LAB ASSIGNMENT-6.4

2303A51337

BATCH-10

➢ TASK-1:

PROMPT:

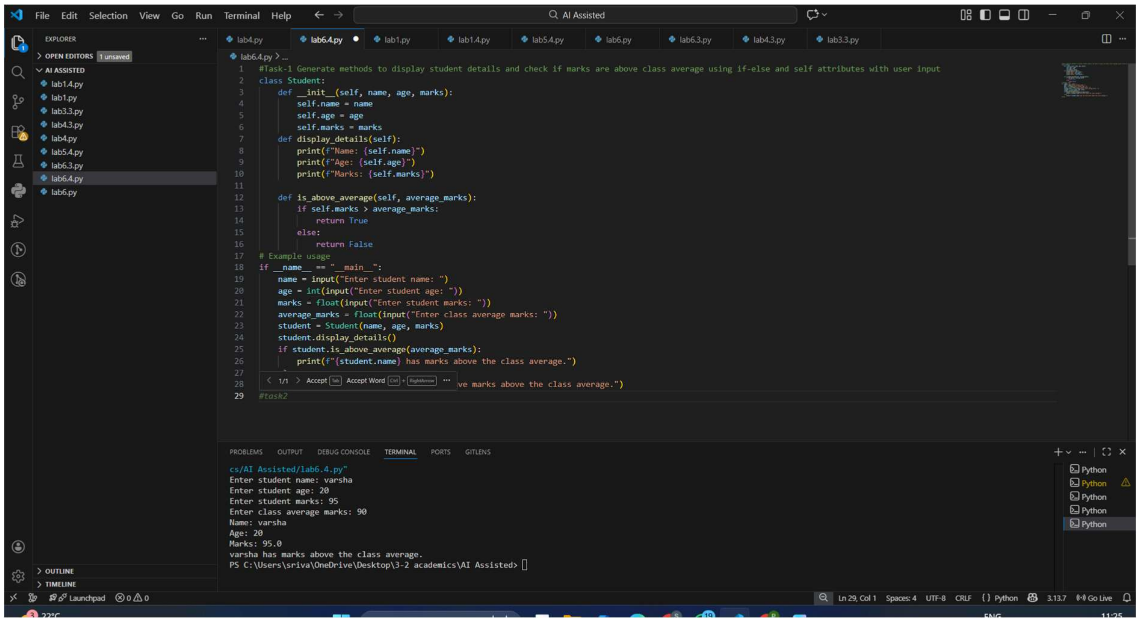
Generate methods to display student details and check if marks are above class average using if-else and self attributes with user input.

CODE:

```
class Student:
  def __init__(self, name, age, marks):
    self.name = name
    self.age = age
    self.marks = marks
  def display_details(self):
    print(f"Name: {self.name}")
    print(f"Age: {self.age}")
    print(f"Marks: {self.marks}")
  def is_above_average(self, average_marks):
    if self.marks > average_marks:
      return True
    else:
      return False
# Example usage
if __name__ == "__main__":
  name = input("Enter student name: ")
  age = int(input("Enter student age: "))
  marks = float(input("Enter student marks: "))
```

```
average_marks = float(input("Enter class average marks: "))

student = Student(name, age, marks)

student.display_details()

if student.is_above_average(average_marks):

    print(f"{student.name} has marks above the class average.")

else:

    print(f"{student.name} does not have marks above the class average.")
```

OUTPUT:



```
#Task-1 Generate methods to display student details and check if marks are above class average using if-else and self attributes with user input
class Student:
    def __init__(self, name, age, marks):
        self.name = name
        self.age = age
        self.marks = marks
    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Marks: {self.marks}")

    def is_above_average(self, average_marks):
        if self.marks > average_marks:
            return True
        else:
            return False
# Example usage
if __name__ == "__main__":
    name = input("Enter student name: ")
    age = int(input("Enter student age: "))
    marks = float(input("Enter student marks: "))
    average_marks = float(input("Enter class average marks: "))
    student = Student(name, age, marks)
    student.display_details()
    if student.is_above_average(average_marks):
        print(f"{student.name} has marks above the class average.")

                                ve marks above the class average.")
#task2
```

```
cs/AI Assisted/lab6.4.py"
Enter student name: varsha
Enter student age: 20
Enter student marks: 95
Enter class average marks: 90
Name: varsha
Age: 20
Marks: 95.0
varsha has marks above the class average.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted> 
```

EXPLANATION:

- This program creates a Student class with attributes for name, age, and marks.
- The constructor initializes these values when an object is created.
- The display_details() method prints the student's information using self.
- The is_above_average() method compares the student's marks with the class average using an if-else condition and returns a result.
- User input is taken to create a student object, and the program displays the details and prints whether the student's marks are above the class average or not.

❖ TASK-2

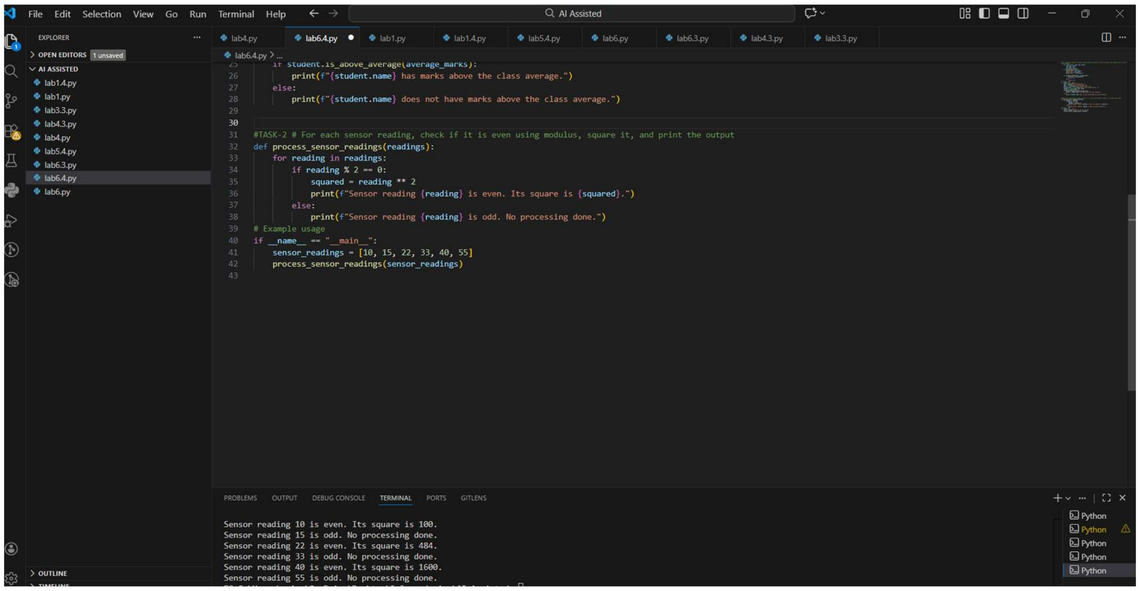
PROMPT:

For each sensor reading, check if it is even using modulus, square it, and print the output.

CODE:

```
def process_sensor_readings(readings):
    for reading in readings:
        if reading % 2 == 0:
            squared = reading ** 2
            print(f"Sensor reading {reading} is even. Its square is {squared}.")
        else:
            print(f"Sensor reading {reading} is odd. No processing done.")
# Example usage
if __name__ == "__main__":
    sensor_readings = [10, 15, 22, 33, 40, 55]
    process_sensor_readings(sensor_readings)
```

OUTPUT:



```
    if student.is_above_average(average_marks):
        print(f"{student.name} has marks above the class average.")
    else:
        print(f"{student.name} does not have marks above the class average.")


#TASK-2 # For each sensor reading, check if it is even using modulus, square it, and print the output
def process_sensor_readings(readings):
    for reading in readings:
        if reading % 2 == 0:
            squared = reading ** 2
            print(f"Sensor reading {reading} is even. Its square is {squared}.")
        else:
            print(f"Sensor reading {reading} is odd. No processing done.")
# Example usage
if __name__ == "__main__":
    sensor_readings = [10, 15, 22, 33, 40, 55]
    process_sensor_readings(sensor_readings)
```

```
Sensor reading 10 is even. Its square is 100.
Sensor reading 15 is odd. No processing done.
Sensor reading 22 is even. Its square is 484.
Sensor reading 33 is odd. No processing done.
Sensor reading 40 is even. Its square is 1600.
Sensor reading 55 is odd. No processing done.
```

EXPLANATION:

- This program processes a list of sensor readings using a loop.
- It checks each reading to see if it is even using the modulus operator.
- If the reading is even, it calculates its square and prints the result. If the reading is odd, it prints a message saying no processing is done.
- This shows the use of loops, conditionals, and arithmetic operations in Python.

❖ TASK-3:

PROMPT:

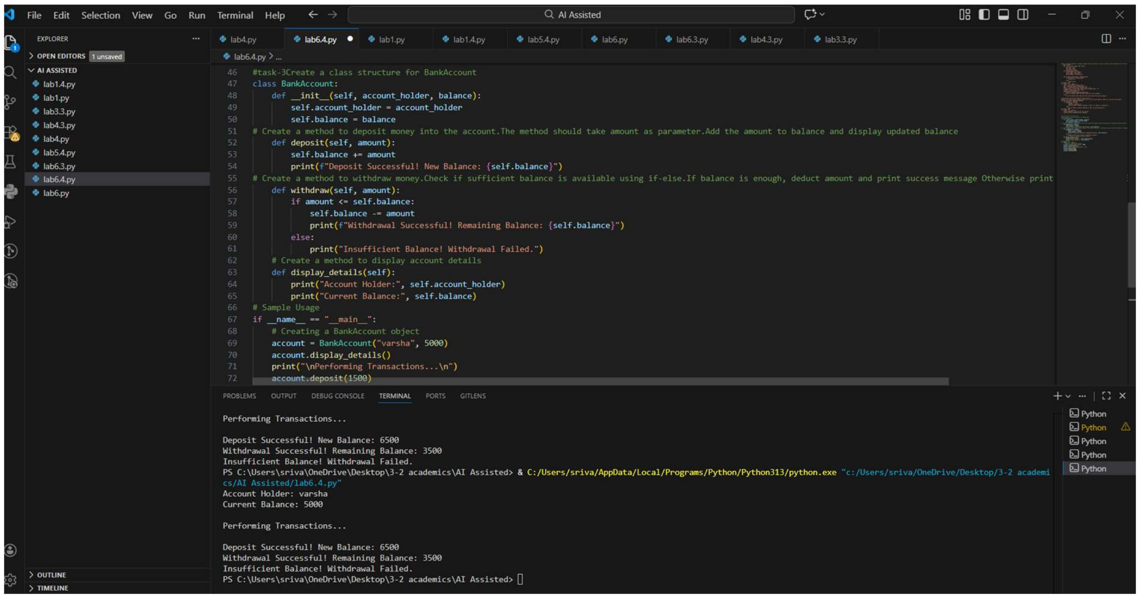
Create a class structure for BankAccount.Create a method to deposit money into the account.The method should take amount as parameter.Add the amount to balance and display updated balance

Create a method to withdraw money.Check if sufficient balance is available using if-else.If balance is enough, deduct amount and print success message Otherwise print 'Insufficient Balance'

CODE:

```
class BankAccount:
    def __init__(self, account_holder, balance):
        self.account_holder = account_holder
        self.balance = balance
# Create a method to deposit money into the account.The method should take amount
as parameter.Add the amount to balance and display updated balance
    def deposit(self, amount):
        self.balance += amount
        print(f"Deposit Successful! New Balance: {self.balance}")
# Create a method to withdraw money.Check if sufficient balance is available using if-
else.If balance is enough, deduct amount and print success message Otherwise print
'Insufficient Balance'
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrawal Successful! Remaining Balance: {self.balance}")
        else:
            print("Insufficient Balance! Withdrawal Failed.")
    # Create a method to display account details
    def display_details(self):
        print("Account Holder:", self.account_holder)
        print("Current Balance:", self.balance)
# Sample Usage
if __name__ == "__main__":
    # Creating a BankAccount object
    account = BankAccount("varsha", 5000)
    account.display_details()
    print("\nPerforming Transactions...\n")
    account.deposit(1500)
    account.withdraw(3000)
    account.withdraw(5000)
```

OUTPUT:



```
#task-3Create a class structure for BankAccount
class BankAccount:
    def __init__(self, account_holder, balance):
        self.account_holder = account_holder
        self.balance = balance
# Create a method to deposit money into the account.The method should take amount as parameter.Add the amount to balance and display updated balance
    def deposit(self, amount):
        self.balance += amount
        print(f"Deposit Successful! New Balance: {self.balance}")
# Create a method to withdraw money.Check if sufficient balance is available using if-else.If balance is enough, deduct amount and print success message Otherwise print
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrawal Successful! Remaining Balance: {self.balance}")
        else:
            print("Insufficient Balance! Withdrawal Failed.")
    # Create a method to display account details
    def display_details(self):
        print("Account Holder:", self.account_holder)
        print("Current Balance:", self.balance)
# Sample Usage
if __name__ == "__main__":
    # Creating a BankAccount object
    account = BankAccount("varsha", 5000)
    account.display_details()
    print("\nPerforming Transactions...\n")
    account.deposit(1500)
```

```
Performing Transactions...

Deposit Successful! New Balance: 6500
Withdrawal Successful! Remaining Balance: 3500
Insufficient Balance! Withdrawal Failed.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted> & C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriva/OneDrive/Desktop/3-2 academi
cs/AI Assisted/lab6.4.py"
Account Holder: varsha
Current Balance: 5000

Performing Transactions...

Deposit Successful! New Balance: 6500
Withdrawal Successful! Remaining Balance: 3500
Insufficient Balance! Withdrawal Failed.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted>
```

EXPLANATION:

This program creates a BankAccount class with account holder and balance attributes. It provides methods to deposit money and withdraw money using if-else to check for sufficient balance. If the balance is not enough, it prints an “Insufficient Balance” message. The program also displays account details and demonstrates transactions using a sample account.

❖ TASK-4:

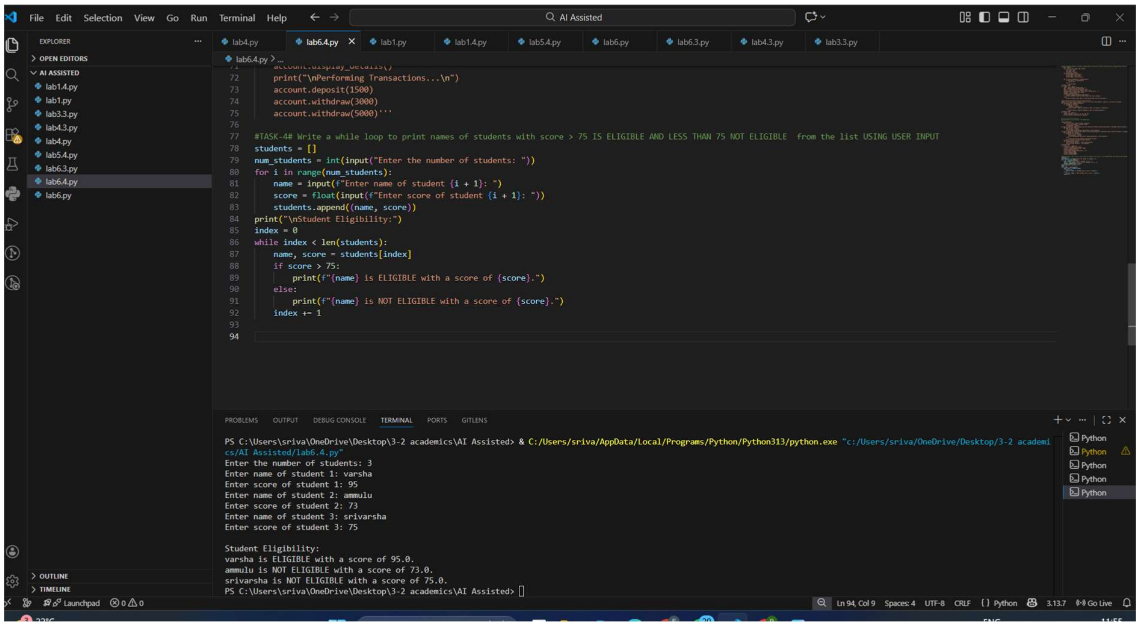
PROMPT:

```
Write a while loop to print names of students with score > 75 is
eligible and less than 75 is not from the list USING USER INPUT
```

CODE:

```
students = []

num_students = int(input("Enter the number of students: "))

for i in range(num_students):

  name = input(f"Enter name of student {i + 1}: ")

  score = float(input(f"Enter score of student {i + 1}: "))

  students.append((name, score))

print("\nStudent Eligibility:")

index = 0

while index < len(students):

  name, score = students[index]

  if score > 75:

    print(f"{name} is ELIGIBLE with a score of {score}.")

  else:

    print(f"{name} is NOT ELIGIBLE with a score of {score}.")

  index += 1
```

OUTPUT:



```
    print("\nPerforming Transactions...\n")
    account.deposit(1500)
    account.withdraw(3000)
    account.withdraw(5000)'''

#TASK-4# Write a while loop to print names of students with score > 75 IS ELIGIBLE AND LESS THAN 75 NOT ELIGIBLE  from the list USING USER INPUT
students = []
num_students = int(input("Enter the number of students: "))
for i in range(num_students):
    name = input(f"Enter name of student {i + 1}: ")
    score = float(input(f"Enter score of student {i + 1}: "))
    students.append((name, score))
print("\nStudent Eligibility:")
index = 0
while index < len(students):
    name, score = students[index]
    if score > 75:
        print(f"{name} is ELIGIBLE with a score of {score}.")
    else:
        print(f"{name} is NOT ELIGIBLE with a score of {score}.")
    index += 1
```

```
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted> & C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriva/OneDrive/Desktop/3-2 academics/AI Assisted/lab6.4.py"
Enter the number of students: 3
Enter name of student 1: varsha
Enter score of student 1: 95
Enter name of student 2: ammulu
Enter score of student 2: 73
Enter name of student 3: srivarsha
Enter score of student 3: 75

Student Eligibility:
varsha is ELIGIBLE with a score of 95.0.
ammulu is NOT ELIGIBLE with a score of 73.0.
srivarsha is NOT ELIGIBLE with a score of 75.0.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted>
```

EXPLANATION:

This program collects student details name and score from the user and stores them in a list. It then uses a while loop to go through each student in the list one by one. For every student, it checks whether the score is greater than 75. If the score is above 75, the student is marked as eligible for the scholarship; otherwise, the student is marked as not eligible.

❖ TASK-5:

PROMPT:

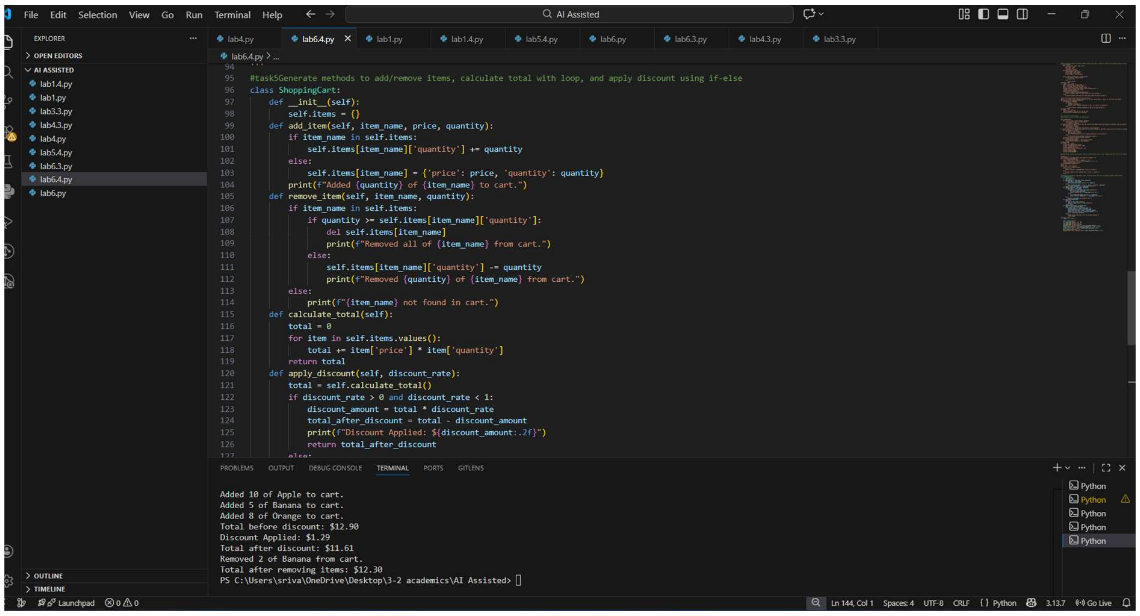
Generate methods to add/remove items, calculate total with loop, and apply discount using if-else

CODE:

```
class ShoppingCart:
    def __init__(self):
        self.items = {}
    def add_item(self, item_name, price, quantity):
        if item_name in self.items:
            self.items[item_name]['quantity'] += quantity
        else:
            self.items[item_name] = {'price': price, 'quantity': quantity}
        print(f"Added {quantity} of {item_name} to cart.")
    def remove_item(self, item_name, quantity):
        if item_name in self.items:
            if quantity >= self.items[item_name]['quantity']:
                del self.items[item_name]
                print(f"Removed all of {item_name} from cart.")
            else:
                self.items[item_name]['quantity'] -= quantity
                print(f"Removed {quantity} of {item_name} from cart.")
        else:
            print(f"{item_name} not found in cart.")
    def calculate_total(self):
        total = 0
        for item in self.items.values():
            total += item['price'] * item['quantity']
        return total
    def apply_discount(self, discount_rate):
        total = self.calculate_total()
        if discount_rate > 0 and discount_rate < 1:
            discount_amount = total * discount_rate
            total_after_discount = total - discount_amount
            print(f"Discount Applied: ${discount_amount:.2f}")
            return total_after_discount
        else:
            print("Invalid discount rate. No discount applied.")
            return total
```

```
# Example usage

if __name__ == "__main__":

    cart = ShoppingCart()

    cart.add_item("Apple", 0.5, 10)

    cart.add_item("Banana", 0.3, 5)

    cart.add_item("Orange", 0.8, 8)

    print(f"Total before discount: ${cart.calculate_total():.2f}")

    total_after_discount = cart.apply_discount(0.1)  # 10% discount

    print(f"Total after discount: ${total_after_discount:.2f}")

    cart.remove_item("Banana", 2)

    print(f"Total after removing items: ${cart.calculate_total():.2f}")
```

OUTPUT:



```python
#task5Generate methods to add/remove items, calculate total with loop, and apply discount using if-else
class ShoppingCart:
    def __init__(self):
        self.items = {}
    def add_item(self, item_name, price, quantity):
        if item_name in self.items:
            self.items[item_name]['quantity'] += quantity
        else:
            self.items[item_name] = {'price': price, 'quantity': quantity}
        print(f"Added {quantity} of {item_name} to cart.")
    def remove_item(self, item_name, quantity):
        if item_name in self.items:
            if quantity >= self.items[item_name]['quantity']:
                del self.items[item_name]
                print(f"Removed all of {item_name} from cart.")
            else:
                self.items[item_name]['quantity'] -= quantity
                print(f"Removed {quantity} of {item_name} from cart.")
        else:
            print(f"{item_name} not found in cart.")
    def calculate_total(self):
        total = 0
        for item in self.items.values():
            total += item['price'] * item['quantity']
        return total
    def apply_discount(self, discount_rate):
        total = self.calculate_total()
        if discount_rate > 0 and discount_rate < 1:
            discount_amount = total * discount_rate
            total_after_discount = total - discount_amount
            print(f"Discount Applied: ${discount_amount:.2f}")
            return total_after_discount
```

```
Added 10 of Apple to cart.
Added 5 of Banana to cart.
Added 8 of Orange to cart.
Total before discount: $12.90
Discount Applied: $1.29
Total after discount: $11.61
Removed 2 of Banana from cart.
Total after removing items: $12.30
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted>
```

EXPLANATION:

This program creates a ShoppingCart class that stores items with their price and quantity. It allows adding and removing items from the cart, calculates the total cost using a loop, and applies a discount using a condition. The example shows item addition, discount application, and total updates after removal.