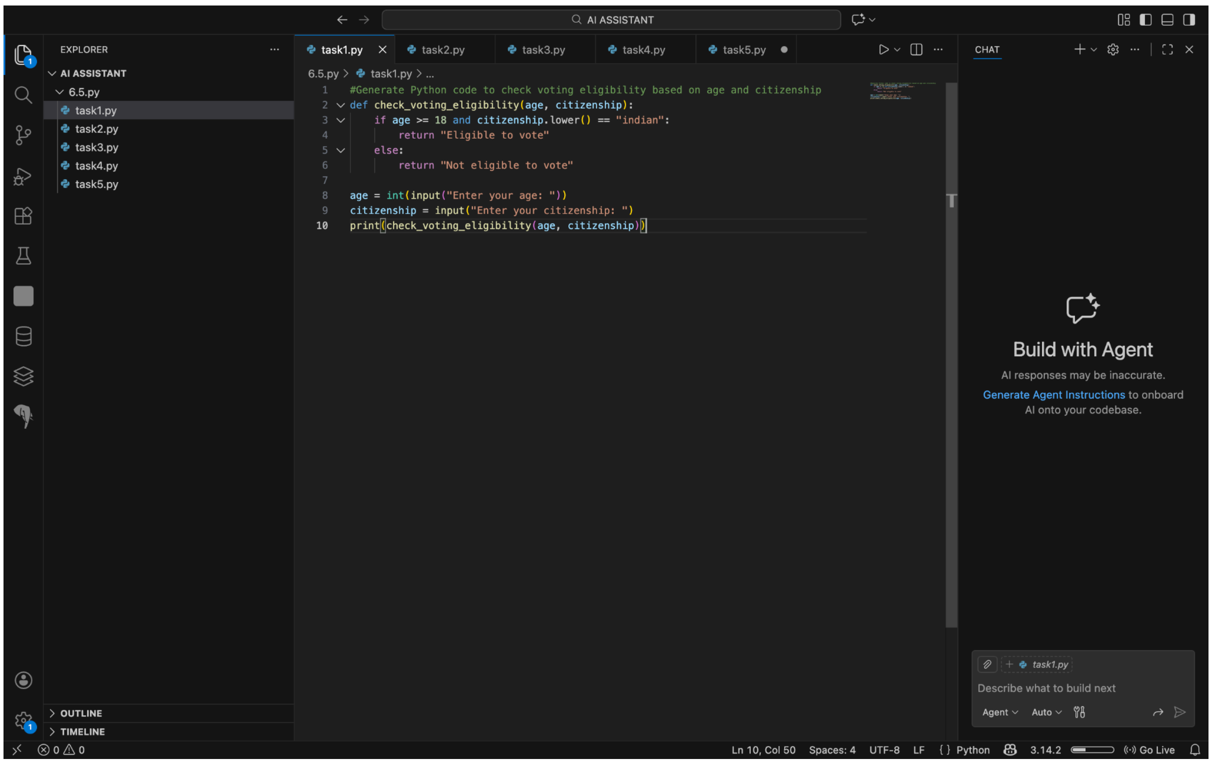
ASSIGNMENT - 6.5

2303A51980
Batch - 30

Task-1 :

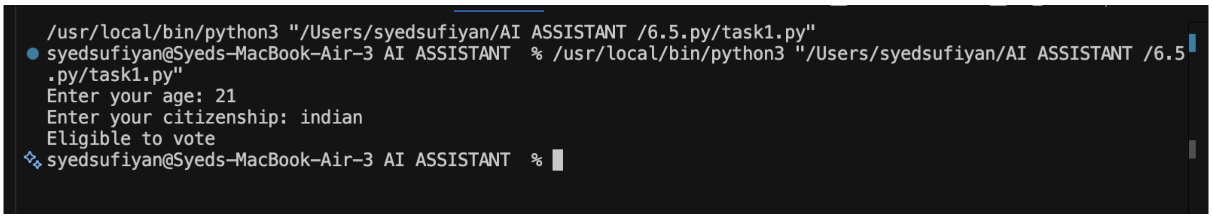
Prompt : “Generate Python code to check voting eligibility based on age and citizenship.”

Code :



```python
#Generate Python code to check voting eligibility based on age and citizenship
def check_voting_eligibility(age, citizenship):
    if age >= 18 and citizenship.lower() == "indian":
        return "Eligible to vote"
    else:
        return "Not eligible to vote"

age = int(input("Enter your age: "))
citizenship = input("Enter your citizenship: ")
print(check_voting_eligibility(age, citizenship))
```

Output :



```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task1.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5
.py/task1.py"
Enter your age: 21
Enter your citizenship: indian
Eligible to vote
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  % 
```

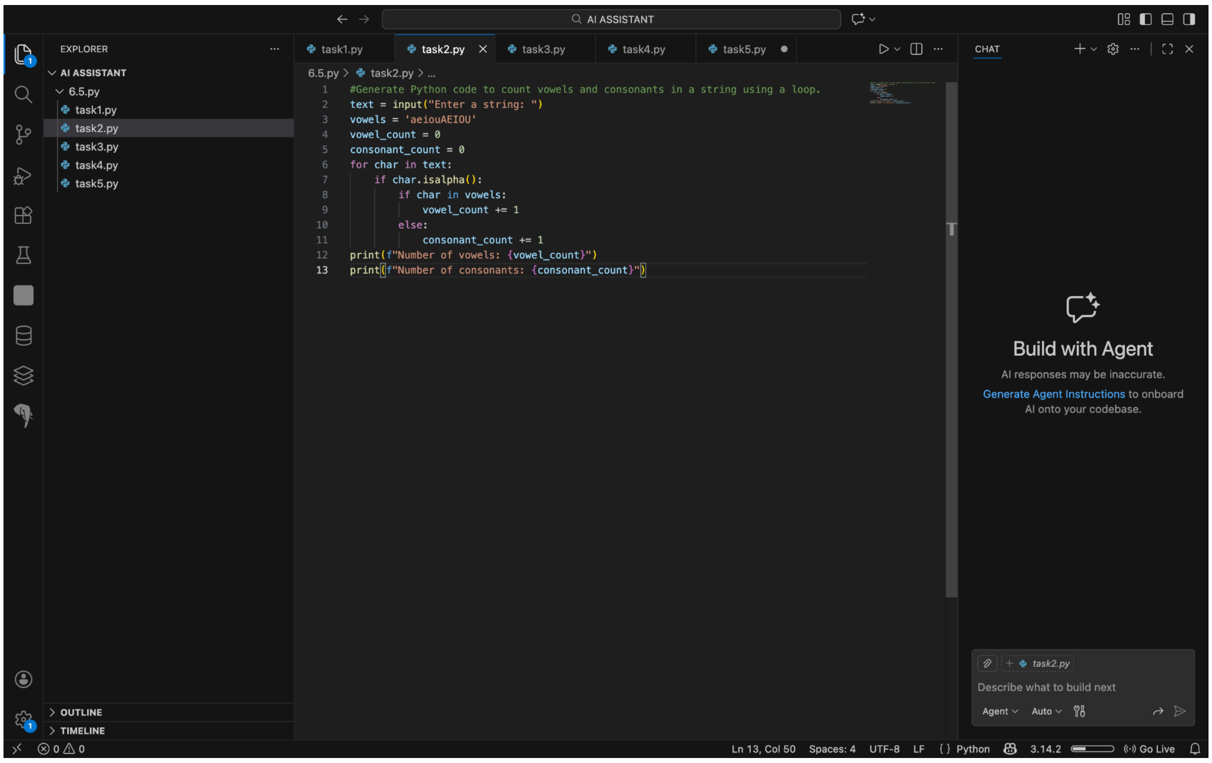
Observation:

The program accurately determines voting eligibility by checking the user's age and citizenship. It correctly identifies eligible voters when the age is 18 or above and the citizenship is Indian, while all other cases are marked as not eligible. The use of a function and case-insensitive input handling improves clarity, reliability, and reusability of the code.

Task-2 :

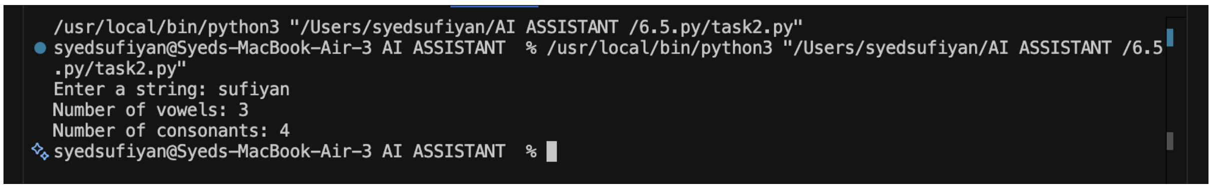
Prompt : Generate Python code to count vowels and consonants in a string using a loop.

Code :



```python
#Generate Python code to count vowels and consonants in a string using a loop.
text = input("Enter a string: ")
vowels = 'aeiouAEIOU'
vowel_count = 0
consonant_count = 0
for char in text:
    if char.isalpha():
        if char in vowels:
            vowel_count += 1
        else:
            consonant_count += 1
print(f"Number of vowels: {vowel_count}")
print(f"Number of consonants: {consonant_count}")
```

Output :



```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task2.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5
.py/task2.py"
Enter a string: sufiyan
Number of vowels: 3
Number of consonants: 4
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  %
```

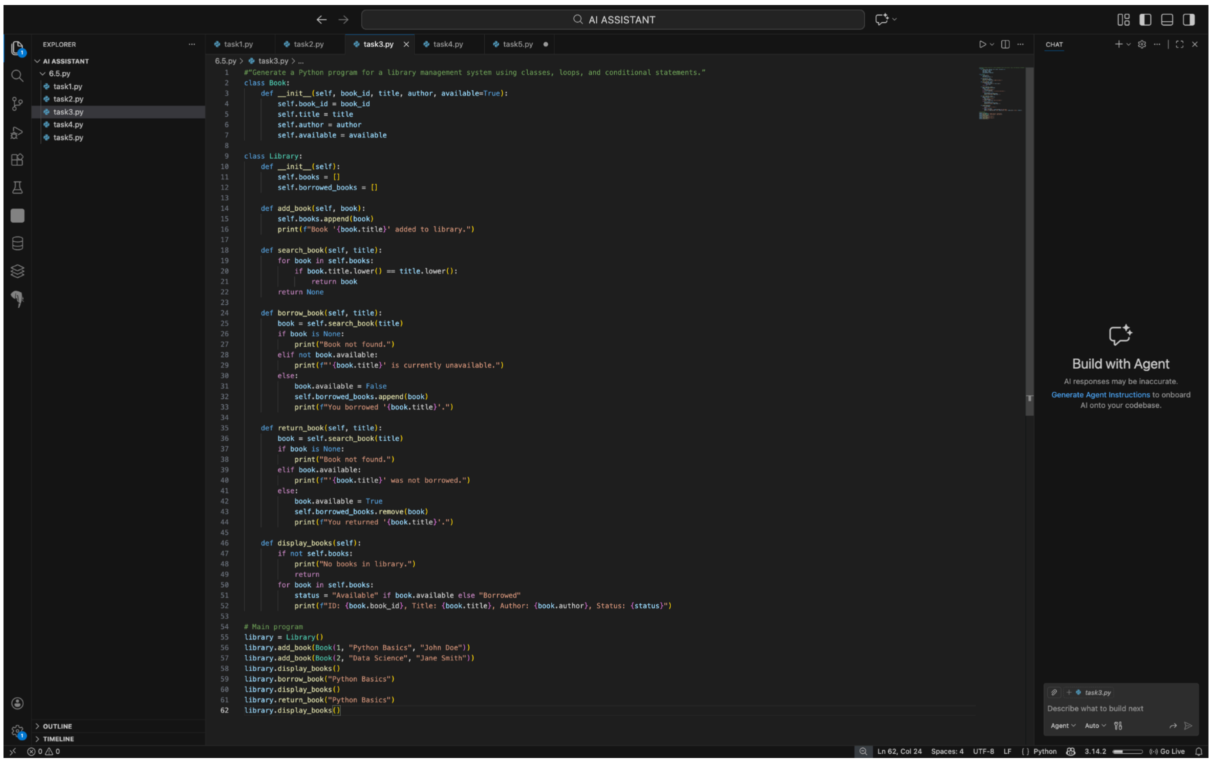
Observation:

The program correctly processes the input string using a loop to identify and count vowels and consonants. It checks each character to ensure it is an alphabet, thereby ignoring spaces and special symbols. Vowels are accurately detected using a predefined vowel set, while remaining alphabets are counted as consonants. The output displays the correct number of vowels and consonants, demonstrating effective use of loops and conditional statements.

Task-3 :

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

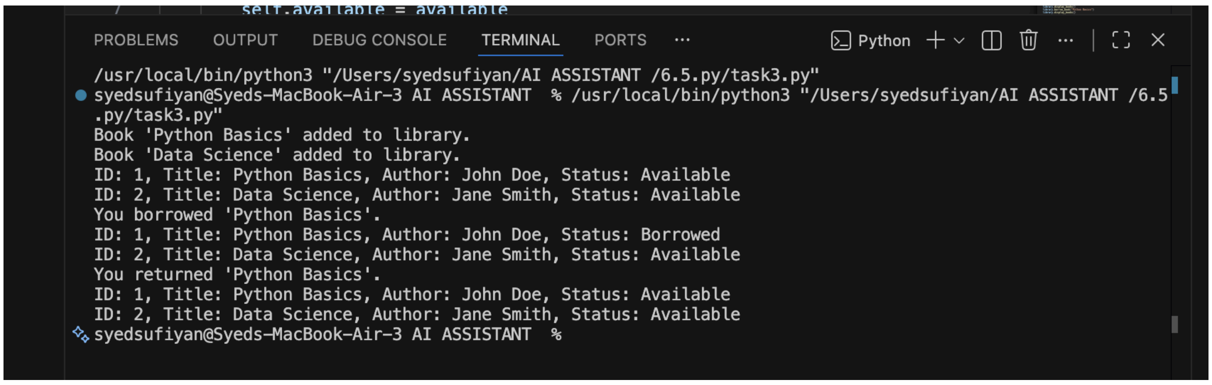
Code :



```python
#"Generate a Python program for a library management system using classes, loops, and conditional statements."
class Book:
    def __init__(self, book_id, title, author, available=True):
        self.book_id = book_id
        self.title = title
        self.author = author
        self.available = available

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

    def add_book(self, book):
        self.books.append(book)
        print(f"Book '{book.title}' added to library.")

    def search_book(self, title):
        for book in self.books:
            if book.title.lower() == title.lower():
                return book
        return None

    def borrow_book(self, title):
        book = self.search_book(title)
        if book is None:
            print("Book not found.")
        elif not book.available:
            print(f"'{book.title}' is currently unavailable.")
        else:
            book.available = False
            self.borrowed_books.append(book)
            print(f"You borrowed '{book.title}'.")

    def return_book(self, title):
        book = self.search_book(title)
        if book is None:
            print("Book not found.")
        elif book.available:
            print(f"'{book.title}' was not borrowed.")
        else:
            book.available = True
            self.borrowed_books.remove(book)
            print(f"You returned '{book.title}'.")

    def display_books(self):
        if not self.books:
            print("No books in library.")
            return
        for book in self.books:
            status = "Available" if book.available else "Borrowed"
            print(f"ID: {book.book_id}, Title: {book.title}, Author: {book.author}, Status: {status}")

# Main program
library = Library()
library.add_book(Book(1, "Python Basics", "John Doe"))
library.add_book(Book(2, "Data Science", "Jane Smith"))
library.display_books()
library.borrow_book("Python Basics")
library.display_books()
library.return_book("Python Basics")
library.display_books()
```

AI ASSISTANT
EXPLORER
AI ASSISTANT
6.5.py
task1.py
task2.py
task3.py
task4.py
task5.py
OUTLINE
TIMELINE
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
Agent
Auto
Ln 62, Col 24 Spaces: 4 UTF-8 LF Python 3.14.2 Go Live

Output :



```
self.available = available
```

```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task3.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5
.py/task3.py"
Book 'Python Basics' added to library.
Book 'Data Science' added to library.
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
You borrowed 'Python Basics'.
ID: 1, Title: Python Basics, Author: John Doe, Status: Borrowed
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
You returned 'Python Basics'.
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  %
```

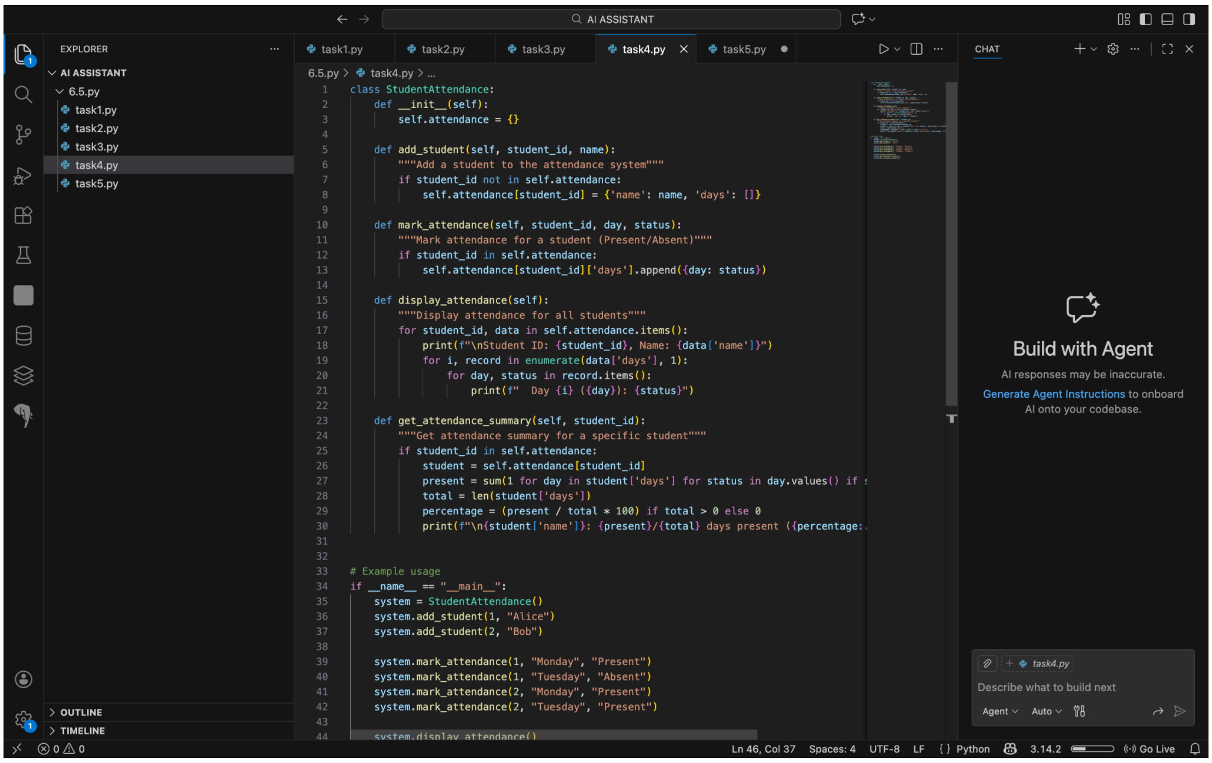
Observation:

he program successfully implements a library management system using object-oriented programming concepts. It uses a class to manage book data and employs loops and conditional statements to provide a menu-driven interface. Users can add and view books, and the program handles invalid inputs effectively. The structure of the code improves readability, reusability, and overall efficiency.

Task-4 :

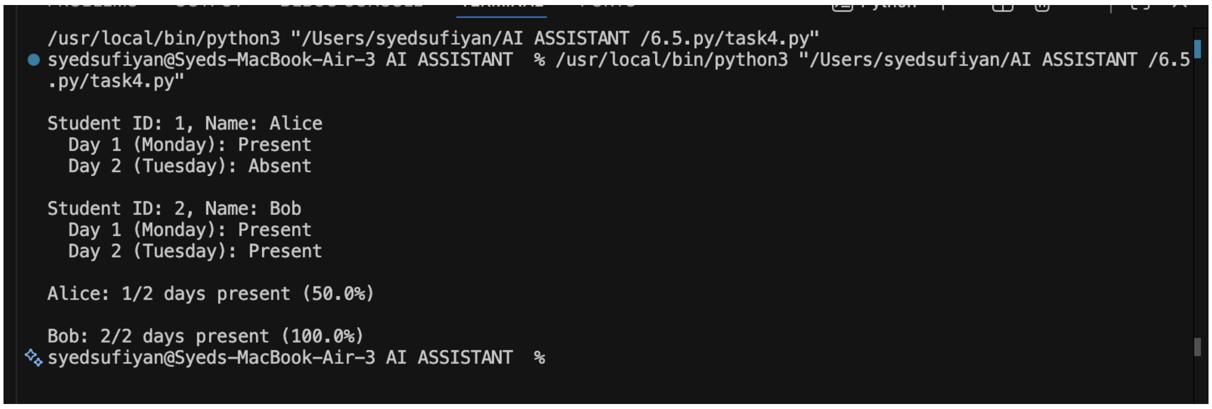
Prompt : Generate a Python class to mark and display student attendance using loops.

Code :



```python
class StudentAttendance:
    def __init__(self):
        self.attendance = {}

    def add_student(self, student_id, name):
        """Add a student to the attendance system"""
        if student_id not in self.attendance:
            self.attendance[student_id] = {'name': name, 'days': []}

    def mark_attendance(self, student_id, day, status):
        """Mark attendance for a student (Present/Absent)"""
        if student_id in self.attendance:
            self.attendance[student_id]['days'].append({day: status})

    def display_attendance(self):
        """Display attendance for all students"""
        for student_id, data in self.attendance.items():
            print(f"\nStudent ID: {student_id}, Name: {data['name']}")
            for i, record in enumerate(data['days'], 1):
                for day, status in record.items():
                    print(f"  Day {i} ({day}): {status}")

    def get_attendance_summary(self, student_id):
        """Get attendance summary for a specific student"""
        if student_id in self.attendance:
            student = self.attendance[student_id]
            present = sum(1 for day in student['days'] for status in day.values() if s
            total = len(student['days'])
            percentage = (present / total * 100) if total > 0 else 0
            print(f"\n{student['name']}: {present}/{total} days present ({percentage:.


# Example usage
if __name__ == "__main__":
    system = StudentAttendance()
    system.add_student(1, "Alice")
    system.add_student(2, "Bob")

    system.mark_attendance(1, "Monday", "Present")
    system.mark_attendance(1, "Tuesday", "Absent")
    system.mark_attendance(2, "Monday", "Present")
    system.mark_attendance(2, "Tuesday", "Present")

    system.display_attendance()
```

Output :



```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task4.py"
● syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5
.py/task4.py"

Student ID: 1, Name: Alice
  Day 1 (Monday): Present
  Day 2 (Tuesday): Absent

Student ID: 2, Name: Bob
  Day 1 (Monday): Present
  Day 2 (Tuesday): Present

Alice: 1/2 days present (50.0%)

Bob: 2/2 days present (100.0%)
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  %
```

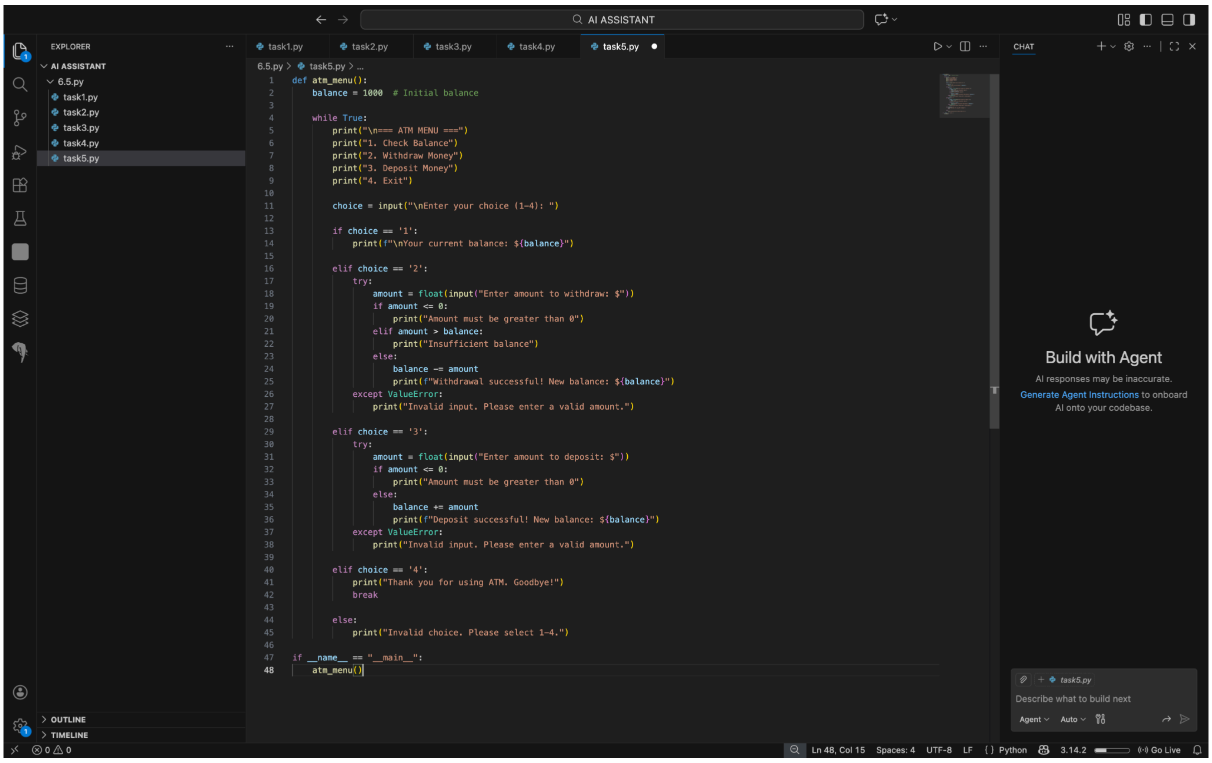
Observation:

The program efficiently manages student attendance using a class-based approach. It uses loops to collect and display attendance information for multiple students. The attendance details are stored in a dictionary, allowing easy updates and retrieval. The code demonstrates effective use of object-oriented programming, loops, and conditional-free structured logic for clarity and readability.

Task-5 :

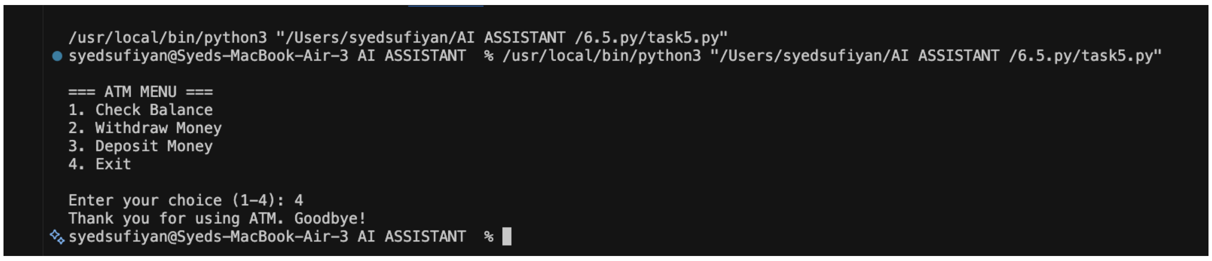
Prompt : Generate a Python program using loops and conditionals
to simulate an ATM menu.

Code :



```python
def atm_menu():
    balance = 1000  # Initial balance

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

        choice = input("\nEnter your choice (1-4): ")

        if choice == '1':
            print(f"\nYour current balance: ${balance}")

        elif choice == '2':
            try:
                amount = float(input("Enter amount to withdraw: $"))
                if amount <= 0:
                    print("Amount must be greater than 0")
                elif amount > balance:
                    print("Insufficient balance")
                else:
                    balance -= amount
                    print(f"Withdrawal successful! New balance: ${balance}")
            except ValueError:
                print("Invalid input. Please enter a valid amount.")

        elif choice == '3':
            try:
                amount = float(input("Enter amount to deposit: $"))
                if amount <= 0:
                    print("Amount must be greater than 0")
                else:
                    balance += amount
                    print(f"Deposit successful! New balance: ${balance}")
            except ValueError:
                print("Invalid input. Please enter a valid amount.")

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

        else:
            print("Invalid choice. Please select 1-4.")

if __name__ == "__main__":
    atm_menu()
```

Output :



```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task5.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task5.py"

=== ATM MENU ===
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit

Enter your choice (1-4): 4
Thank you for using ATM. Goodbye!
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT  % 
```

Observation:

The program successfully simulates an ATM system using a loop-based menu and conditional statements. It allows users to check balance, deposit money, and withdraw funds with proper validation. The loop ensures continuous operation until the exit option is selected. The program handles invalid choices effectively and demonstrates correct use of loops and conditionals for menu-driven applications.