# ASSIGNMENT-6.5

**NAME:** M. Akshaya

**H.T.NO:** 2303A52449

**BATCH:** 35

**Experiment 6:**

**Task Description #1** (AI-Based Code Completion for Conditional Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

Prompt:

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

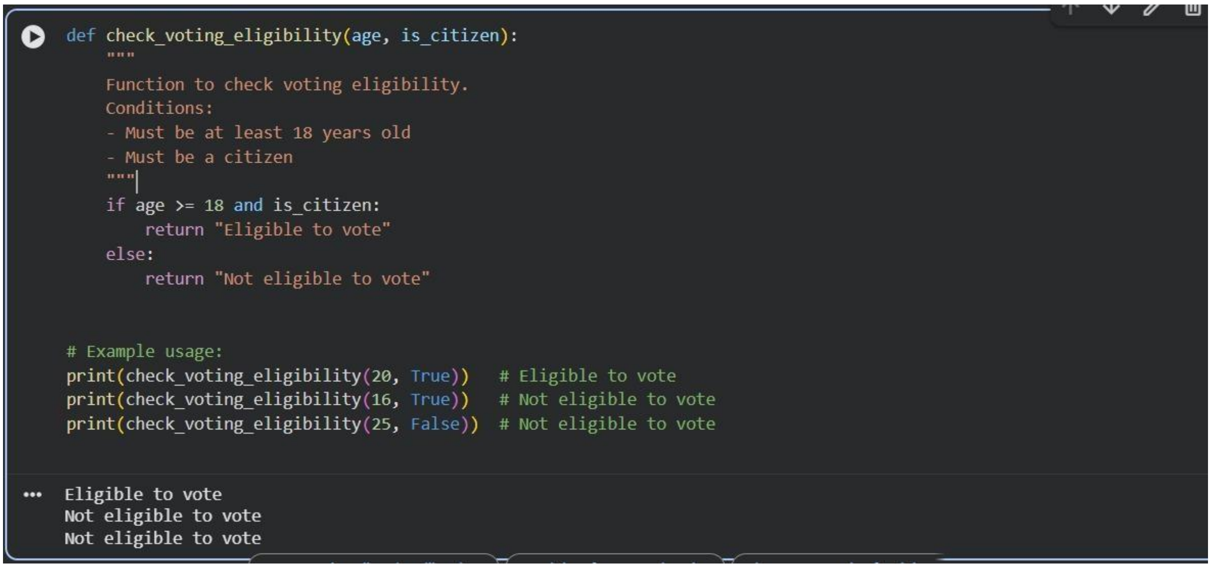
Expected Output:

• AI-generated conditional logic.

• Correct eligibility decisions.

• Explanation of conditions.

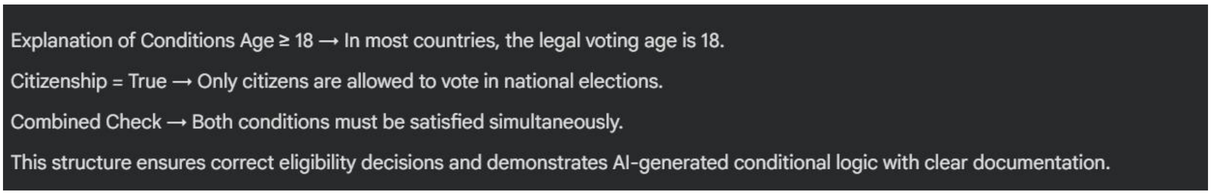
**CODE:**



```python
def check_voting_eligibility(age, is_citizen):
    """
    Function to check voting eligibility.
    Conditions:
    - Must be at least 18 years old
    - Must be a citizen
    """
    if age >= 18 and is_citizen:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"


# Example usage:
print(check_voting_eligibility(20, True))   # Eligible to vote
print(check_voting_eligibility(16, True))   # Not eligible to vote
print(check_voting_eligibility(25, False))  # Not eligible to vote
```

```
Eligible to vote
Not eligible to vote
Not eligible to vote
```

**EXPLANATION:**



Explanation of Conditions Age ≥ 18 → In most countries, the legal voting age is 18.

Citizenship = True → Only citizens are allowed to vote in national elections.

Combined Check → Both conditions must be satisfied simultaneously.

This structure ensures correct eligibility decisions and demonstrates AI-generated conditional logic with clear documentation.

**Task Description #2(**AI-Based Code Completion for Loop-Based String Processing)

Task: Use an AI tool to process strings using loops.

Prompt:

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

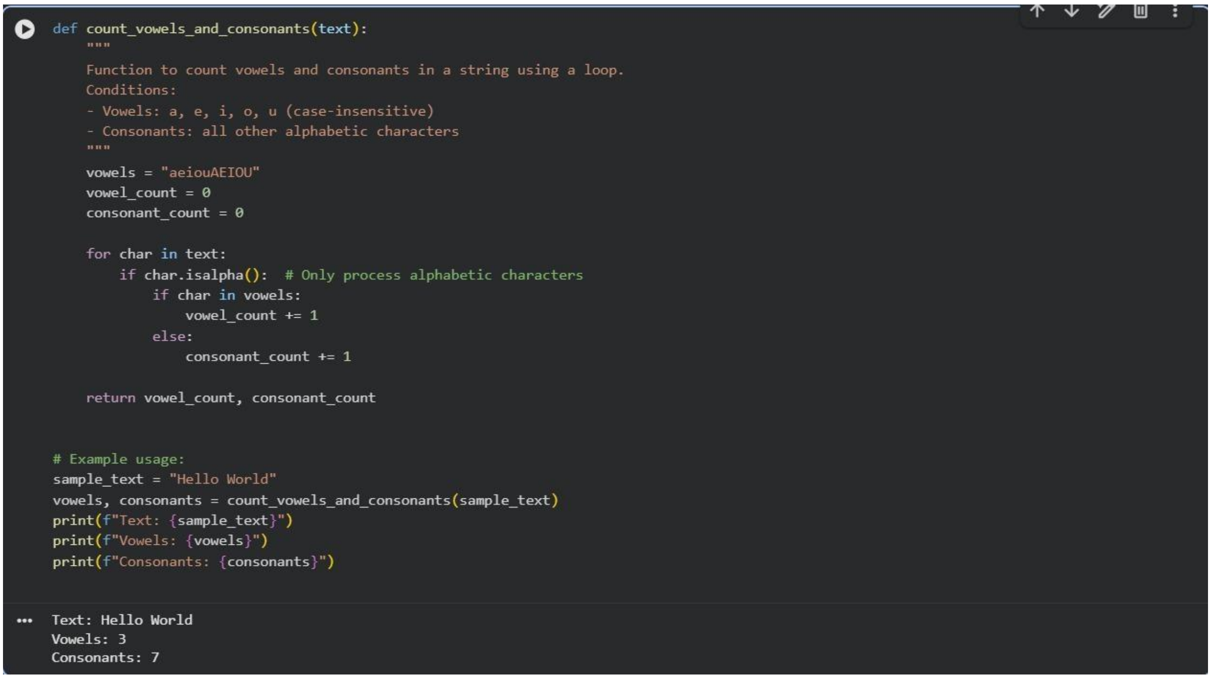
Expected Output:

• AI-generated string processing logic.

• Correct counts.

• Output verification.

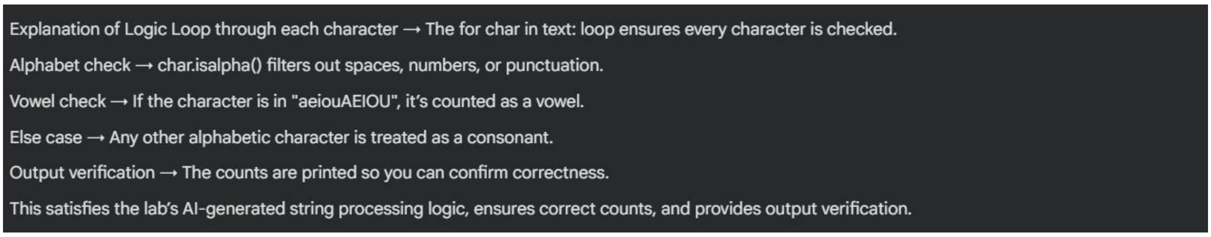
**CODE:**



```
def count_vowels_and_consonants(text):
    """
    Function to count vowels and consonants in a string using a loop.
    Conditions:
    - Vowels: a, e, i, o, u (case-insensitive)
    - Consonants: all other alphabetic characters
    """
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in text:
        if char.isalpha():  # Only process alphabetic characters
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count


# Example usage:
sample_text = "Hello World"
vowels, consonants = count_vowels_and_consonants(sample_text)
print(f"Text: {sample_text}")
print(f"Vowels: {vowels}")
print(f"Consonants: {consonants}")
```

```
Text: Hello World
Vowels: 3
Consonants: 7
```

**EXPLANATION:**



Explanation of Logic Loop through each character → The for char in text: loop ensures every character is checked.

Alphabet check → char.isalpha() filters out spaces, numbers, or punctuation.

Vowel check → If the character is in "aeiouAEIOU", it’s counted as a vowel.

Else case → Any other alphabetic character is treated as a consonant.

Output verification → The counts are printed so you can confirm correctness.

This satisfies the lab’s AI-generated string processing logic, ensures correct counts, and provides output verification.

**Task Description #3** (AI-Assisted Code Completion Reflection Task)

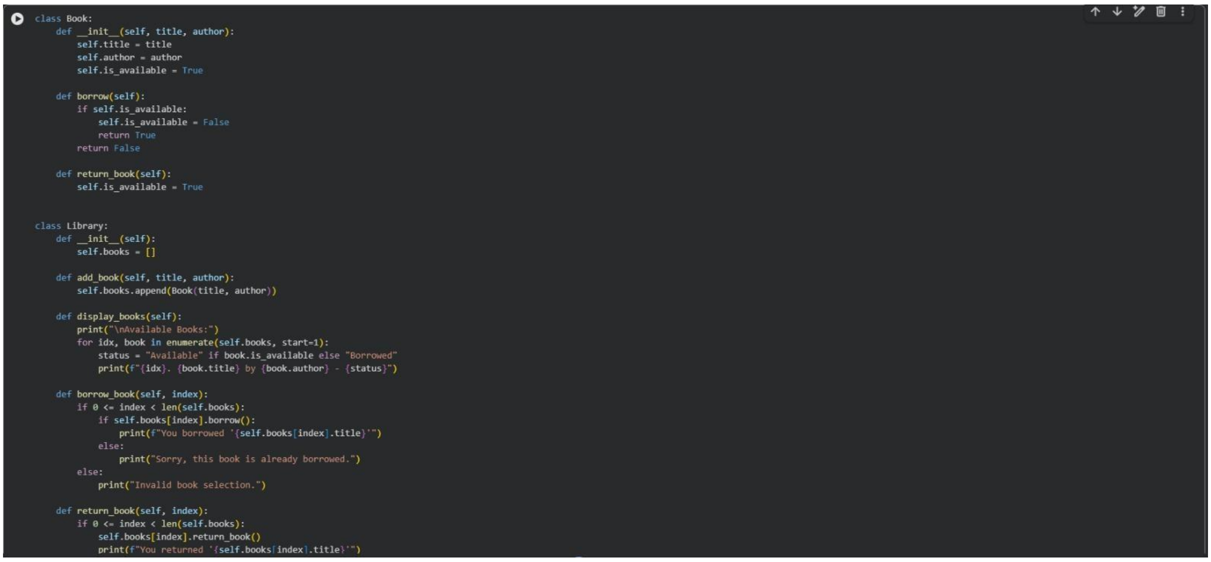
Task: Use an AI tool to generate a complete program using classes, loops, and conditionals.

Prompt:

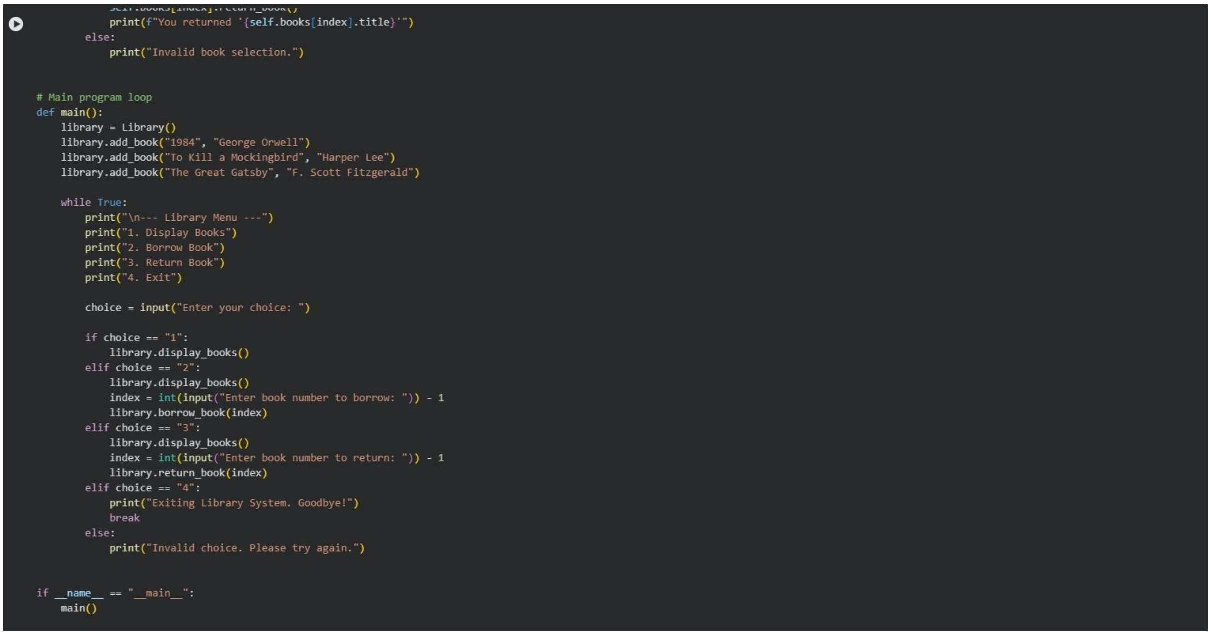
“Generate a Python program for a library management system using classes, loops, and conditional statements.” Expected Output:

- Complete AI-generated program.
- Review of AI suggestions quality.
- Short reflection on AI-assisted coding experience.

**CODE:**

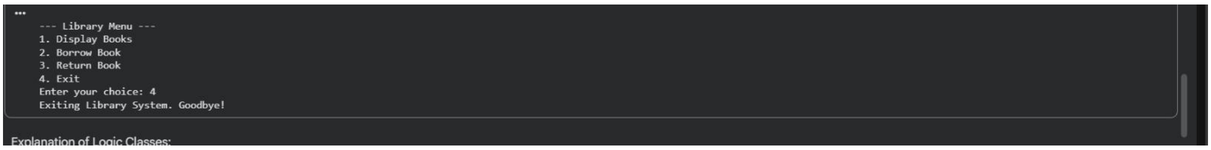


```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_available = True

    def borrow(self):
        if self.is_available:
            self.is_available = False
            return True
        return False

    def return_book(self):
        self.is_available = True


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

    def add_book(self, title, author):
        self.books.append(Book(title, author))

    def display_books(self):
        print("\nAvailable Books:")
        for idx, book in enumerate(self.books, start=1):
            status = "Available" if book.is_available else "Borrowed"
            print(f"{idx}. {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 borrowed '{self.books[index].title}'")
            else:
                print("Sorry, this book is already borrowed.")
        else:
            print("Invalid book selection.")

    def return_book(self, index):
        if 0 <= index < len(self.books):
            self.books[index].return_book()
            print(f"You returned '{self.books[index].title}'")
```



```
            self.books[index].return_book()
            print(f"You returned '{self.books[index].title}'")
        else:
            print("Invalid book selection.")


# Main program loop
def main():
    library = Library()
    library.add_book("1984", "George Orwell")
    library.add_book("To Kill a Mockingbird", "Harper Lee")
    library.add_book("The Great Gatsby", "F. Scott Fitzgerald")

    while True:
        print("\n--- Library 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":
            library.display_books()
            index = int(input("Enter book number to borrow: ")) - 1
            library.borrow_book(index)
        elif choice == "3":
            library.display_books()
            index = int(input("Enter book number to return: ")) - 1
            library.return_book(index)
        elif choice == "4":
            print("Exiting Library System. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")


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

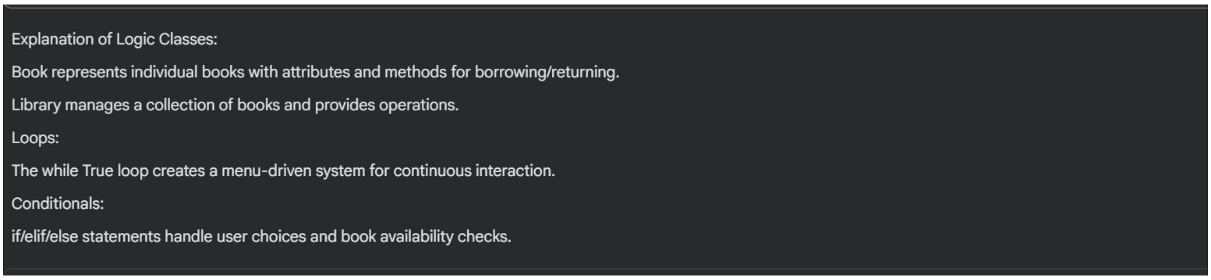
**OUTPUT:**



```
--- Library Menu ---
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 4
Exiting Library System. Goodbye!
```

Explanation of Logic Classes:

**EXPLANATION:**



Explanation of Logic Classes:

Book represents individual books with attributes and methods for borrowing/returning.

Library manages a collection of books and provides operations.

Loops:

The while True loop creates a menu-driven system for continuous interaction.

Conditionals:

if/elif/else statements handle user choices and book availability checks.

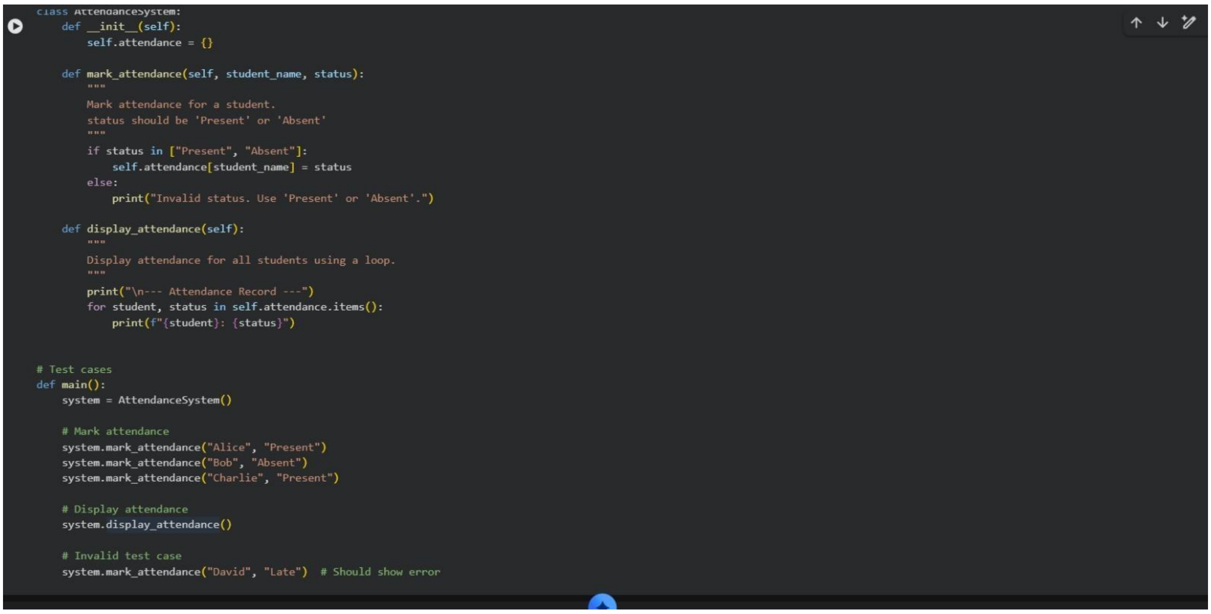
**Task Description #4** (AI-Assisted Code Completion for Class-Based Attendance System)

Task: Use an AI tool to generate an attendance management class.

Prompt: “Generate a Python class to mark and display student attendance using loops.” Expected Output:

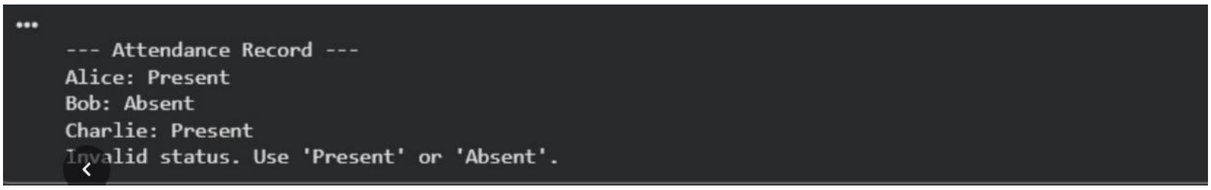
- AI-generated attendance logic.
- Correct display of attendance.
- Test cases.

**CODE:**



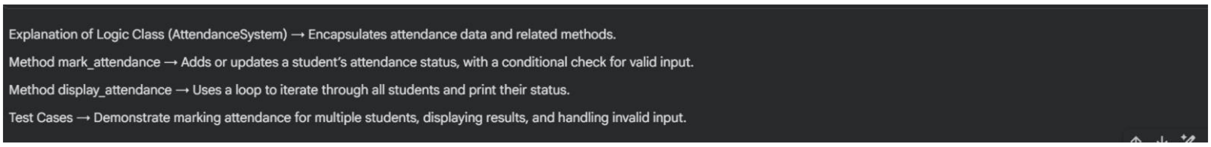
```python
class AttendanceSystem:
    def __init__(self):
        self.attendance = {}

    def mark_attendance(self, student_name, status):
        """
        Mark attendance for a student.
        status should be 'Present' or 'Absent'
        """
        if status in ["Present", "Absent"]:
            self.attendance[student_name] = status
        else:
            print("Invalid status. Use 'Present' or 'Absent'.")

    def display_attendance(self):
        """
        Display attendance for all students using a loop.
        """
        print("\n--- Attendance Record ---")
        for student, status in self.attendance.items():
            print(f"{student}: {status}")


# Test cases
def main():
    system = AttendanceSystem()

    # Mark attendance
    system.mark_attendance("Alice", "Present")
    system.mark_attendance("Bob", "Absent")
    system.mark_attendance("Charlie", "Present")

    # Display attendance
    system.display_attendance()

    # Invalid test case
    system.mark_attendance("David", "Late")  # Should show error
```

**OUTPUT:**



```
--- Attendance Record ---
Alice: Present
Bob: Absent
Charlie: Present
Invalid status. Use 'Present' or 'Absent'.
```

**EXPLANATION:**



Explanation of Logic Class (AttendanceSystem) → Encapsulates attendance data and related methods.

Method mark_attendance → Adds or updates a student's attendance status, with a conditional check for valid input.

Method display_attendance → Uses a loop to iterate through all students and print their status.

Test Cases → Demonstrate marking attendance for multiple students, displaying results, and handling invalid input.

**Task Description #5** (AI-Based Code Completion for Conditional

Menu Navigation)

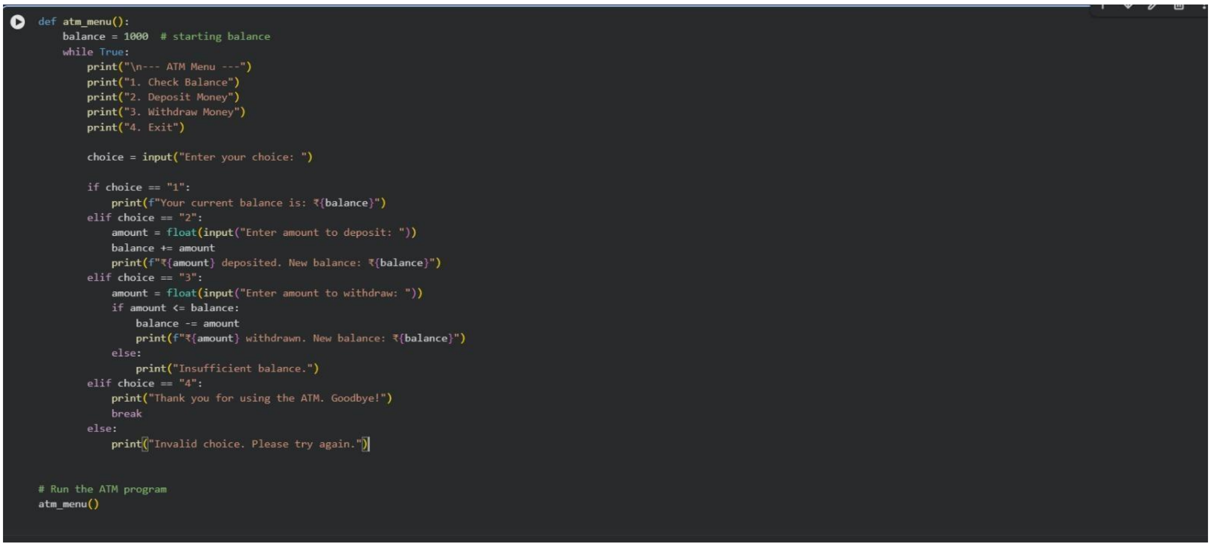
Task: Use an AI tool to complete a navigation menu.

Prompt: "Generate a Python program using loops and conditionals

to simulate an ATM menu." Expected Output:

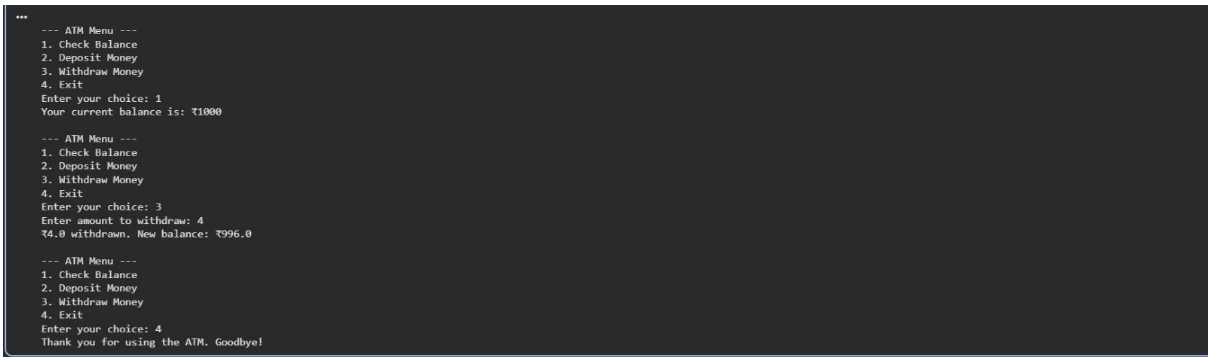
- AI-generated menu logic.
- Correct option handling.
- Output verification.

**CODE:**



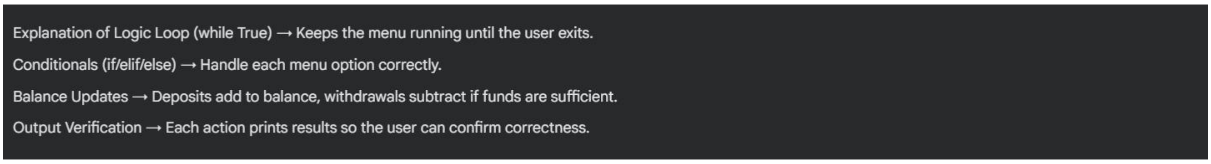
```
def atm_menu():
    balance = 1000  # starting balance
    while True:
        print("\n--- ATM Menu ---")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            print(f"Your current balance is: ₹{balance}")
        elif choice == "2":
            amount = float(input("Enter amount to deposit: "))
            balance += amount
            print(f"₹{amount} deposited. New balance: ₹{balance}")
        elif choice == "3":
            amount = float(input("Enter amount to withdraw: "))
            if amount <= balance:
                balance -= amount
                print(f"₹{amount} withdrawn. New balance: ₹{balance}")
            else:
                print("Insufficient balance.")
        elif choice == "4":
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")


# Run the ATM program
atm_menu()
```

# OUTPUT:



```
...
--- ATM Menu ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 1
Your current balance is: ₹1000

--- ATM Menu ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 3
Enter amount to withdraw: 4
₹4.0 withdrawn. New balance: ₹996.0

--- ATM Menu ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 4
Thank you for using the ATM. Goodbye!
```

# EXPLANATION:



Explanation of Logic Loop (while True) → Keeps the menu running until the user exits.

Conditionals (if/elif/else) → Handle each menu option correctly.

Balance Updates → Deposits add to balance, withdrawals subtract if funds are sufficient.

Output Verification → Each action prints results so the user can confirm correctness.