AI ASSISTANT CODING

ASSIGNMENT-6

Name: Kommu Madhupriya

Hallticket:2303A51583

Batch:22

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.”

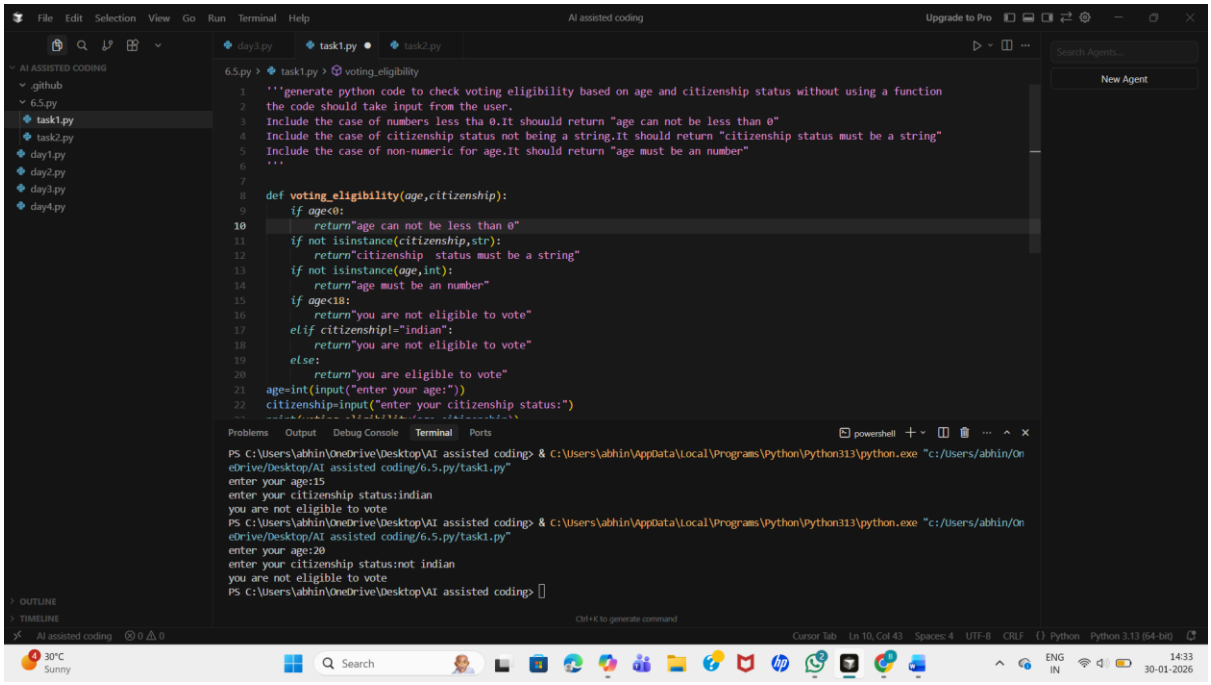
Expected Output:

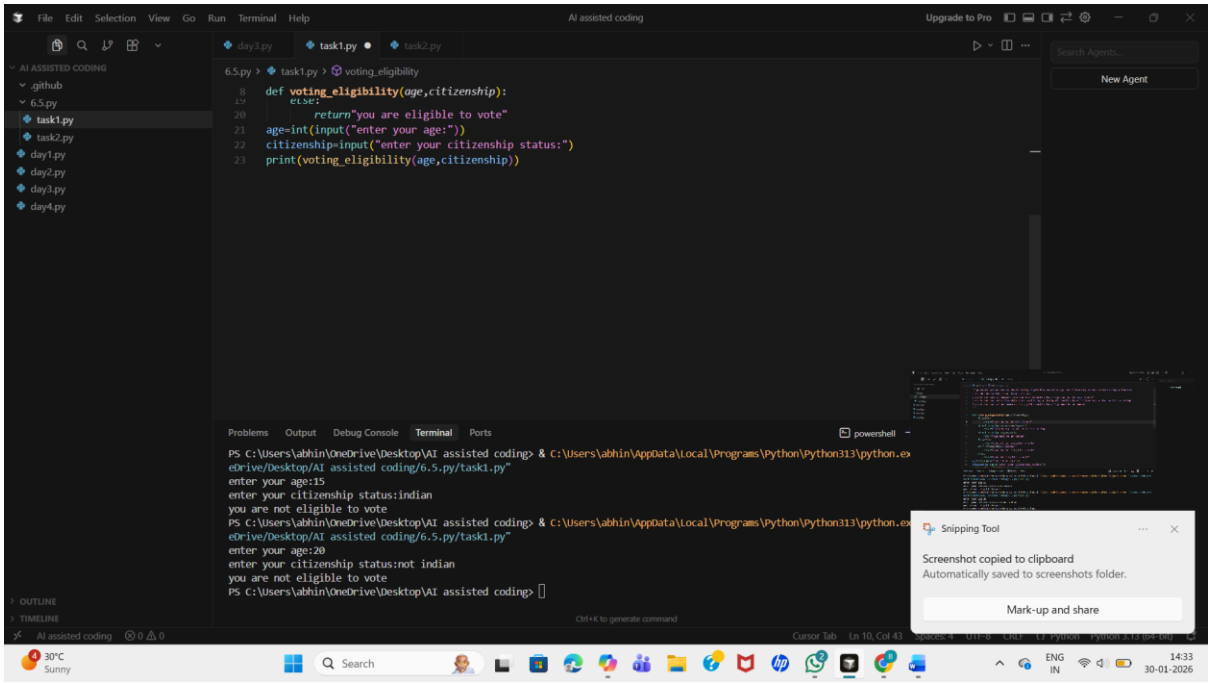
• AI-generated conditional logic.

• Correct eligibility decisions.

• Explanation of conditions.

code:





EXPLANATION:

- The AI-generated Python program was used to design a voting eligibility check with proper input validation and clear decision logic.
- The code was carefully reviewed line by line to understand how age and citizenship inputs are validated, including handling missing, invalid, and incorrect values.
- Logical flaws and potential errors were identified and addressed by adding appropriate condition checks and exception handling.
- The program was further refined to improve readability and maintainability through meaningful variable names, structured validation, and clear comments.
- Responsible use of AI tools was ensured by verifying the logic, correcting mistakes, and fully understanding the code rather than copying the AI-generated output without evaluation.

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.”

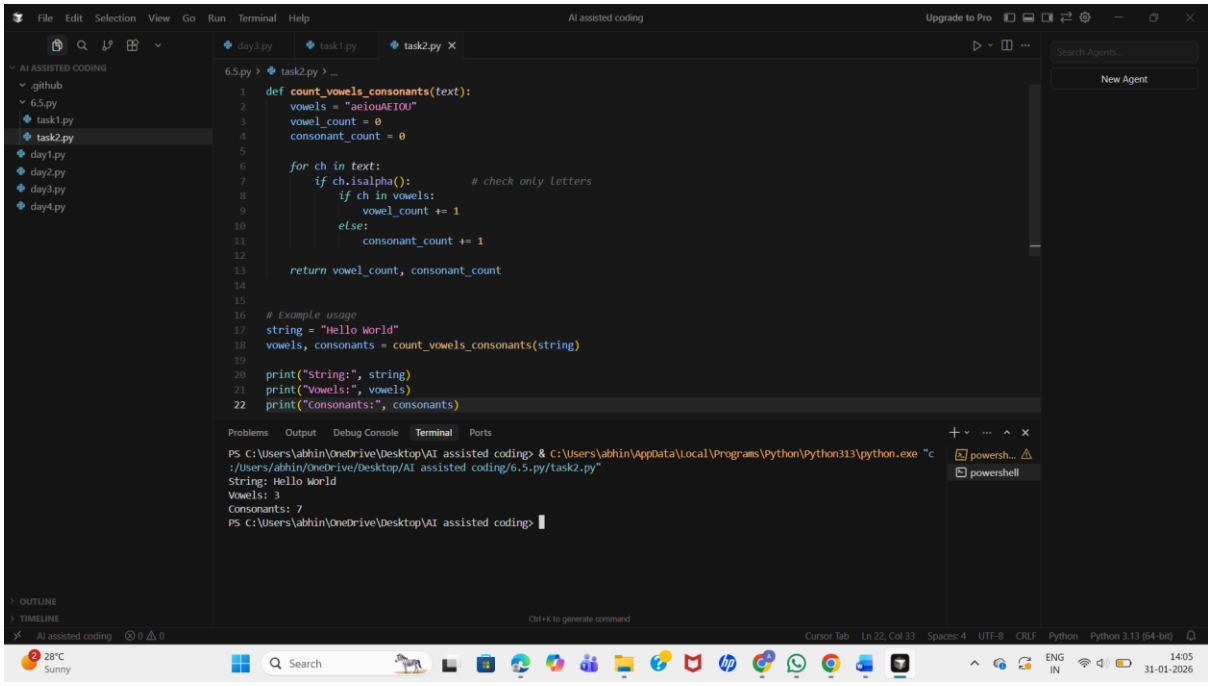
Expected Output:

• AI-generated string processing logic.

• Correct counts.

• Output verification.

code:



EXPLANATION:

- The AI tool was used to generate a Python program that applies a user-defined function, loop-based logic, and conditional statements to count vowels and consonants in a string.
- The generated code was carefully examined line by line to understand how input validation, character checking, and counting logic work together to produce correct results.
- During the review process, potential issues such as empty inputs, non-string values, and invalid characters were identified and handled appropriately to ensure logical correctness.
- The program was further refined to improve readability and efficiency through clear comments, structured conditions, and meaningful variable names.

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.”

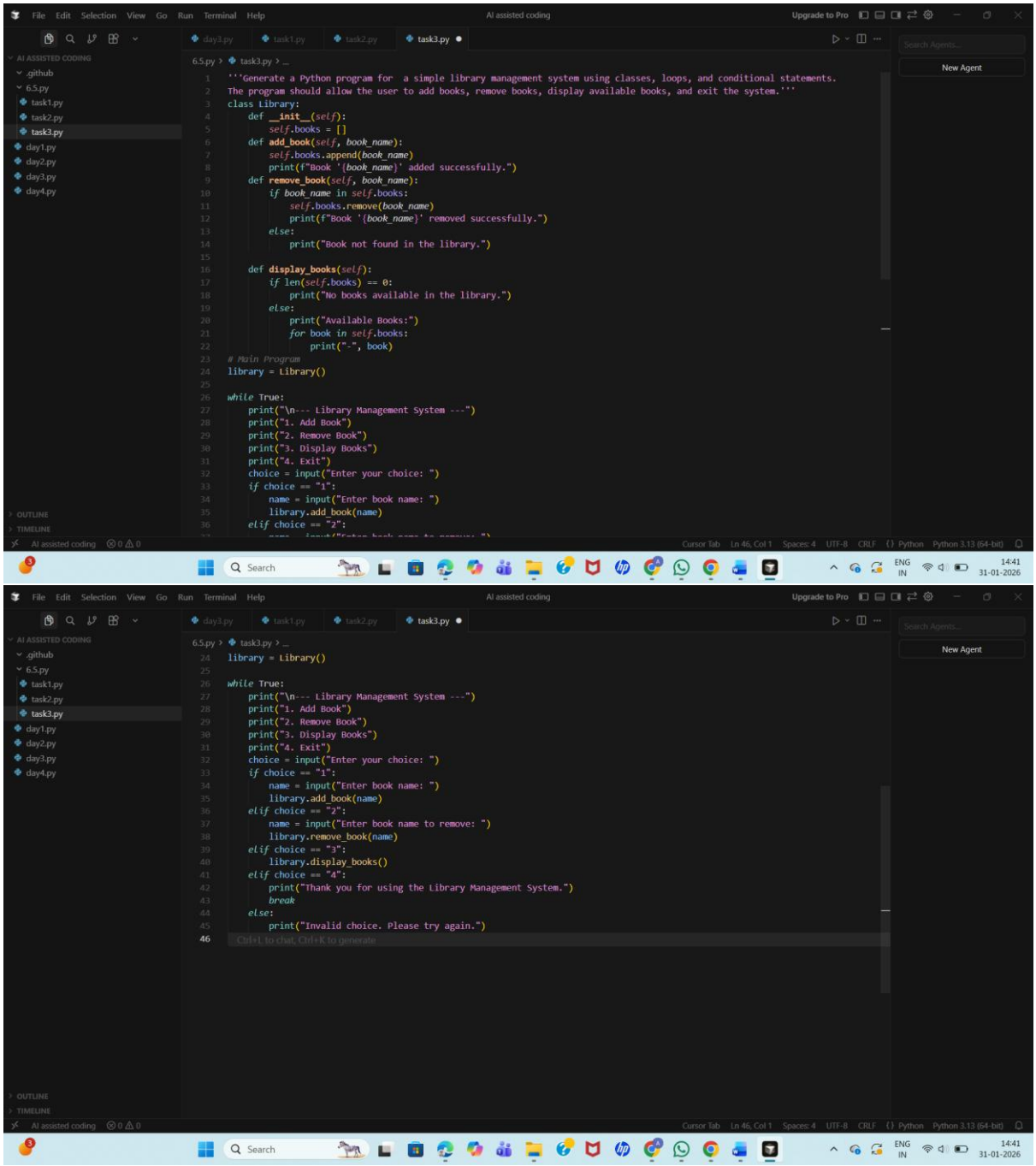
Expected Output:

• Complete AI-generated program.

• Review of AI suggestions quality.

• Short reflection on AI-assisted coding experience.

code:

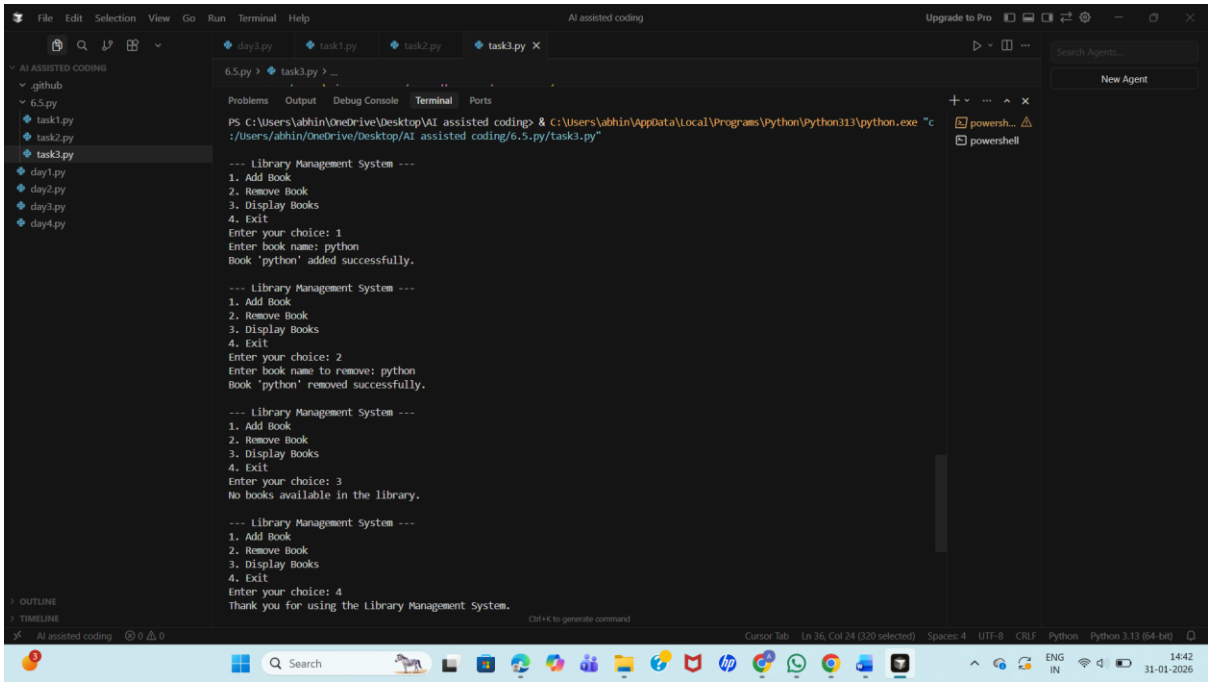


```
'''Generate a Python program for  a simple library management system using classes, loops, and conditional statements.
The program should allow the user to add books, remove books, display available books, and exit the system.'''
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book_name):
        self.books.append(book_name)
        print(f"Book '{book_name}' added successfully.")
    def remove_book(self, book_name):
        if book_name in self.books:
            self.books.remove(book_name)
            print(f"Book '{book_name}' removed successfully.")
        else:
            print("Book not found in the library.")

    def display_books(self):
        if len(self.books) == 0:
            print("No books available in the library.")
        else:
            print("Available Books:")
            for book in self.books:
                print("-", book)
# Main Program
library = Library()

while True:
    print("\n--- Library Management System ---")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Display Books")
    print("4. Exit")
    choice = input("Enter your choice: ")
    if choice == "1":
        name = input("Enter book name: ")
        library.add_book(name)
    elif choice == "2":
```



```
library = Library()

while True:
    print("\n--- Library Management System ---")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Display Books")
    print("4. Exit")
    choice = input("Enter your choice: ")
    if choice == "1":
        name = input("Enter book name: ")
        library.add_book(name)
    elif choice == "2":
        name = input("Enter book name to remove: ")
        library.remove_book(name)
    elif choice == "3":
        library.display_books()
    elif choice == "4":
        print("Thank you for using the Library Management System.")
        break
    else:
        print("Invalid choice. Please try again.")
```

output:



Explanation:

- The AI-generated Python program was used to design a simple library management system using a class-based approach along with loops and conditional statements for menu-driven operations.
- The code was carefully reviewed line by line to understand how object-oriented concepts, such as classes, methods, and instance variables, manage book records and availability status.
- Logical conditions were examined to ensure correct handling of operations like adding, displaying, issuing, and returning books, while also preventing invalid actions such as issuing an already issued book.
- The program was refined for better readability and maintainability through meaningful method names, clear comments, and structured input validation within the menu loop.
- Responsible use of AI tools was demonstrated by verifying the generated logic, handling user input errors properly, and ensuring a clear understanding of the program's functionality rather than relying on the AI output without evaluation

---

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."

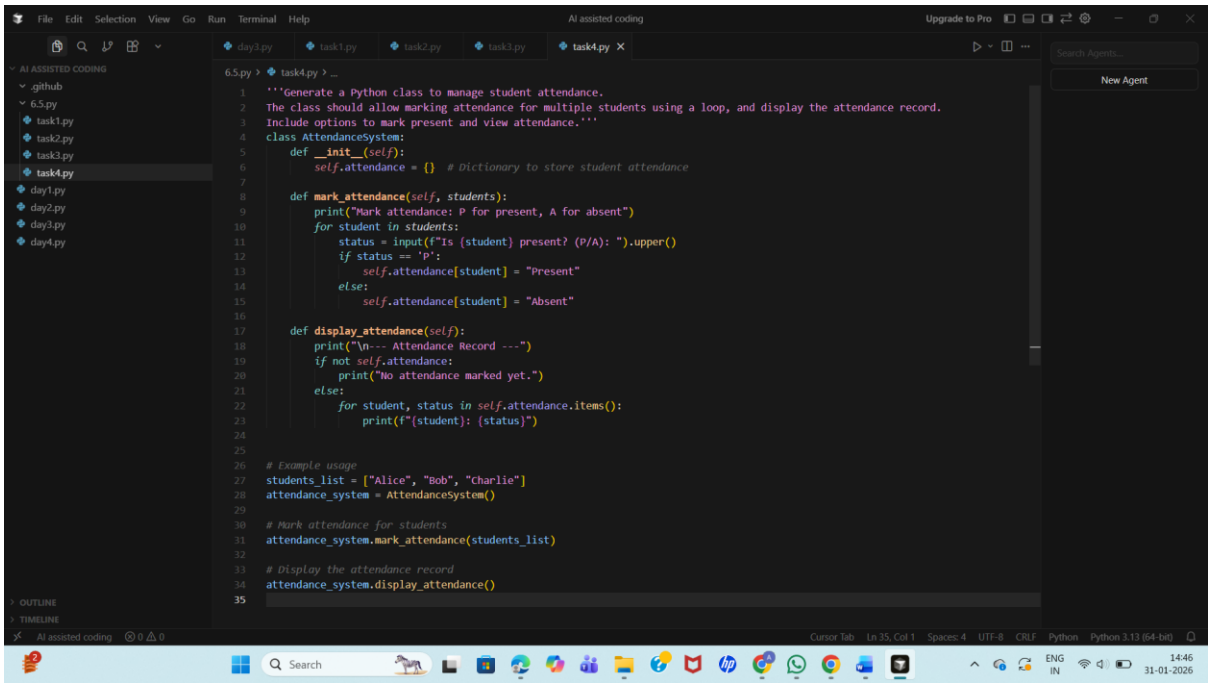
Expected Output:

• AI-generated attendance logic.

• Correct display of attendance.

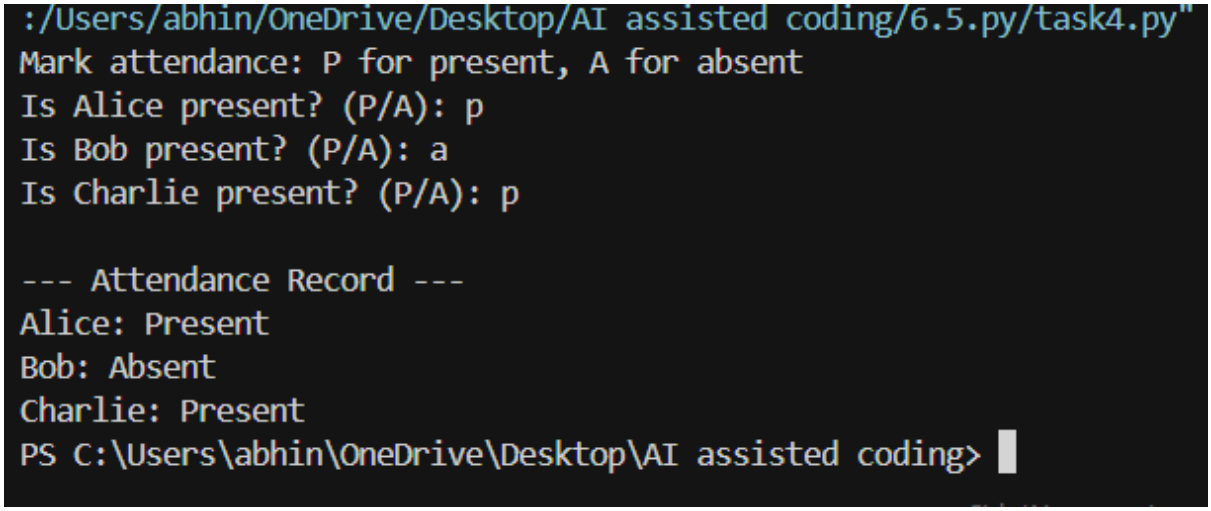
• Test cases.

code:



```
'''Generate a Python class to manage student attendance.
The class should allow marking attendance for multiple students using a loop, and display the attendance record.
Include options to mark present and view attendance.'''
class AttendanceSystem:
    def __init__(self):
        self.attendance = {}  # Dictionary to store student attendance

    def mark_attendance(self, students):
        print("Mark attendance: P for present, A for absent")
        for student in students:
            status = input(f"Is {student} present? (P/A): ").upper()
            if status == 'P':
                self.attendance[student] = "Present"
            else:
                self.attendance[student] = "Absent"

    def display_attendance(self):
        print("\n--- Attendance Record ---")
        if not self.attendance:
            print("No attendance marked yet.")
        else:
            for student, status in self.attendance.items():
                print(f"{student}: {status}")


# Example usage
students_list = ["Alice", "Bob", "Charlie"]
attendance_system = AttendanceSystem()

# Mark attendance for students
attendance_system.mark_attendance(students_list)

# Display the attendance record
attendance_system.display_attendance()
```

output:



```
:/Users/abhin/OneDrive/Desktop/AI assisted coding/6.5.py/task4.py"
Mark attendance: P for present, A for absent
Is Alice present? (P/A): p
Is Bob present? (P/A): a
Is Charlie present? (P/A): p

--- Attendance Record ---
Alice: Present
Bob: Absent
Charlie: Present
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding>
```

Explanation:

- The AI-generated Python program was used to develop an attendance management system using a class-based structure along with loops and conditional statements for menu-driven operations.
- The code was reviewed in detail to understand how methods are used to add students, mark attendance as present or absent, and display attendance records using dictionary-based storage.
- Conditional checks were analysed to ensure proper handling of invalid inputs such as empty names, incorrect attendance status, and non-existing students. The implementation was

refined to improve readability and maintainability by using meaningful method names, clear comments, and structured control flow within the loop.

- Responsible use of AI tools was demonstrated by validating the generated logic, handling edge cases correctly, and ensuring a clear understanding of the program's functionality rather than relying on the AI output without verification.

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."

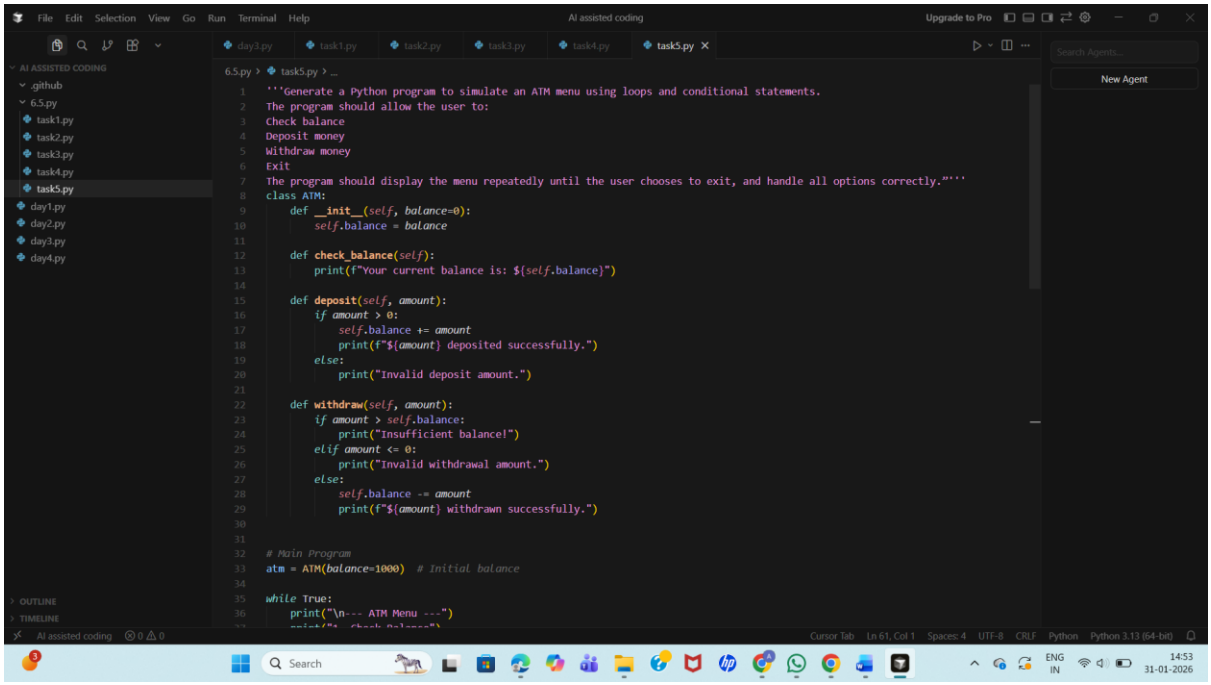
Expected Output:

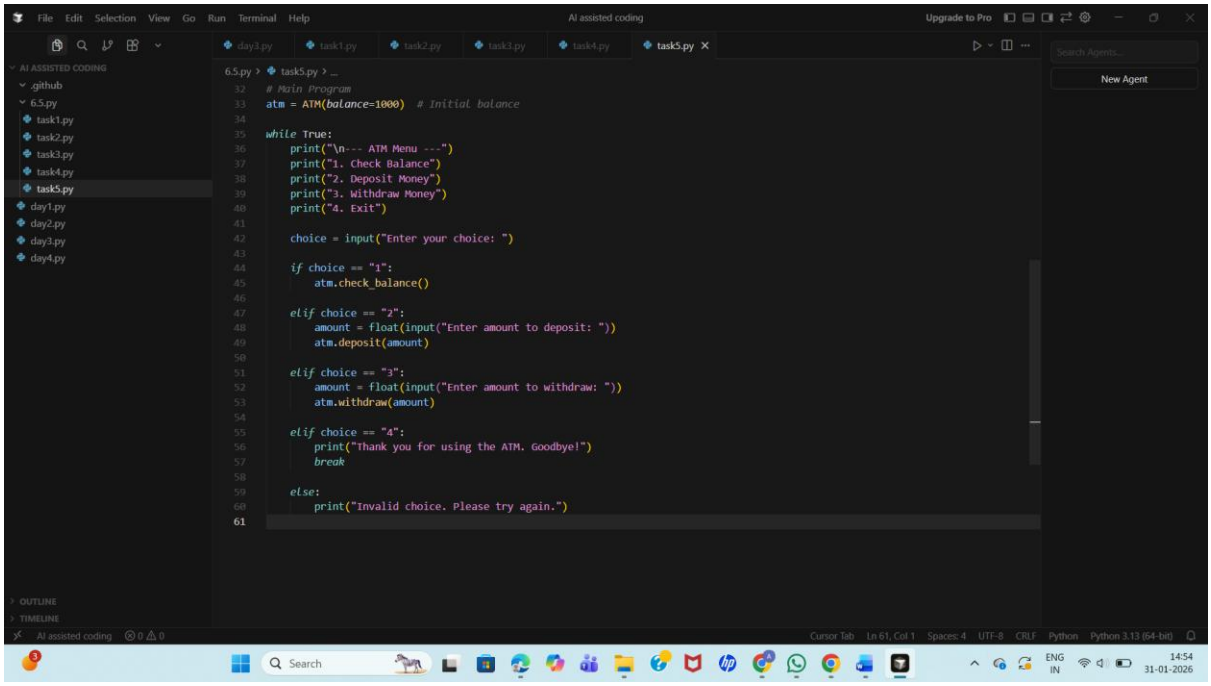
• AI-generated menu logic.

• Correct option handling.

• Output verification.

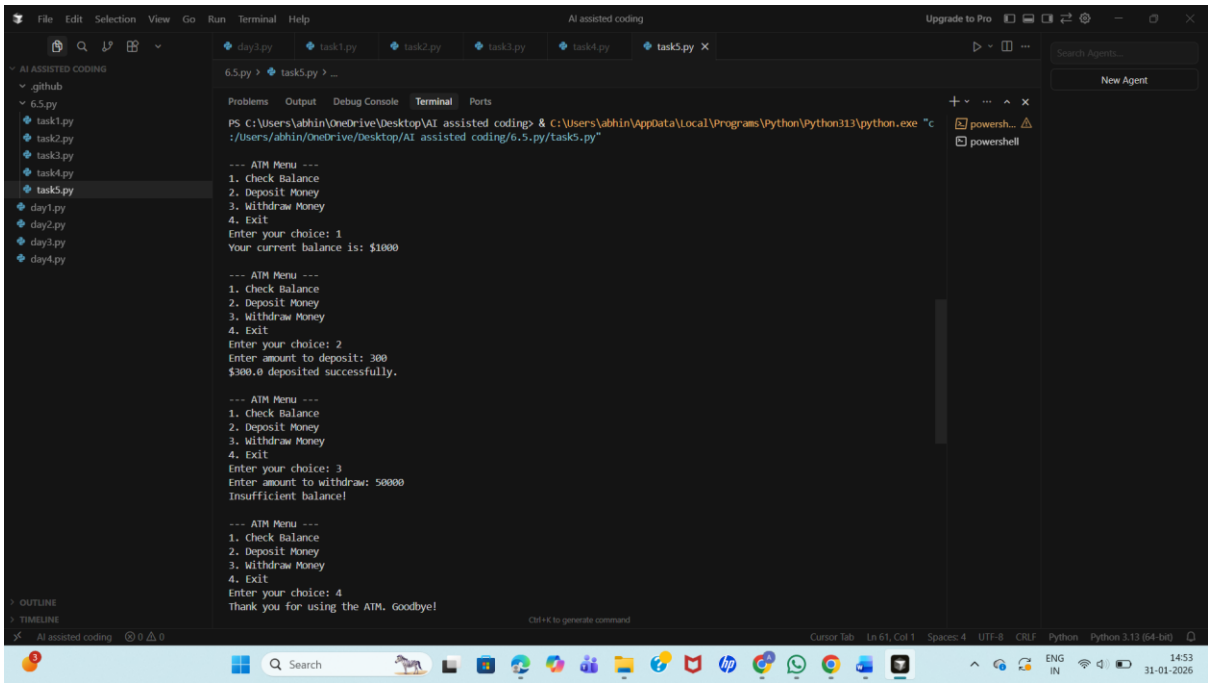
Code:





```python
# Main Program
atm = ATM(balance=1000)  # Initial 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":
        atm.check_balance()

    elif choice == "2":
        amount = float(input("Enter amount to deposit: "))
        atm.deposit(amount)

    elif choice == "3":
        amount = float(input("Enter amount to withdraw: "))
        atm.withdraw(amount)

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

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

output:



```
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding> & C:\Users\abhin\AppData\Local\Programs\Python\Python313\python.exe "c
:/Users/abhin/OneDrive/Desktop/AI assisted coding/6.5.py/task5.py"

--- 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: 2
Enter amount to deposit: 300
$300.0 deposited successfully.

--- ATM Menu ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 3
Enter amount to withdraw: 50000
Insufficient balance!

--- 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:

- The AI-generated Python program was used to simulate an ATM system using loops and conditional statements to provide a menu-driven interface.
- The code was carefully examined to understand how the loop keeps the menu running until the user chooses to exit and how conditional branches handle balance inquiry, deposit, and withdrawal operations. Input validation was analysed to ensure that invalid menu selections, non-numeric inputs, negative amounts, and insufficient balance cases are handled correctly.

- The program structure was reviewed to identify and prevent logical errors, while clear comments and meaningful variable names were used to improve readability and maintainability.
- Responsible use of AI tools was demonstrated by verifying the generated logic, testing different transaction scenarios, and ensuring correct behaviour rather than relying on the AI output without evaluation.