SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 6.5

Program	**: B. Tech (CSE)**
Specialization	**: -**
Course Title	**: AI Assisted Coding**
Course Code	**: 23CS002PC304**
Semester	**II**
Academic Session	**: 2025-2026**
Name of Student	**: R.Akash Reddy**
Enrollment No.	**: 2403A51L30**
Batch No.	**51**
Date	**: 23/01/26**

Submission Starts here

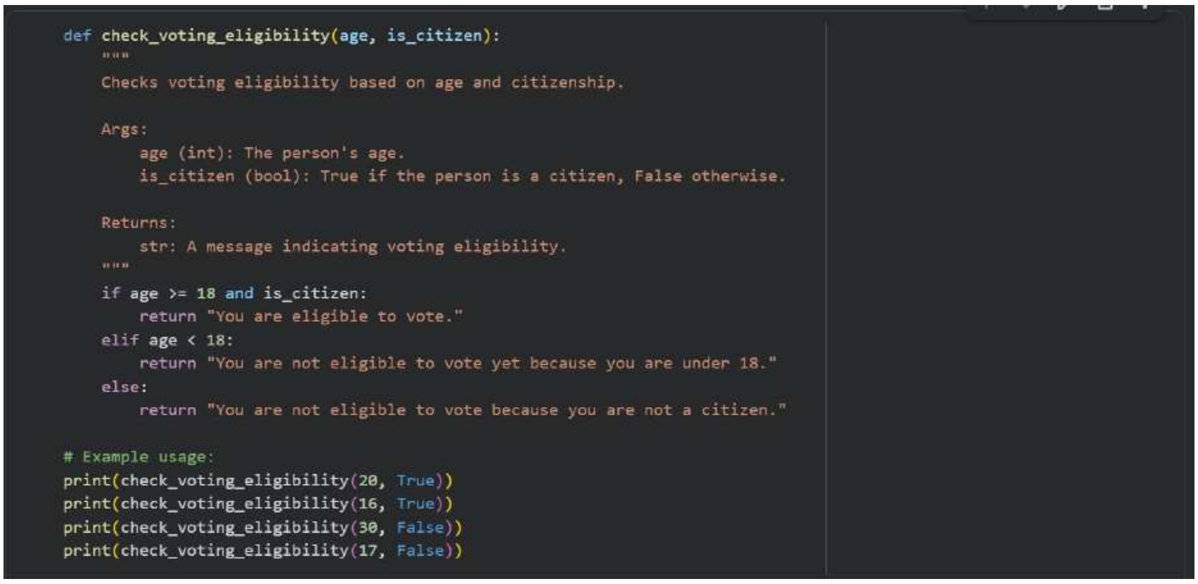
Screenshots:

Task Description #1 (AI-Based Code Completion for Conditional Eligibility Check)
Task: Use an AI tool to generate eligibility logic.

Prompt:

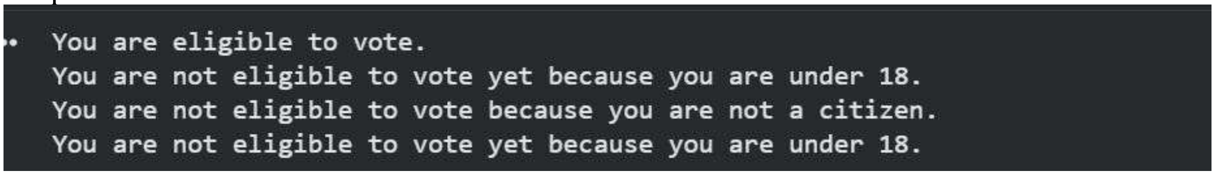
give a python program to check voting eligibility based on age and citizenship

Code:



```python
def check_voting_eligibility(age, is_citizen):
    """
    Checks voting eligibility based on age and citizenship.

    Args:
        age (int): The person's age.
        is_citizen (bool): True if the person is a citizen, False otherwise.

    Returns:
        str: A message indicating voting eligibility.
    """
    if age >= 18 and is_citizen:
        return "You are eligible to vote."
    elif age < 18:
        return "You are not eligible to vote yet because you are under 18."
    else:
        return "You are not eligible to vote because you are not a citizen."

# Example usage:
print(check_voting_eligibility(20, True))
print(check_voting_eligibility(16, True))
print(check_voting_eligibility(30, False))
print(check_voting_eligibility(17, False))
```

Output:



```
•  You are eligible to vote.
   You are not eligible to vote yet because you are under 18.
   You are not eligible to vote because you are not a citizen.
   You are not eligible to vote yet because you are under 18.
```

Explaination:
1. if age >= 18 and is_citizen:

This is the primary condition for eligibility. A person is eligible to vote if both their age is 18 or greater, and they are a citizen (meaning is_citizen is True).
If both these parts are true, the function returns: "You are eligible to vote."

2. elif age < 18:
This condition is checked only if the first if condition was false. It specifically checks if the person's age is less than 18.
If this is true, it means the person is too young to vote, regardless of citizenship. The function returns: "You are not eligible to vote yet because you are under 18."

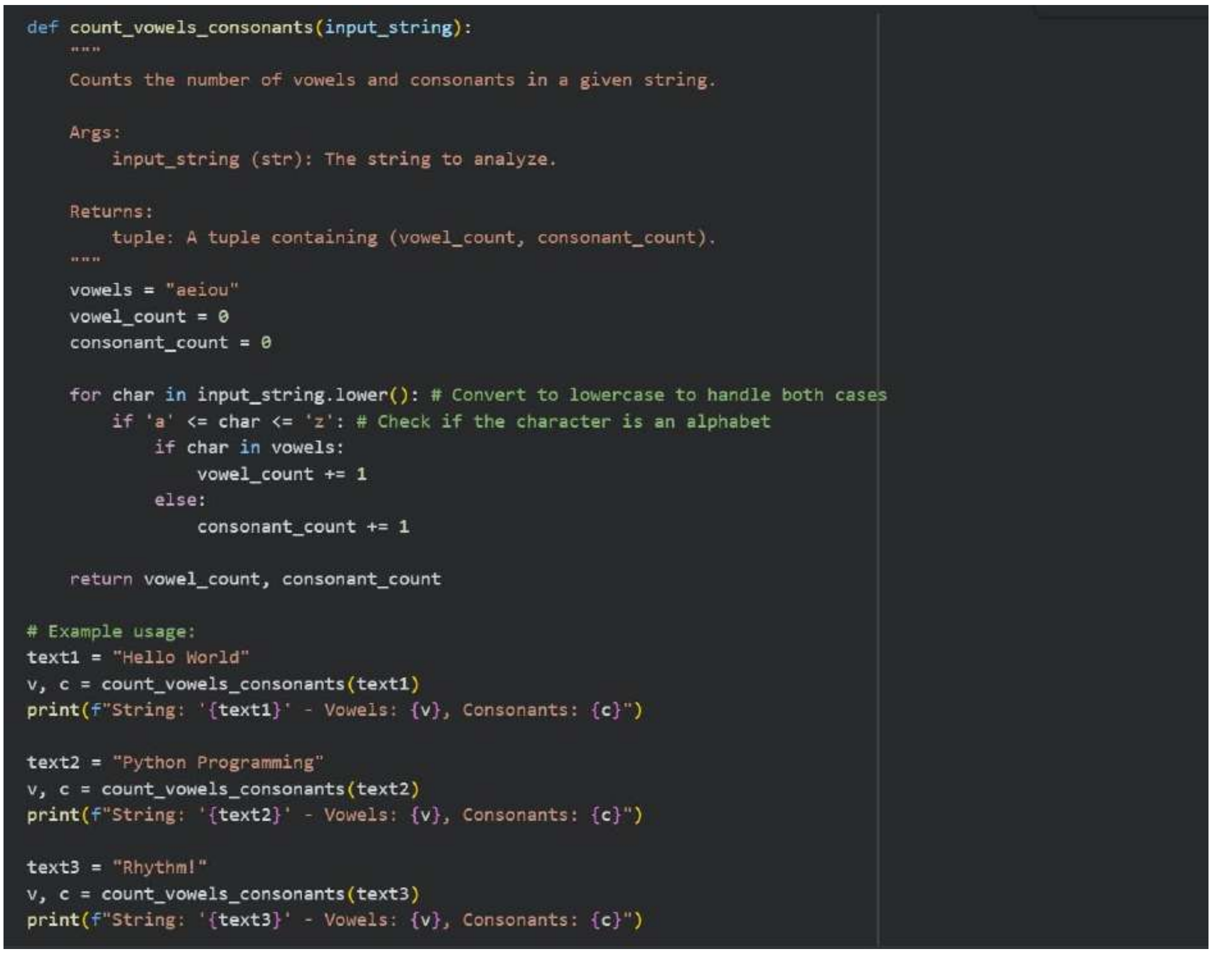
3. else:
This else block is reached only if both the if and elif conditions above were false.
If the first if (age >= 18 and is_citizen) was false, and the second elif (age < 18) was also false, it implies that the person is 18 or older but not a citizen (is_citizen must be False).
In this case, the function returns: "You are not eligible to vote because you are not a citizen."

Task Description #2(AI-Based Code Completion for Loop-Based String Processing)
Task: Use an AI tool to process strings using loops.

Prompt:
program to count vowels and consonants in a string

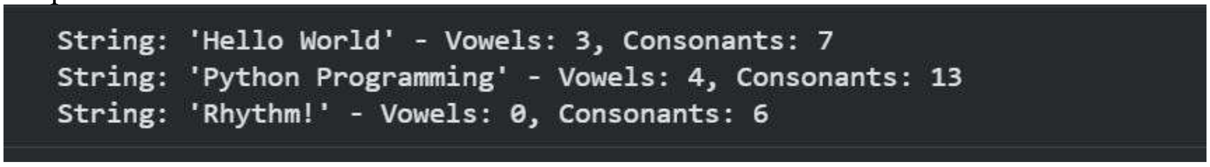
SR UNIVERSITY

Code:



```python
def count_vowels_consonants(input_string):
    """
    Counts the number of vowels and consonants in a given string.

    Args:
        input_string (str): The string to analyze.

    Returns:
        tuple: A tuple containing (vowel_count, consonant_count).
    """
    vowels = "aeiou"
    vowel_count = 0
    consonant_count = 0

    for char in input_string.lower(): # Convert to lowercase to handle both cases
        if 'a' <= char <= 'z': # Check if the character is an alphabet
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count

# Example usage:
text1 = "Hello World"
v, c = count_vowels_consonants(text1)
print(f"String: '{text1}' - Vowels: {v}, Consonants: {c}")

text2 = "Python Programming"
v, c = count_vowels_consonants(text2)
print(f"String: '{text2}' - Vowels: {v}, Consonants: {c}")

text3 = "Rhythm!"
v, c = count_vowels_consonants(text3)
print(f"String: '{text3}' - Vowels: {v}, Consonants: {c}")
```

Output:



```
String: 'Hello World' - Vowels: 3, Consonants: 7
String: 'Python Programming' - Vowels: 4, Consonants: 13
String: 'Rhythm!' - Vowels: 0, Consonants: 6
```

Explaination:

1. Initialization:

vowels = "aeiou": A string vowels is defined to easily check if a character is a vowel.
vowel_count = 0 and consonant_count = 0: Two variables are initialized to zero to keep track of the counts.

2. Lowercase Conversion:

for char in input_string.lower():: The input_string is first converted to lowercase using .lower(). This is crucial because it ensures that the vowel check (char in vowels) works for both 'A' and 'a', 'E' and 'e', and so on, without needing to check for uppercase vowels separately.

3. Character Iteration:

The for loop then iterates through each character (char) of this lowercase string.

4. Alphabet Check:

if 'a' <= char <= 'z':: Inside the loop, the first condition checks if the current char is an alphabet (between 'a' and 'z'). This step is important to ignore non-alphabetic characters like spaces, numbers, or punctuation (e.g., '!', ' ', '1') when counting vowels and consonants.

5. Vowel or Consonant Determination:

if char in vowels:: If the character is indeed an alphabet, this nested if condition checks if the char is present in our vowels string. If it is, vowel_count is incremented.
else:: If the character is an alphabet but not a vowel, it must be a consonant. In this case, consonant_count is incremented.

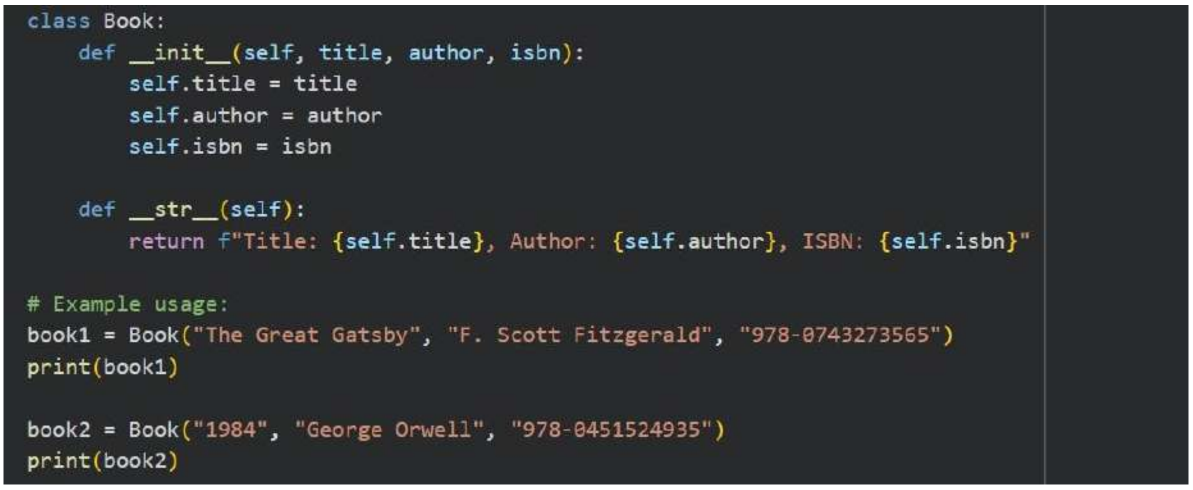
6. Return Value:

return vowel_count, consonant_count: After iterating through all characters, the function returns a tuple containing the final vowel_count and consonant_count.

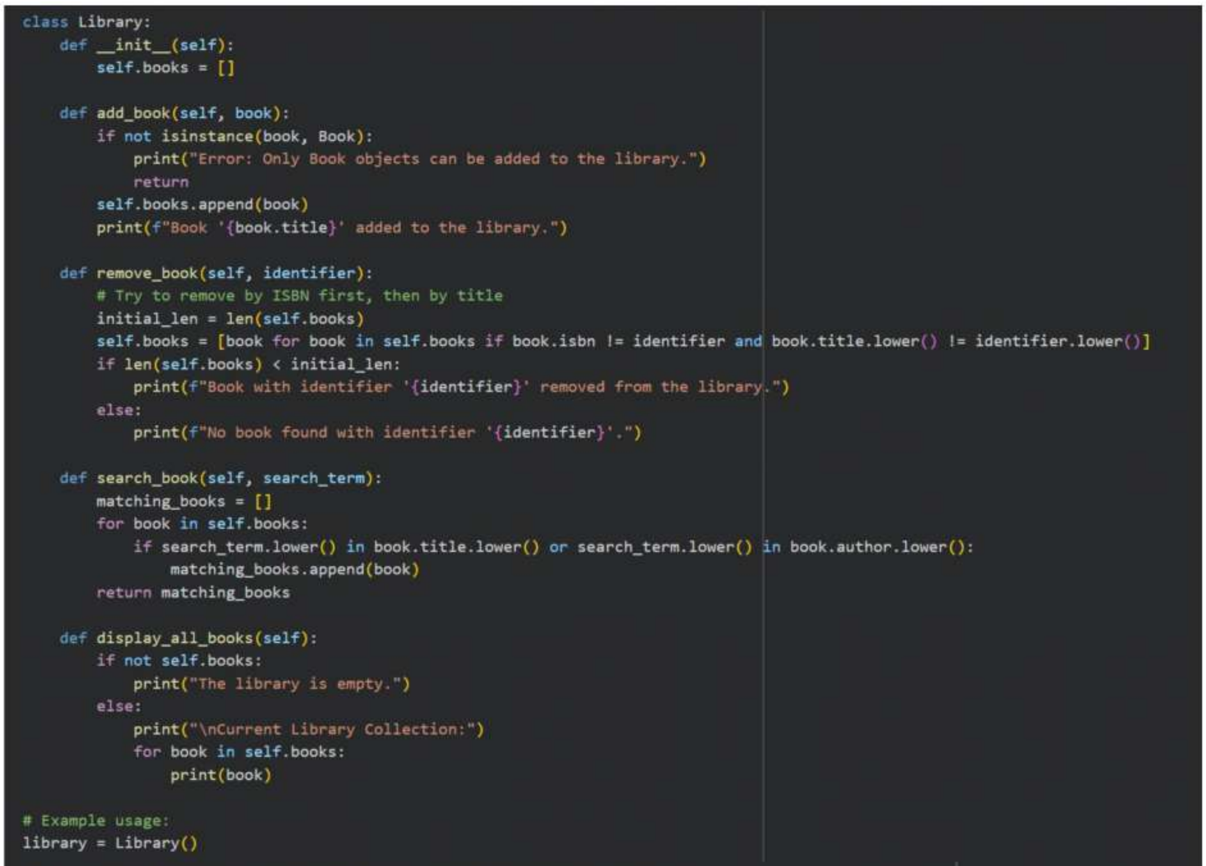
Task Description #3 (AI-Assisted Code Completion Reflection Task)
Task: Use an AI tool to generate a complete program using classes, loops, and conditionals.

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

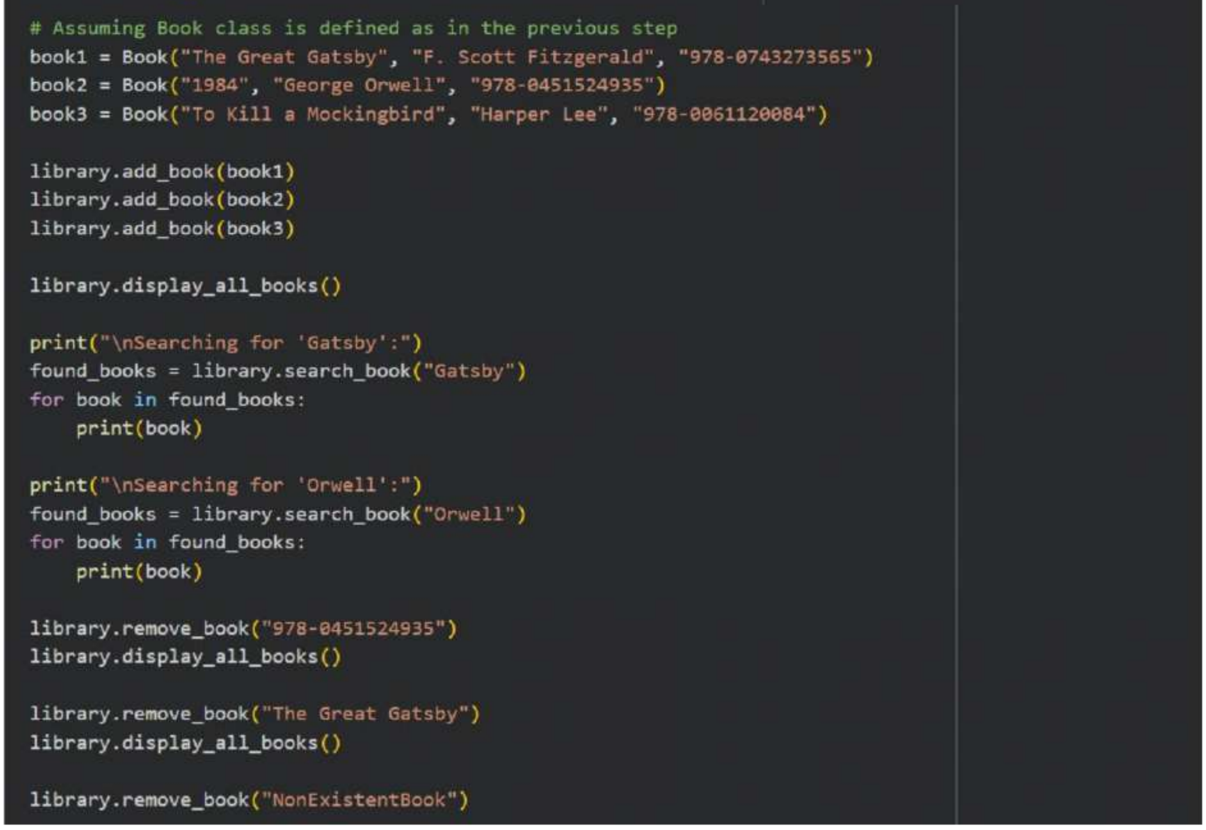
Code:



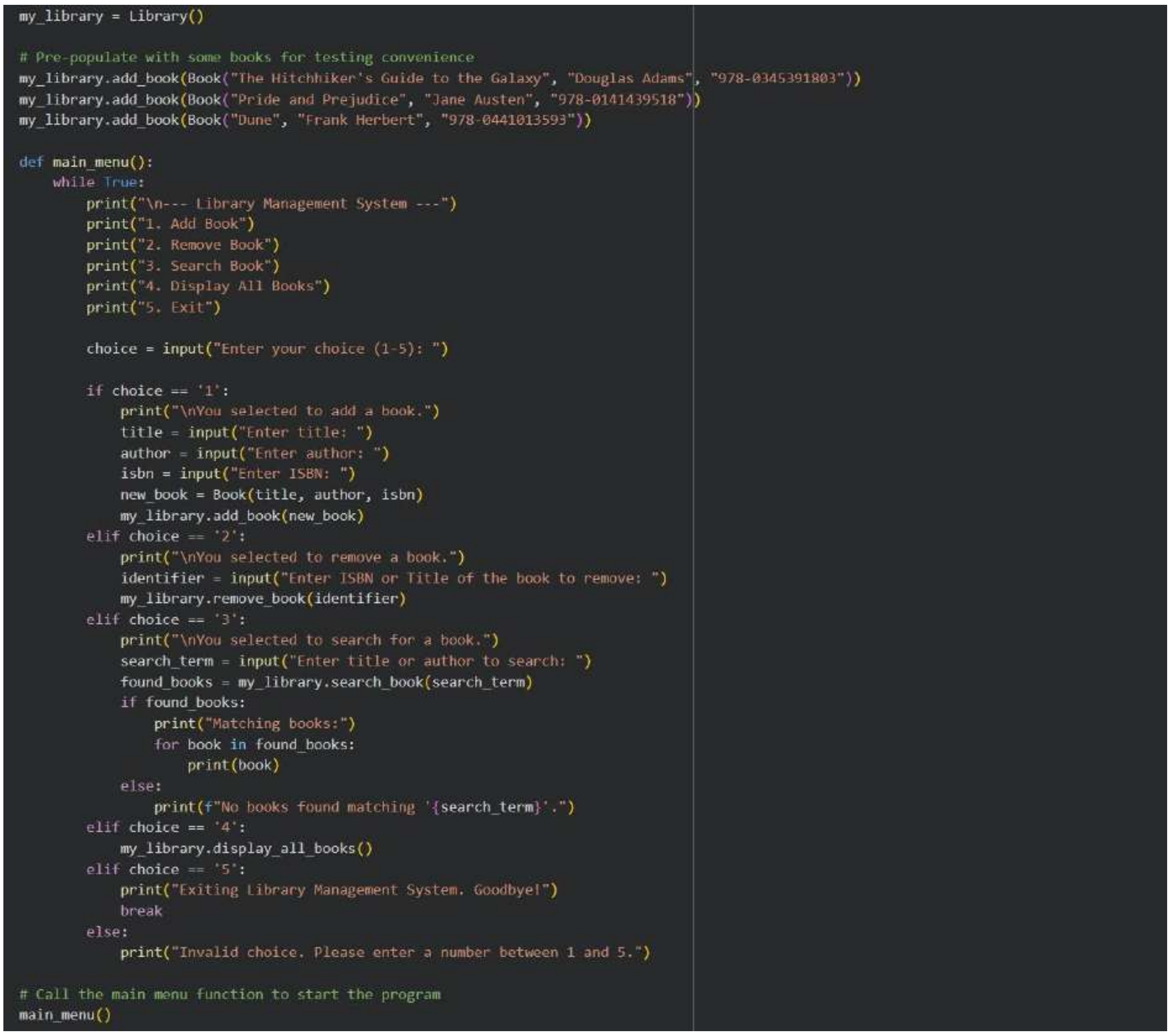
```python
class Book:
    def __init__(self, title, author, isbn):
        self.title = title
        self.author = author
        self.isbn = isbn

    def __str__(self):
        return f"Title: {self.title}, Author: {self.author}, ISBN: {self.isbn}"

# Example usage:
book1 = Book("The Great Gatsby", "F. Scott Fitzgerald", "978-0743273565")
print(book1)

book2 = Book("1984", "George Orwell", "978-0451524935")
print(book2)
```



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

    def add_book(self, book):
        if not isinstance(book, Book):
            print("Error: Only Book objects can be added to the library.")
            return
        self.books.append(book)
        print(f"Book '{book.title}' added to the library.")

    def remove_book(self, identifier):
        # Try to remove by ISBN first, then by title
        initial_len = len(self.books)
        self.books = [book for book in self.books if book.isbn != identifier and book.title.lower() != identifier.lower()]
        if len(self.books) < initial_len:
            print(f"Book with identifier '{identifier}' removed from the library.")
        else:
            print(f"No book found with identifier '{identifier}'.")

    def search_book(self, search_term):
        matching_books = []
        for book in self.books:
            if search_term.lower() in book.title.lower() or search_term.lower() in book.author.lower():
                matching_books.append(book)
        return matching_books

    def display_all_books(self):
        if not self.books:
            print("The library is empty.")
        else:
            print("\nCurrent Library Collection:")
            for book in self.books:
                print(book)

# Example usage:
library = Library()
```

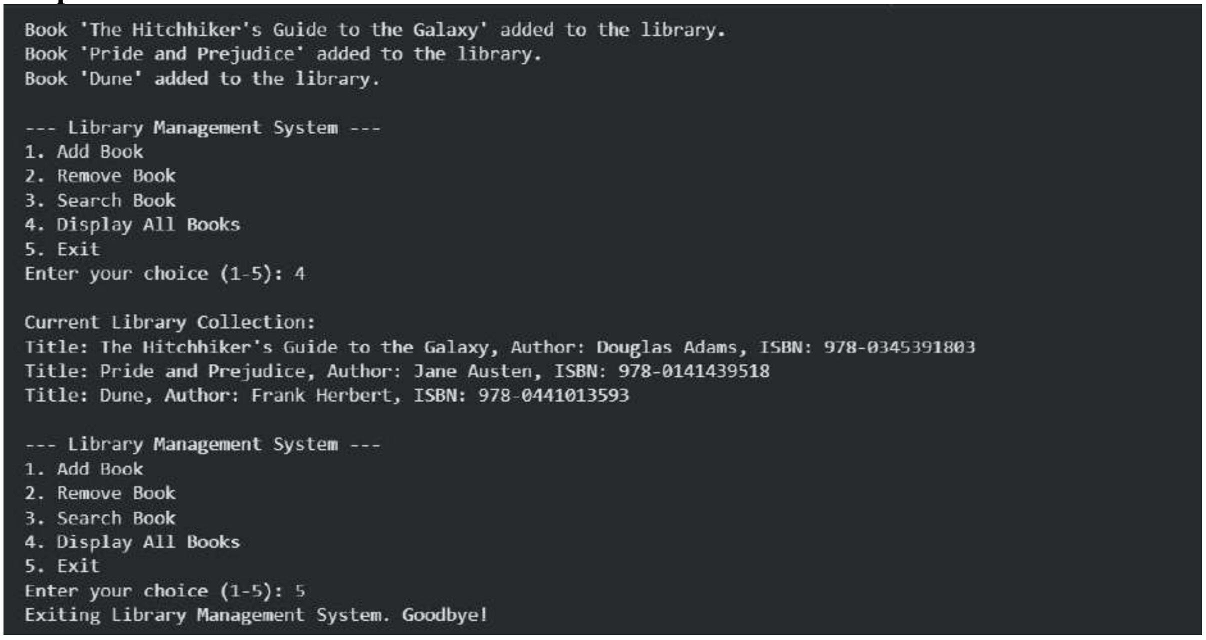


```python
# Assuming Book class is defined as in the previous step
book1 = Book("The Great Gatsby", "F. Scott Fitzgerald", "978-0743273565")
book2 = Book("1984", "George Orwell", "978-0451524935")
book3 = Book("To Kill a Mockingbird", "Harper Lee", "978-0061120084")

library.add_book(book1)
library.add_book(book2)
library.add_book(book3)

library.display_all_books()

print("\nSearching for 'Gatsby':")
found_books = library.search_book("Gatsby")
for book in found_books:
    print(book)

print("\nSearching for 'Orwell':")
found_books = library.search_book("Orwell")
for book in found_books:
    print(book)

library.remove_book("978-0451524935")
library.display_all_books()

library.remove_book("The Great Gatsby")
library.display_all_books()

library.remove_book("NonExistentBook")
```



```python
my_library = Library()

# Pre-populate with some books for testing convenience
my_library.add_book(Book("The Hitchhiker's Guide to the Galaxy", "Douglas Adams", "978-0345391803"))
my_library.add_book(Book("Pride and Prejudice", "Jane Austen", "978-0141439518"))
my_library.add_book(Book("Dune", "Frank Herbert", "978-0441013593"))

def main_menu():
    while True:
        print("\n--- Library Management System ---")
        print("1. Add Book")
        print("2. Remove Book")
        print("3. Search Book")
        print("4. Display All Books")
        print("5. Exit")

        choice = input("Enter your choice (1-5): ")

        if choice == '1':
            print("\nYou selected to add a book.")
            title = input("Enter title: ")
            author = input("Enter author: ")
            isbn = input("Enter ISBN: ")
            new_book = Book(title, author, isbn)
            my_library.add_book(new_book)
        elif choice == '2':
            print("\nYou selected to remove a book.")
            identifier = input("Enter ISBN or Title of the book to remove: ")
            my_library.remove_book(identifier)
        elif choice == '3':
            print("\nYou selected to search for a book.")
            search_term = input("Enter title or author to search: ")
            found_books = my_library.search_book(search_term)
            if found_books:
                print("Matching books:")
                for book in found_books:
                    print(book)
            else:
                print(f"No books found matching '{search_term}'.")
        elif choice == '4':
            my_library.display_all_books()
        elif choice == '5':
            print("Exiting Library Management System. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 5.")

# Call the main menu function to start the program
main_menu()
```

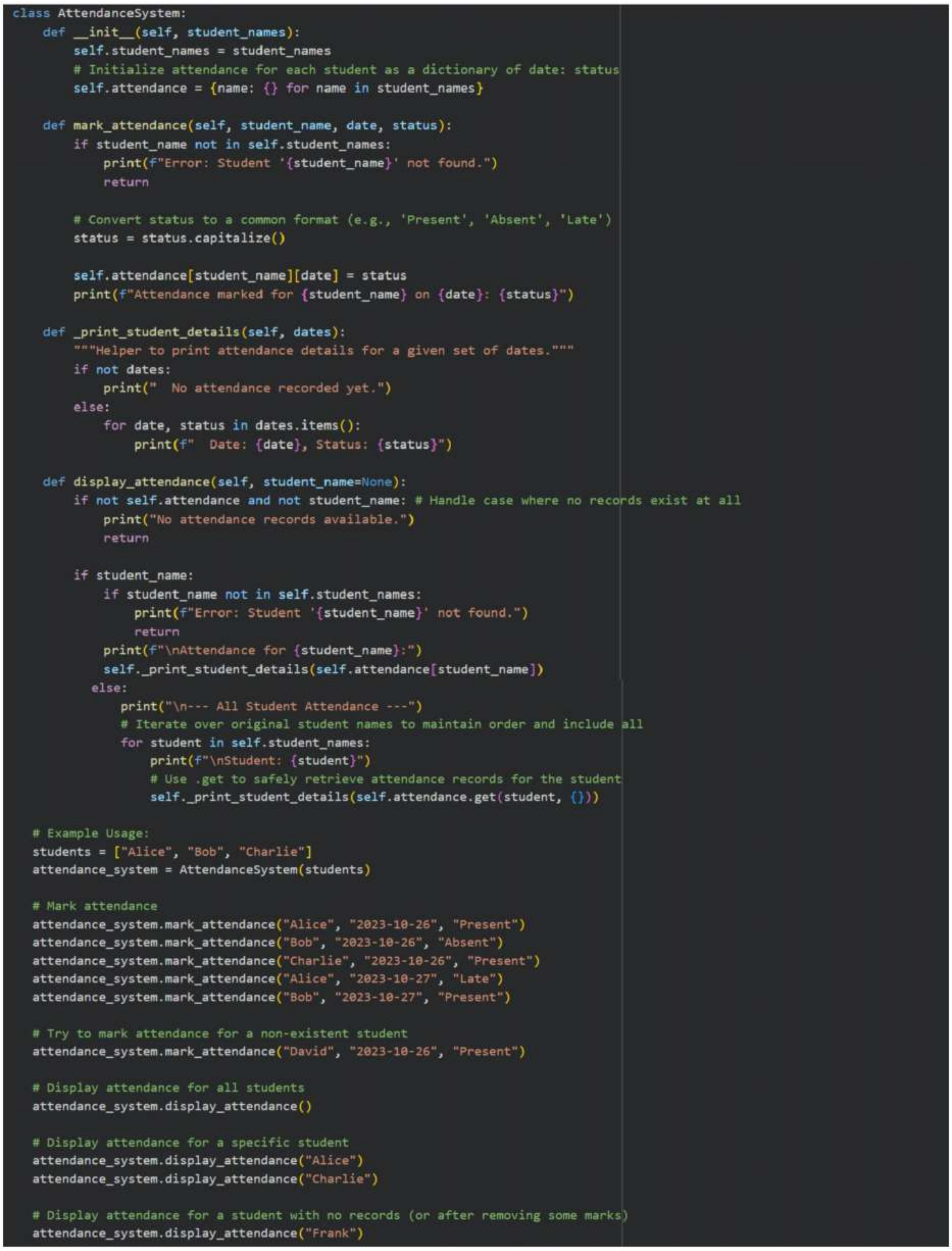
Output:



```
Book 'The Hitchhiker's Guide to the Galaxy' added to the library.
Book 'Pride and Prejudice' added to the library.
Book 'Dune' added to the library.

--- Library Management System ---
1. Add Book
2. Remove Book
3. Search Book
4. Display All Books
5. Exit
Enter your choice (1-5): 4

Current Library Collection:
Title: The Hitchhiker's Guide to the Galaxy, Author: Douglas Adams, ISBN: 978-0345391803
Title: Pride and Prejudice, Author: Jane Austen, ISBN: 978-0141439518
Title: Dune, Author: Frank Herbert, ISBN: 978-0441013593

--- Library Management System ---
1. Add Book
2. Remove Book
3. Search Book
4. Display All Books
5. Exit
Enter your choice (1-5): 5
Exiting Library Management System. Goodbye!
```

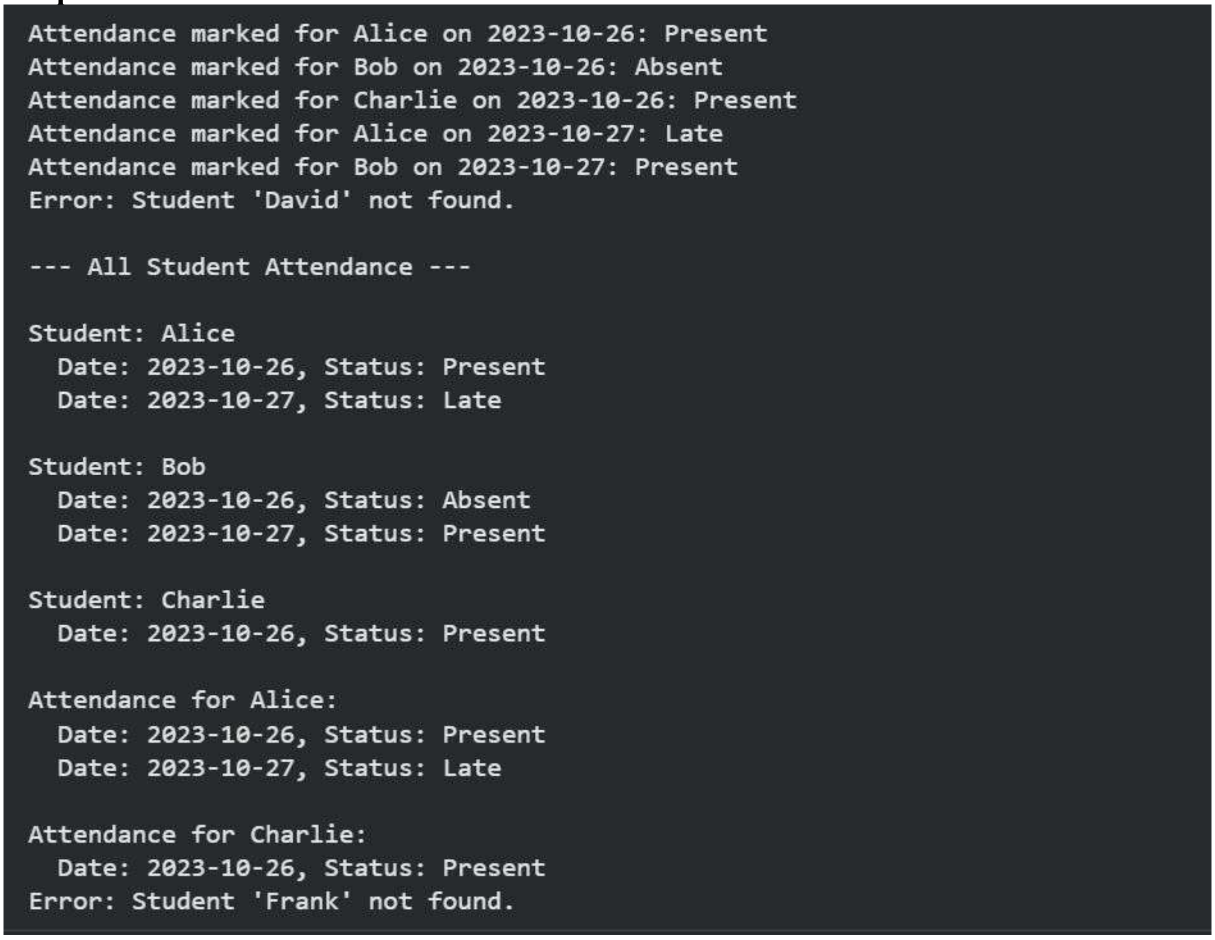
Task Description #4 (AI-Assisted Code Completion for Class-Based Attendance System)
Task: Use an AI tool to generate an attendance management class.

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



```python
class AttendanceSystem:
    def __init__(self, student_names):
        self.student_names = student_names
        # Initialize attendance for each student as a dictionary of date: status
        self.attendance = {name: {} for name in student_names}

    def mark_attendance(self, student_name, date, status):
        if student_name not in self.student_names:
            print(f"Error: Student '{student_name}' not found.")
            return

        # Convert status to a common format (e.g., 'Present', 'Absent', 'Late')
        status = status.capitalize()

        self.attendance[student_name][date] = status
        print(f"Attendance marked for {student_name} on {date}: {status}")

    def _print_student_details(self, dates):
        """Helper to print attendance details for a given set of dates."""
        if not dates:
            print("  No attendance recorded yet.")
        else:
            for date, status in dates.items():
                print(f"  Date: {date}, Status: {status}")

    def display_attendance(self, student_name=None):
        if not self.attendance and not student_name: # Handle case where no records exist at all
            print("No attendance records available.")
            return

        if student_name:
            if student_name not in self.student_names:
                print(f"Error: Student '{student_name}' not found.")
                return
            print(f"\nAttendance for {student_name}:")
            self._print_student_details(self.attendance[student_name])
        else:
            print("\n--- All Student Attendance ---")
            # Iterate over original student names to maintain order and include all
            for student in self.student_names:
                print(f"\nStudent: {student}")
                # Use .get to safely retrieve attendance records for the student
                self._print_student_details(self.attendance.get(student, {}))

# Example Usage:
students = ["Alice", "Bob", "Charlie"]
attendance_system = AttendanceSystem(students)

# Mark attendance
attendance_system.mark_attendance("Alice", "2023-10-26", "Present")
attendance_system.mark_attendance("Bob", "2023-10-26", "Absent")
attendance_system.mark_attendance("Charlie", "2023-10-26", "Present")
attendance_system.mark_attendance("Alice", "2023-10-27", "Late")
attendance_system.mark_attendance("Bob", "2023-10-27", "Present")

# Try to mark attendance for a non-existent student
attendance_system.mark_attendance("David", "2023-10-26", "Present")

# Display attendance for all students
attendance_system.display_attendance()

# Display attendance for a specific student
attendance_system.display_attendance("Alice")
attendance_system.display_attendance("Charlie")

# Display attendance for a student with no records (or after removing some marks)
attendance_system.display_attendance("Frank")
```

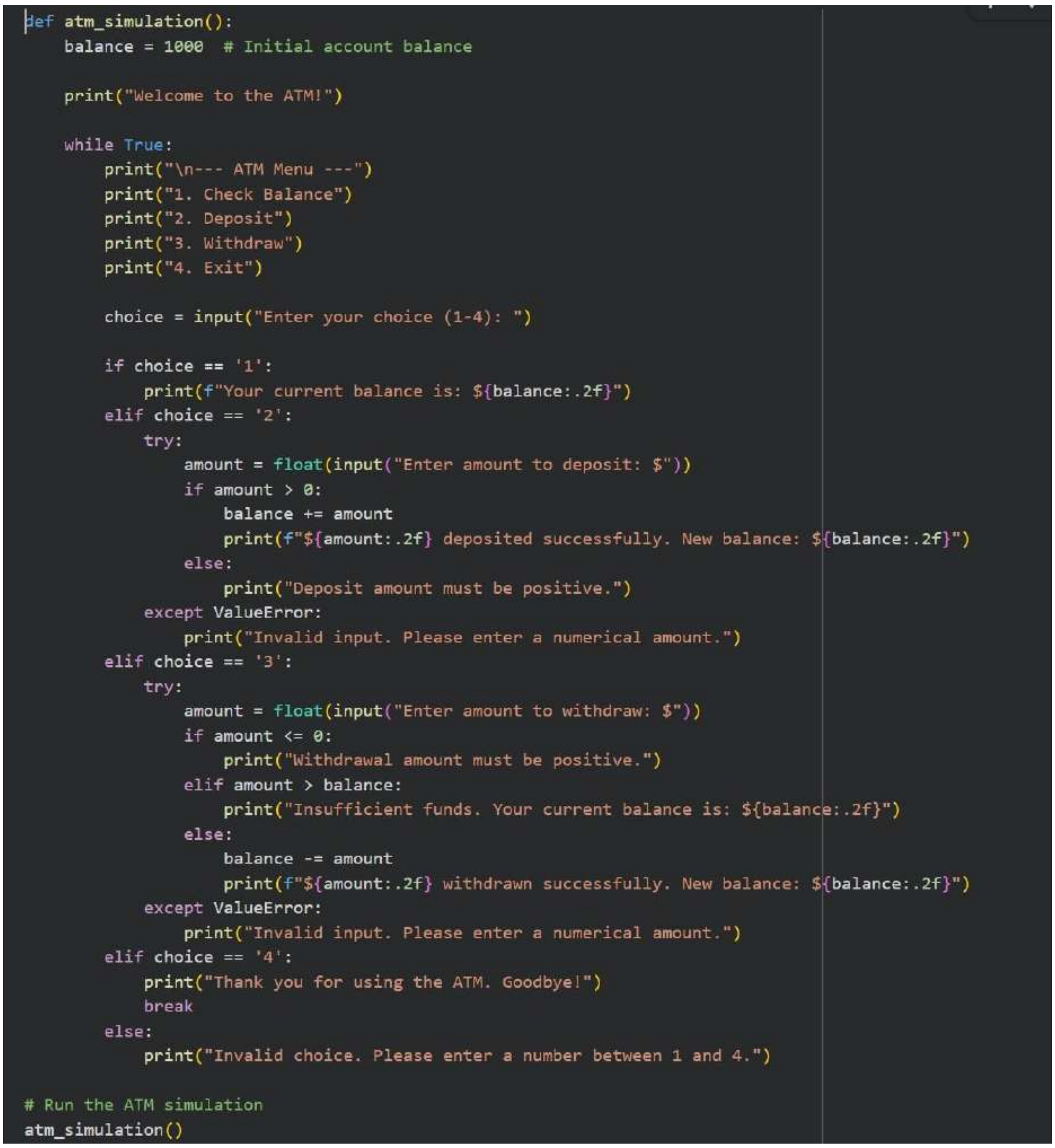
Output:


```
Attendance marked for Alice on 2023-10-26: Present
Attendance marked for Bob on 2023-10-26: Absent
Attendance marked for Charlie on 2023-10-26: Present
Attendance marked for Alice on 2023-10-27: Late
Attendance marked for Bob on 2023-10-27: Present
Error: Student 'David' not found.

--- All Student Attendance ---

Student: Alice
  Date: 2023-10-26, Status: Present
  Date: 2023-10-27, Status: Late

Student: Bob
  Date: 2023-10-26, Status: Absent
  Date: 2023-10-27, Status: Present

Student: Charlie
  Date: 2023-10-26, Status: Present

Attendance for Alice:
  Date: 2023-10-26, Status: Present
  Date: 2023-10-27, Status: Late

Attendance for Charlie:
  Date: 2023-10-26, Status: Present
Error: Student 'Frank' not found.
```

Task Description #5 (AI-Based Code Completion for Conditional Menu Navigation)
Task: Use an AI tool to complete a navigation menu.

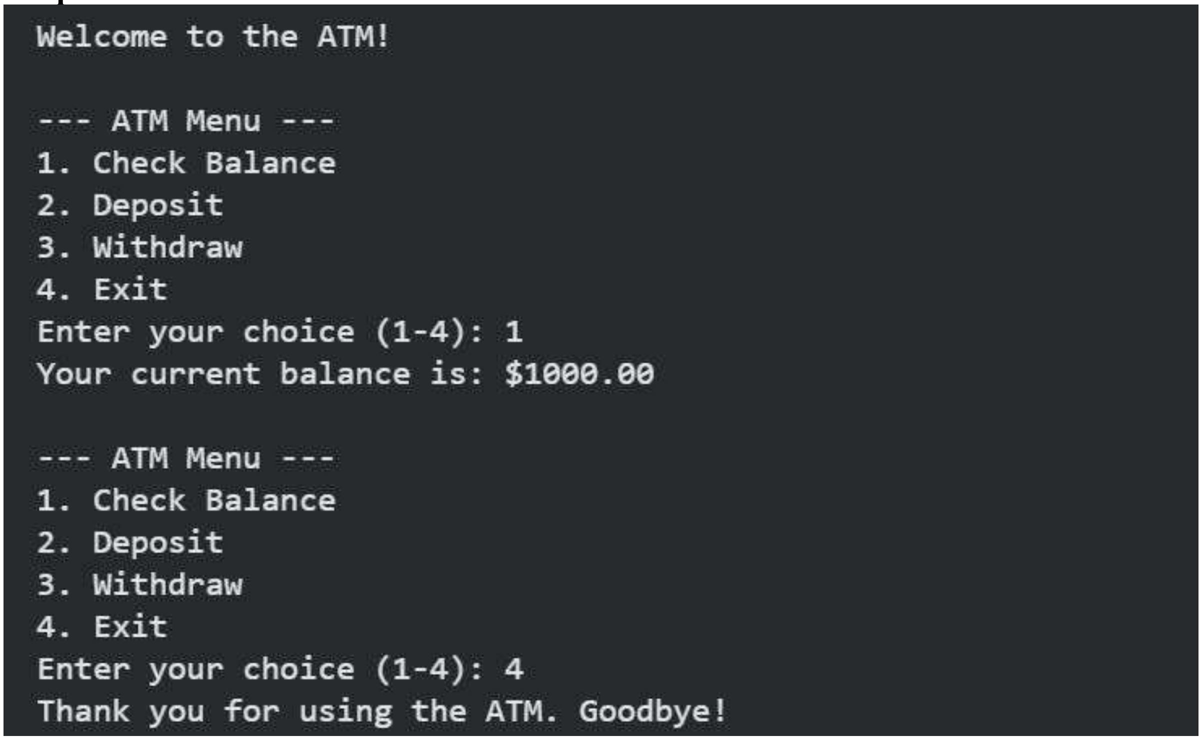
Prompt:
Generate a Python program using loops and conditionals to simulate an ATM menu.

Code:



```python
def atm_simulation():
    balance = 1000  # Initial account balance

    print("Welcome to the ATM!")

    while True:
        print("\n--- ATM Menu ---")
        print("1. Check Balance")
        print("2. Deposit")
        print("3. Withdraw")
        print("4. Exit")

        choice = input("Enter your choice (1-4): ")

        if choice == '1':
            print(f"Your current balance is: ${balance:.2f}")
        elif choice == '2':
            try:
                amount = float(input("Enter amount to deposit: $"))
                if amount > 0:
                    balance += amount
                    print(f"${amount:.2f} deposited successfully. New balance: ${balance:.2f}")
                else:
                    print("Deposit amount must be positive.")
            except ValueError:
                print("Invalid input. Please enter a numerical amount.")
        elif choice == '3':
            try:
                amount = float(input("Enter amount to withdraw: $"))
                if amount <= 0:
                    print("Withdrawal amount must be positive.")
                elif amount > balance:
                    print("Insufficient funds. Your current balance is: ${balance:.2f}")
                else:
                    balance -= amount
                    print(f"${amount:.2f} withdrawn successfully. New balance: ${balance:.2f}")
            except ValueError:
                print("Invalid input. Please enter a numerical amount.")
        elif choice == '4':
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 4.")

# Run the ATM simulation
atm_simulation()
```

Output:



```
Welcome to the ATM!

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

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