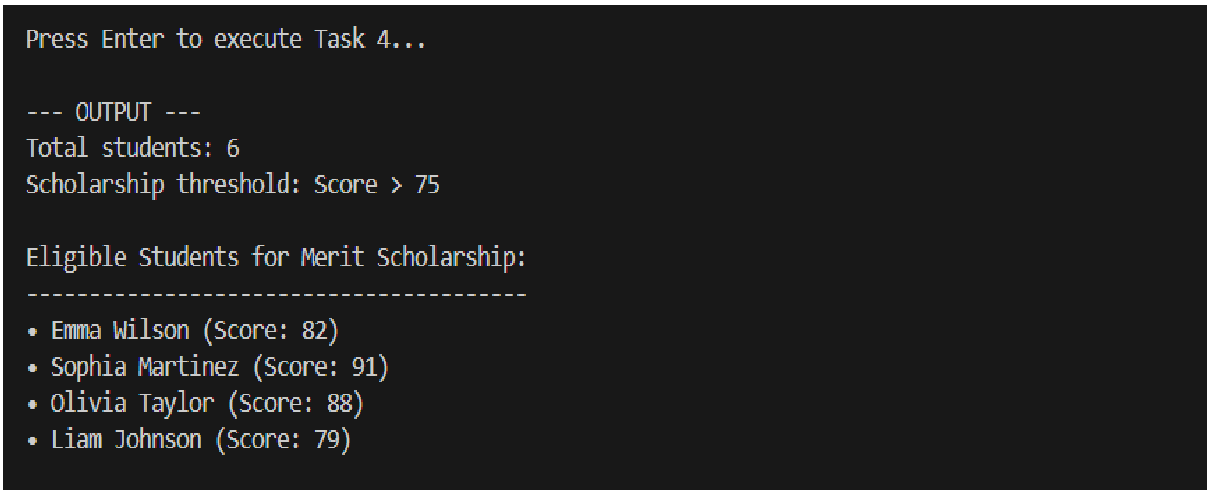
Justification:
This task illustrates secure and logical transaction handling using classes, conditionals, and validation checks. AI-based code completion ensures error prevention, such as avoiding negative deposits and insufficient balance withdrawals, which reflects real-world banking systems.

Task 4: Student Scholarship Eligibility Check
Scenario
A university wants to identify students eligible for a merit-based scholarship based on their scores.
Task Description
Define a list of dictionaries where each dictionary represents a student with:
• name
• score
Write the initialization and list structure yourself.
Then, prompt GitHub Copilot to generate a while loop that:
• Iterates through the list
• Prints the names of students who scored more than 75
Use comments to guide Copilot's code completion.
PROMPT: Create a list of dictionaries with student name and score.Write a while loop to iterate through the student list.Print names of students who scored more than 75
CODE:



```python
# TASK 4: Student Scholarship Eligibility Check
# ============================================================================

# PROMPT: "Create a list of dictionaries with student name and score"
# PROMPT: "Write a while loop to iterate through the student list"
# PROMPT: "Print names of students who scored more than 75"

# List of students with their scores
students = [
    {"name": "Emma Wilson", "score": 82},
    {"name": "Michael Chen", "score": 68},
    {"name": "Sophia Martinez", "score": 91},
    {"name": "James Anderson", "score": 74},
    {"name": "Olivia Taylor", "score": 88},
    {"name": "Liam Johnson", "score": 79}
]

# Execute Task 4
input("\nPress Enter to execute Task 4...")
print("\n--- OUTPUT ---")
print(f"Total students: {len(students)}")
print("Scholarship threshold: Score > 75\n")
print("Eligible Students for Merit Scholarship:")
print("-" * 40)

# AI Copilot Completion: While loop to check scholarship eligibility
index = 0
while index < len(students):
    # Check if student's score is above 75
    if students[index]["score"] > 75:
        print(f"• {students[index]['name']} (Score: {students[index]['score']})")
    index += 1
```

OUTPUT:

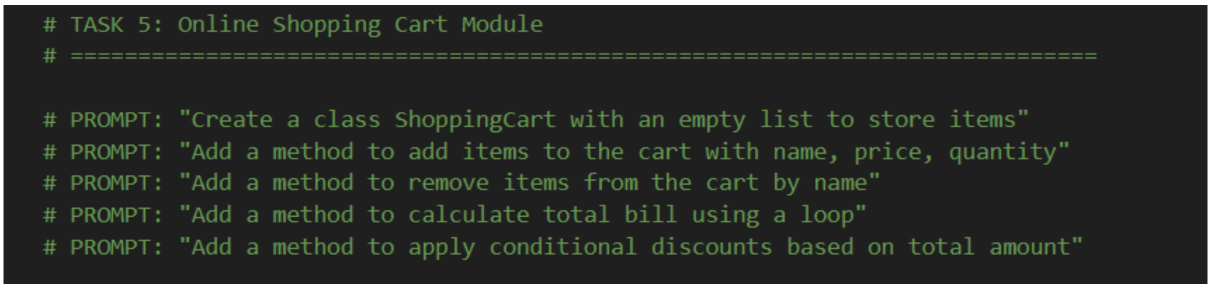


```
Press Enter to execute Task 4...

--- OUTPUT ---
Total students: 6
Scholarship threshold: Score > 75

Eligible Students for Merit Scholarship:
----------------------------------------
• Emma Wilson (Score: 82)
• Sophia Martinez (Score: 91)
• Olivia Taylor (Score: 88)
• Liam Johnson (Score: 79)
```

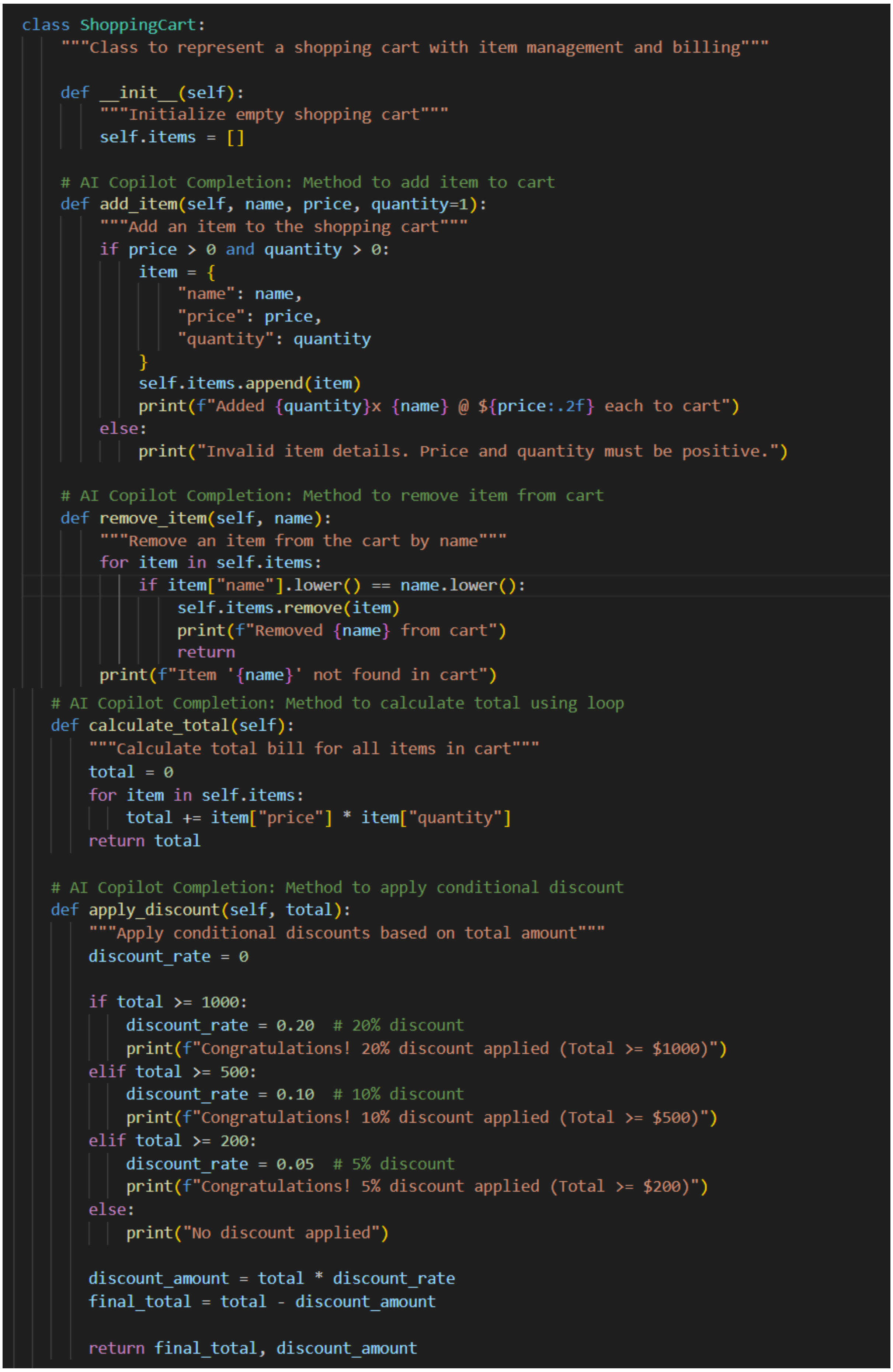
Justification:
This task focuses on the use of **while loops and list traversal** to evaluate eligibility criteria. It demonstrates how AI-generated control structures help in managing datasets and making decisions based on predefined conditions.

Task 5: Online Shopping Cart Module
Scenario
You are designing a simplified shopping cart system for an e-commerce website that supports item management and discount calculation.
Task Description
Begin writing a Python class named ShoppingCart with:
• An empty list to store items (each item may include name, price, quantity)
Use GitHub Copilot to generate methods that:
• Add items to the cart
• Remove items from the cart
• Calculate the total bill using a loop
• Apply conditional discounts (e.g., discount if total exceeds a certain amount)
PROMPT: Create a class ShoppingCart with an empty list to store items.Add a method to add items to the cart with name, price, quantity.Add a method to remove items from the cart by name.Add a method to calculate total bill using a loop.Add a method to apply conditional discounts based on total amount.

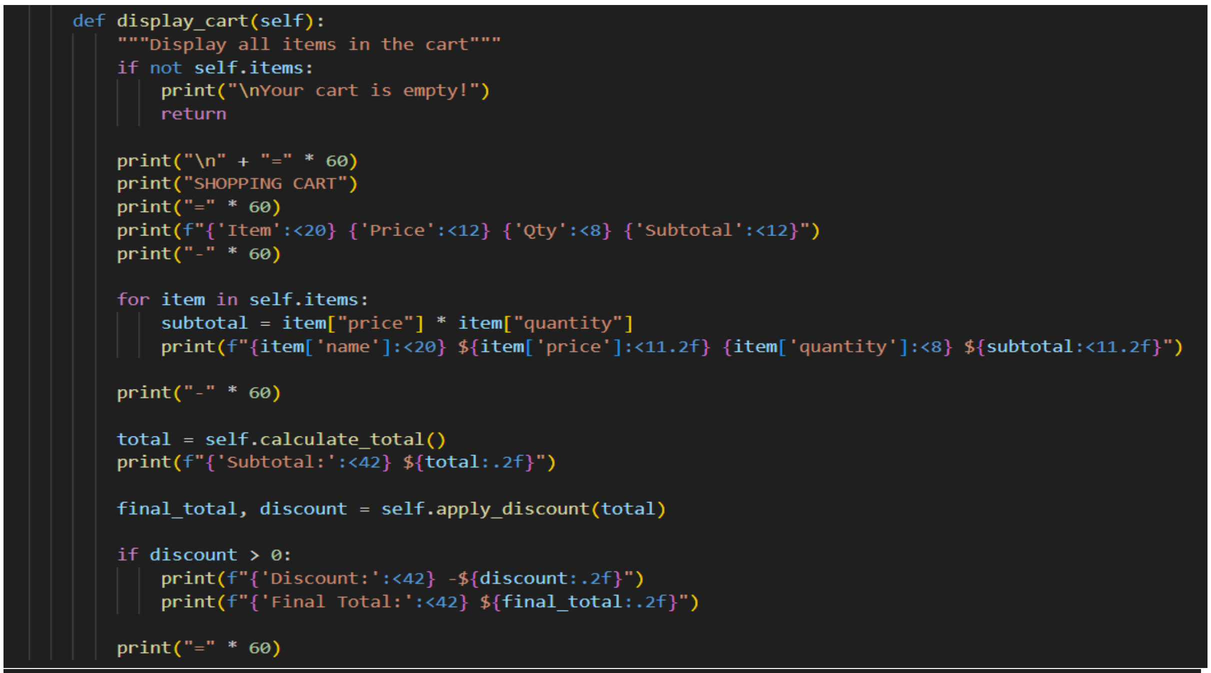
CODE:



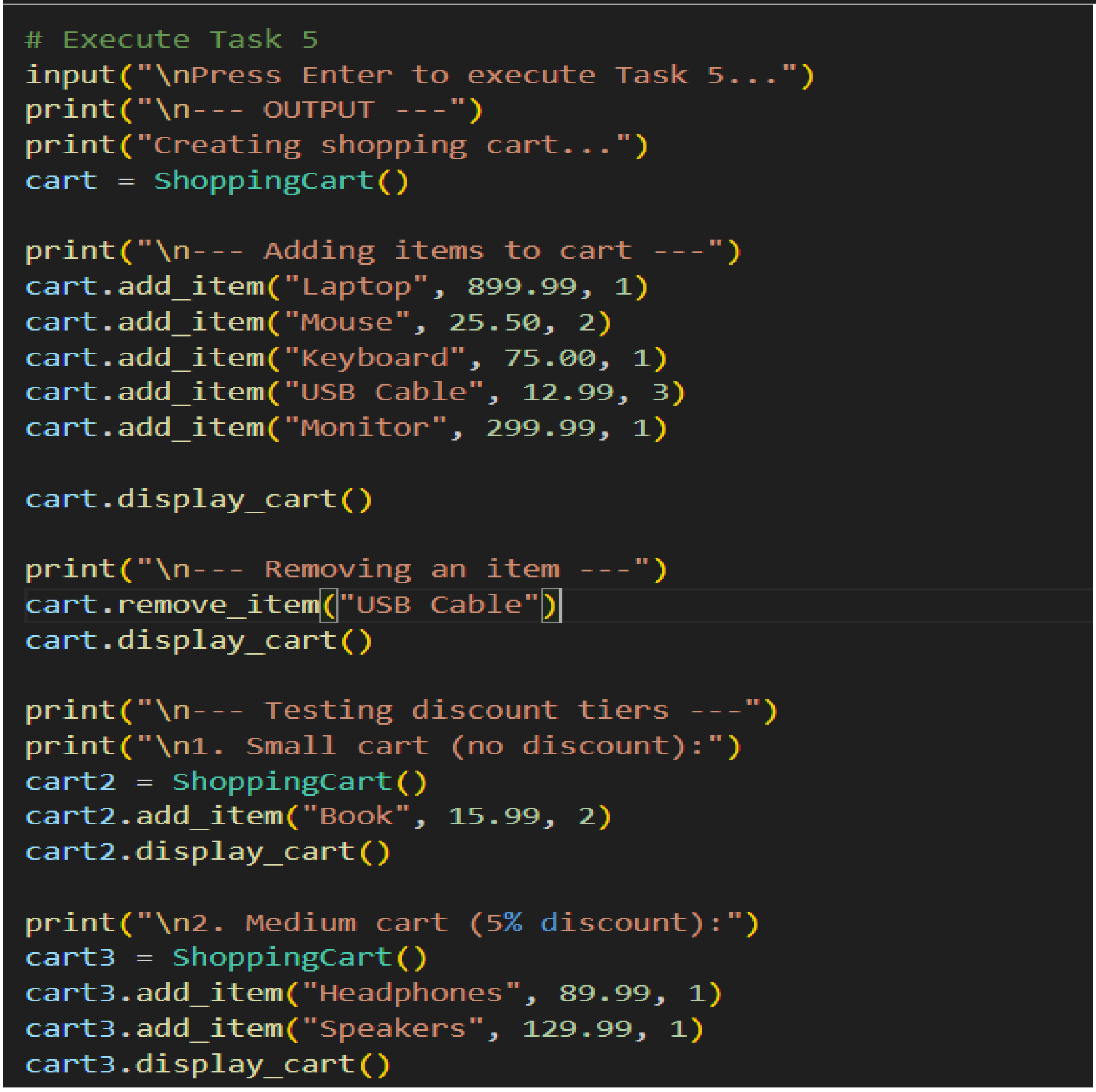
```
# TASK 5: Online Shopping Cart Module
# ============================================================================

# PROMPT: "Create a class ShoppingCart with an empty list to store items"
# PROMPT: "Add a method to add items to the cart with name, price, quantity"
# PROMPT: "Add a method to remove items from the cart by name"
# PROMPT: "Add a method to calculate total bill using a loop"
# PROMPT: "Add a method to apply conditional discounts based on total amount"
```



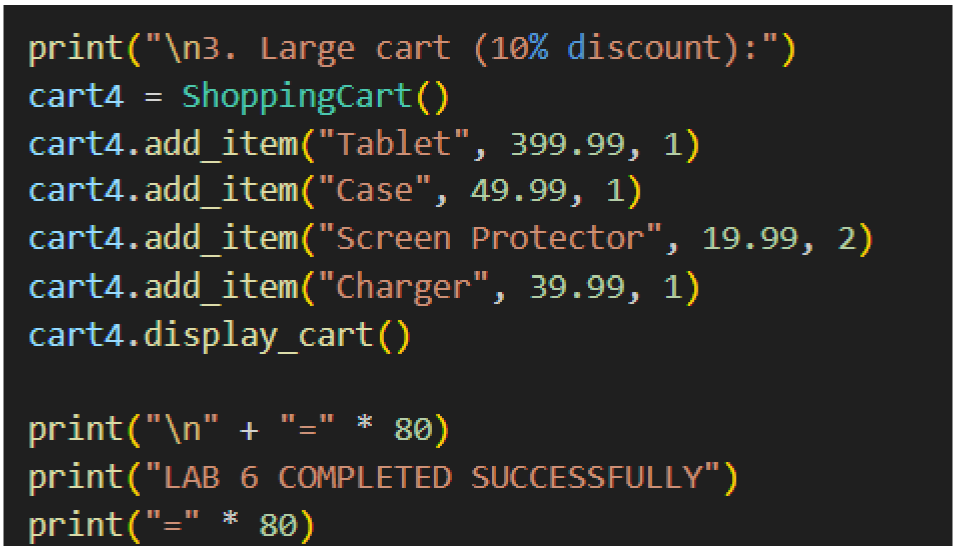
```python
class ShoppingCart:
    """Class to represent a shopping cart with item management and billing"""

    def __init__(self):
        """Initialize empty shopping cart"""
        self.items = []

    # AI Copilot Completion: Method to add item to cart
    def add_item(self, name, price, quantity=1):
        """Add an item to the shopping cart"""
        if price > 0 and quantity > 0:
            item = {
                "name": name,
                "price": price,
                "quantity": quantity
            }
            self.items.append(item)
            print(f"Added {quantity}x {name} @ ${price:.2f} each to cart")
        else:
            print("Invalid item details. Price and quantity must be positive.")

    # AI Copilot Completion: Method to remove item from cart
    def remove_item(self, name):
        """Remove an item from the cart by name"""
        for item in self.items:
            if item["name"].lower() == name.lower():
                self.items.remove(item)
                print(f"Removed {name} from cart")
                return
        print(f"Item '{name}' not found in cart")
    # AI Copilot Completion: Method to calculate total using loop
    def calculate_total(self):
        """Calculate total bill for all items in cart"""
        total = 0
        for item in self.items:
            total += item["price"] * item["quantity"]
        return total

    # AI Copilot Completion: Method to apply conditional discount
    def apply_discount(self, total):
        """Apply conditional discounts based on total amount"""
        discount_rate = 0

        if total >= 1000:
            discount_rate = 0.20  # 20% discount
            print(f"Congratulations! 20% discount applied (Total >= $1000)")
        elif total >= 500:
            discount_rate = 0.10  # 10% discount
            print(f"Congratulations! 10% discount applied (Total >= $500)")
        elif total >= 200:
            discount_rate = 0.05  # 5% discount
            print(f"Congratulations! 5% discount applied (Total >= $200)")
        else:
            print("No discount applied")

        discount_amount = total * discount_rate
        final_total = total - discount_amount

        return final_total, discount_amount
```



```python
    def display_cart(self):
        """Display all items in the cart"""
        if not self.items:
            print("\nYour cart is empty!")
            return

        print("\n" + "=" * 60)
        print("SHOPPING CART")
        print("=" * 60)
        print(f"{'Item':<20} {'Price':<12} {'Qty':<8} {'Subtotal':<12}")
        print("-" * 60)

        for item in self.items:
            subtotal = item["price"] * item["quantity"]
            print(f"{item['name']:<20} ${item['price']:<11.2f} {item['quantity']:<8} ${subtotal:<11.2f}")

        print("-" * 60)

        total = self.calculate_total()
        print(f"{'Subtotal:':<42} ${total:.2f}")

        final_total, discount = self.apply_discount(total)

        if discount > 0:
            print(f"{'Discount:':<42} -${discount:.2f}")
            print(f"{'Final Total:':<42} ${final_total:.2f}")

        print("=" * 60)
```

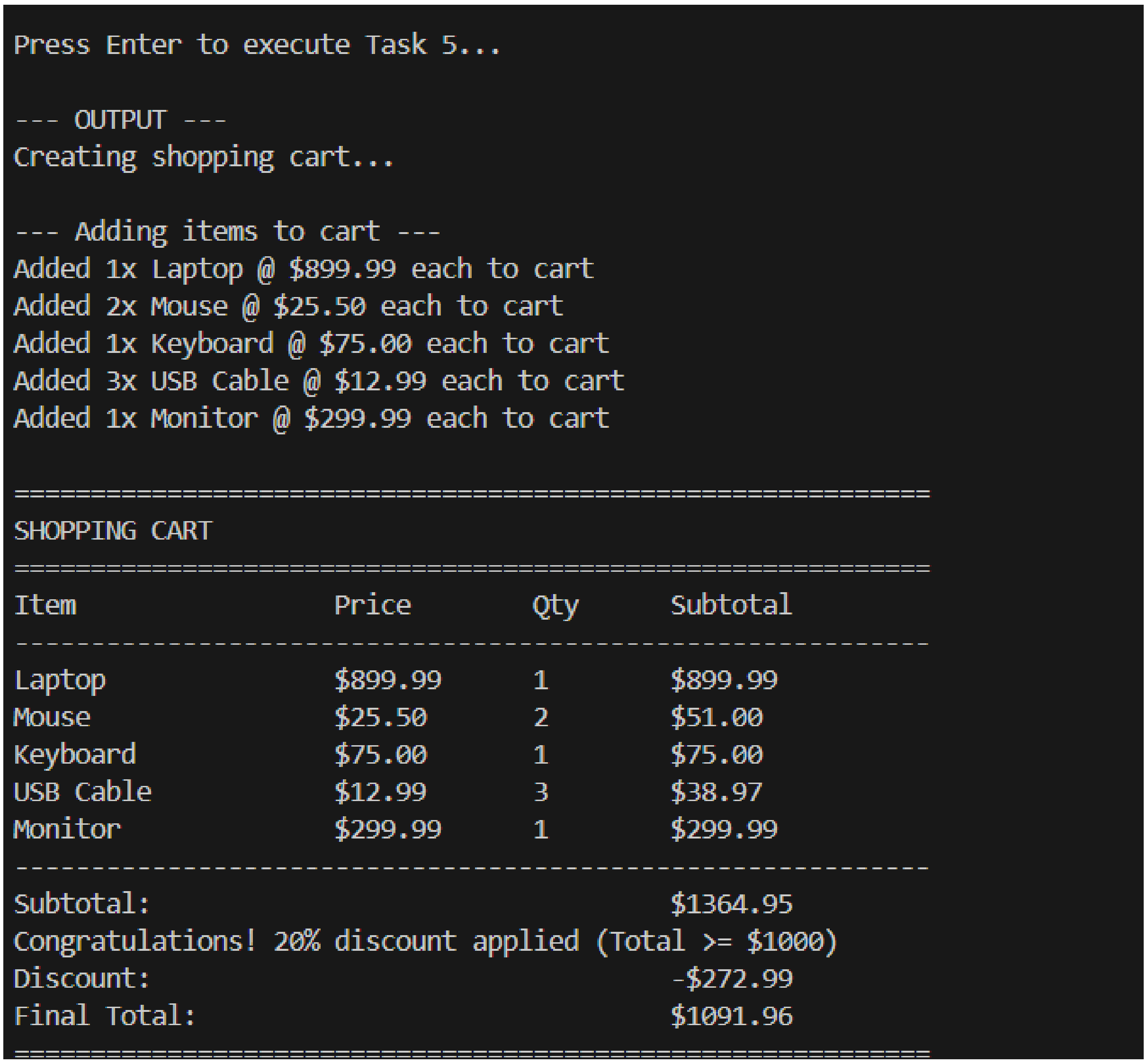


```python
# Execute Task 5
input("\nPress Enter to execute Task 5...")
print("\n--- OUTPUT ---")
print("Creating shopping cart...")
cart = ShoppingCart()

print("\n--- Adding items to cart ---")
cart.add_item("Laptop", 899.99, 1)
cart.add_item("Mouse", 25.50, 2)
cart.add_item("Keyboard", 75.00, 1)
cart.add_item("USB Cable", 12.99, 3)
cart.add_item("Monitor", 299.99, 1)

cart.display_cart()

print("\n--- Removing an item ---")
cart.remove_item("USB Cable")
cart.display_cart()

print("\n--- Testing discount tiers ---")
print("\n1. Small cart (no discount):")
cart2 = ShoppingCart()
cart2.add_item("Book", 15.99, 2)
cart2.display_cart()

print("\n2. Medium cart (5% discount):")
cart3 = ShoppingCart()
cart3.add_item("Headphones", 89.99, 1)
cart3.add_item("Speakers", 129.99, 1)
cart3.display_cart()
```

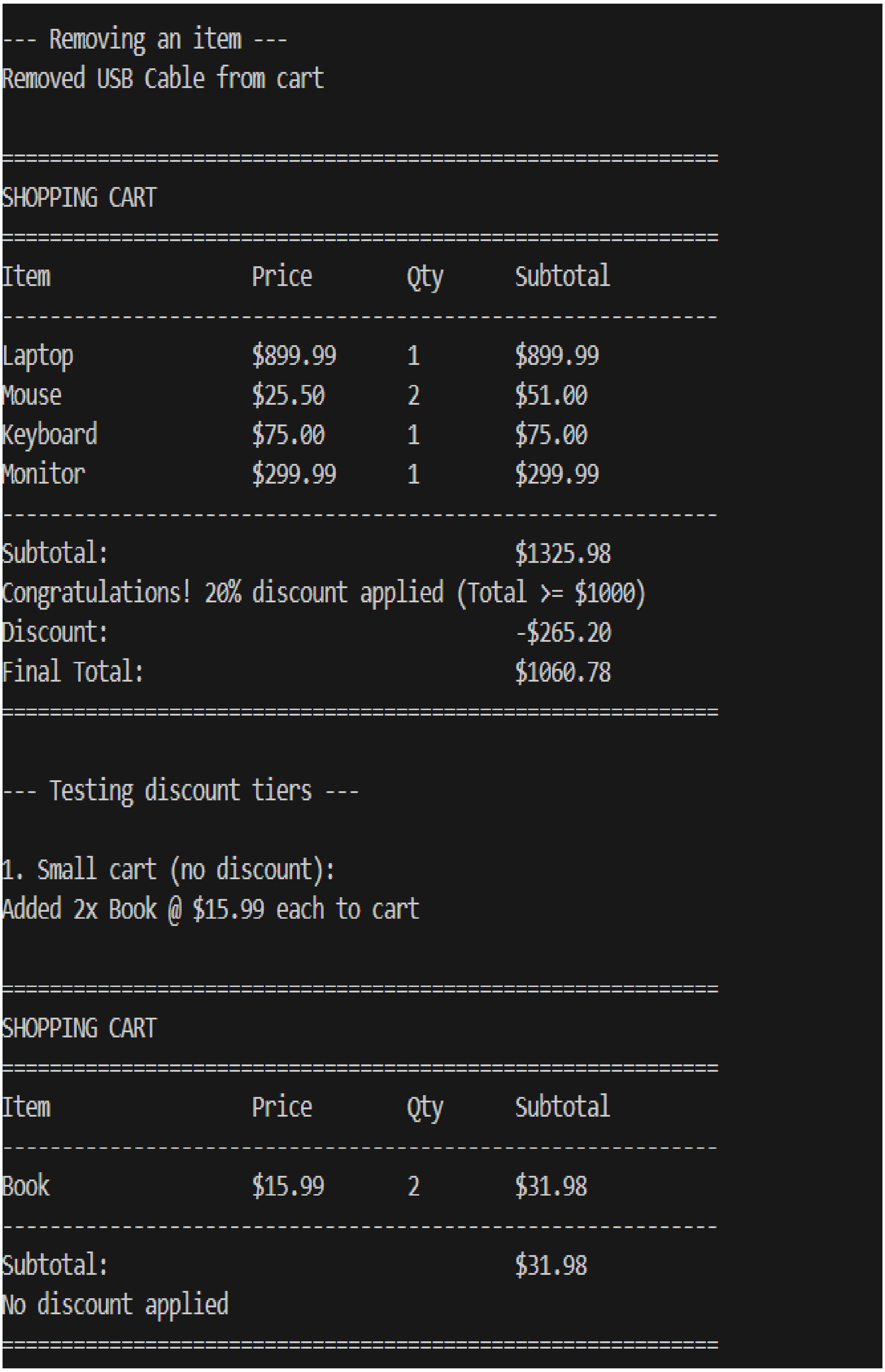


```python
print("\n3. Large cart (10% discount):")
cart4 = ShoppingCart()
cart4.add_item("Tablet", 399.99, 1)
cart4.add_item("Case", 49.99, 1)
cart4.add_item("Screen Protector", 19.99, 2)
cart4.add_item("Charger", 39.99, 1)
cart4.display_cart()

print("\n" + "=" * 80)
print("LAB 6 COMPLETED SUCCESSFULLY")
print("=" * 80)
```

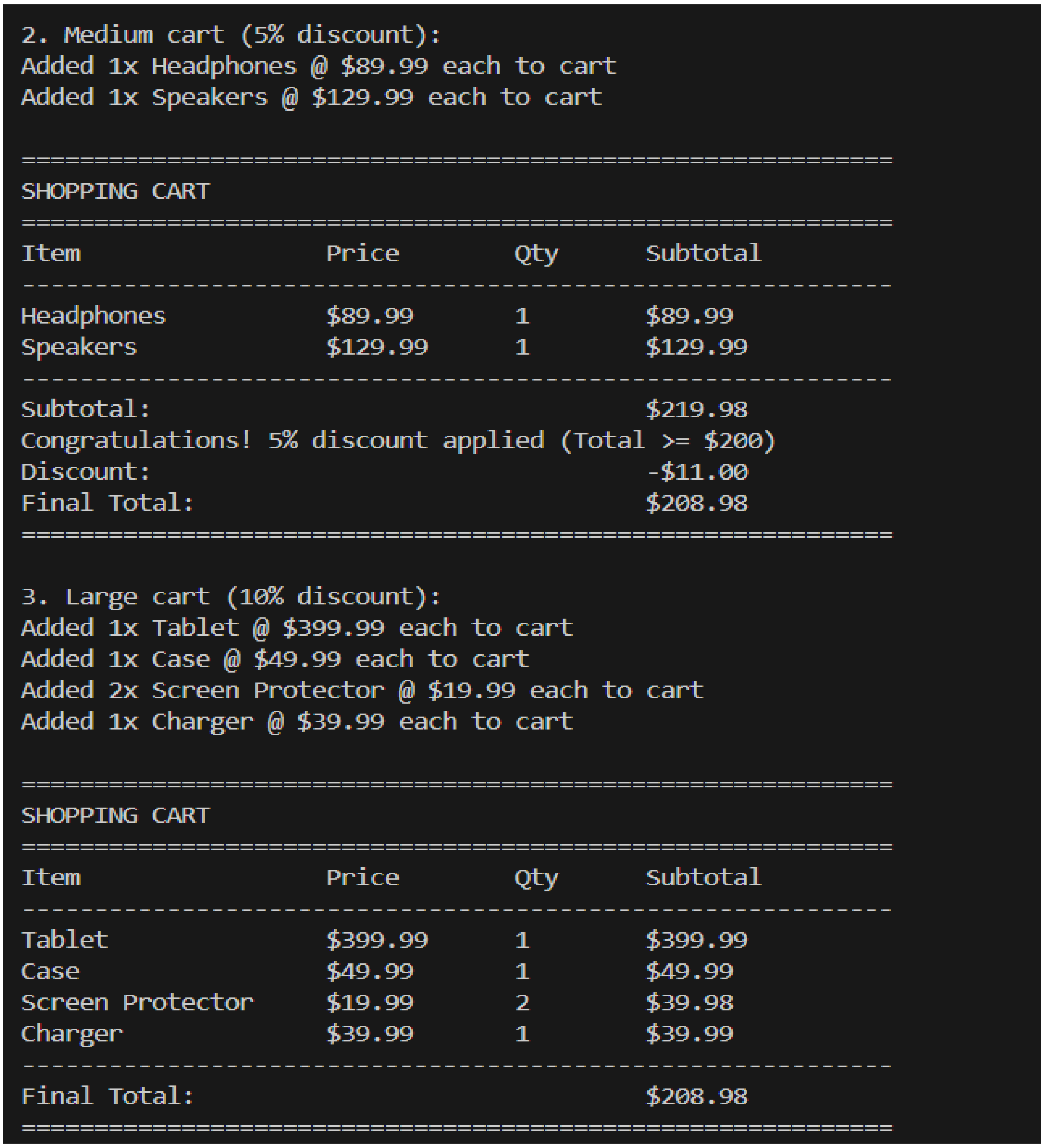
OUTPUT:



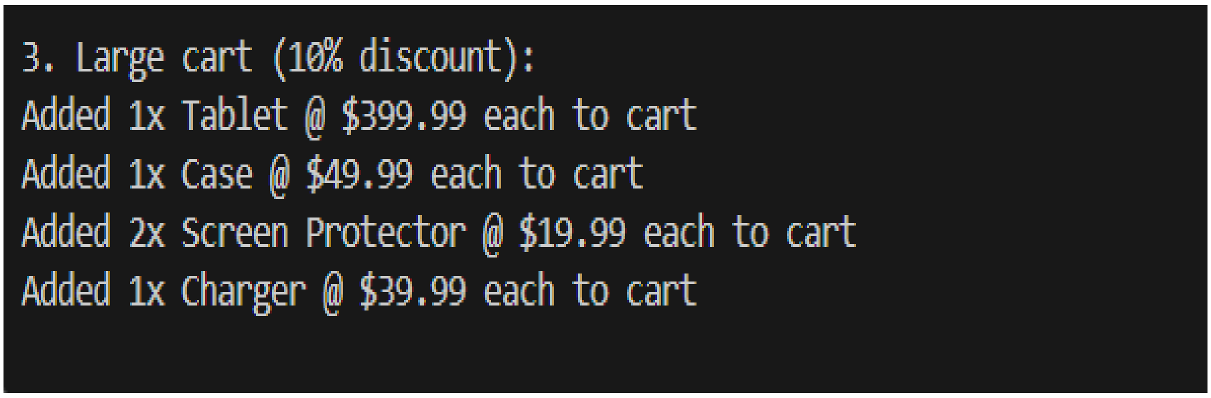
```
Press Enter to execute Task 5...

--- OUTPUT ---
Creating shopping cart...

--- Adding items to cart ---
Added 1x Laptop @ $899.99 each to cart
Added 2x Mouse @ $25.50 each to cart
Added 1x Keyboard @ $75.00 each to cart
Added 3x USB Cable @ $12.99 each to cart
Added 1x Monitor @ $299.99 each to cart

============================================================
SHOPPING CART
============================================================
Item                 Price        Qty      Subtotal
------------------------------------------------------------
Laptop               $899.99      1        $899.99
Mouse                $25.50       2        $51.00
Keyboard             $75.00       1        $75.00
USB Cable            $12.99       3        $38.97
Monitor              $299.99      1        $299.99
------------------------------------------------------------
Subtotal:                                  $1364.95
Congratulations! 20% discount applied (Total >= $1000)
Discount:                                  -$272.99
Final Total:                               $1091.96
============================================================
```



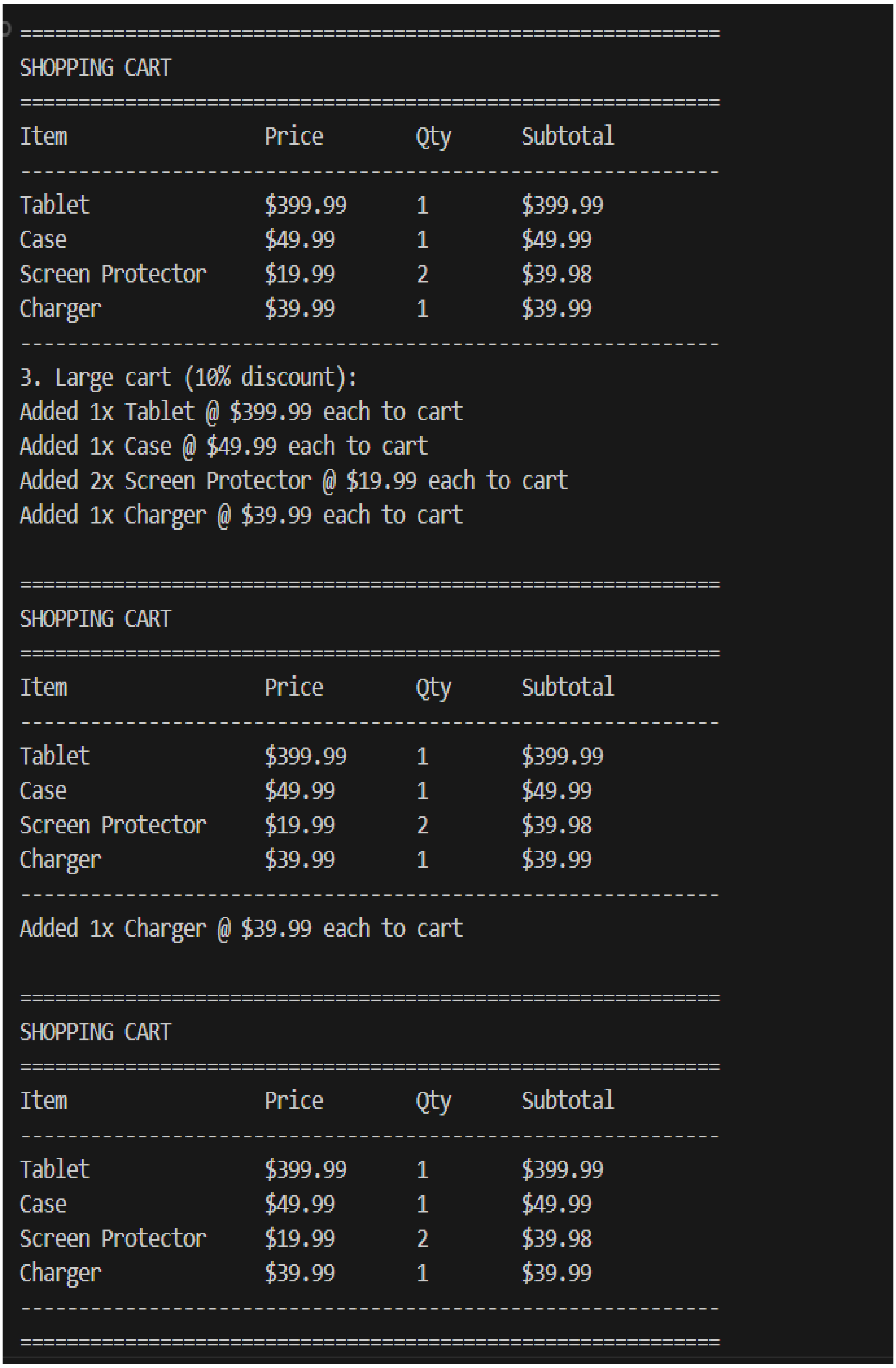
```
--- Removing an item ---
Removed USB Cable from cart

============================================================
SHOPPING CART
============================================================
Item                 Price        Qty      Subtotal
------------------------------------------------------------
Laptop               $899.99      1        $899.99
Mouse                $25.50       2        $51.00
Keyboard             $75.00       1        $75.00
Monitor              $299.99      1        $299.99
------------------------------------------------------------
Subtotal:                                  $1325.98
Congratulations! 20% discount applied (Total >= $1000)
Discount:                                  -$265.20
Final Total:                               $1060.78
============================================================

--- Testing discount tiers ---

1. Small cart (no discount):
Added 2x Book @ $15.99 each to cart

============================================================
SHOPPING CART
============================================================
Item                 Price        Qty      Subtotal
------------------------------------------------------------
Book                 $15.99       2        $31.98
------------------------------------------------------------
Subtotal:                                  $31.98
No discount applied
============================================================
```



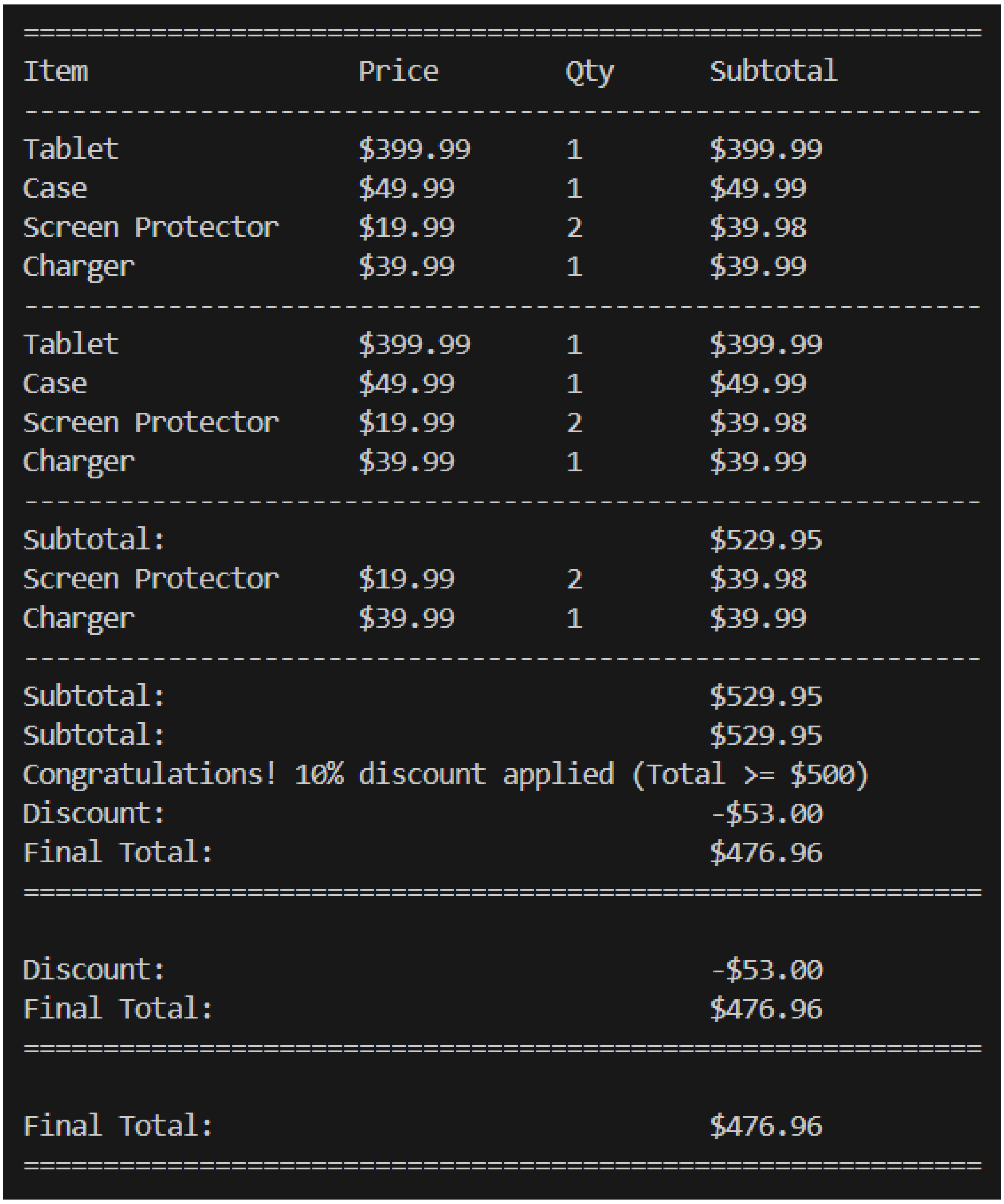
```
2. Medium cart (5% discount):
Added 1x Headphones @ $89.99 each to cart
Added 1x Speakers @ $129.99 each to cart

============================================================
SHOPPING CART
============================================================
Item                 Price        Qty      Subtotal
------------------------------------------------------------
Headphones           $89.99       1        $89.99
Speakers             $129.99      1        $129.99
------------------------------------------------------------
Subtotal:                                  $219.98
Congratulations! 5% discount applied (Total >= $200)
Discount:                                  -$11.00
Final Total:                               $208.98
============================================================

3. Large cart (10% discount):
Added 1x Tablet @ $399.99 each to cart
Added 1x Case @ $49.99 each to cart
Added 2x Screen Protector @ $19.99 each to cart
Added 1x Charger @ $39.99 each to cart

============================================================
SHOPPING CART
============================================================
Item                 Price        Qty      Subtotal
------------------------------------------------------------
Tablet               $399.99      1        $399.99
Case                 $49.99       1        $49.99
Screen Protector     $19.99       2        $39.98
Charger              $39.99       1        $39.99
------------------------------------------------------------
Final Total:                               $208.98
============================================================
```



```
3. Large cart (10% discount):
Added 1x Tablet @ $399.99 each to cart
Added 1x Case @ $49.99 each to cart
Added 2x Screen Protector @ $19.99 each to cart
Added 1x Charger @ $39.99 each to cart
```



```
============================================================
SHOPPING CART
============================================================
Item                 Price        Qty      Subtotal
------------------------------------------------------------
Tablet               $399.99      1        $399.99
Case                 $49.99       1        $49.99
Screen Protector     $19.99       2        $39.98
Charger              $39.99       1        $39.99
------------------------------------------------------------
3. Large cart (10% discount):
Added 1x Tablet @ $399.99 each to cart
Added 1x Case @ $49.99 each to cart
Added 2x Screen Protector @ $19.99 each to cart
Added 1x Charger @ $39.99 each to cart

============================================================
SHOPPING CART
============================================================
Item                 Price        Qty      Subtotal
------------------------------------------------------------
Tablet               $399.99      1        $399.99
Case                 $49.99       1        $49.99
Screen Protector     $19.99       2        $39.98
Charger              $39.99       1        $39.99
------------------------------------------------------------
Added 1x Charger @ $39.99 each to cart

============================================================
SHOPPING CART
============================================================
Item                 Price        Qty      Subtotal
------------------------------------------------------------
Tablet               $399.99      1        $399.99
Case                 $49.99       1        $49.99
Screen Protector     $19.99       2        $39.98
Charger              $39.99       1        $39.99
------------------------------------------------------------
============================================================
```



```
============================================================
Item                 Price        Qty       Subtotal
------------------------------------------------------------
Tablet               $399.99      1         $399.99
Case                 $49.99       1         $49.99
Screen Protector     $19.99       2         $39.98
Charger              $39.99       1         $39.99
------------------------------------------------------------
Tablet               $399.99      1         $399.99
Case                 $49.99       1         $49.99
Screen Protector     $19.99       2         $39.98
Charger              $39.99       1         $39.99
------------------------------------------------------------
Subtotal:                                   $529.95
Screen Protector     $19.99       2         $39.98
Charger              $39.99       1         $39.99
------------------------------------------------------------
Subtotal:                                   $529.95
Subtotal:                                   $529.95
Congratulations! 10% discount applied (Total >= $500)
Discount:                                   -$53.00
Final Total:                                $476.96
============================================================

Discount:                                   -$53.00
Final Total:                                $476.96
============================================================

Final Total:                                $476.96
============================================================
```

Justification:

This task integrates classes, loops, conditionals, and data structures to simulate an e-commerce shopping cart. AI-assisted code completion improves productivity by efficiently implementing item management, billing, and discount logic, reflecting real-world online shopping systems.