# AI Assisted Coding

## Assignment 6.3

Name: R.Bharadwaj

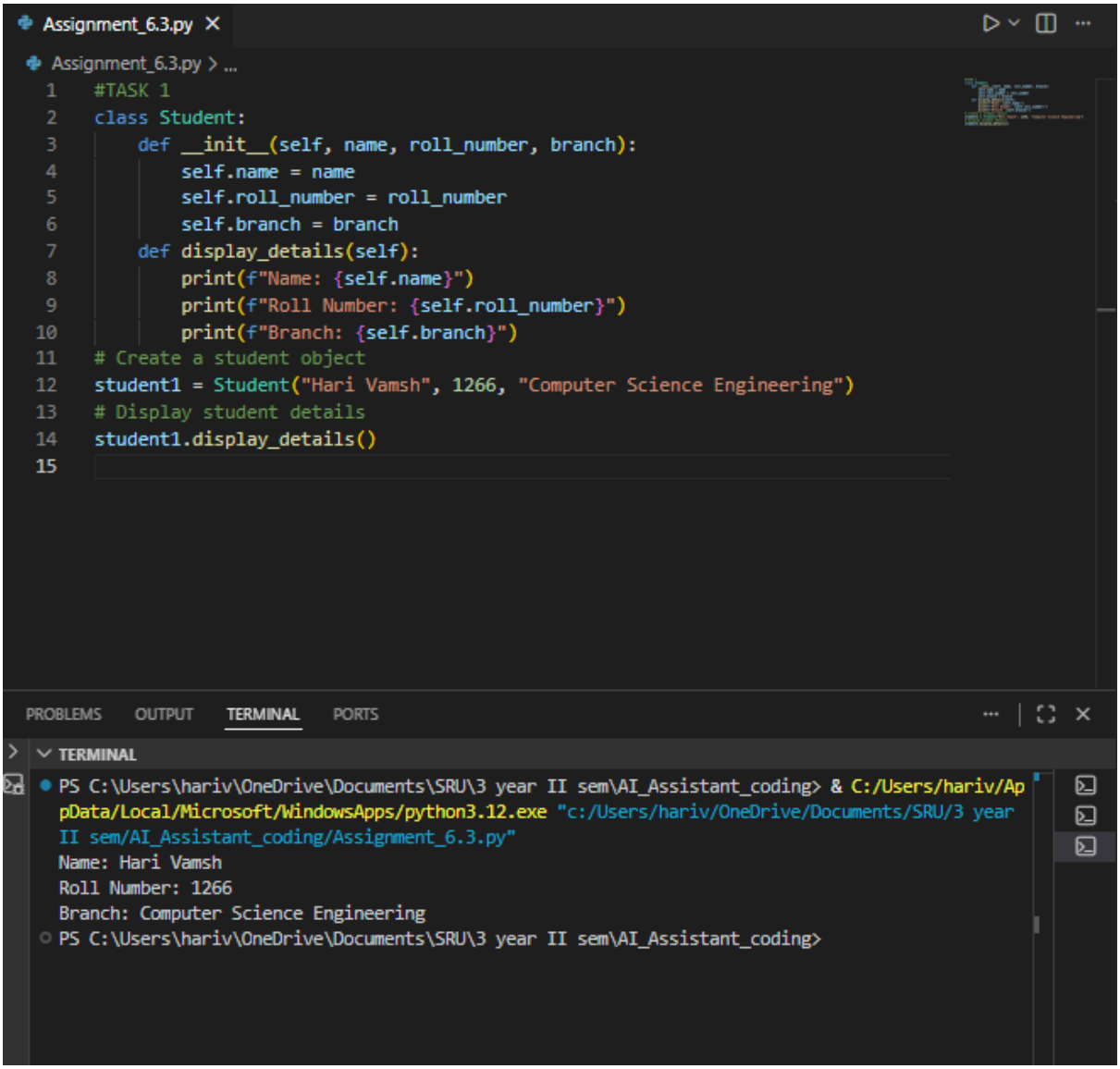
Hall ticket no: 2303A51610

Batch no: 19

**Task 1:**

**Prompt:**

Generate a Python Student class with attributes name, roll number, and branch, and a method to display student details.




**Explanation:**

The AI-generated code properly implements a Student class by following object-oriented programming concepts. The constructor is used to initialize the student's details, and the

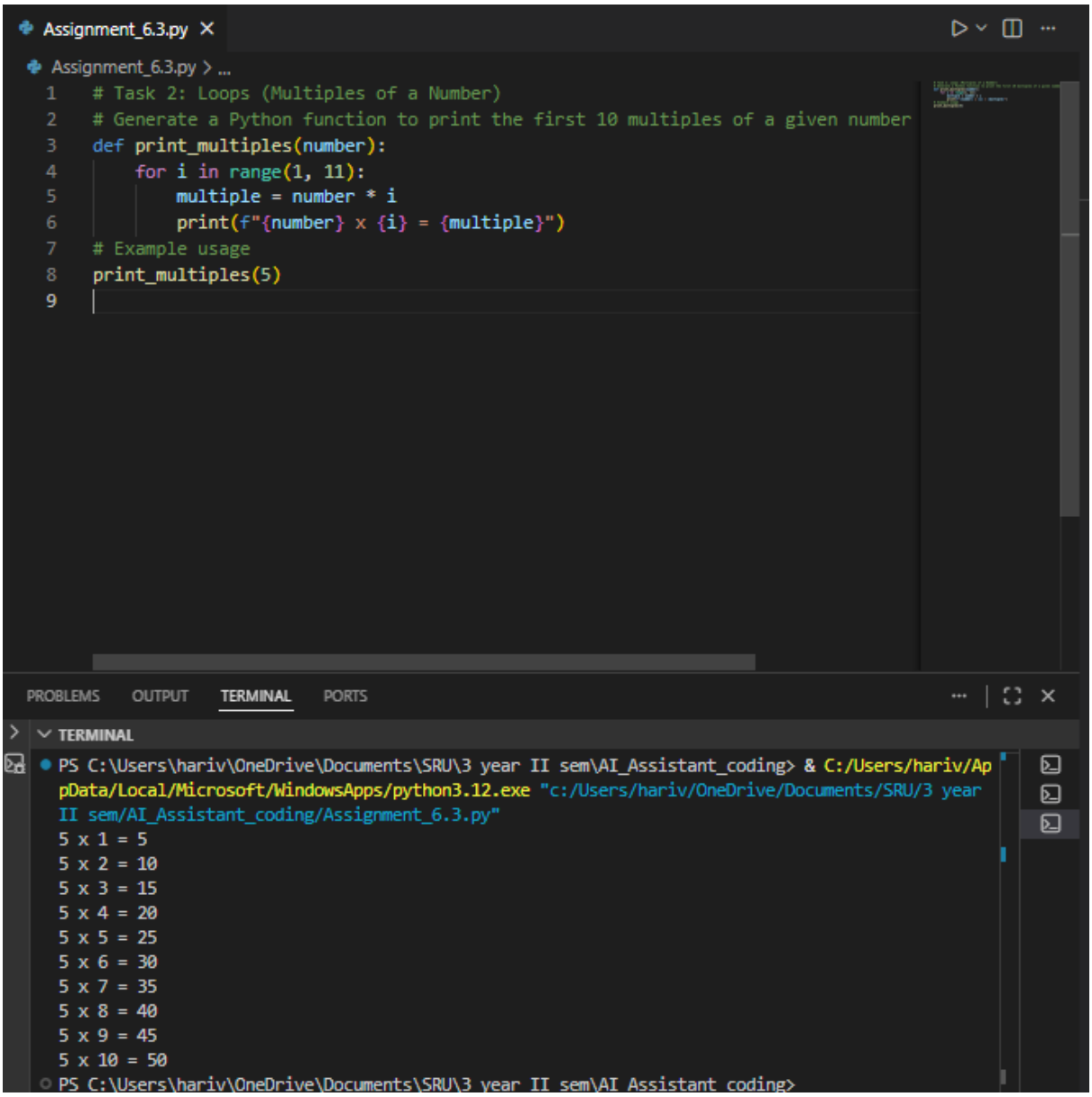
display_details() method neatly outputs this information. Overall, the class design is clean, easy to understand, and works as expected when a student object is created and run.

### Task 2: Loops (Multiples of a Number)

**Prompt:**

Generate a Python function to print the first 10 multiples of a given number using a for loop.

**Code & Output:**




**Explanation:**

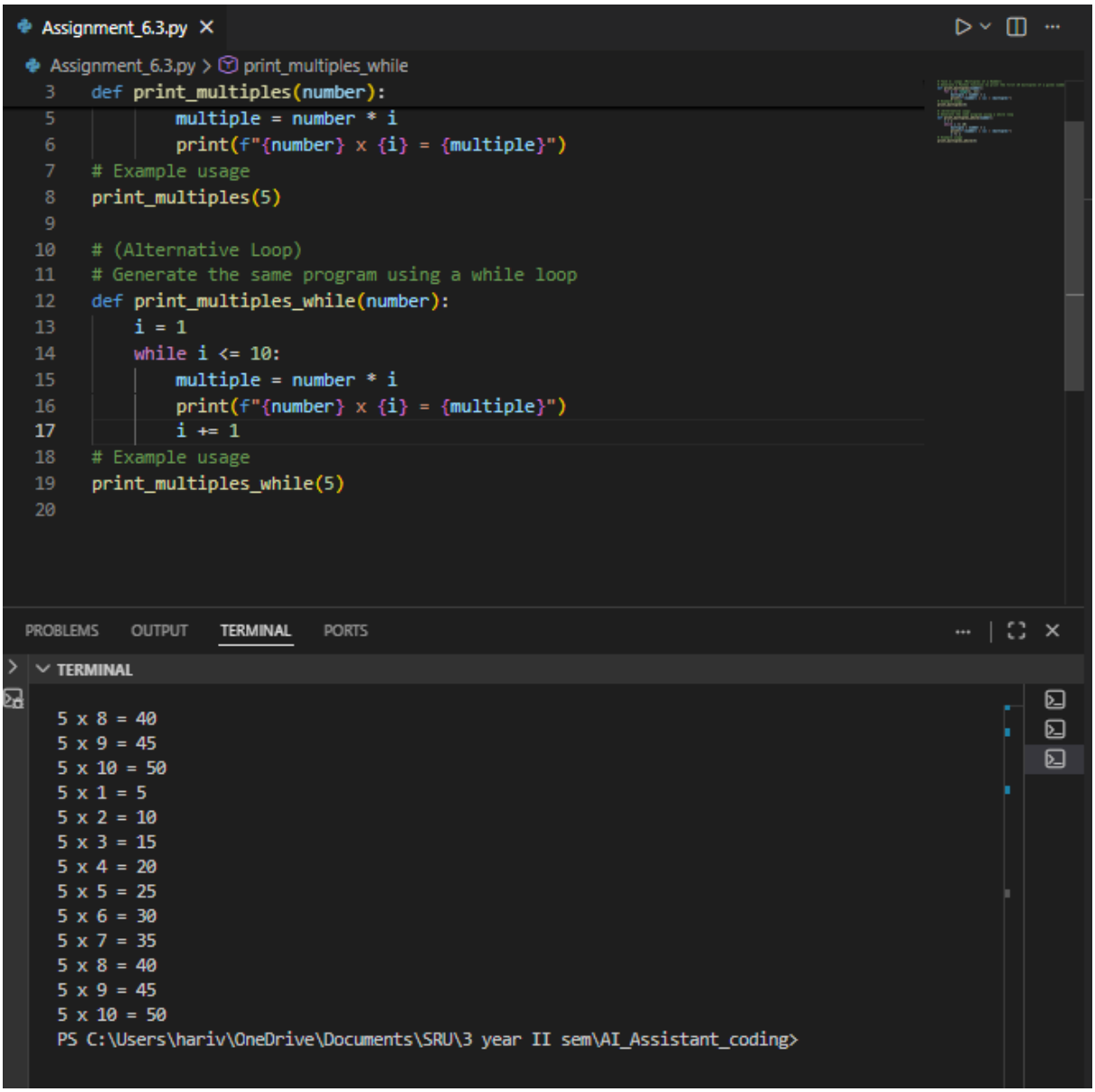
The AI-generated function uses a for loop to loop through values from 1 to 10 and display the multiples of the given number. During each loop cycle, the current value is multiplied by the input number to generate the result. The loop limits are clearly set, and the logic works

correctly to produce accurate output. Overall, the code is simple, efficient, and easy to read, making it well-suited for situations that require a fixed number of repetitions.

**Prompt (Alternative Loop):**

Generate the same program using a while loop

**Code & Output:**




**Explanation:**

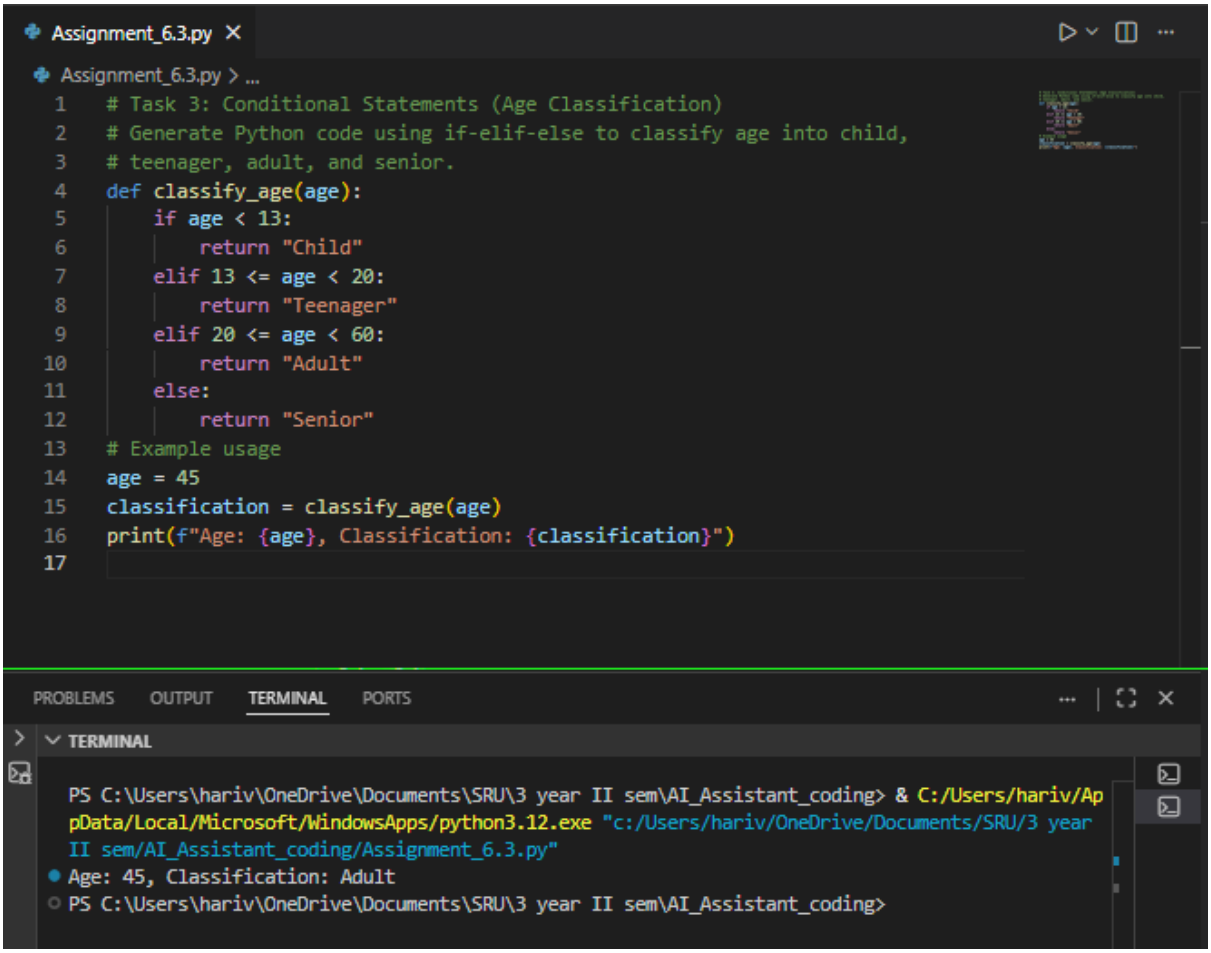
The while-loop version produces the same output as the for-loop version. While loops require manual control of the counter variable, making the for loop slightly cleaner and more readable for fixed iterations.

**Task 3: Conditional Statements (Age Classification)**

**Prompt:**

Generate Python code using if-elif-else to classify age into child, teenager, adult, and senior.

**Code & Output:**




**Explanation:**

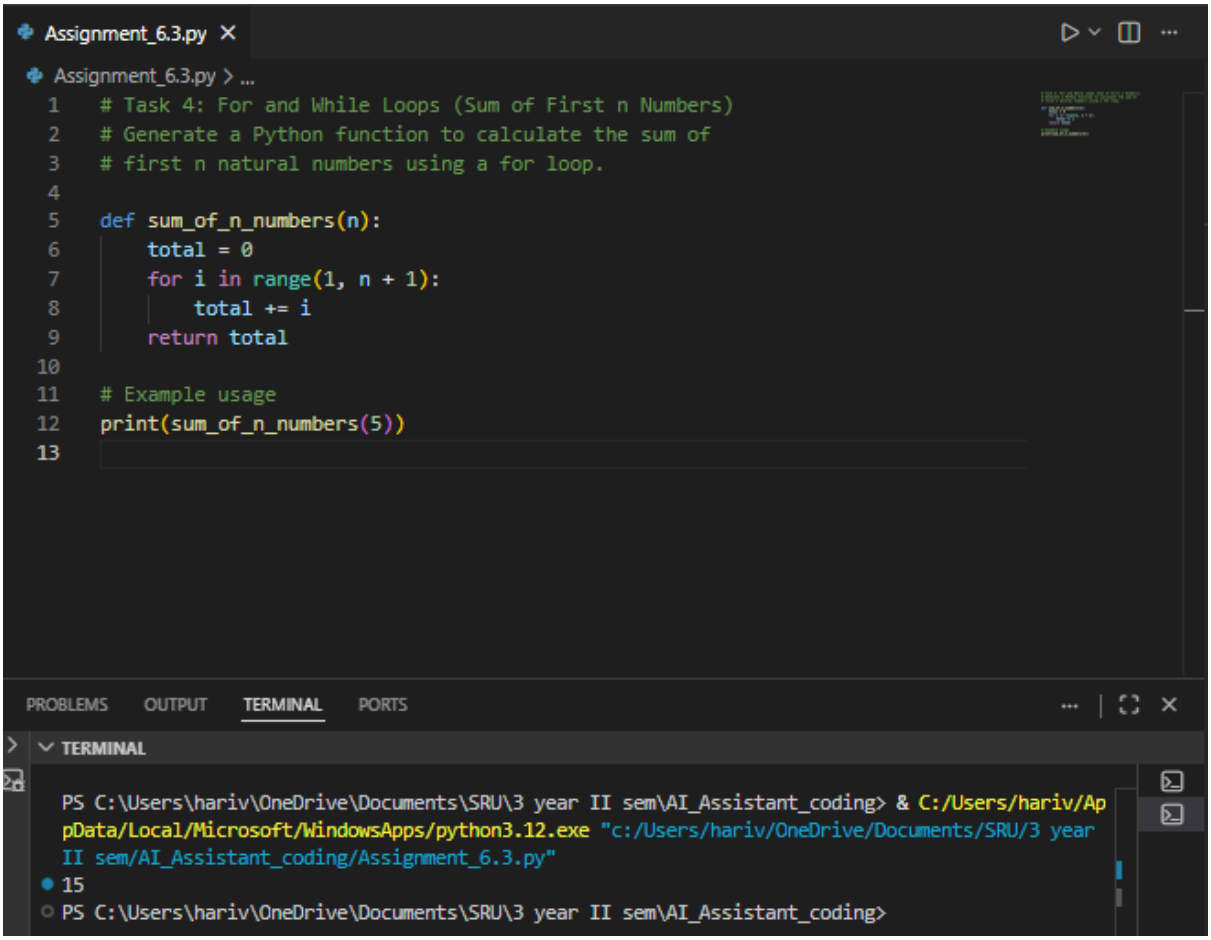
The AI-generated function applies nested if-elif-else statements to categorize different age groups. Each condition evaluates a particular age range in a logical sequence, ensuring that only one group is assigned for any given age. The approach is straightforward and accurate, making the logic easy to follow and verify. This clear structure makes the code beginner-friendly and practical for real-world age classification scenarios.

**Task 4: For and While Loops (Sum of First n Numbers)**

**Prompt:**

Generate a Python function to calculate the sum of first n natural numbers using a for loop.

**Code & Output:**




**Explanation:**

The AI-generated function finds the sum by looping through the numbers from 1 to n and adding each value to a running total. The loop is implemented correctly and gives the right result. This method is simple, easy to follow, and works well for small to medium values of n.

**Task 5: Classes (Bank Account Class)**

**Prompt:**

Generate a Python Bank Account class with methods to deposit, withdraw, and check balance.

**Code & Output:**

Assignment_6.3.py

```python
# Task 5: Classes (Bank Account Class)
# Generate a Python Bank Account class with methods to deposit, withdraw,
# and check balance.

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

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited: {amount}. New Balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew: {amount}. New Balance: {self.balance}")
        else:
            print("Insufficient balance or invalid withdrawal amount.")

    def check_balance(self):
        print(f"Account Holder: {self.account_holder}, Balance: {self.balance}"

# Example usage
account = BankAccount("John Doe", 1000)
account.check_balance()
account.deposit(500)
account.withdraw(200)
account.check_balance()
```

```
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/Assignment_6.3.py"
Account Holder: John Doe, Balance: 1000
Deposited: 500. New Balance: 1500
Withdrew: 200. New Balance: 1300
Account Holder: John Doe, Balance: 1300
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

**Explanation:**

The AI-generated Bank Account class effectively showcases object-oriented programming concepts. The constructor sets the initial balance, while the class methods handle deposits, withdrawals, and balance inquiries. Built-in conditional checks ensure that withdrawals are blocked when there are insufficient funds. Overall, the code is well-structured, logically correct, and easy to expand, making it a good fit for a simple banking system.