ASSIGNMENT- 6.3

Name: EDDE.POOJITHA

HT.No: 2303A51356

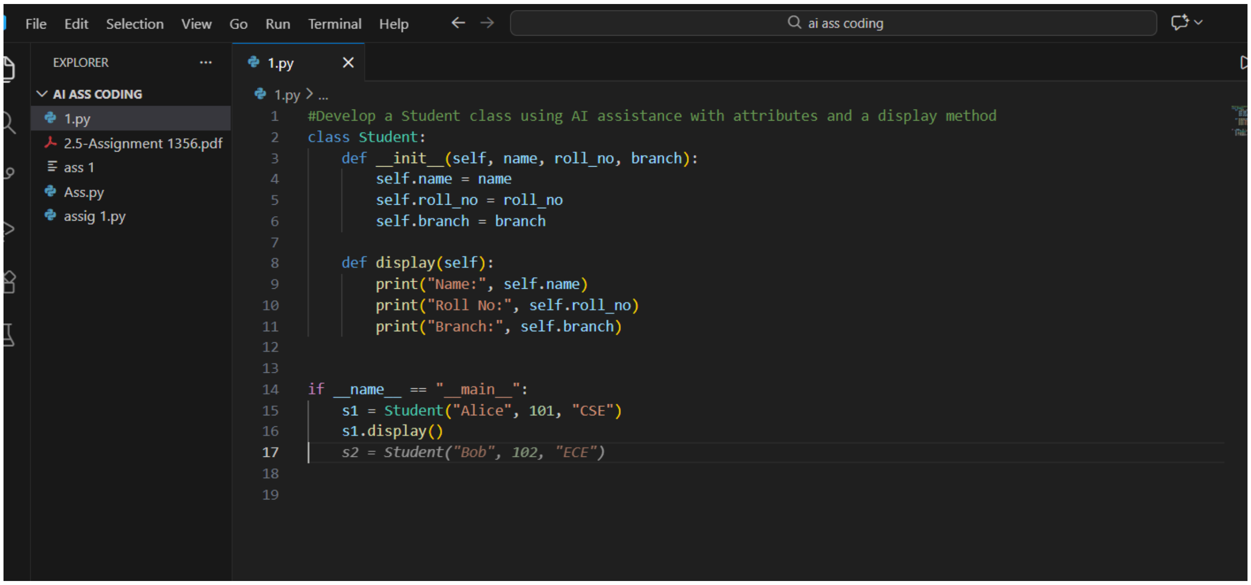
Batch: 20

Task 1: Classes – Student Class

Develop a Student class using AI assistance with attributes and a display method

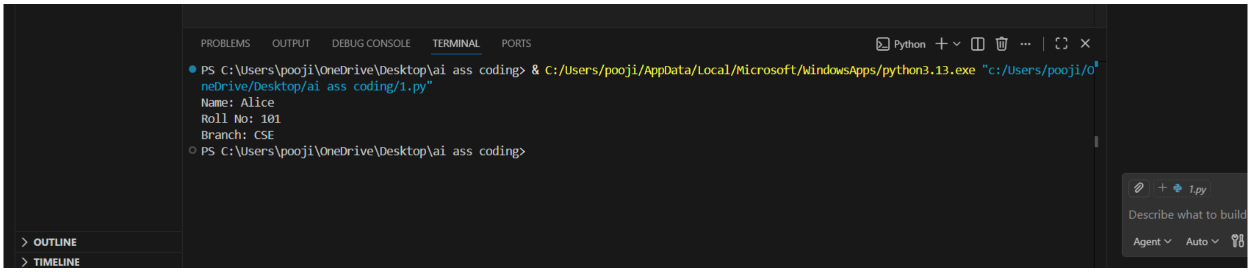
.Prompt: #Generate a Python Student class with name, roll number, and branch. Include a method to display student details..

Code:



```python
#Develop a Student class using AI assistance with attributes and a display method
class Student:
    def __init__(self, name, roll_no, branch):
        self.name = name
        self.roll_no = roll_no
        self.branch = branch

    def display(self):
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Branch:", self.branch)


if __name__ == "__main__":
    s1 = Student("Alice", 101, "CSE")
    s1.display()
    s2 = Student("Bob", 102, "ECE")
```

Result:



```
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/OneDrive/Desktop/ai ass coding/1.py"
Name: Alice
Roll No: 101
Branch: CSE
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>
```

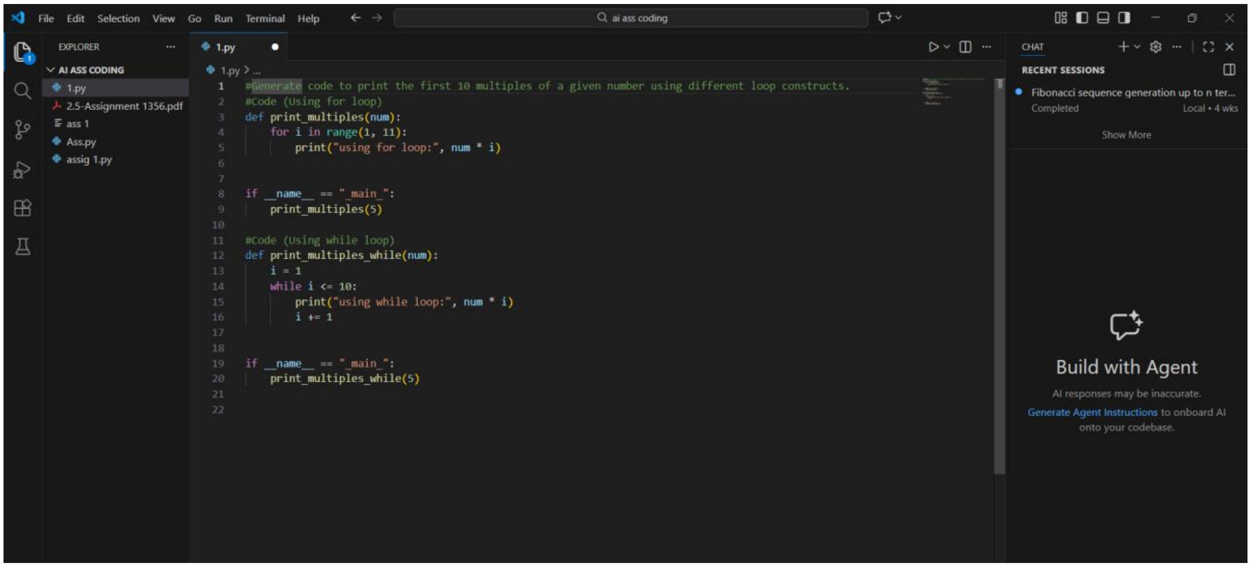
Observation:
The AI-generated class structure is clear and logically organized. The constructor correctly initializes attributes, and the display method outputs student details in a readable format. The code is simple, correct, and suitable for beginner-level object-oriented programming.

Task 2: Loops – Multiples of a Number. Generate code to print the first 10 multiples of a given number using different loop constructs.

Prompt: #Generate Python code to print the first 10

multiples of a number using a loop.

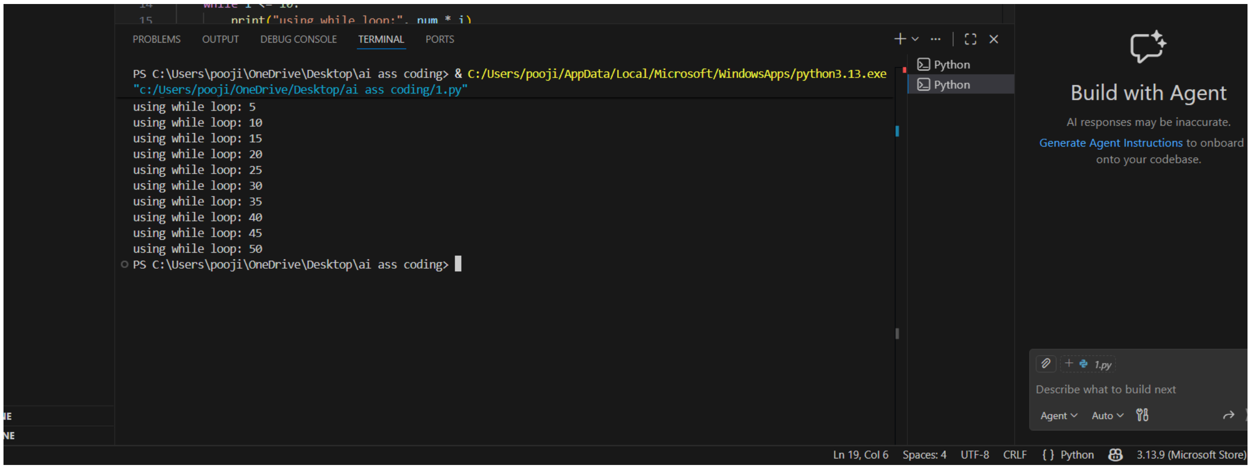
Code:



File Edit Selection View Go Run Terminal Help
ai ass coding
EXPLORER
AI ASS CODING
1.py
2.5-Assignment 1356.pdf
ass 1
Ass.py
assig 1.py
CHAT
RECENT SESSIONS
Fibonacci sequence generation up to n ter...
Completed
Local • 4 wks
Show More
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

```python
#Generate code to print the first 10 multiples of a given number using different loop constructs.
#Code (Using for loop)
def print_multiples(num):
    for i in range(1, 11):
        print("using for loop:", num * i)


if __name__ == "__main__":
    print_multiples(5)

#Code (Using while loop)
def print_multiples_while(num):
    i = 1
    while i <= 10:
        print("using while loop:", num * i)
        i += 1


if __name__ == "__main__":
    print_multiples_while(5)
```

Result:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Python
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard
onto your codebase.
1.py
Describe what to build next
Agent
Auto
Ln 19, Col 6 Spaces: 4 UTF-8 CRLF { } Python 3.13.9 (Microsoft Store)

```
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe
"c:/Users/pooji/OneDrive/Desktop/ai ass coding/1.py"
using while loop: 5
using while loop: 10
using while loop: 15
using while loop: 20
using while loop: 25
using while loop: 30
using while loop: 35
using while loop: 40
using while loop: 45
using while loop: 50
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> 
```

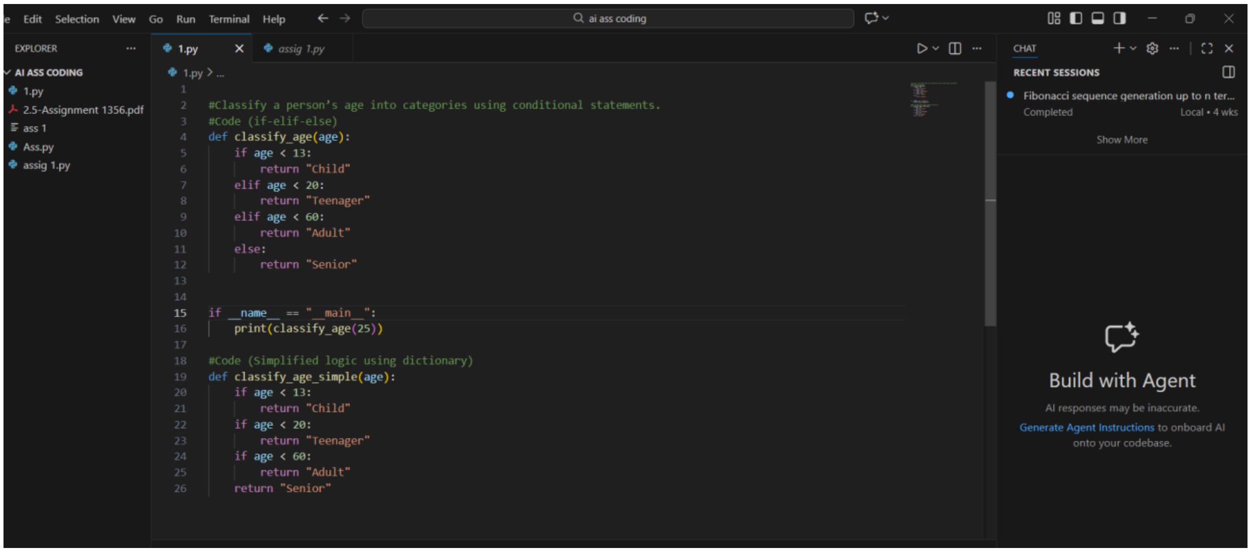
Observation:

Both loop implementations correctly generate the required output. The for-loop version is more concise and readable, while the while-loop version provides better insight into loop control and iteration. AI suggestions for both approaches are correct and efficient.

Task 3: Conditional Statements – Age Classification. Classify a person's age into categories using conditional statements.

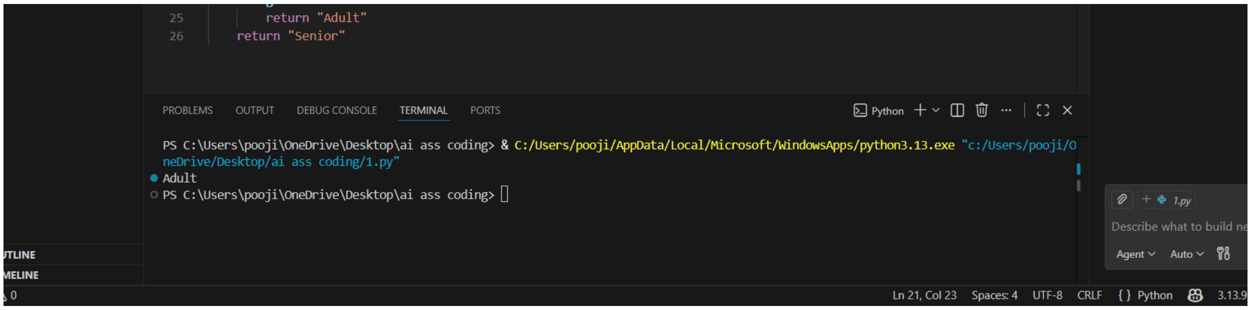
Prompt: # Generate Python code to classify age into child, teenager, adult, and senior using if-elif-else..

Code:



```python

#Classify a person’s age into categories using conditional statements.
#Code (if-elif-else)
def classify_age(age):
    if age < 13:
        return "Child"
    elif age < 20:
        return "Teenager"
    elif age < 60:
        return "Adult"
    else:
        return "Senior"


if __name__ == "__main__":
    print(classify_age(25))

#Code (Simplified logic using dictionary)
def classify_age_simple(age):
    if age < 13:
        return "Child"
    if age < 20:
        return "Teenager"
    if age < 60:
        return "Adult"
    return "Senior"
```

Result:



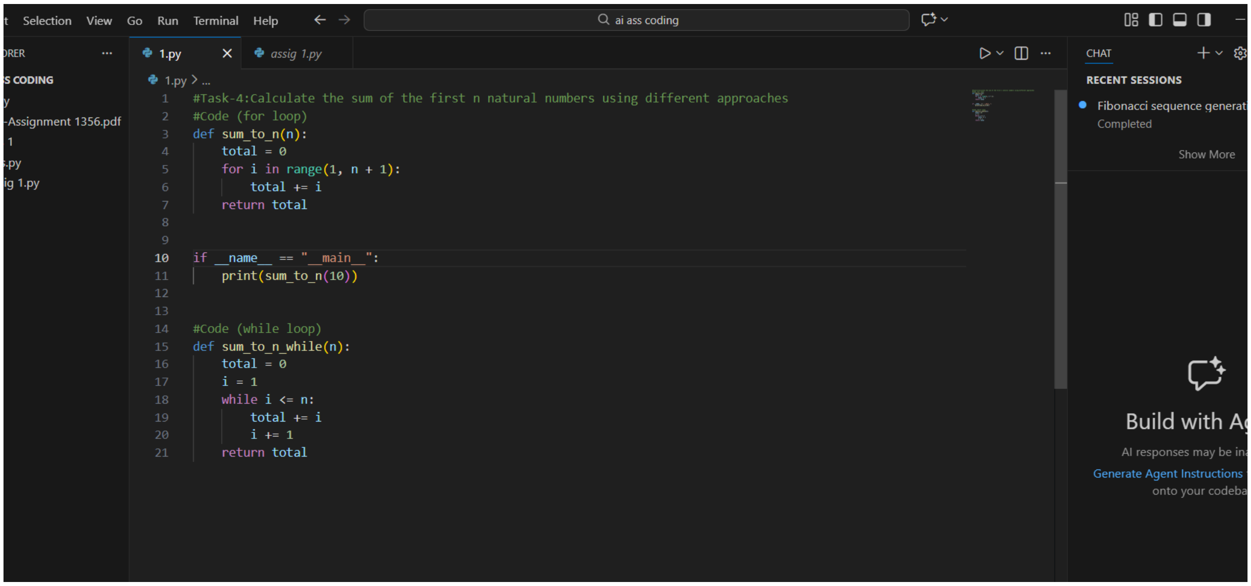
```python
        return "Adult"
    return "Senior"
```

```
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/OneDrive/Desktop/ai ass coding/1.py"
Adult
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> 
```

Observation:
The AI-generated conditions correctly classify age groups. The if-elif-else structure is clear and readable, while the simplified version reduces nesting and improves clarity. Both approaches are logically sound.
Task 4: For and While Loops – Sum of First n Numbers. Calculate the sum of the first n natural numbers using different approaches.

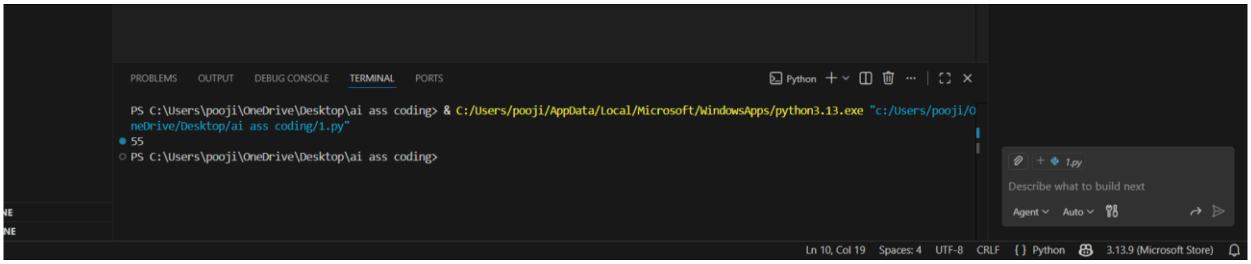
Prompt: #Generate Python code to find the sum of the first n natural numbers using loops.

Code:



```python
#Task-4:Calculate the sum of the first n natural numbers using different approaches
#Code (for loop)
def sum_to_n(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total


if __name__ == "__main__":
    print(sum_to_n(10))


#Code (while loop)
def sum_to_n_while(n):
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total
```

Result:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
1.py
Describe what to build next
Agent Auto
Ln 10, Col 19 Spaces: 4 UTF-8 CRLF {} Python 3.13.9 (Microsoft Store)

```
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/O
neDrive/Desktop/ai ass coding/1.py"
55
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>
```

Observation

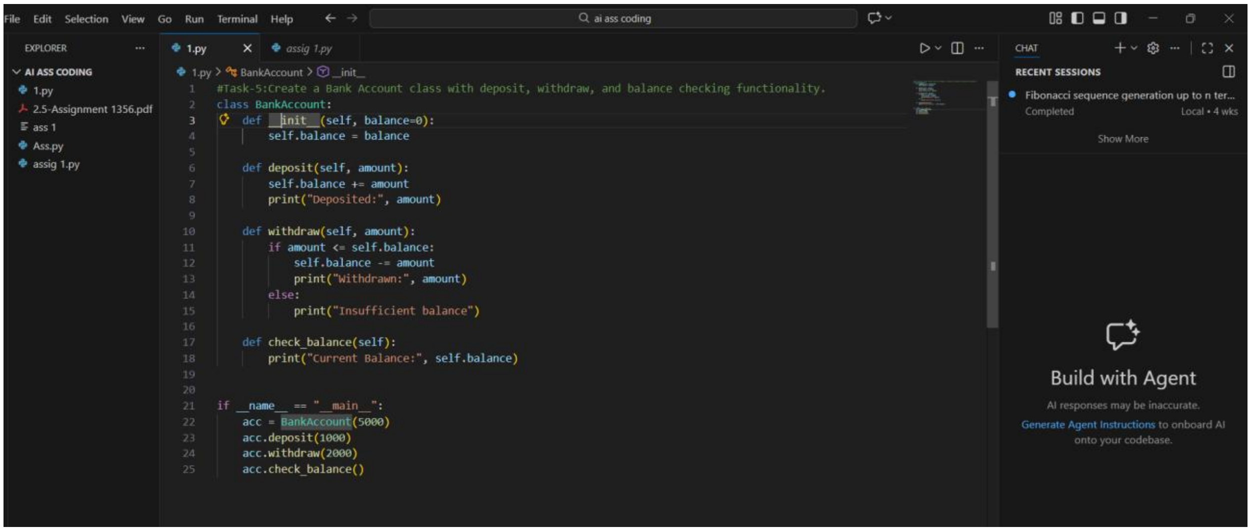
Both loop-based solutions produce the correct result. The for-loop version is more concise, while the while-loop version offers explicit control over iteration. AI-generated logic is correct and easy to understand

Task 5: Classes – Bank Account Class

Create a Bank Account class with deposit, withdraw, and balance checking functionality.

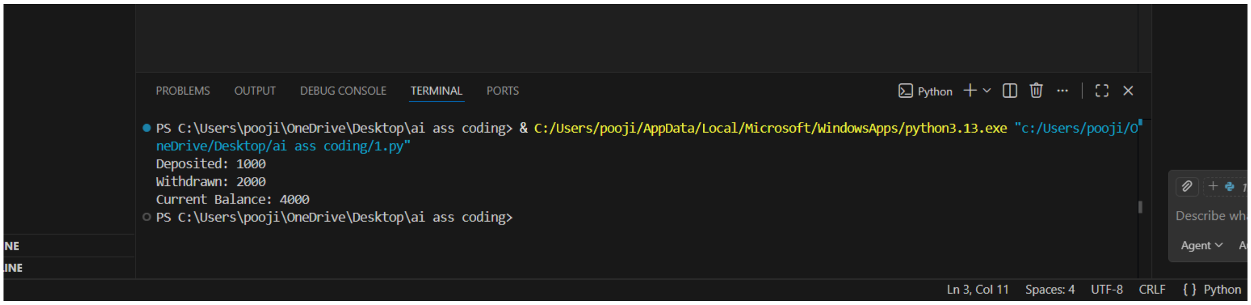
Prompt: #Generate a Python Bank Account class with deposit, withdraw, and check balance methods.

Code:



```python
#Task-5:Create a Bank Account class with deposit, withdraw, and balance checking functionality.
class BankAccount:
    def __init__(self, balance=0):
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposited:", amount)

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawn:", amount)
        else:
            print("Insufficient balance")

    def check_balance(self):
        print("Current Balance:", self.balance)


if __name__ == "__main__":
    acc = BankAccount(5000)
    acc.deposit(1000)
    acc.withdraw(2000)
    acc.check_balance()
```

Result:



```
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/O
neDrive/Desktop/ai ass coding/1.py"
Deposited: 1000
Withdrawn: 2000
Current Balance: 4000
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>
```

Observation:

The AI-generated class structure is well organized and logically correct. Methods perform expected operations, and balance updates are accurate. The code is readable, maintainable, and suitable for a basic banking application.