AI ASSISTANT CODING ASSIGNMENT-6.5

Name: Sanjay Karupothula

Enrollment no: 2303A52337

Batch: 34

Semester: VI

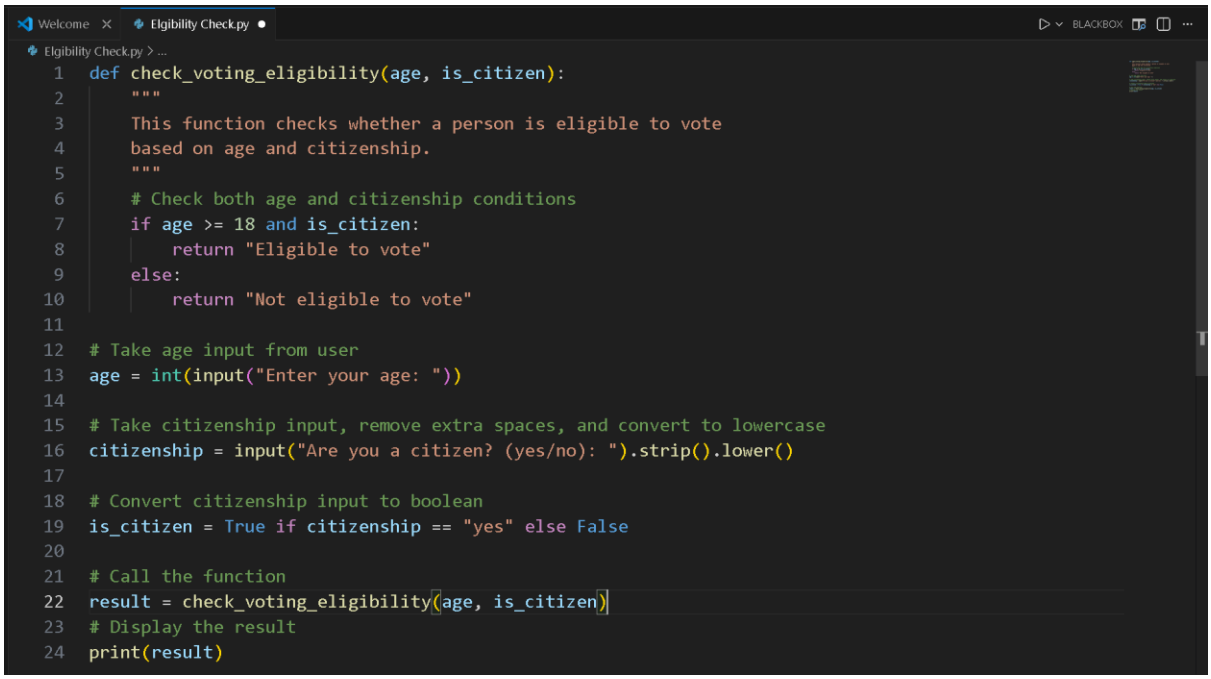
Branch: CSE(AIML)

Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

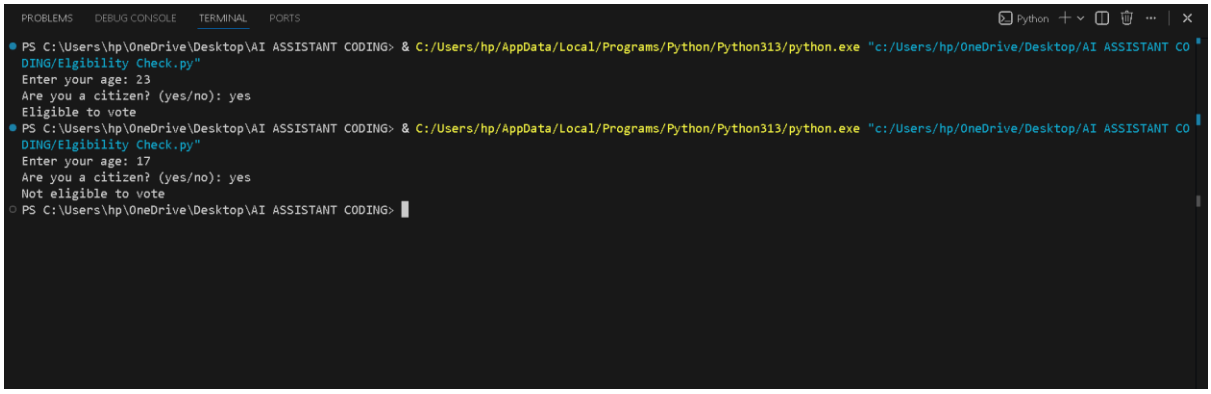
Task Description #1 (AI-Based Code Completion for Conditional Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

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



```python
def check_voting_eligibility(age, is_citizen):
    """
    This function checks whether a person is eligible to vote
    based on age and citizenship.
    """
    # Check both age and citizenship conditions
    if age >= 18 and is_citizen:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"

# Take age input from user
age = int(input("Enter your age: "))

# Take citizenship input, remove extra spaces, and convert to lowercase
citizenship = input("Are you a citizen? (yes/no): ").strip().lower()

# Convert citizenship input to boolean
is_citizen = True if citizenship == "yes" else False

# Call the function
result = check_voting_eligibility(age, is_citizen)
# Display the result
print(result)
```



Condition 1: Age Check (age >= 18)

- The person must be 18 years or older to vote.
- If age is less than 18, the person is not eligible, even if they are a citizen.

Condition 2: Citizenship Check (is_citizen)

- The person must be a citizen to vote.
- The user enters "yes" or "no", which is converted into a Boolean value:
    - "yes" → True
    - "no" → False

Decision Outcome

- If both conditions are true → "Eligible to vote"
- If any condition is false → "Not eligible to vote"

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

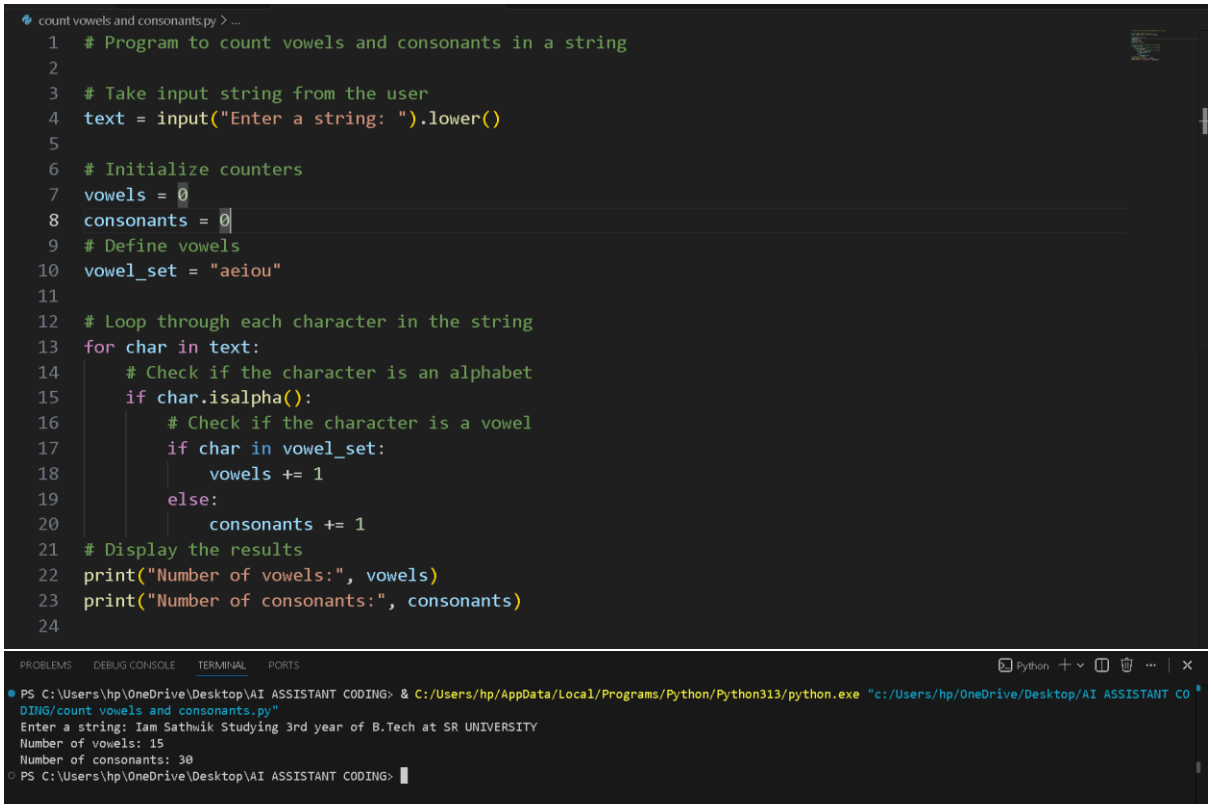
using a loop.”

Expected Output:

• AI-generated string processing logic.

• Correct counts.

• Output verification.



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

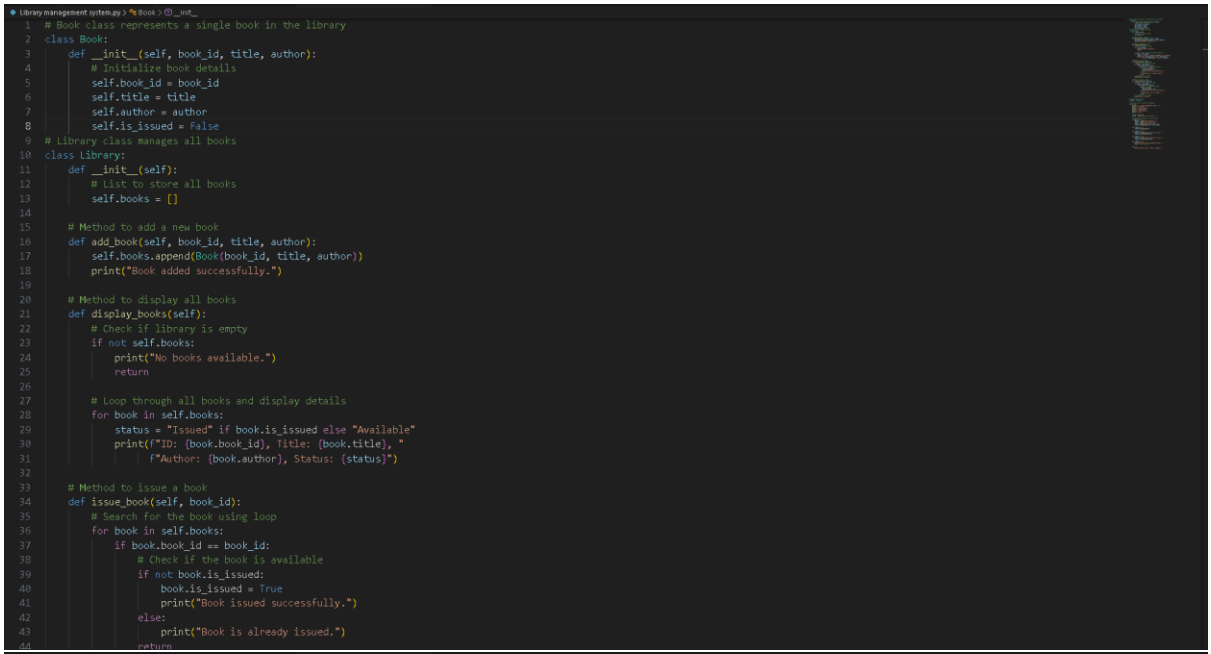
using classes, loops, and conditional statements.”

1) Review of AI suggestions quality.

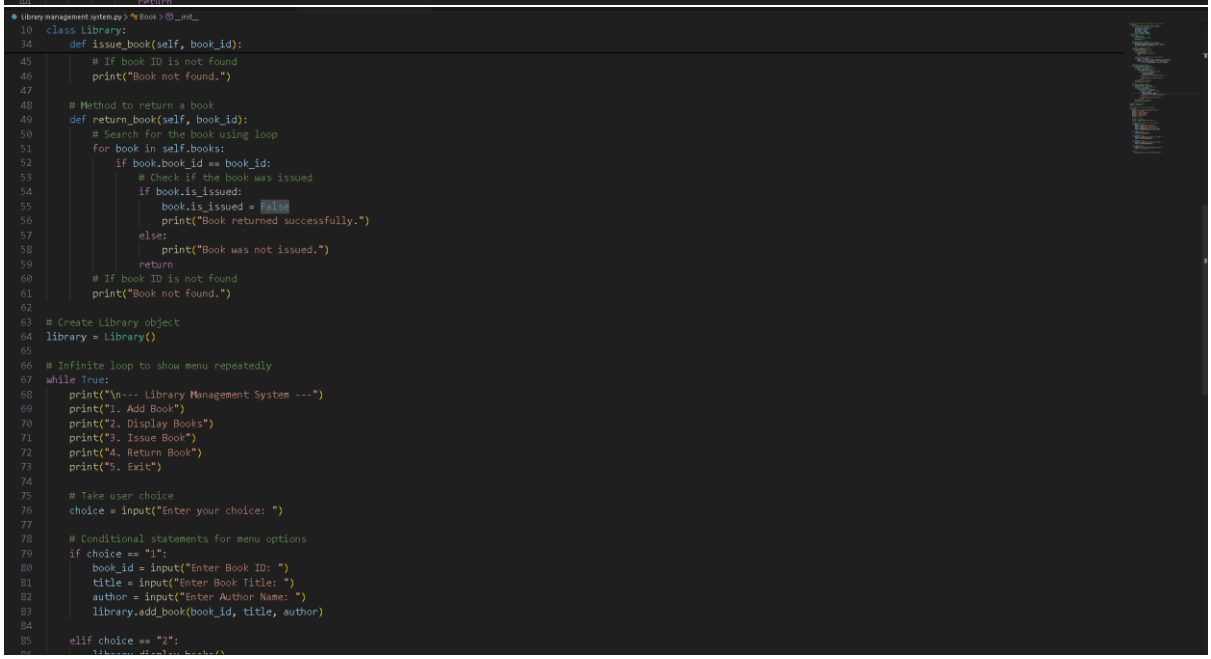
The AI suggestions are accurate and reliable. The generated code effectively uses classes, loops, and conditional statements to build a functional library management system. The logic is clear, well-structured, and easy to understand, making the solution suitable for beginners and academic use.

2) Short reflection on AI-assisted coding experience.

AI-assisted coding helped in developing the program efficiently by providing a clear structure and correct logic. It reduced the time required to write code and minimized errors during implementation. Reviewing the AI-generated solution improved understanding of object-oriented concepts, loops, and conditional statements, making the overall coding experience easier and more effective.



```python
# Book class represents a single book in the library
class Book:
    def __init__(self, book_id, title, author):
        # Initialize book details
        self.book_id = book_id
        self.title = title
        self.author = author
        self.is_issued = False
# Library class manages all books
class Library:
    def __init__(self):
        # List to store all books
        self.books = []

    # Method to add a new book
    def add_book(self, book_id, title, author):
        self.books.append(Book(book_id, title, author))
        print("Book added successfully.")

    # Method to display all books
    def display_books(self):
        # Check if library is empty
        if not self.books:
            print("No books available.")
            return

        # Loop through all books and display details
        for book in self.books:
            status = "Issued" if book.is_issued else "Available"
            print(f"ID: {book.book_id}, Title: {book.title}, "
                  f"Author: {book.author}, Status: {status}")

    # Method to issue a book
    def issue_book(self, book_id):
        # Search for the book using loop
        for book in self.books:
            if book.book_id == book_id:
                # Check if the book is available
                if not book.is_issued:
                    book.is_issued = True
                    print("Book issued successfully.")
                else:
                    print("Book is already issued.")
                return
```



```python
class Library:
    def issue_book(self, book_id):
        # If book ID is not found
        print("Book not found.")

    # Method to return a book
    def return_book(self, book_id):
        # Search for the book using loop
        for book in self.books:
            if book.book_id == book_id:
                # Check if the book was issued
                if book.is_issued:
                    book.is_issued = False
                    print("Book returned successfully.")
                else:
                    print("Book was not issued.")
                return
        # If book ID is not found
        print("Book not found.")

# Create Library object
library = Library()

# Infinite loop to show menu repeatedly
while True:
    print("\n--- Library Management System ---")
    print("1. Add Book")
    print("2. Display Books")
    print("3. Issue Book")
    print("4. Return Book")
    print("5. Exit")

    # Take user choice
    choice = input("Enter your choice: ")

    # Conditional statements for menu options
    if choice == "1":
        book_id = input("Enter Book ID: ")
        title = input("Enter Book Title: ")
        author = input("Enter Author Name: ")
        library.add_book(book_id, title, author)

    elif choice == "2":
```

```python
    elif choice == "2":
        library.display_books()

    elif choice == "3":
        book_id = input("Enter Book ID to issue: ")
        library.issue_book(book_id)

    elif choice == "4":
        book_id = input("Enter Book ID to return: ")
        library.return_book(book_id)

    elif choice == "5":
        print("Exiting Library Management System.")
        break  # Exit the loop

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

```
PS C:\Users\hp\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/hp/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/hp/OneDrive/Desktop/AI ASSISTANT CODING/Library management system.py"

--- Library Management System ---
1. Add Book
2. Display Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice: 1
Enter Book ID: 248109
Enter Book Title: Sense and Sensibility
Enter Author Name: Jane Austen
Book added successfully.

--- Library Management System ---
1. Add Book
2. Display Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice: 2
ID: 248109, Title: Sense and Sensibility, Author: Jane Austen, Status: Available

--- Library Management System ---
1. Add Book
2. Display Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice: 3
Enter Book ID to issue: 248109
Book issued successfully.

--- Library Management System ---
1. Add Book
2. Display Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice: 4
Enter Book ID to return: 248109
Book returned successfully.

--- Library Management System ---
1. Add Book
2. Display Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice: 5
Exiting Library Management System.
PS C:\Users\hp\OneDrive\Desktop\AI ASSISTANT CODING>
```

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.”

display student attendance.py > ...

```python
# Attendance class to manage student attendance
class Attendance:
    def __init__(self):
        # Dictionary to store student attendance
        # Key: student name, Value: Present/Absent
        self.records = {}

    # Method to mark attendance
    def mark_attendance(self, students):
        # Loop through student list
        for student in students:
            status = input(f"Enter attendance for {student} (P/A): ")
            # Conditional statement to check input
            if status.upper() == "P":
                self.records[student] = "Present"
            else:
                self.records[student] = "Absent"

    # Method to display attendance
    def display_attendance(self):
        print("\n--- Attendance Report ---")
        # Loop to display attendance
        for student, status in self.records.items():
            print(f"{student}: {status}")

# Test Cases

# Create Attendance object
attendance = Attendance()
```

```python
# Create Attendance object
attendance = Attendance()

# Test case 1: List of students
students_list = ["Alice", "Bob", "Charlie"]

# Mark attendance
attendance.mark_attendance(students_list)

# Display attendance
attendance.display_attendance()
```

```
PS C:\Users\hp\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/hp/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/hp/OneDrive/Desktop/AI ASSISTANT CODING/display student attendance.py"
Enter attendance for Alice (P/A): p
Enter attendance for Bob (P/A): a
Enter attendance for Charlie (P/A): p

--- Attendance Report ---
Alice: Present
Bob: Absent
Charlie: Present
PS C:\Users\hp\OneDrive\Desktop\AI ASSISTANT CODING>
```

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.”

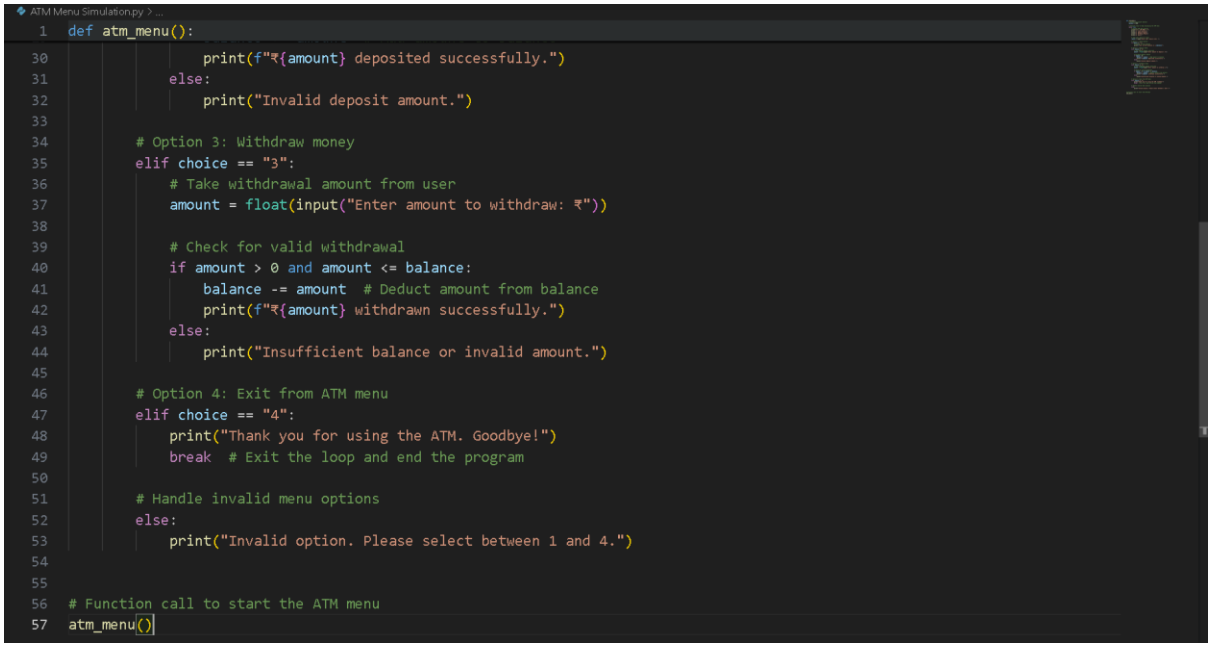
Expected Output:

• AI-generated menu logic.

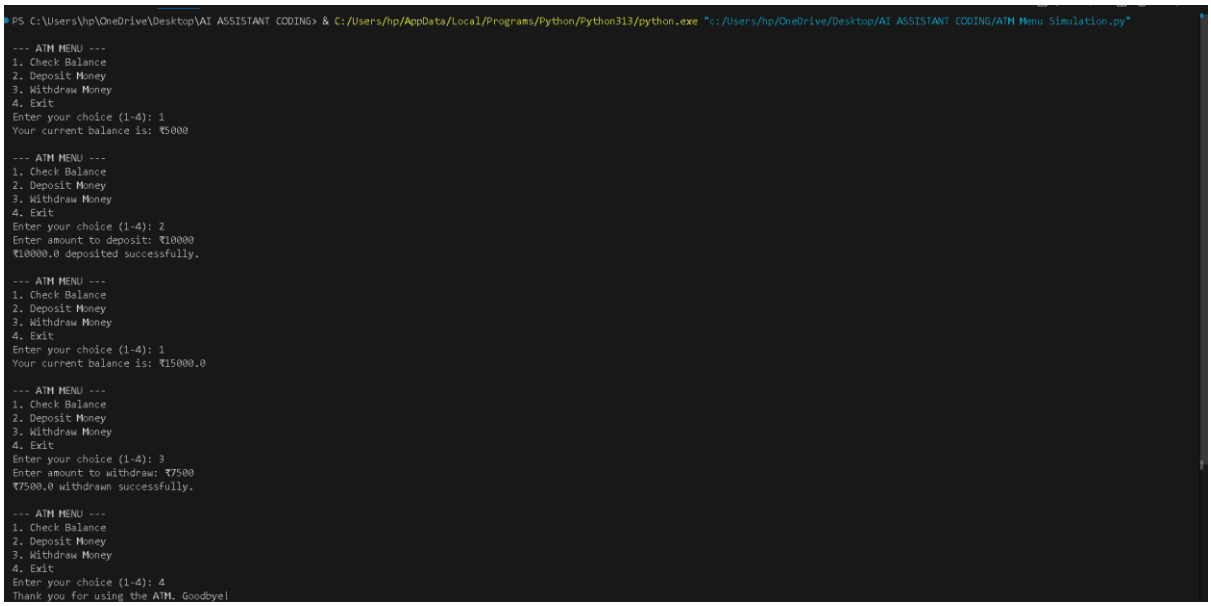
• Correct option handling.

• Output verification.

```python
def atm_menu():
    # Initialize account balance
    balance = 5000

    # Infinite loop to keep displaying the ATM menu
    while True:
        # Display ATM menu options
        print("\n--- ATM MENU ---")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

        # Take user choice as input
        choice = input("Enter your choice (1-4): ")

        # Option 1: Check balance
        if choice == "1":
            # Display current balance
            print(f"Your current balance is: ₹{balance}")

        # Option 2: Deposit money
        elif choice == "2":
            # Take deposit amount from user
            amount = float(input("Enter amount to deposit: ₹"))

            # Validate deposit amount
            if amount > 0:
                balance += amount  # Add amount to balance
                print(f"₹{amount} deposited successfully.")
```



```
ATM Menu Simulation.py > ...
  1  def atm_menu():
 30                  print(f"₹{amount} deposited successfully.")
 31              else:
 32                  print("Invalid deposit amount.")
 33
 34          # Option 3: Withdraw money
 35          elif choice == "3":
 36              # Take withdrawal amount from user
 37              amount = float(input("Enter amount to withdraw: ₹"))
 38
 39              # Check for valid withdrawal
 40              if amount > 0 and amount <= balance:
 41                  balance -= amount  # Deduct amount from balance
 42                  print(f"₹{amount} withdrawn successfully.")
 43              else:
 44                  print("Insufficient balance or invalid amount.")
 45
 46          # Option 4: Exit from ATM menu
 47          elif choice == "4":
 48              print("Thank you for using the ATM. Goodbye!")
 49              break  # Exit the loop and end the program
 50
 51          # Handle invalid menu options
 52          else:
 53              print("Invalid option. Please select between 1 and 4.")
 54
 55
 56  # Function call to start the ATM menu
 57  atm_menu()
```



```
PS C:\Users\hp\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/hp/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/hp/OneDrive/Desktop/AI ASSISTANT CODING/ATM Menu Simulation.py"

--- ATM MENU ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 1
Your current balance is: ₹5000

--- ATM MENU ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 2
Enter amount to deposit: ₹10000
₹10000.0 deposited successfully.

--- ATM MENU ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 1
Your current balance is: ₹15000.0

--- ATM MENU ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 3
Enter amount to withdraw: ₹7500
₹7500.0 withdrawn successfully.

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