Assignment-6.3

Type your text

Hall ticket no:2303A51523

Task Description #1: Classes (Student Class)

Scenario

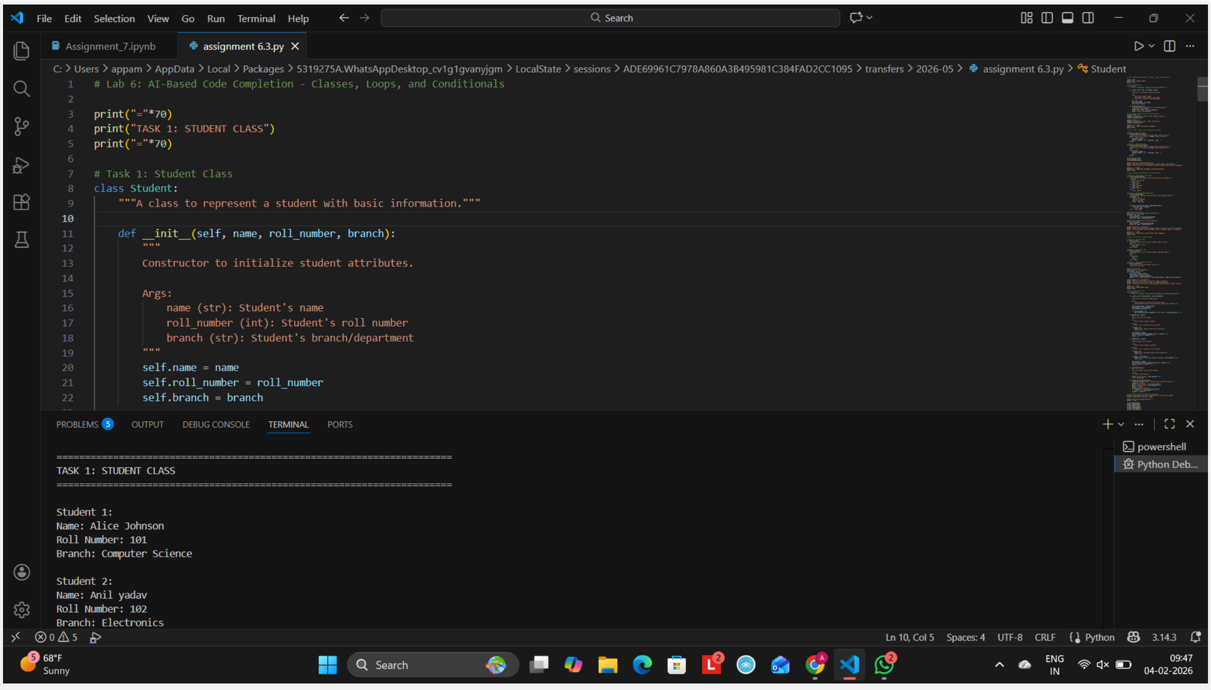
You are developing a simple student information management module.

Task

- Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
- The class should include attributes such as name, roll number, and branch.
- Add a method display_details() to print student information.
- Execute the code and verify the output.
- Analyze the code generated by the AI tool for correctness and clarity.

Prompt:develop a simple student information management module using python

In this task, an AI tool was used to generate a Python `Student` class with attributes such as name, roll number, and branch. The class included a constructor to initialize values and a `display_details()` method to print student information. The generated code was clear, correct, and followed basic object-oriented programming principles.



```python
# Lab 6: AI-Based Code Completion - Classes, Loops, and Conditionals

print("="*70)
print("TASK 1: STUDENT CLASS")
print("="*70)

# Task 1: Student Class
class Student:
    """A class to represent a student with basic information."""

    def __init__(self, name, roll_number, branch):
        """
        Constructor to initialize student attributes.

        Args:
            name (str): Student's name
            roll_number (int): Student's roll number
            branch (str): Student's branch/department
        """
        self.name = name
        self.roll_number = roll_number
        self.branch = branch
```

```
======================================================================
TASK 1: STUDENT CLASS
======================================================================

Student 1:
Name: Alice Johnson
Roll Number: 101
Branch: Computer Science

Student 2:
Name: Anil yadav
Roll Number: 102
Branch: Electronics
```

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

Scenario

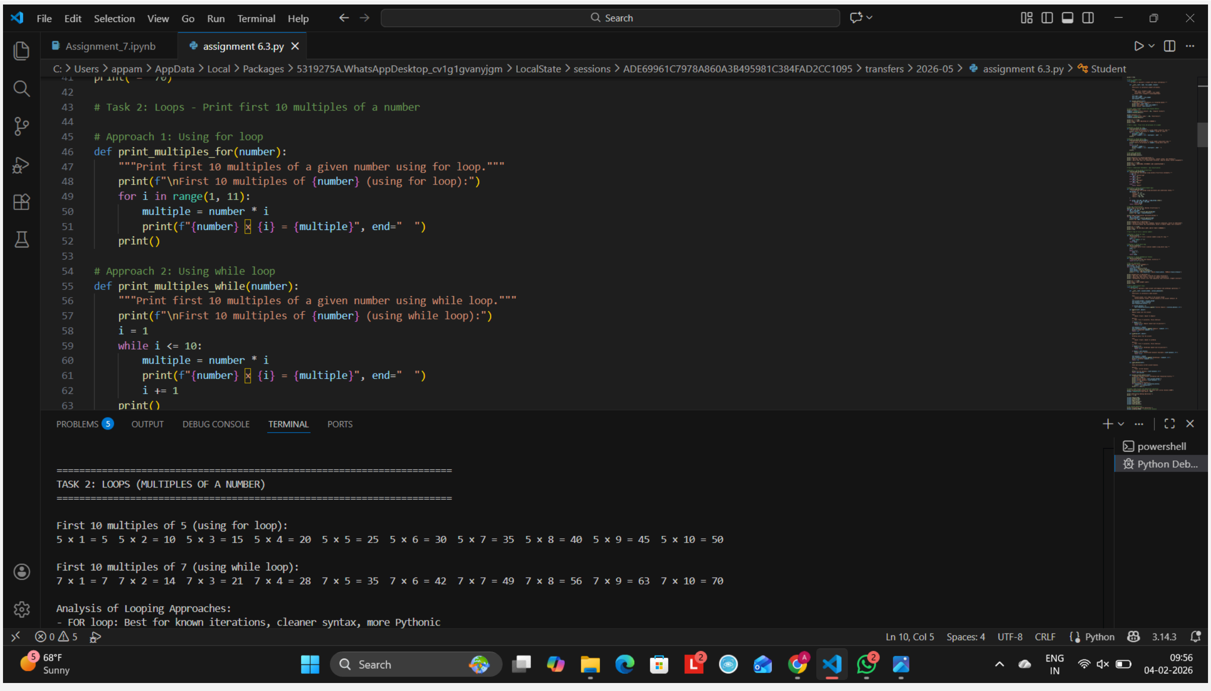
You are writing a utility function to display multiples of a given number.

Task

- Prompt the AI tool to generate a function that prints the first 10 multiples of a given number using a loop.
- Analyze the generated loop logic.
- Ask the AI to generate the same functionality using another controlled looping structure (e.g., while instead of for).

Prompt:print first 10 multiples of a number

AI assistance was used to create a function that prints the first ten multiples of a given number using a `for` loop. The same logic was also generated using a `while` loop. Both approaches produced correct results, with the `for` loop being simpler and more readable, while the `while` loop offered more control.



```python
    print("=" * 70)

# Task 2: Loops - Print first 10 multiples of a number

# Approach 1: Using for loop
def print_multiples_for(number):
    """Print first 10 multiples of a given number using for loop."""
    print(f"\nFirst 10 multiples of {number} (using for loop):")
    for i in range(1, 11):
        multiple = number * i
        print(f"{number} × {i} = {multiple}", end="  ")
    print()

# Approach 2: Using while loop
def print_multiples_while(number):
    """Print first 10 multiples of a given number using while loop."""
    print(f"\nFirst 10 multiples of {number} (using while loop):")
    i = 1
    while i <= 10:
        multiple = number * i
        print(f"{number} × {i} = {multiple}", end="  ")
        i += 1
    print()
```

```
======================================================================
TASK 2: LOOPS (MULTIPLES OF A NUMBER)
======================================================================

First 10 multiples of 5 (using for loop):
5 × 1 = 5  5 × 2 = 10  5 × 3 = 15  5 × 4 = 20  5 × 5 = 25  5 × 6 = 30  5 × 7 = 35  5 × 8 = 40  5 × 9 = 45  5 × 10 = 50

First 10 multiples of 7 (using while loop):
7 × 1 = 7  7 × 2 = 14  7 × 3 = 21  7 × 4 = 28  7 × 5 = 35  7 × 6 = 42  7 × 7 = 49  7 × 8 = 56  7 × 9 = 63  7 × 10 = 70

Analysis of Looping Approaches:
- FOR loop: Best for known iterations, cleaner syntax, more Pythonic
```

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

Task

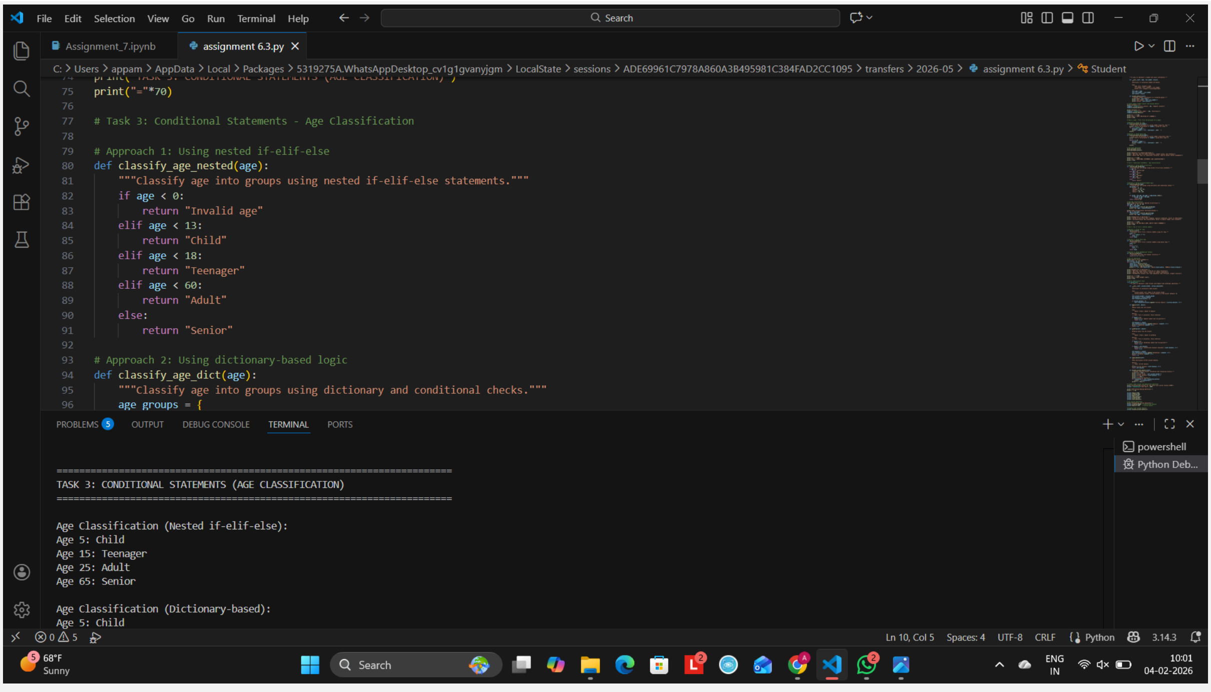
- Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).

- Analyze the generated conditions and logic.

- Ask the AI to generate the same classification using alternative conditional structures (e.g., simplified conditions or dictionary-based logic).

Prompt:conditional statements and age qualification using nested if-elif-else

explanation:The AI generated a function using `if-elif-else` statements to classify age into child, teenager, adult, and senior categories. The conditions were logically ordered and easy to understand. An alternative approach showed how the same classification could be done using simplified or dictionary-based logic.



```python
print("TASK 3: CONDITIONAL STATEMENTS (AGE CLASSIFICATION)")
print("="*70)

# Task 3: Conditional Statements - Age Classification

# Approach 1: Using nested if-elif-else
def classify_age_nested(age):
    """Classify age into groups using nested if-elif-else statements."""
    if age < 0:
        return "Invalid age"
    elif age < 13:
        return "Child"
    elif age < 18:
        return "Teenager"
    elif age < 60:
        return "Adult"
    else:
        return "Senior"

# Approach 2: Using dictionary-based logic
def classify_age_dict(age):
    """Classify age into groups using dictionary and conditional checks."""
    age_groups = {
```

```
======================================================================
TASK 3: CONDITIONAL STATEMENTS (AGE CLASSIFICATION)
======================================================================

Age Classification (Nested if-elif-else):
Age 5: Child
Age 15: Teenager
Age 25: Adult
Age 65: Senior

Age Classification (Dictionary-based):
Age 5: Child
```

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

t

task4:Scenario

You need to calculate the sum of the first n natural numbers.

Task

- Use AI assistance to generate a sum_to_n() function using a for loop.

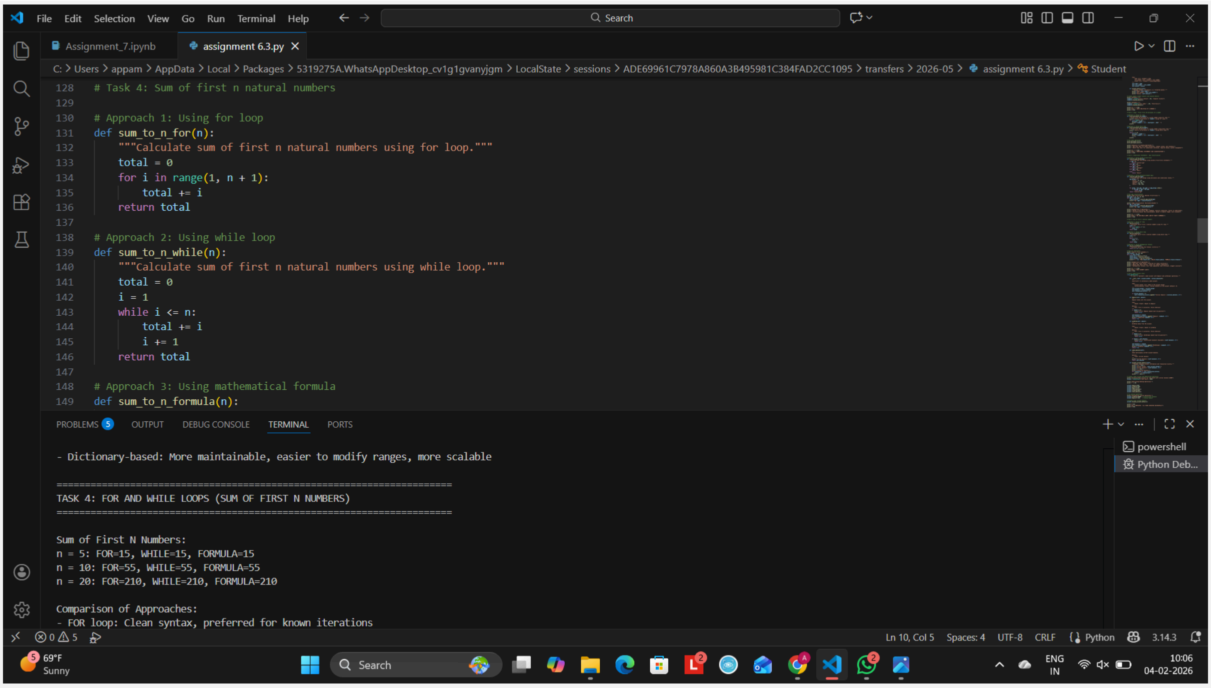
- Analyze the generated code.

- Ask the AI to suggest an alternative implementation using a while loop or a mathematical

Formula.

Prompt:give first sum of n natural numbers using for loop and while loop

explanation:AI was used to generate a function that calculates the sum of the first `n` natural numbers using a `for` loop. Alternative solutions using a `while` loop and a mathematical formula were also suggested. The formula-based approach was the most efficient, while loop-based solutions were easier forbeginners.



```python
# Task 4: Sum of first n natural numbers

# Approach 1: Using for loop
def sum_to_n_for(n):
    """Calculate sum of first n natural numbers using for loop."""
    total = 0
    for i in range(1, n + 1):
        total += i
    return total

# Approach 2: Using while loop
def sum_to_n_while(n):
    """Calculate sum of first n natural numbers using while loop."""
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total

# Approach 3: Using mathematical formula
def sum_to_n_formula(n):
```

```
- Dictionary-based: More maintainable, easier to modify ranges, more scalable

======================================================================
TASK 4: FOR AND WHILE LOOPS (SUM OF FIRST N NUMBERS)
======================================================================

Sum of First N Numbers:
n = 5: FOR=15, WHILE=15, FORMULA=15
n = 10: FOR=55, WHILE=55, FORMULA=55
n = 20: FOR=210, WHILE=210, FORMULA=210

Comparison of Approaches:
- FOR loop: Clean syntax, preferred for known iterations
```

File Edit Selection View Go Run Terminal Help
Search
Assignment_7.ipynb
assignment 6.3.py
C: > Users > appam > AppData > Local > Packages > 5319275A.WhatsAppDesktop_cv1g1gvanyjgm > LocalState > sessions > ADE69961C7978A860A3B495981C384FAD2CC1095 > transfers > 2026-05 > assignment 6.3.py > Student
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python Deb...
Ln 10, Col 5 Spaces: 4 UTF-8 CRLF Python 3.14.3
69°F Sunny
Search
ENG IN
10:06 04-02-2026

task5:Classes (Bank Account Class)

Scenario

You are designing a basic banking application.

Task

- Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(),

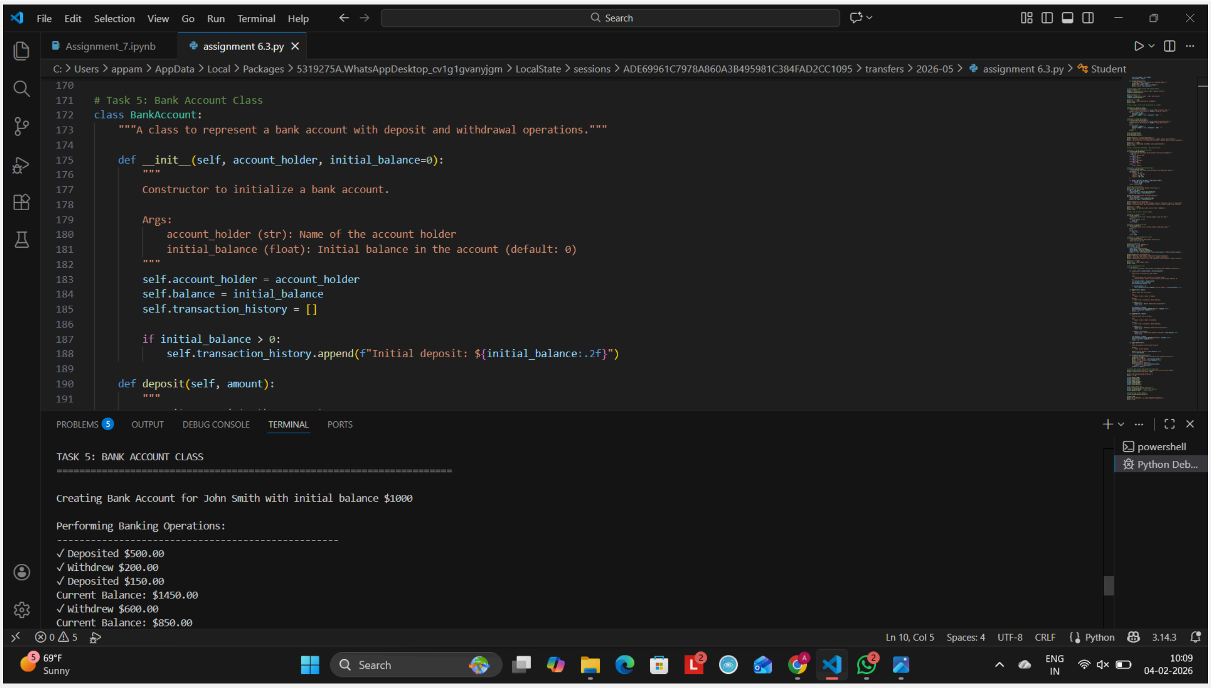
and check_balance().

-

Analyze the AI-generated class structure and logic.

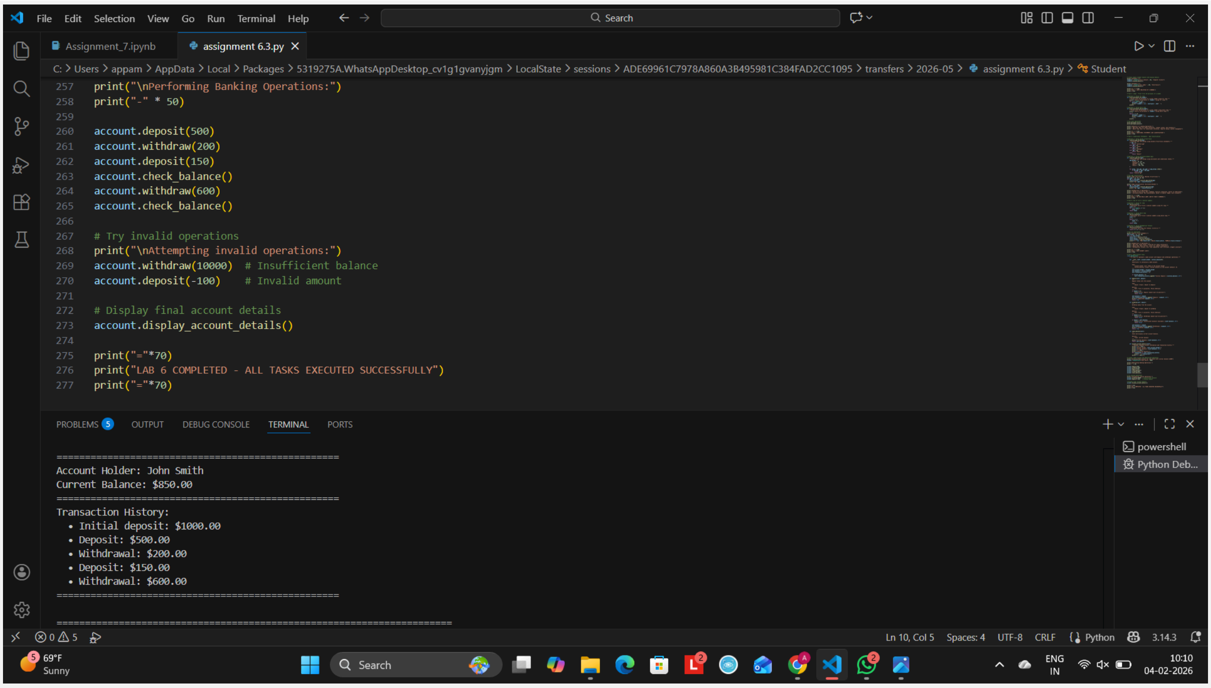
- Add meaningful comments and explain the working of the code

Prompt:create bank account class and demonstrate



```python

# Task 5: Bank Account Class
class BankAccount:
    """A class to represent a bank account with deposit and withdrawal operations."""

    def __init__(self, account_holder, initial_balance=0):
        """
        Constructor to initialize a bank account.

        Args:
            account_holder (str): Name of the account holder
            initial_balance (float): Initial balance in the account (default: 0)
        """
        self.account_holder = account_holder
        self.balance = initial_balance
        self.transaction_history = []

        if initial_balance > 0:
            self.transaction_history.append(f"Initial deposit: ${initial_balance:.2f}")

    def deposit(self, amount):
        """
```

```
TASK 5: BANK ACCOUNT CLASS
======================================================================

Creating Bank Account for John Smith with initial balance $1000

Performing Banking Operations:
--------------------------------------------------
✓ Deposited $500.00
✓ Withdrew $200.00
✓ Deposited $150.00
Current Balance: $1450.00
✓ Withdrew $600.00
Current Balance: $850.00
```



```python
print("\nPerforming Banking Operations:")
print("-" * 50)

account.deposit(500)
account.withdraw(200)
account.deposit(150)
account.check_balance()
account.withdraw(600)
account.check_balance()

# Try invalid operations
print("\nAttempting invalid operations:")
account.withdraw(10000)  # Insufficient balance
account.deposit(-100)    # Invalid amount

# Display final account details
account.display_account_details()

print("="*70)
print("LAB 6 COMPLETED - ALL TASKS EXECUTED SUCCESSFULLY")
print("="*70)
```

```
==================================================
Account Holder: John Smith
Current Balance: $850.00
==================================================
Transaction History:
  • Initial deposit: $1000.00
  • Deposit: $500.00
  • Withdrawal: $200.00
  • Deposit: $150.00
  • Withdrawal: $600.00
==================================================

======================================================================
```

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

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