AI ASSISTANT CODEING

Assignment 6.5

Name: S Srinidhi

HT NO:2303A51342

BATCH:20

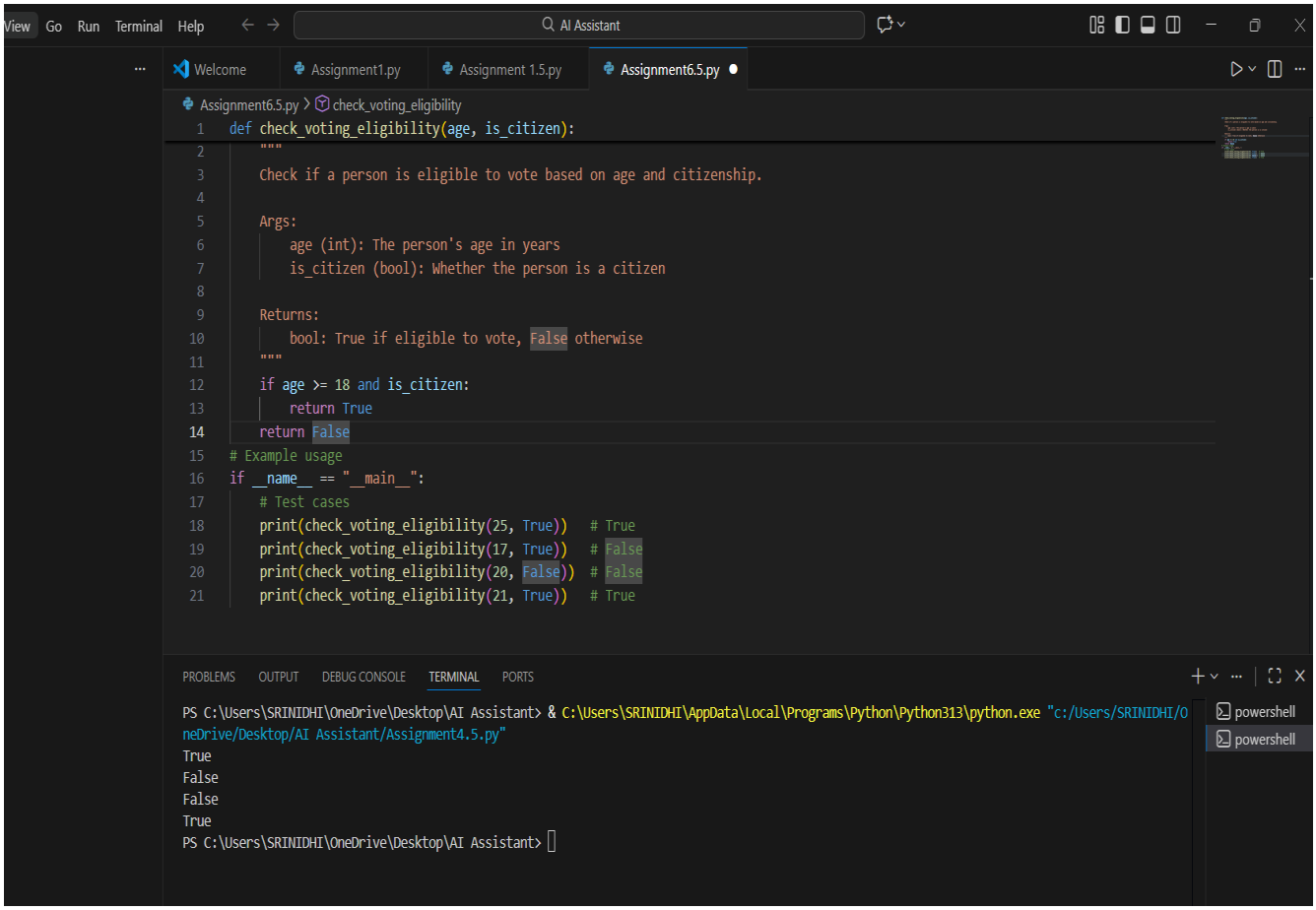
Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

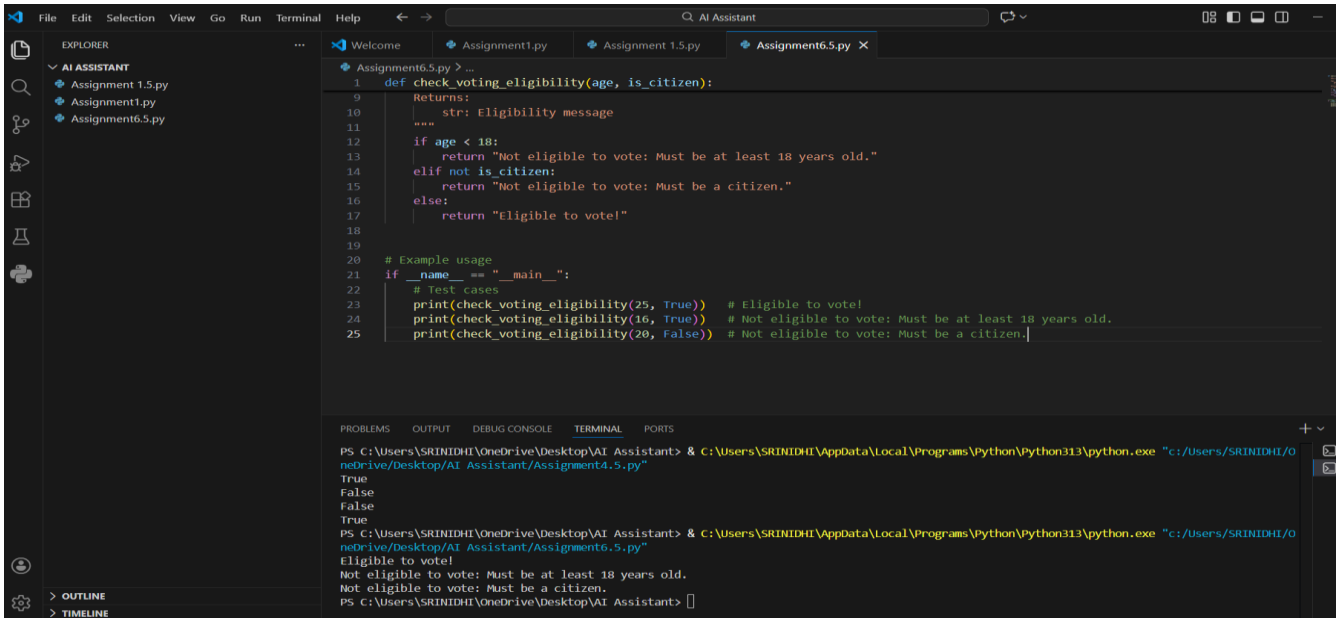
Task Description 1 (AI-Based Code Completion for Conditional Eligibility Check)

Prompt:

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

#Code & output





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 ≥ 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)

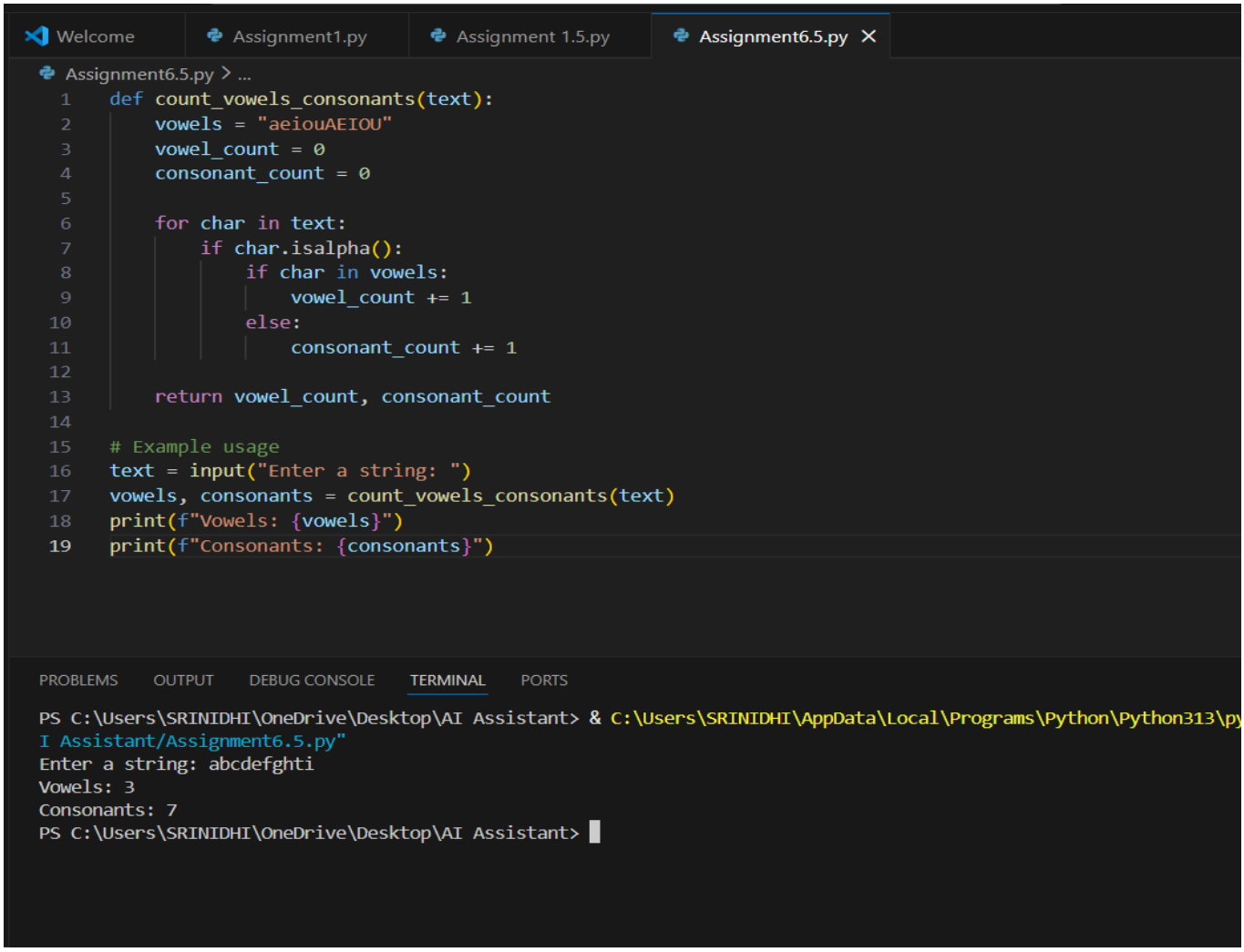
- The AI-generated code correctly uses **conditional statements** to check age and citizenship.
- Eligibility is accurately determined based on the given conditions.
- The program produces clear and correct output messages for all test cases.
- Logical flow of conditions avoids unnecessary checks.

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



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)

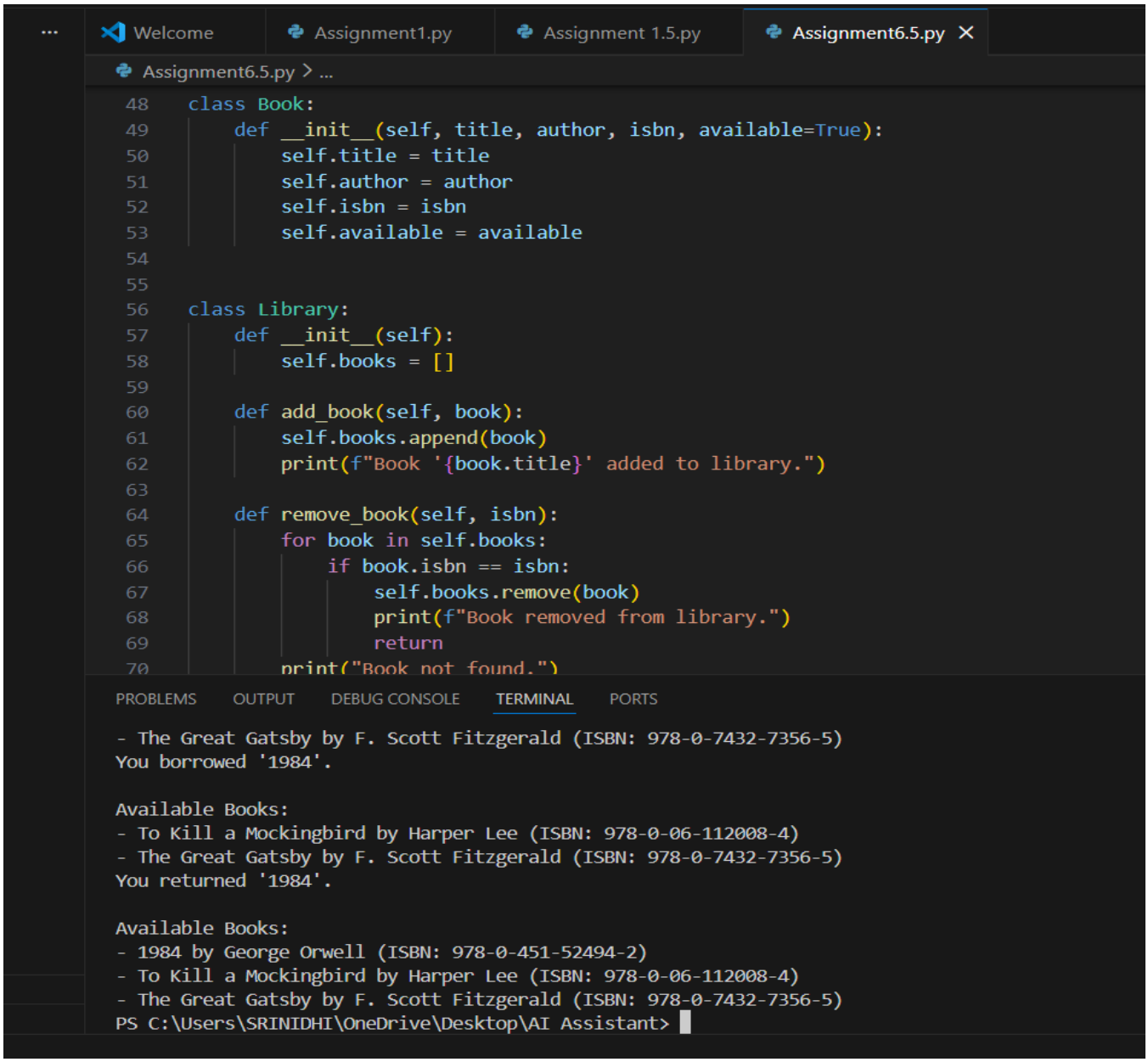
- The AI-generated code efficiently processes the string using a **loop**.
- Only alphabetic characters are considered for counting.
- Vowels and consonants are counted correctly.
- The output matches the expected results for different inputs.

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



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:
150            self.display_attendance(student_name)
151
152        def get_attendance_percentage(self, student_name):
153            if student_name in self.attendance:
154                records = self.attendance[student_name]
155                if len(records) == 0:
156                    return 0
157                present_days = sum(1 for r in records if r["present"])
158                percentage = (present_days / len(records)) * 100
159                print(f"{student_name}'s attendance: {percentage:.2f}%")
160                return percentage
161            else:
162                print(f"Student '{student_name}' not found.")
163
164
165    # Example usage
166    if __name__ == "__main__":
167        attendance = StudentAttendance()
168        attendance.add_student("Alice")
169        attendance.add_student("Bob")
170        attendance.add_student("Charlie")
171
172        attendance.mark_attendance("Alice", "2024-01-15", True)
173        attendance.mark_attendance("Alice", "2024-01-16", True)
174        attendance.mark_attendance("Alice", "2024-01-17", False)
175        attendance.mark_attendance("Bob", "2024-01-15", True)
176        attendance.mark_attendance("Bob", "2024-01-16", False)
177
178        attendance.display_all_attendance()
179        attendance.get_attendance_percentage("Alice")
180        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)

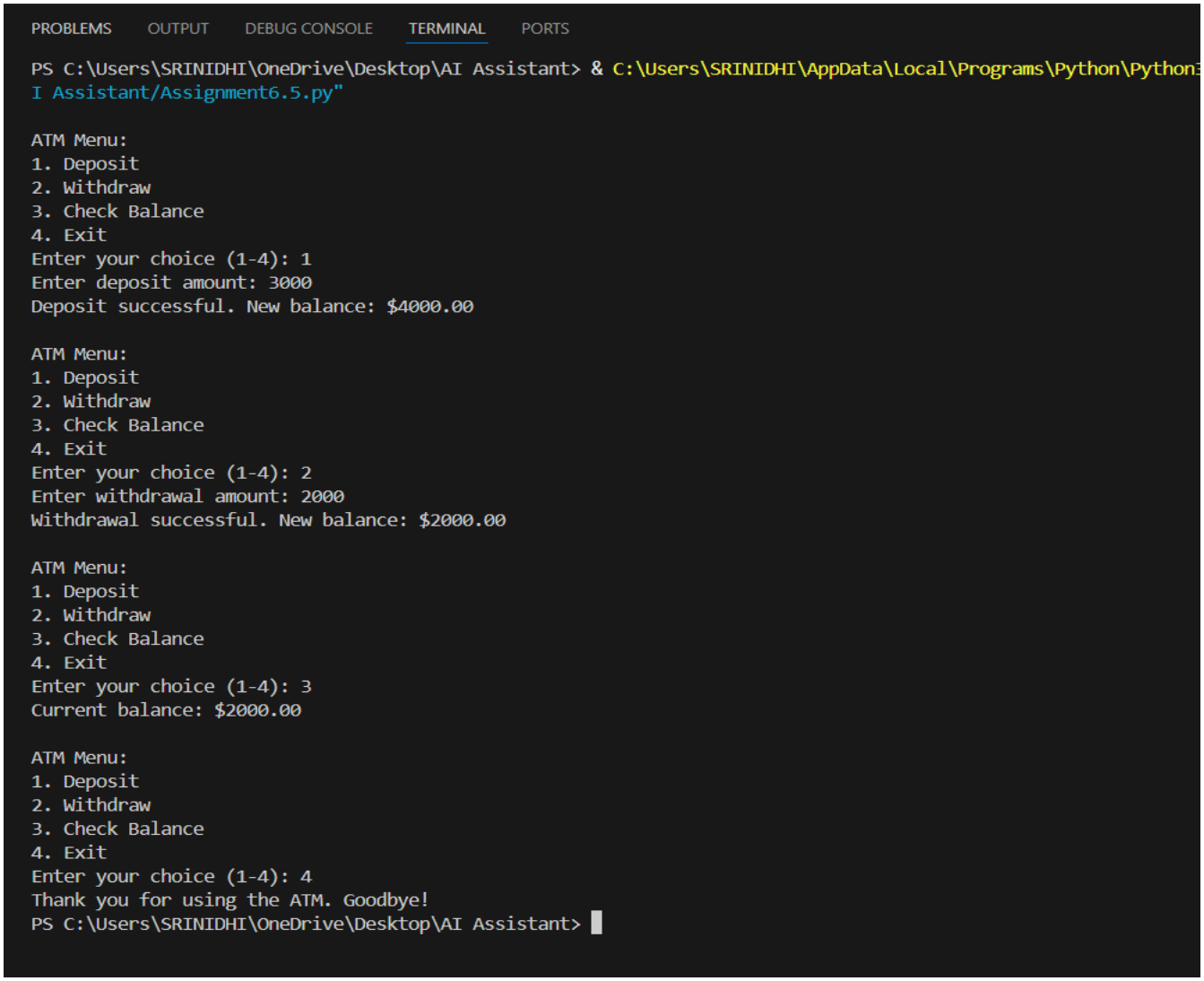
- The attendance system accurately stores and displays student attendance.
- Loops are used to iterate through attendance records efficiently.
- Conditional checks prevent invalid operations.
- Attendance percentage is calculated correctly.

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

```
account_balance = 1000

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

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

    if choice == '1':
        try:
            deposit_amount = float(input("Enter deposit amount: "))
            if deposit_amount > 0:
                account_balance += deposit_amount
                print(f"Deposit successful. New balance: ${account_balance:.2f}")
            else:
                print("Deposit amount must be positive.")
        except ValueError:
            print("Invalid input. Please enter a number.")

    elif choice == '2':
        try:
            withdraw_amount = float(input("Enter withdrawal amount: "))
            if withdraw_amount > 0:
                if account_balance >= withdraw_amount:
                    account_balance -= withdraw_amount
                    print(f"Withdrawal successful. New balance: ${account_balance:.2f}")
                else:
                    print("Insufficient balance.")
            else:
                print("Withdrawal amount must be positive.")
        except ValueError:
            print("Invalid input. Please enter a number.")

```



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.