Friday Lab: AI-Assisted Coding Using Python

Hall Ticket: 2303a51722

Subject: AI-Assisted Programming

Tool Used: GitHub Copilot / AI Code Completion

Learning Outcomes

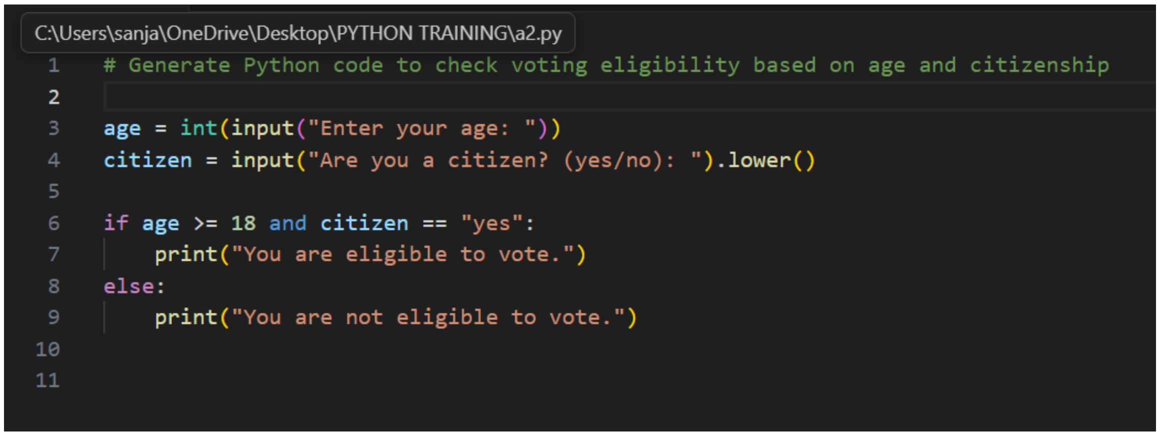
- LO1: Use AI-based code completion tools to generate Python code involving conditionals and loops
- LO2: Interpret and explain AI-generated code
- LO3: Identify logical issues or inefficiencies
- LO4: Optimize AI-generated code
- LO5: Demonstrate ethical and responsible use of AI tools

Task 1: AI-Based Code Completion for Conditional Eligibility Check

Scenario:
You are developing a simple eligibility-checking module for a voting system.

Prompt / Comments:

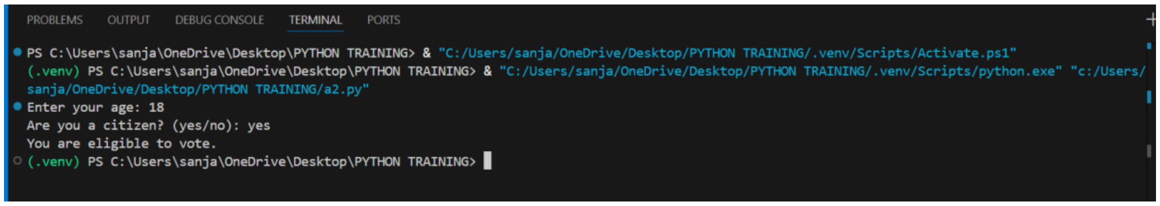
# Generate Python code to check voting eligibility based on age and citizenship



C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\a2.py

```python
# Generate Python code to check voting eligibility based on age and citizenship

age = int(input("Enter your age: "))
citizen = input("Are you a citizen? (yes/no): ").lower()

if age >= 18 and citizen == "yes":
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

Output:



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/a2.py"
Enter your age: 18
Are you a citizen? (yes/no): yes
You are eligible to vote.
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING>
```

Observation:

- The program correctly checks voting eligibility using conditional statements.
- Both age and citizenship conditions are validated.
- The logic is simple and easy to understand.
- AI-generated code was reviewed and validated.

Task 2: AI-Based Code Completion for Loop-Based String Processing

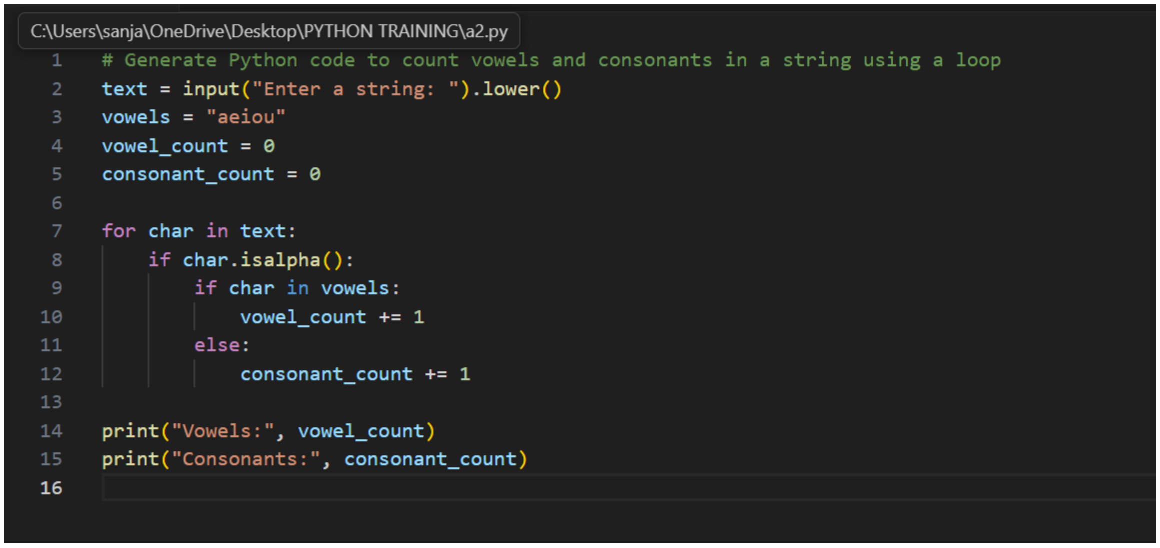
Scenario:

You are working on a text-processing utility that analyzes user input.

Prompt / Comments:

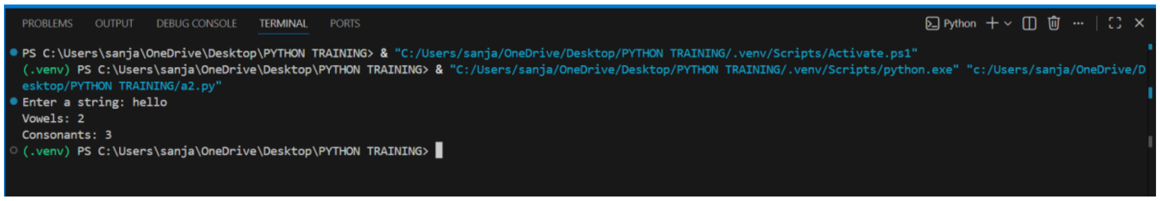
# Generate Python code to count vowels and consonants in a string using a loop

Code:



```
C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\a2.py
# Generate Python code to count vowels and consonants in a string using a loop
text = input("Enter a string: ").lower()
vowels = "aeiou"
vowel_count = 0
consonant_count = 0

for char in text:
    if char.isalpha():
        if char in vowels:
            vowel_count += 1
        else:
            consonant_count += 1

print("Vowels:", vowel_count)
print("Consonants:", consonant_count)
```

Output:



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/a2.py"
Enter a string: hello
Vowels: 2
Consonants: 3
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING>
```

Observation:

- The program correctly counts vowels and consonants.
- Non-alphabetic characters are ignored.
- Time complexity is O(n).
- AI-assisted code was ethically and responsibly used.

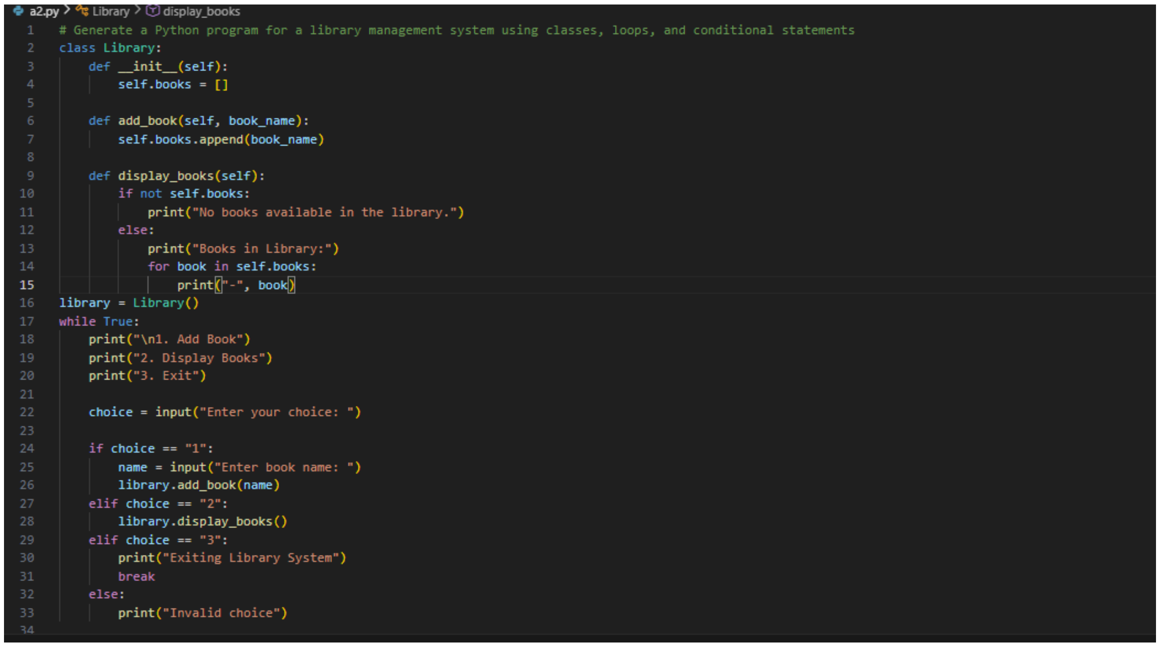
Task 3: AI-Assisted Code Completion Reflection Task

Scenario:
You are developing a small library management system using AI assistance.

Prompt / Comments:

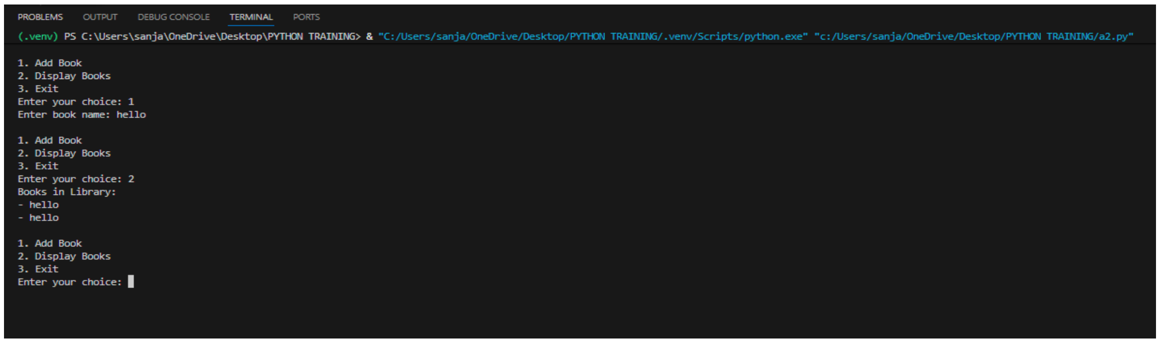
# Generate a Python program for a library management system using classes, loops, and conditional statements

Code:



```python
# Generate a Python program for a library management system using classes, loops, and conditional statements
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book_name):
        self.books.append(book_name)

    def display_books(self):
        if not self.books:
            print("No books available in the library.")
        else:
            print("Books in Library:")
            for book in self.books:
                print("-", book)
library = Library()
while True:
    print("\n1. Add Book")
    print("2. Display Books")
    print("3. Exit")

    choice = input("Enter your choice: ")

    if choice == "1":
        name = input("Enter book name: ")
        library.add_book(name)
    elif choice == "2":
        library.display_books()
    elif choice == "3":
        print("Exiting Library System")
        break
    else:
        print("Invalid choice")
```

Output:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/a2.py"

1. Add Book
2. Display Books
3. Exit
Enter your choice: 1
Enter book name: hello

1. Add Book
2. Display Books
3. Exit
Enter your choice: 2
Books in Library:
- hello
- hello

1. Add Book
2. Display Books
3. Exit
Enter your choice: 
```

Observation:

- The AI-generated program uses classes, loops, and conditional statements correctly.
- The Library class manages book data efficiently.
- The menu-driven loop allows repeated operations.
- AI suggestions were logical and required minimal correction.

Reflection:

AI-assisted coding helped in quickly generating a structured program. Human understanding was required to verify logic and test outputs. AI is best used as a supportive coding assistant.

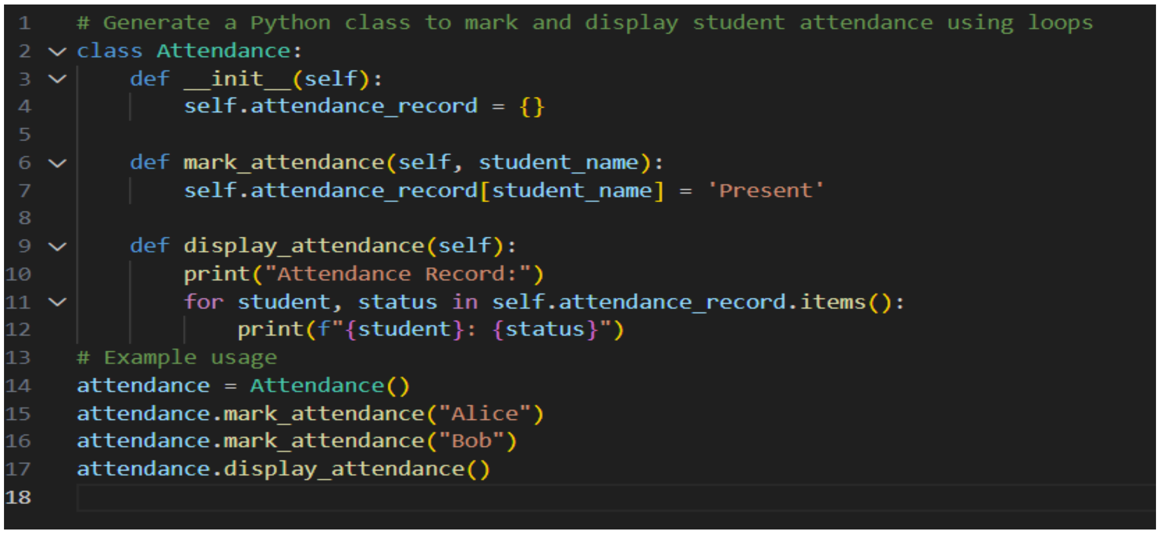
Task 4: AI-Assisted Code Completion for Class-Based Attendance System

Scenario:
You are developing a simple attendance tracking system for students.

Prompt / Comments:

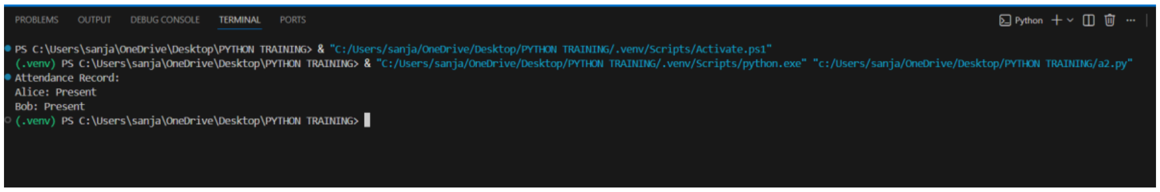
# Generate a Python class to mark and display student attendance using loops

Code:



```python
# Generate a Python class to mark and display student attendance using loops
class Attendance:
    def __init__(self):
        self.attendance_record = {}

    def mark_attendance(self, student_name):
        self.attendance_record[student_name] = 'Present'

    def display_attendance(self):
        print("Attendance Record:")
        for student, status in self.attendance_record.items():
            print(f"{student}: {status}")
# Example usage
attendance = Attendance()
attendance.mark_attendance("Alice")
attendance.mark_attendance("Bob")
attendance.display_attendance()
```

Output:



```
PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/a2.py"
Attendance Record:
Alice: Present
Bob: Present
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> 
```

Observation:

- The Attendance class stores attendance data using a dictionary.
- Loops are used to display attendance records.
- Conditional logic correctly shows Present or Absent.
- The code is reusable and easy to maintain.

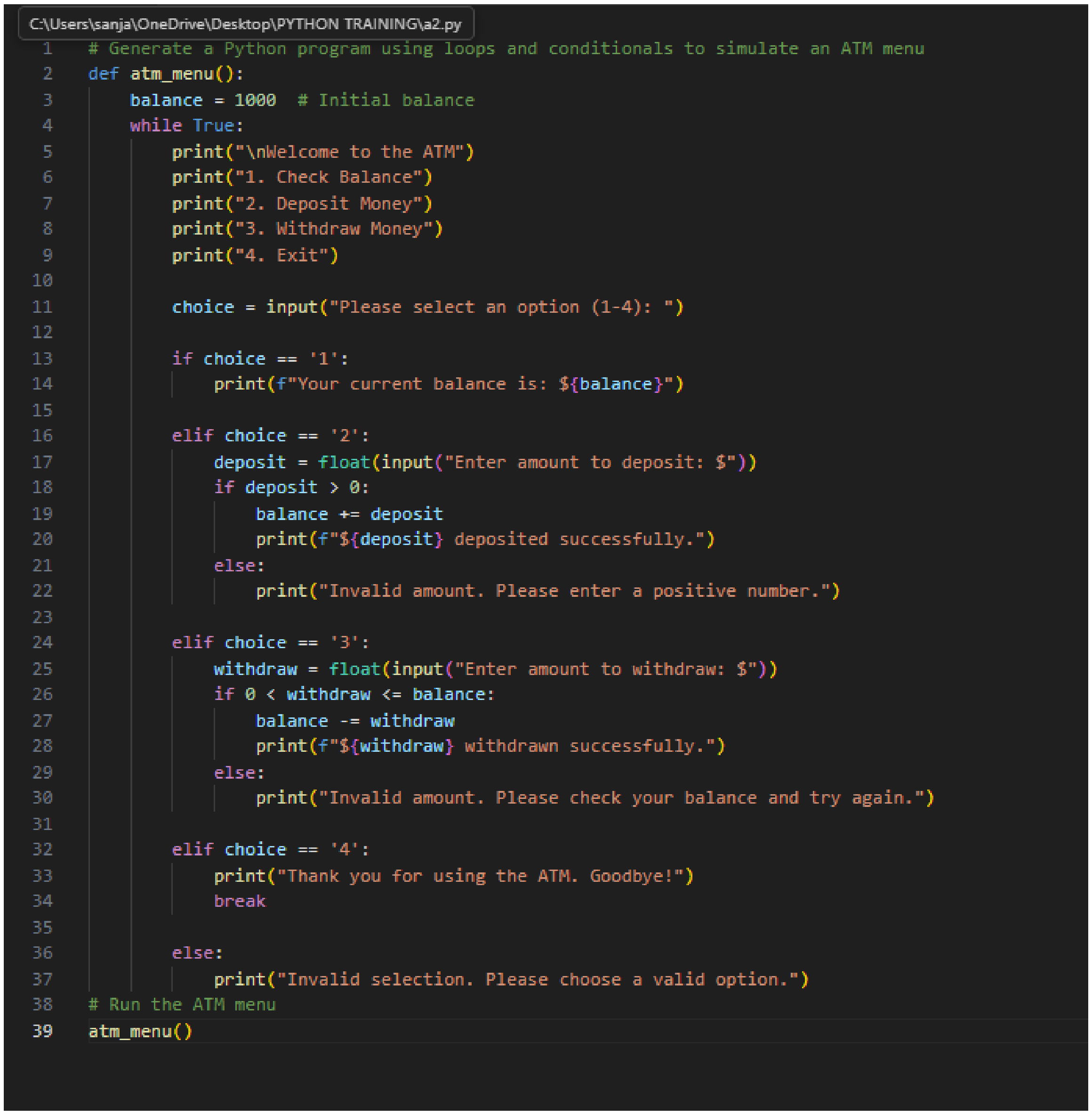
Task 5: AI-Based Code Completion for Conditional Menu Navigation

Scenario:
You are simulating a simple ATM system using Python.

Prompt / Comments:

# Generate a Python program using loops and conditionals to simulate an ATM menu

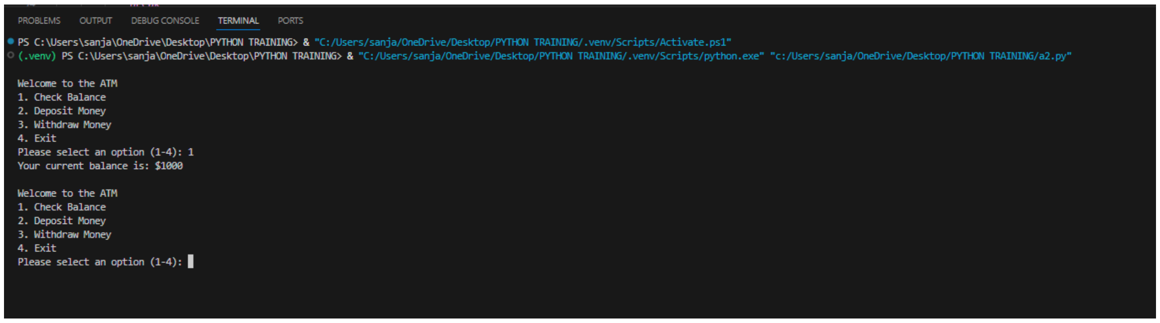
Code:



C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\a2.py

```python
# Generate a Python program using loops and conditionals to simulate an ATM menu
def atm_menu():
    balance = 1000  # Initial balance
    while True:
        print("\nWelcome to the ATM")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

        choice = input("Please select an option (1-4): ")

        if choice == '1':
            print(f"Your current balance is: ${balance}")

        elif choice == '2':
            deposit = float(input("Enter amount to deposit: $"))
            if deposit > 0:
                balance += deposit
                print(f"${deposit} deposited successfully.")
            else:
                print("Invalid amount. Please enter a positive number.")

        elif choice == '3':
            withdraw = float(input("Enter amount to withdraw: $"))
            if 0 < withdraw <= balance:
                balance -= withdraw
                print(f"${withdraw} withdrawn successfully.")
            else:
                print("Invalid amount. Please check your balance and try again.")

        elif choice == '4':
            print("Thank you for using the ATM. Goodbye!")
            break

        else:
            print("Invalid selection. Please choose a valid option.")
# Run the ATM menu
atm_menu()
```

Output:



```
PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/a2.py"

Welcome to the ATM
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4): 1
Your current balance is: $1000

Welcome to the ATM
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4):
```

Observation:

- The program correctly simulates an ATM menu.

- User options are handled using conditional statements.
- The loop allows continuous transactions.
- AI-generated logic was verified and tested.