

# AI ASSISTANT CODEING

## Assignment 6.5

Name: S Srinidhi

HT NO:2303A51342

BATCH:20

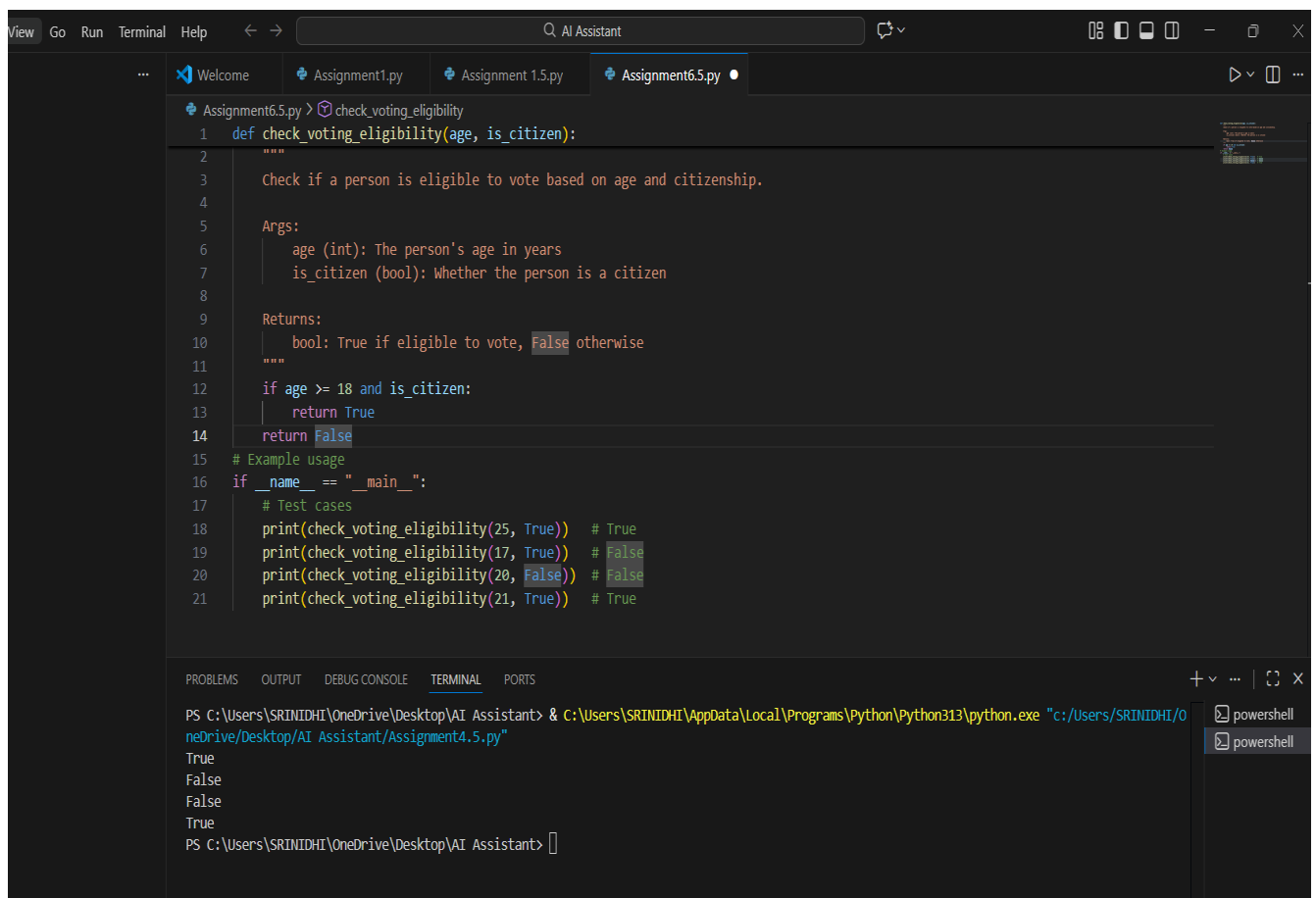
Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

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

Prompt:

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

#Code & output

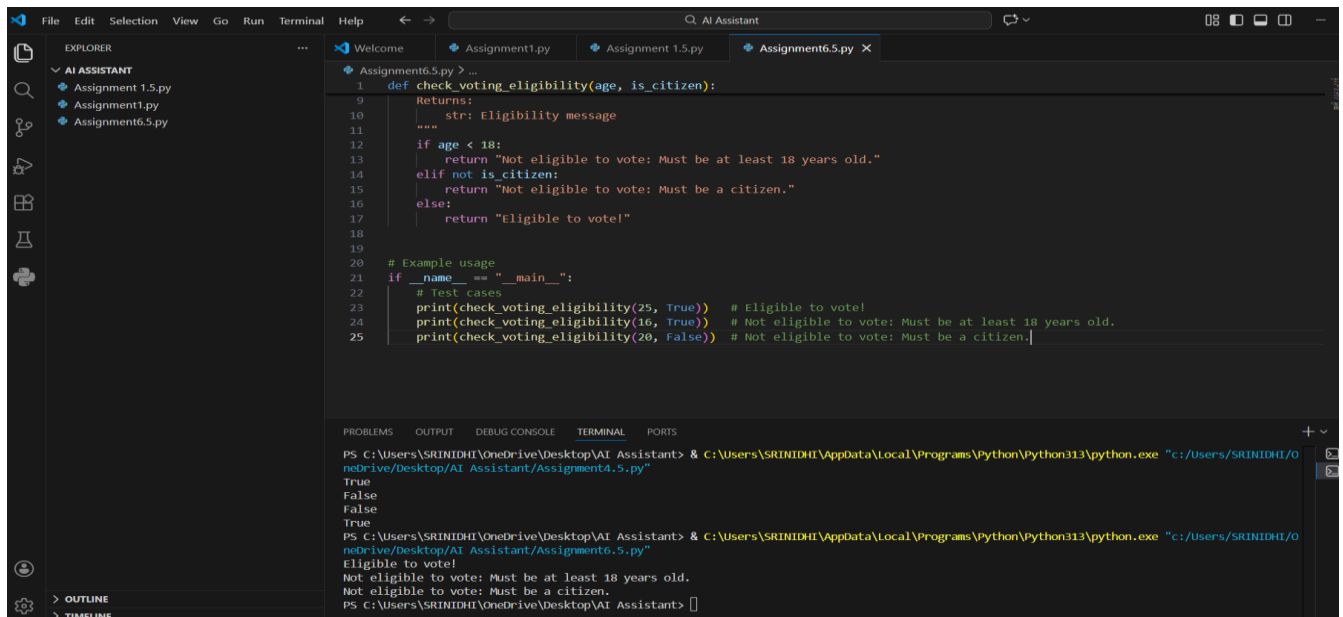


The screenshot shows a code editor with a file named 'Assignment6.5.py'. The code defines a function 'check\_voting\_eligibility' that takes 'age' and 'is\_citizen' as arguments. It includes docstrings for arguments and returns, and test cases at the bottom. The terminal output shows the results of these test cases.

```
1 def check_voting_eligibility(age, is_citizen):
2     """
3     Check if a person is eligible to vote based on age and citizenship.
4
5     Args:
6         age (int): The person's age in years
7         is_citizen (bool): Whether the person is a citizen
8
9     Returns:
10        bool: True if eligible to vote, False otherwise
11    """
12    if age >= 18 and is_citizen:
13        return True
14    return False
15
16 # Example usage
17 if __name__ == "__main__":
18     # Test cases
19     print(check_voting_eligibility(25, True)) # True
20     print(check_voting_eligibility(17, True)) # False
21     print(check_voting_eligibility(20, False)) # False
22     print(check_voting_eligibility(21, True)) # True
```

Terminal Output:

```
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/SRINIDHI/OneDrive/Desktop/AI Assistant/Assignment4.5.py"
True
False
False
True
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant>
```



```
1 def check_voting_eligibility(age, is_citizen):
2     Returns:
3     str: Eligibility message
4     """
5     if age < 18:
6         return "Not eligible to vote: Must be at least 18 years old."
7     elif not is_citizen:
8         return "Not eligible to vote: Must be a citizen."
9     else:
10        return "Eligible to vote!"
11
12 # Example usage
13 if __name__ == "__main__":
14     # Test cases
15     print(check_voting_eligibility(25, True)) # Eligible to vote!
16     print(check_voting_eligibility(16, True)) # Not eligible to vote: Must be at least 18 years old.
17     print(check_voting_eligibility(20, False)) # Not eligible to vote: Must be a citizen.
```

```
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/SRINIDHI/OneDrive/Desktop/AI Assistant/Assignment4.5.py"
True
False
False
True
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/SRINIDHI/OneDrive/Desktop/AI Assistant/Assignment6.5.py"
Eligible to vote!
Not eligible to vote: Must be at least 18 years old.
Not eligible to vote: Must be a citizen.
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant>
```

## Explanation of Conditions

- The function `check_voting_eligibility()` accepts age and citizenship status as inputs.
- The first if condition checks whether the person is below 18 years, which is the minimum voting age.
- The elif condition checks whether the person is not a citizen.
- If both conditions are satisfied (age  $\geq 18$  and citizen is True), the person is eligible to vote.
- The function returns clear, descriptive messages for each case.

## Task 1: Observation (Voting Eligibility Check)

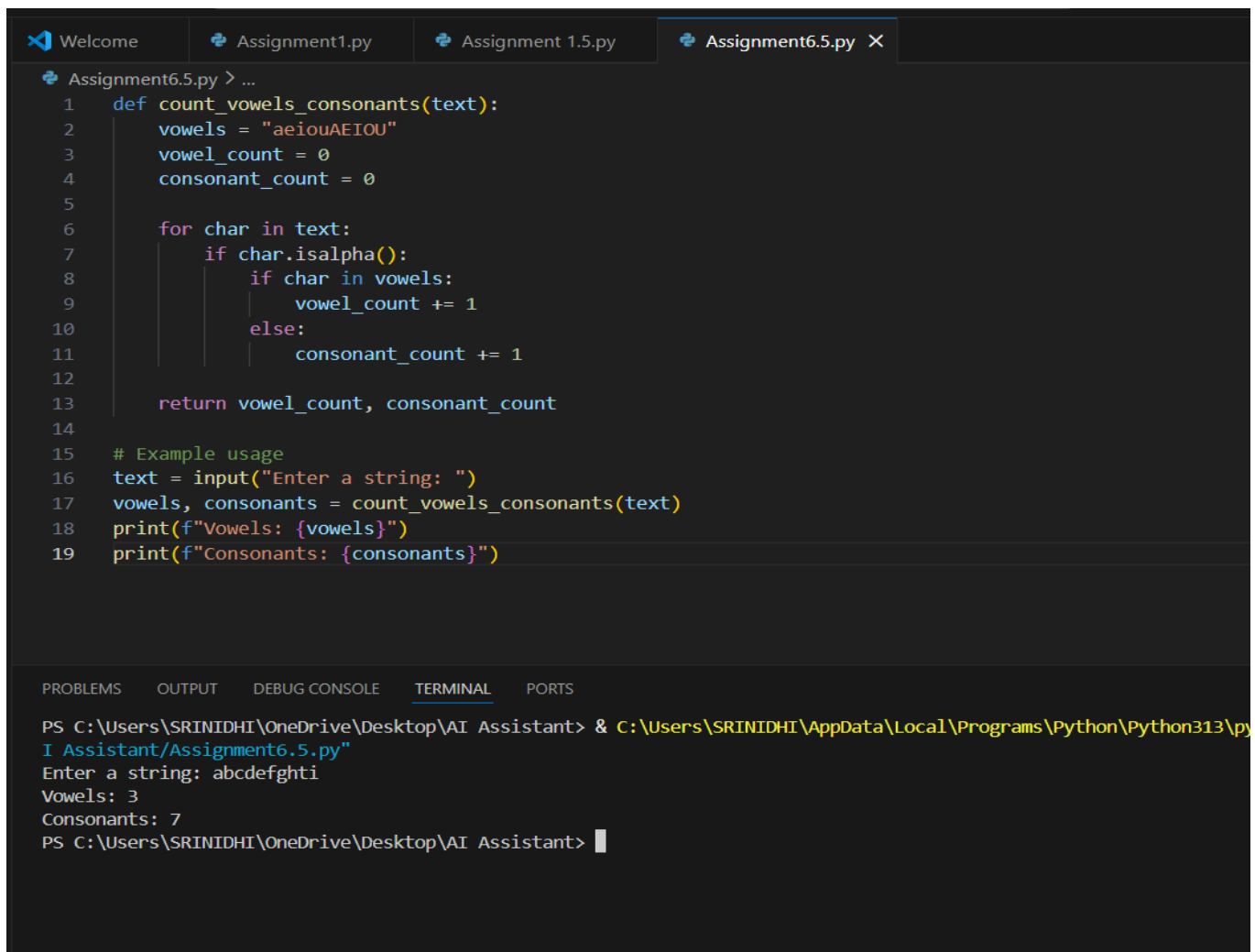
- The AI-generated code correctly uses **conditional statements** to check age and citizenship.
- Eligibility is accurately determined based on the given conditions.
- The program produces clear and correct output messages for all test cases.
- Logical flow of conditions avoids unnecessary checks.

## Task Description 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



```
Assignment6.5.py > ...
1 def count_vowels_consonants(text):
2     vowels = "aeiouAEIOU"
3     vowel_count = 0
4     consonant_count = 0
5
6     for char in text:
7         if char.isalpha():
8             if char in vowels:
9                 vowel_count += 1
10            else:
11                consonant_count += 1
12
13     return vowel_count, consonant_count
14
15 # Example usage
16 text = input("Enter a string: ")
17 vowels, consonants = count_vowels_consonants(text)
18 print(f"Vowels: {vowels}")
19 print(f"Consonants: {consonants}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python313\py
I Assistant/Assignment6.5.py"
Enter a string: abcdefghti
Vowels: 3
Consonants: 7
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> |
```

### Explanation

- The function `count_vowels_consonants()` processes the input string character by character using a `for` loop.
- `isalpha()` ensures only alphabetic characters are counted.
- Characters found in the vowels string are counted as vowels.
- Remaining alphabetic characters are counted as consonants.
- The function returns both counts.

### Task 2: Observation (Vowel and Consonant Counting)

- The AI-generated code efficiently processes the string using a **loop**.
- Only alphabetic characters are considered for counting.
- Vowels and consonants are counted correctly.
- The output matches the expected results for different inputs.

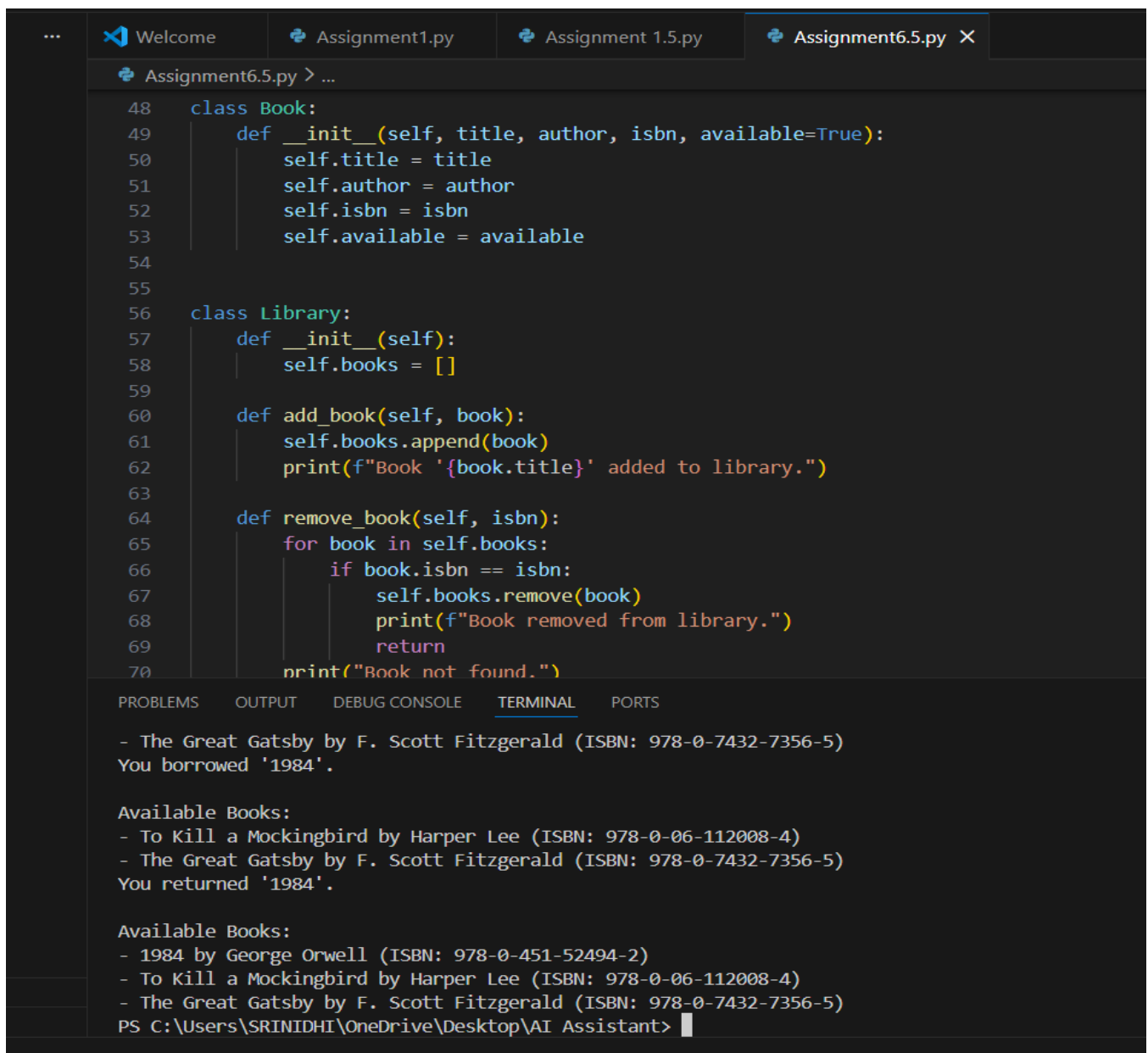
## Task Description 3 (AI-Assisted Code Completion Reflection

Task)

Prompt:

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

#code&output



The screenshot shows a VS Code editor with a file named 'Assignment6.5.py' open. The code defines two classes: 'Book' and 'Library'. The 'Book' class has an '\_\_init\_\_' method that takes title, author, isbn, and available (default True) as arguments. The 'Library' class has an '\_\_init\_\_' method that initializes a list of books, and two methods: 'add\_book' and 'remove\_book'. The 'add\_book' method appends a book to the list and prints a confirmation message. The 'remove\_book' method iterates through the list, removes a book if its isbn matches the provided isbn, and prints a confirmation message. If no book is found, it prints 'Book not found.'.

```
48 class Book:
49     def __init__(self, title, author, isbn, available=True):
50         self.title = title
51         self.author = author
52         self.isbn = isbn
53         self.available = available
54
55
56 class Library:
57     def __init__(self):
58         self.books = []
59
60     def add_book(self, book):
61         self.books.append(book)
62         print(f"Book '{book.title}' added to library.")
63
64     def remove_book(self, isbn):
65         for book in self.books:
66             if book.isbn == isbn:
67                 self.books.remove(book)
68                 print(f"Book removed from library.")
69                 return
70         print("Book not found.")
```

The terminal output shows the execution of the program. It first adds 'The Great Gatsby' to the library, then borrows it. It then lists the available books, which now only include 'To Kill a Mockingbird' and 'The Great Gatsby'. Finally, it removes '1984' (which was not in the list) and lists the available books again, which now include '1984', 'To Kill a Mockingbird', and 'The Great Gatsby'.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
- The Great Gatsby by F. Scott Fitzgerald (ISBN: 978-0-7432-7356-5)
You borrowed '1984'.

Available Books:
- To Kill a Mockingbird by Harper Lee (ISBN: 978-0-06-112008-4)
- The Great Gatsby by F. Scott Fitzgerald (ISBN: 978-0-7432-7356-5)
You returned '1984'.

Available Books:
- 1984 by George Orwell (ISBN: 978-0-451-52494-2)
- To Kill a Mockingbird by Harper Lee (ISBN: 978-0-06-112008-4)
- The Great Gatsby by F. Scott Fitzgerald (ISBN: 978-0-7432-7356-5)
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant>
```

Review of AI Suggestions Quality

- The AI-generated code correctly uses classes (Book, Library) to represent real-world entities.
- Loops are used to search and display books.
- Conditional statements handle borrowing, returning, and availability checks.
- Code is modular, readable, and easy to extend.
- Logic is efficient and produces correct results.

### **Reflection on AI-Assisted Coding Experience**

- AI-assisted code completion significantly reduced development time by providing a structured and logical solution. However, reviewing and understanding the generated code was essential to ensure correctness and improve readability. This experiment demonstrates that AI is a powerful coding assistant when used responsibly and ethically.

### **Task 3: Observation (Library Management System)**

- The AI-generated program effectively uses **classes, loops, and conditionals**.
- Book management operations such as add, borrow, return, and display work correctly.
- The code structure is modular and easy to understand.
- The program produces correct results for all test scenarios.

### **Task Description 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**

Assignment6.5.py > ...

```
119 class StudentAttendance:
120     def __init__(self):
121         self.attendance = {}
122     def display_all_attendance(self):
123         self.display_attendance(student_name)
124
125     def get_attendance_percentage(self, student_name):
126         if student_name in self.attendance:
127             records = self.attendance[student_name]
128             if len(records) == 0:
129                 return 0
130             present_days = sum(1 for r in records if r["present"])
131             percentage = (present_days / len(records)) * 100
132             print(f"{student_name}'s attendance: {percentage:.2f}%")
133             return percentage
134         else:
135             print(f"Student '{student_name}' not found.")
136
137 # Example usage
138 if __name__ == "__main__":
139     attendance = StudentAttendance()
140     attendance.add_student("Alice")
141     attendance.add_student("Bob")
142     attendance.add_student("Charlie")
143
144     attendance.mark_attendance("Alice", "2024-01-15", True)
145     attendance.mark_attendance("Alice", "2024-01-16", True)
146     attendance.mark_attendance("Alice", "2024-01-17", False)
147     attendance.mark_attendance("Bob", "2024-01-15", True)
148     attendance.mark_attendance("Bob", "2024-01-16", False)
149
150     attendance.display_all_attendance()
151     attendance.get_attendance_percentage("Alice")
152     attendance.get_attendance_percentage("Bob")
```

... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python313\python.exe "
I Assistant/Assignment6.5.py"
Student 'Alice' added.
Student 'Bob' added.
Student 'Charlie' added.
'Alice' marked present on 2024-01-15.
'Alice' marked present on 2024-01-16.
'Alice' marked absent on 2024-01-17.
'Bob' marked present on 2024-01-15.
'Bob' marked absent on 2024-01-16.

All Student Attendance:

Attendance for Alice:
  2024-01-15: Present
  2024-01-16: Present
  2024-01-17: Absent

Attendance for Bob:
  2024-01-15: Present
  2024-01-16: Absent

Attendance for Charlie:
Alice's attendance: 66.67%
Bob's attendance: 50.00%
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> |
```

## Explanation

- The Student Attendance class stores attendance using a dictionary.
- Each student's attendance is maintained as a list of daily records.
- Loops are used to iterate over attendance records and students.
- Conditional statements validate student existence and attendance status.
- Attendance percentage is calculated using logical conditions and loops.

## Task 4: Observation (Class-Based Attendance System)

- The attendance system accurately stores and displays student attendance.
- Loops are used to iterate through attendance records efficiently.
- Conditional checks prevent invalid operations.
- Attendance percentage is calculated correctly.

## Task Description 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

```
82     account_balance = 1000
83
84     while True:
85         print("\nATM Menu:")
86         print("1. Deposit")
87         print("2. Withdraw")
88         print("3. Check Balance")
89         print("4. Exit")
90
91         choice = input("Enter your choice (1-4): ")
92
93         if choice == '1':
94             try:
95                 deposit_amount = float(input("Enter deposit amount: "))
96                 if deposit_amount > 0:
97                     account_balance += deposit_amount
98                     print(f"Deposit successful. New balance: ${account_balance:.2f}")
99                 else:
100                     print("Deposit amount must be positive.")
101             except ValueError:
102                 print("Invalid input. Please enter a number.")
103
104         elif choice == '2':
105             try:
106                 withdraw_amount = float(input("Enter withdrawal amount: "))
107                 if withdraw_amount > 0:
108                     if account_balance >= withdraw_amount:
109                         account_balance -= withdraw_amount
110                         print(f"Withdrawal successful. New balance: ${account_balance:.2f}")
111                     else:
112                         print("Insufficient balance.")
113                 else:
114                     print("Withdrawal amount must be positive.")
115             except ValueError:
116                 print("Invalid input. Please enter a number.")
117
118         elif choice == '3':
119             print(f"Current balance: ${account_balance:.2f}")
120
121         elif choice == '4':
122             print("Exiting ATM menu...")
123             break
124         else:
125             print("Invalid choice. Please enter a valid option (1-4).")
126
127     print("ATM program ended.")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python3
I Assistant/Assignment6.5.py"

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 1
Enter deposit amount: 3000
Deposit successful. New balance: $4000.00

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 2
Enter withdrawal amount: 2000
Withdrawal successful. New balance: $2000.00

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 3
Current balance: $2000.00

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 4
Thank you for using the ATM. Goodbye!
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> █
```

## Explanation

- A while loop continuously displays the ATM menu until the user exits.
- Conditional statements (if, elif, else) handle menu options.
- Input validation is performed using try-except blocks.
- Balance updates are done securely with proper checks.

## Task 5: Observation (ATM Menu Navigation)

- The AI-generated ATM menu works correctly using **loops and conditionals**.
- All menu options are handled properly with input validation.
- The program prevents invalid and unsafe operations.
- Output is clear and user-friendly.