AI Assisted Coding

Assignment 6.5

Name: V.Harivamsh

Hall ticket no: 2303A51266

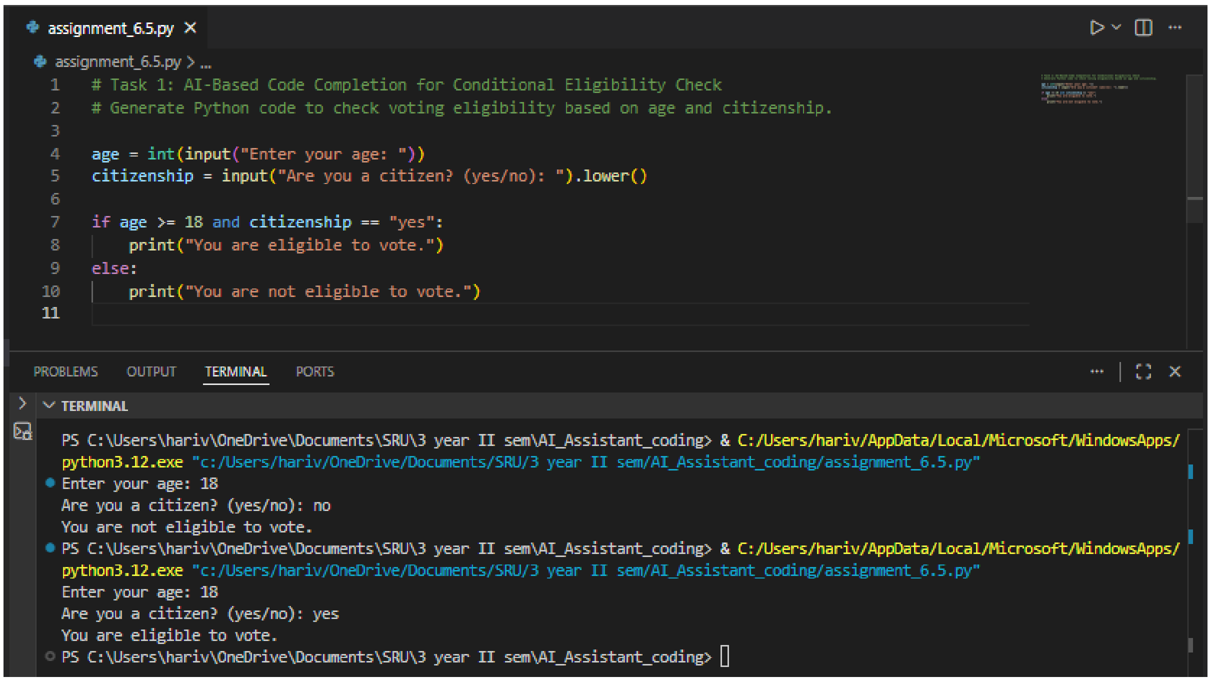
Batch no: 19

Task 1:

Prompt:

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

Code & Output:



```python
# Task 1: AI-Based Code Completion for Conditional Eligibility Check
# Generate Python code to check voting eligibility based on age and citizenship.

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

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

```

```
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_6.5.py"
Enter your age: 18
Are you a citizen? (yes/no): no
You are not eligible to vote.
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_6.5.py"
Enter your age: 18
Are you a citizen? (yes/no): yes
You are eligible to vote.
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> 
```

Explanation:

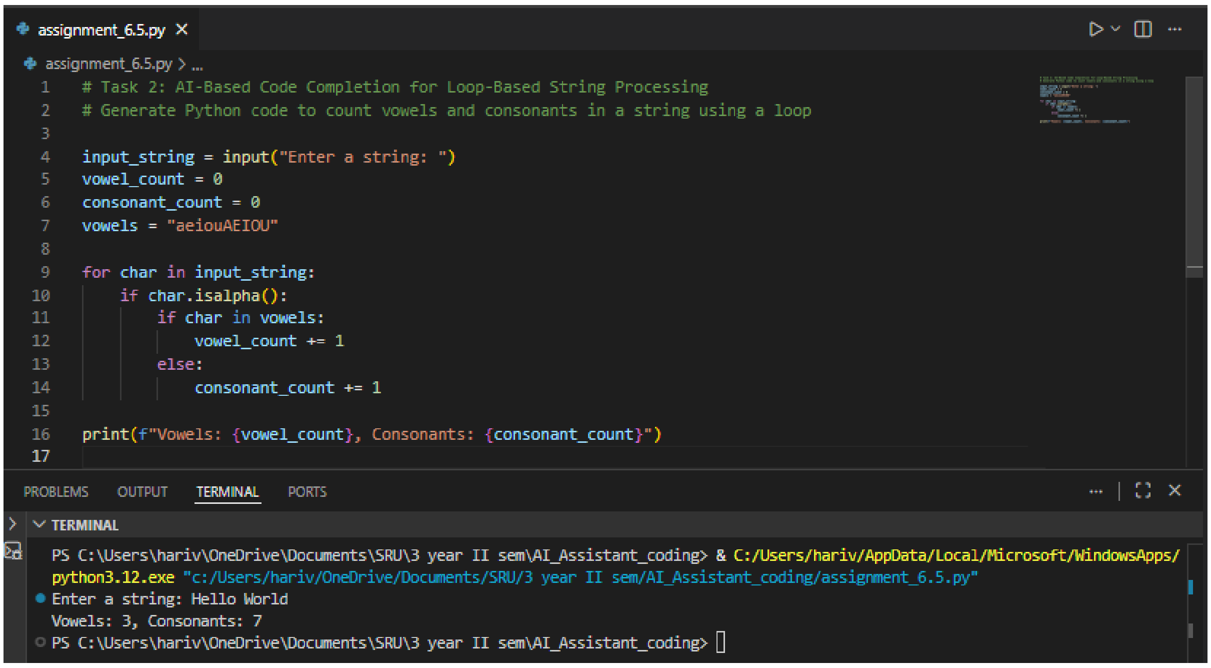
The AI-generated code applies conditional statements to determine voting eligibility. It checks if the person's age is 18 or above and confirms their citizenship status. Only when both conditions are satisfied is the user considered eligible. This shows an accurate use of conditional logic produced through AI-based code generation.

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

Prompt:

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

Code & Output:



```python
# Task 2: AI-Based Code Completion for Loop-Based String Processing
# Generate Python code to count vowels and consonants in a string using a loop

input_string = input("Enter a string: ")
vowel_count = 0
consonant_count = 0
vowels = "aeiouAEIOU"

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

print(f"Vowels: {vowel_count}, Consonants: {consonant_count}")
```

```
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_6.5.py"
Enter a string: Hello World
Vowels: 3, Consonants: 7
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

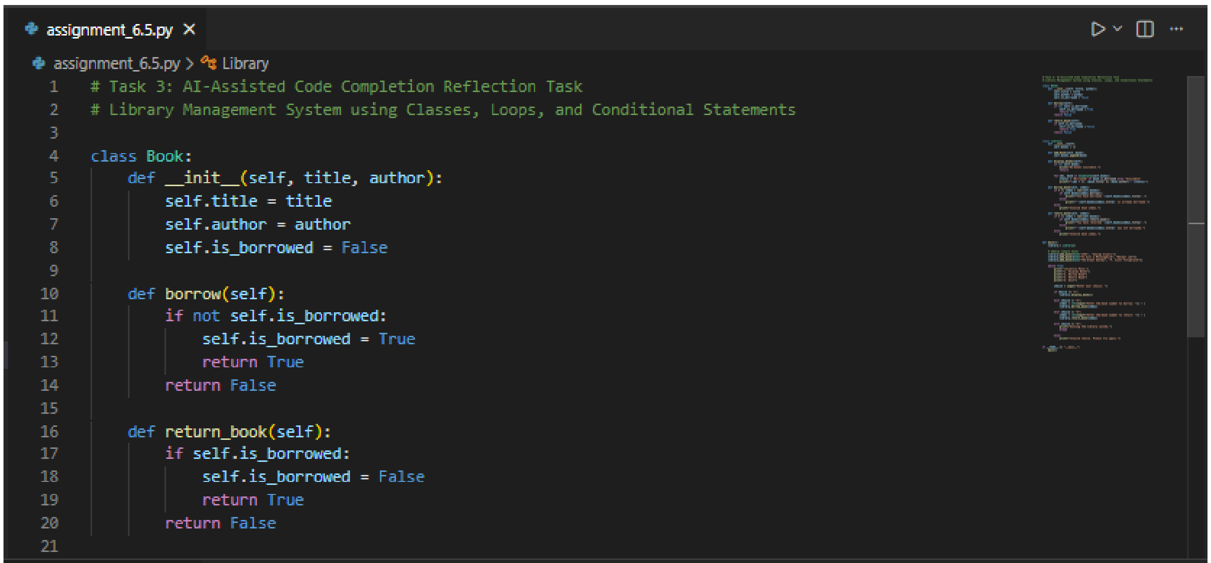
The AI-generated code handles the input string by looping through each character one by one. It checks whether each character is a vowel or a consonant and counts only the alphabetic characters, ignoring any non-letter symbols. The final output confirms that the logic functions correctly as intended.

Task 3: AI-Assisted Code Completion Reflection Task

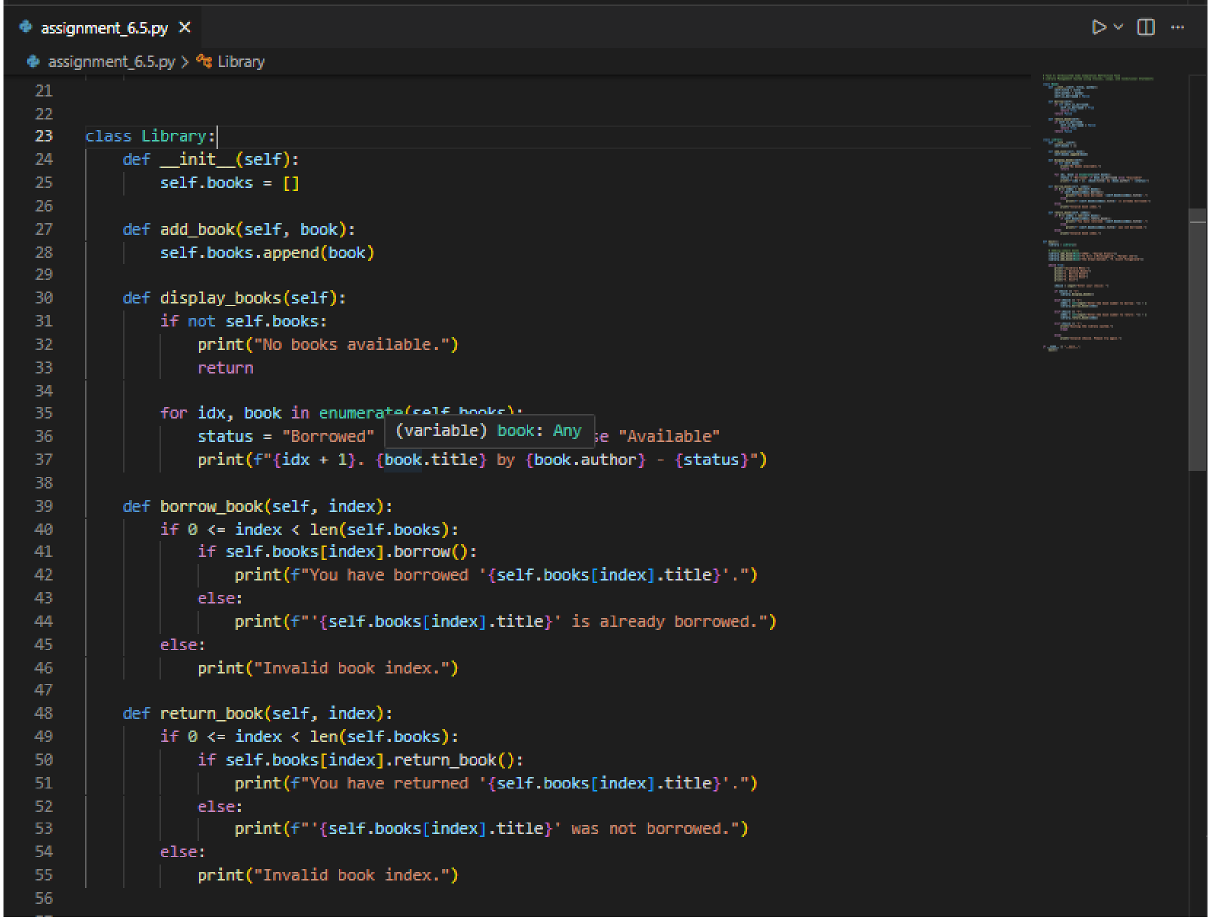
Prompt:

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

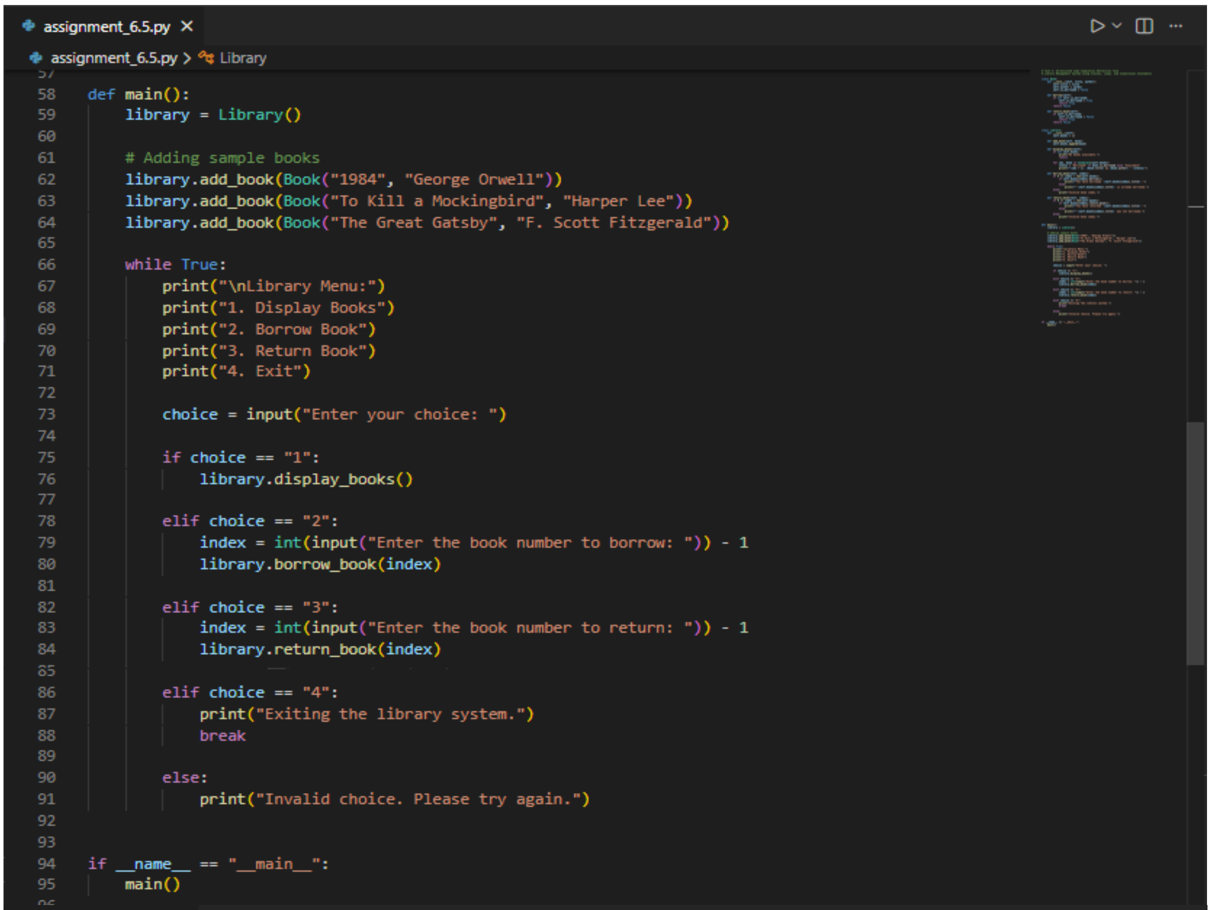
Code & Output:



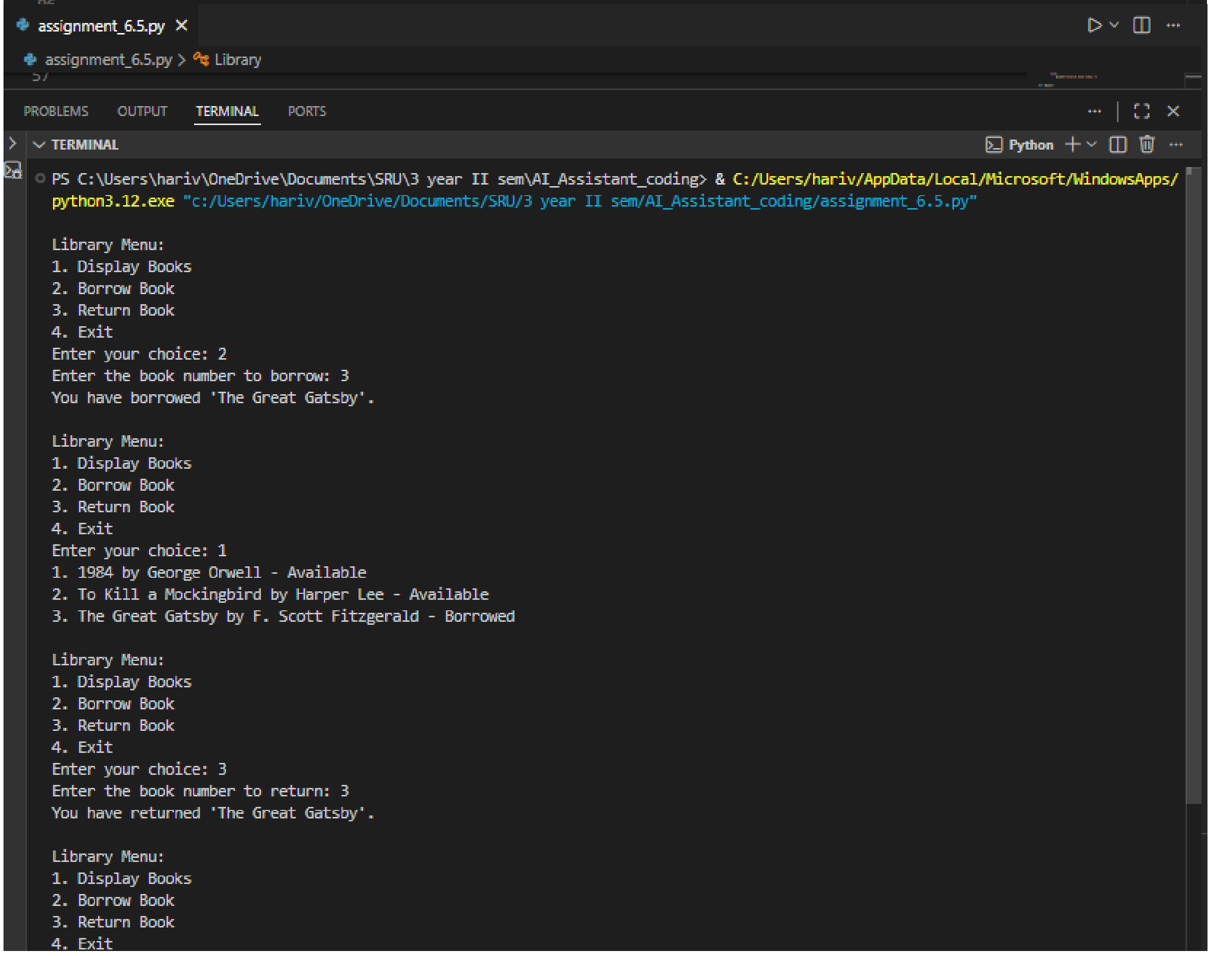
```python
# Task 3: AI-Assisted Code Completion Reflection Task
# Library Management System using Classes, Loops, and Conditional Statements

class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_borrowed = False

    def borrow(self):
        if not self.is_borrowed:
            self.is_borrowed = True
            return True
        return False

    def return_book(self):
        if self.is_borrowed:
            self.is_borrowed = False
            return True
        return False

```



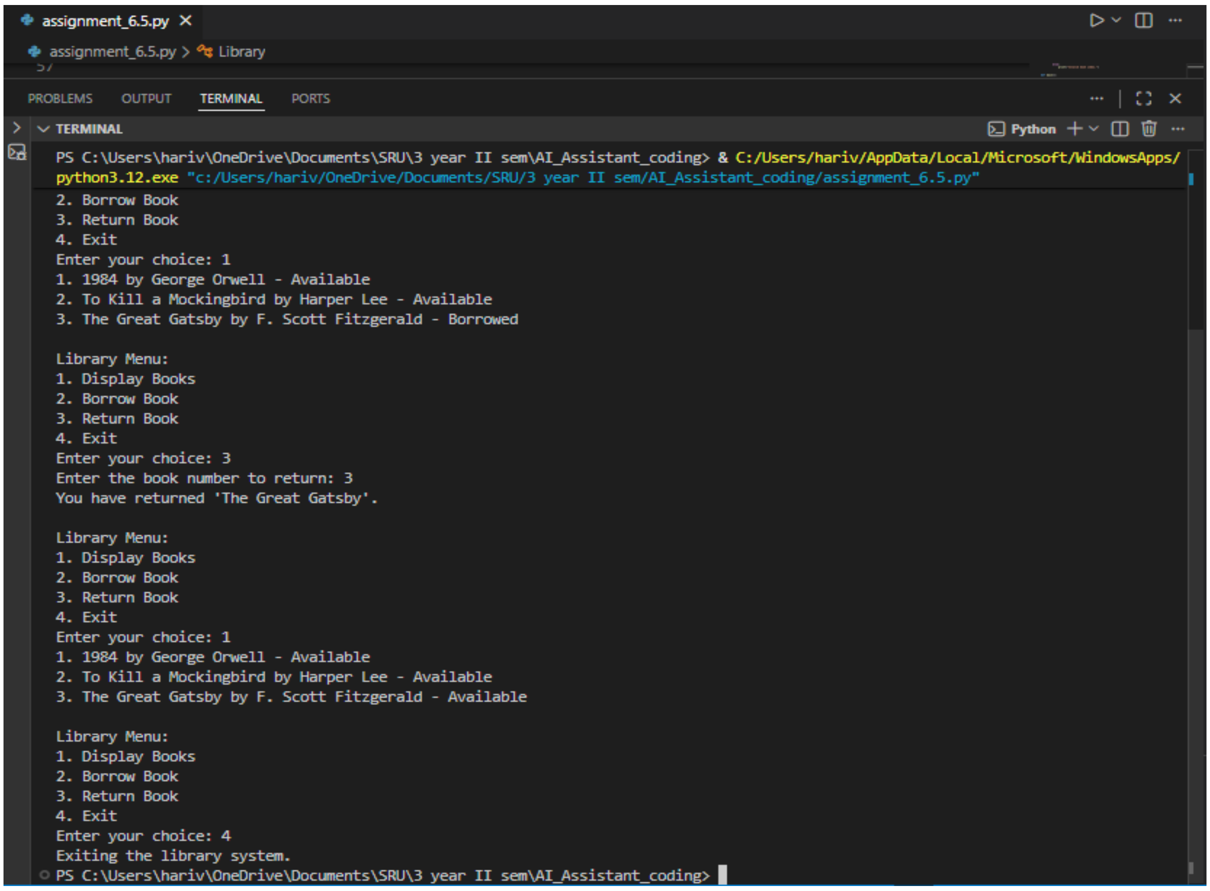
```python


class Library:
    def __init__(self):
        self.books = []

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

    def display_books(self):
        if not self.books:
            print("No books available.")
            return

        for idx, book in enumerate(self.books):
            status = "Borrowed"                      se "Available"
            print(f"{idx + 1}. {book.title} by {book.author} - {status}")

    def borrow_book(self, index):
        if 0 <= index < len(self.books):
            if self.books[index].borrow():
                print(f"You have borrowed '{self.books[index].title}'.")
            else:
                print(f"'{self.books[index].title}' is already borrowed.")
        else:
            print("Invalid book index.")

    def return_book(self, index):
        if 0 <= index < len(self.books):
            if self.books[index].return_book():
                print(f"You have returned '{self.books[index].title}'.")
            else:
                print(f"'{self.books[index].title}' was not borrowed.")
        else:
            print("Invalid book index.")

```



```python
def main():
    library = Library()

    # Adding sample books
    library.add_book(Book("1984", "George Orwell"))
    library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
    library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))

    while True:
        print("\nLibrary Menu:")
        print("1. Display Books")
        print("2. Borrow Book")
        print("3. Return Book")
        print("4. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            library.display_books()

        elif choice == "2":
            index = int(input("Enter the book number to borrow: ")) - 1
            library.borrow_book(index)

        elif choice == "3":
            index = int(input("Enter the book number to return: ")) - 1
            library.return_book(index)

        elif choice == "4":
            print("Exiting the library system.")
            break

        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    main()
```



```
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_6.5.py"

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 2
Enter the book number to borrow: 3
You have borrowed 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book number to return: 3
You have returned 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
```



```
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_6.5.py"
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book number to return: 3
You have returned 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Available

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 4
Exiting the library system.
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

The AI-generated program models a library using a class and incorporates loops and conditional statements to create a menu-driven interface. The loop enables the program to repeatedly accept user input, while the conditional logic manages the flow of operations. Overall, the program clearly demonstrates the effective use of object-oriented programming concepts with AI assistance.

Reflection on AI-Assisted Coding:

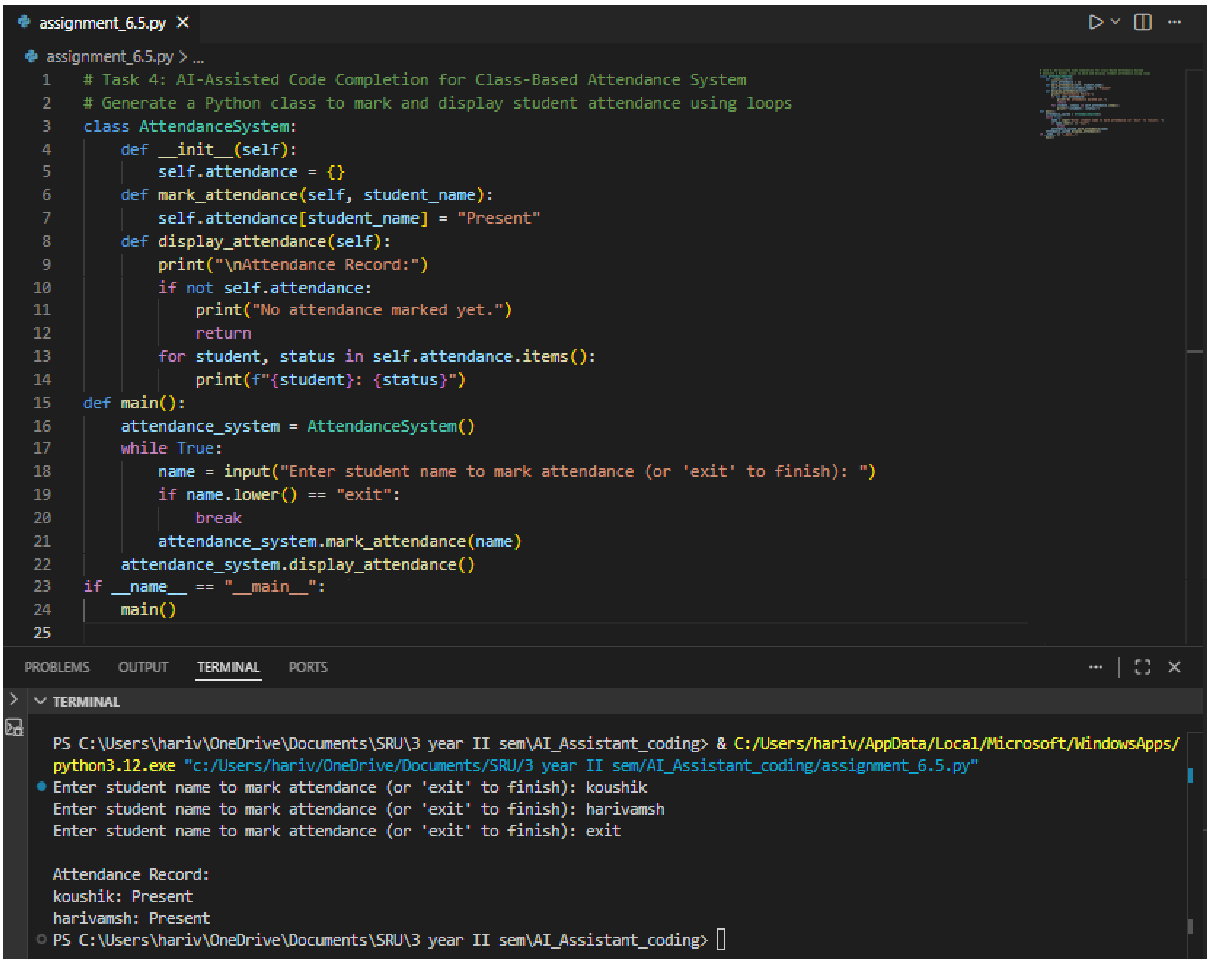
The AI tool was able to generate a complete and working program in a short time. Although the logic is correct, there is still room for improvement by adding input validation and more advanced features. This task highlights how AI can accelerate development, while human review and refinement are still important for achieving better quality.

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

Prompt:

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

Code & Output:


assignment_6.5.py

```python
# Task 4: AI-Assisted Code Completion for Class-Based Attendance System
# Generate a Python class to mark and display student attendance using loops
class AttendanceSystem:
    def __init__(self):
        self.attendance = {}
    def mark_attendance(self, student_name):
        self.attendance[student_name] = "Present"
    def display_attendance(self):
        print("\nAttendance Record:")
        if not self.attendance:
            print("No attendance marked yet.")
            return
        for student, status in self.attendance.items():
            print(f"{student}: {status}")
def main():
    attendance_system = AttendanceSystem()
    while True:
        name = input("Enter student name to mark attendance (or 'exit' to finish): ")
        if name.lower() == "exit":
            break
        attendance_system.mark_attendance(name)
    attendance_system.display_attendance()
if __name__ == "__main__":
    main()
```

```
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_6.5.py"
Enter student name to mark attendance (or 'exit' to finish): koushik
Enter student name to mark attendance (or 'exit' to finish): harivamsh
Enter student name to mark attendance (or 'exit' to finish): exit

Attendance Record:
koushik: Present
harivamsh: Present
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

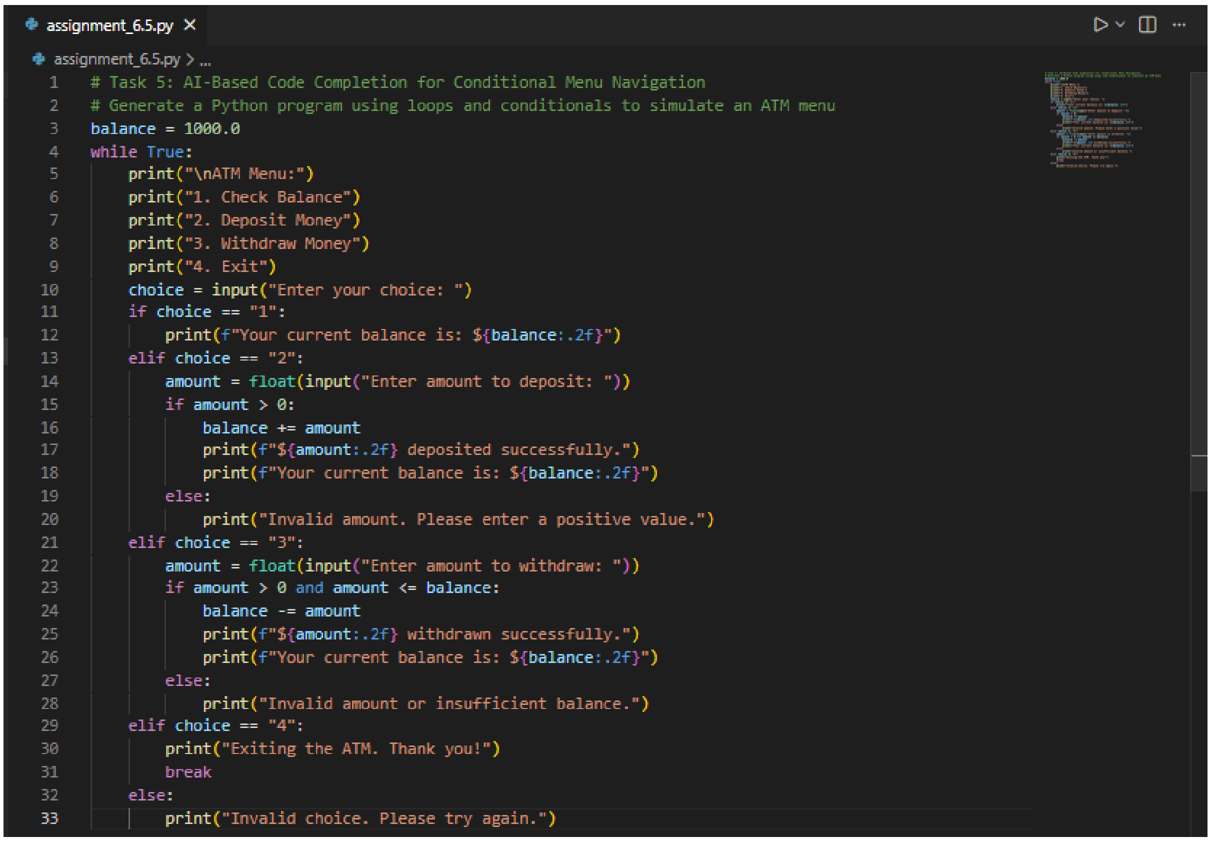
The AI-generated attendance system uses a class to manage and store attendance information. It includes a loop to collect details for multiple students and another loop to display the recorded attendance. The program functions correctly and clearly demonstrates the use of class-based code generated with AI assistance.

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

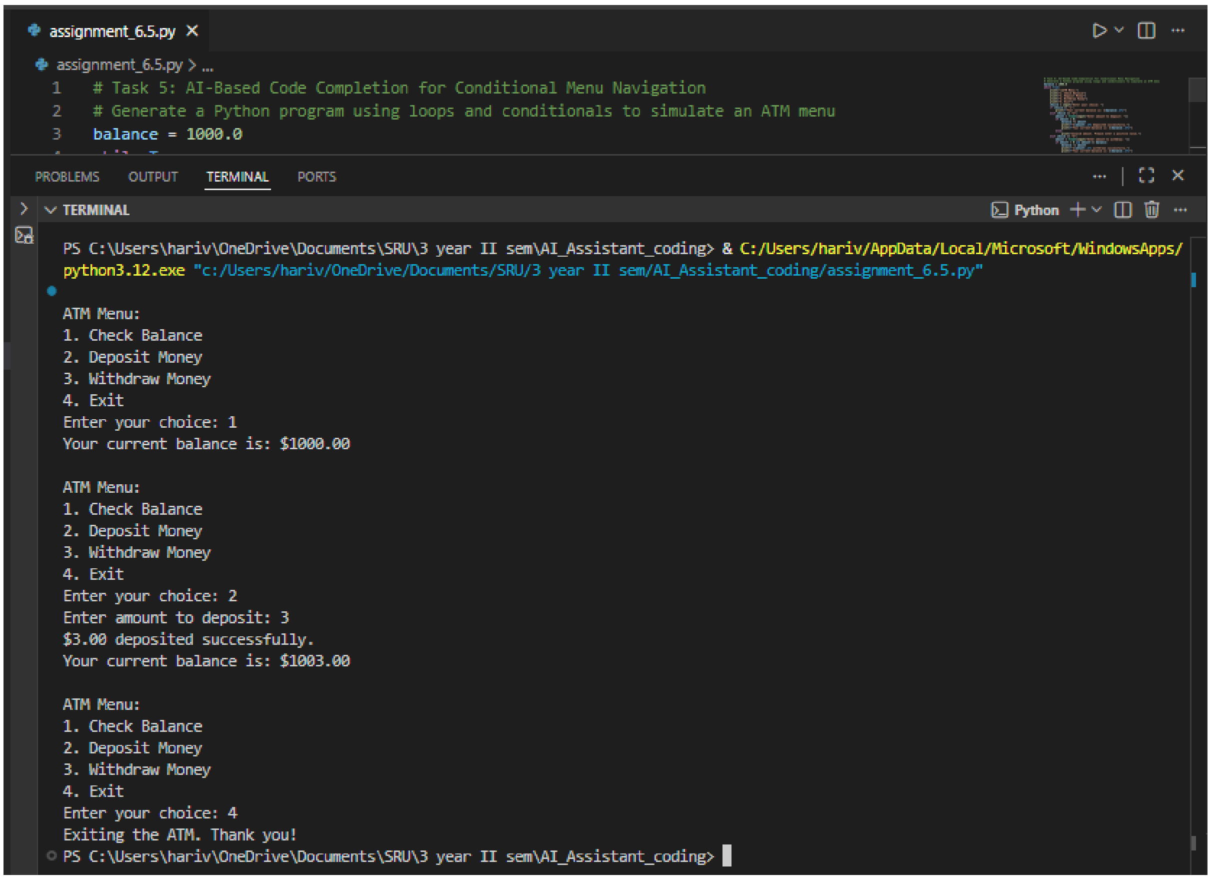
Prompt:

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

Code & Output:



```python
# Task 5: AI-Based Code Completion for Conditional Menu Navigation
# Generate a Python program using loops and conditionals to simulate an ATM menu
balance = 1000.0
while True:
    print("\nATM Menu:")
    print("1. Check Balance")
    print("2. Deposit Money")
    print("3. Withdraw Money")
    print("4. Exit")
    choice = input("Enter your choice: ")
    if choice == "1":
        print(f"Your current balance is: ${balance:.2f}")
    elif choice == "2":
        amount = float(input("Enter amount to deposit: "))
        if amount > 0:
            balance += amount
            print(f"${amount:.2f} deposited successfully.")
            print(f"Your current balance is: ${balance:.2f}")
        else:
            print("Invalid amount. Please enter a positive value.")
    elif choice == "3":
        amount = float(input("Enter amount to withdraw: "))
        if amount > 0 and amount <= balance:
            balance -= amount
            print(f"${amount:.2f} withdrawn successfully.")
            print(f"Your current balance is: ${balance:.2f}")
        else:
            print("Invalid amount or insufficient balance.")
    elif choice == "4":
        print("Exiting the ATM. Thank you!")
        break
    else:
        print("Invalid choice. Please try again.")
```



```python
# Task 5: AI-Based Code Completion for Conditional Menu Navigation
# Generate a Python program using loops and conditionals to simulate an ATM menu
balance = 1000.0
```

```
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_6.5.py"

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 1
Your current balance is: $1000.00

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 2
Enter amount to deposit: 3
$3.00 deposited successfully.
Your current balance is: $1003.00

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 4
Exiting the ATM. Thank you!
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

The AI-generated ATM program uses a loop to repeatedly show the menu and applies conditional statements to process the user's choices. The logic updates the account balance correctly and blocks withdrawals when they are not valid. Overall, this task shows how AI-based code completion can be effectively used to build menu-driven programs.

Final Conclusion:

This experiment demonstrates that AI-based code completion tools are capable of generating practical Python programs that make use of classes, loops, and conditional statements. Although AI significantly accelerates the development process, it is still the responsibility of developers to review the logic, address edge cases, and ensure that the generated code is used ethically and responsibly.