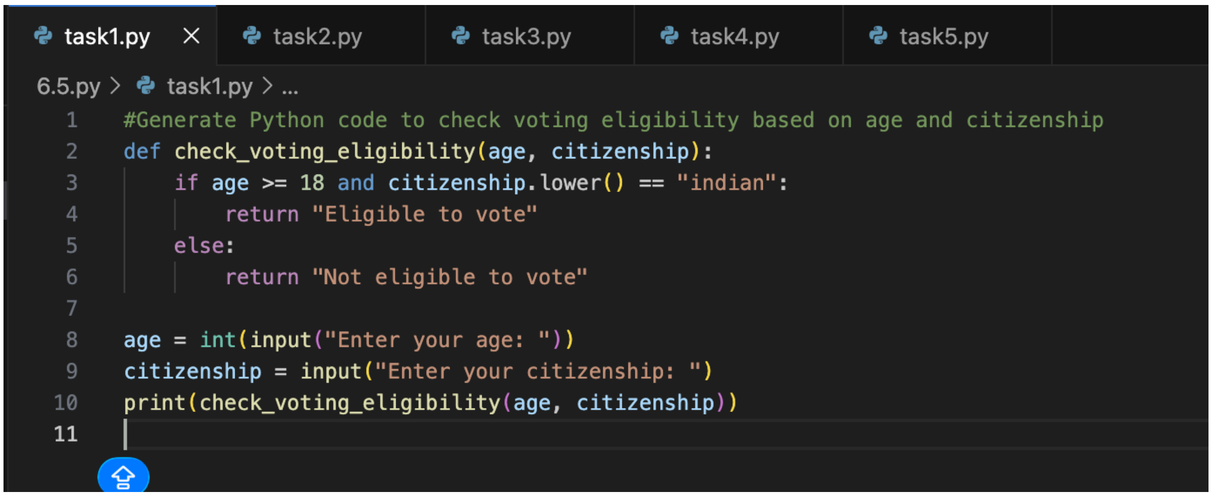
LAB ASSIGNMENT 6.5

NAME : B.GNANESHWAR
HT NO : 2303A51T01

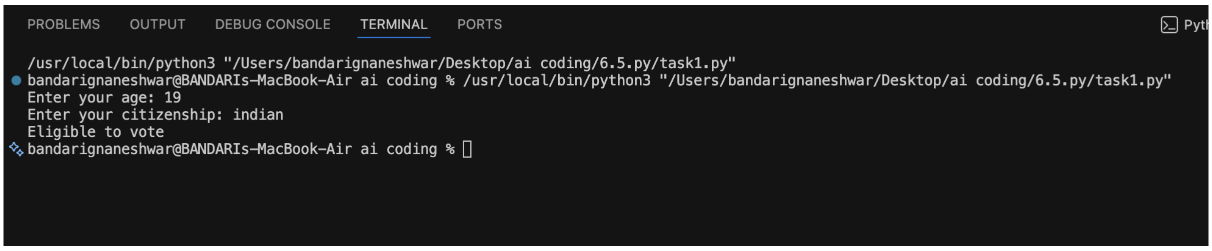
Task1: Use an AI tool to generate eligibility logic.
Prompt:
“Generate Python code to check voting eligibility based on age and citizenship.”
Expected Output:
• AI-generated conditional logic.
• Correct eligibility decisions.
• Explanation of conditions.

CODE :



```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/bandarignaneshwar/Desktop/ai coding/6.5.py/task1.py"
bandarignaneshwar@BANDARIs-MacBook-Air ai coding % /usr/local/bin/python3 "/Users/bandarignaneshwar/Desktop/ai coding/6.5.py/task1.py"
Enter your age: 19
Enter your citizenship: indian
Eligible to vote
bandarignaneshwar@BANDARIs-MacBook-Air ai coding % 
```

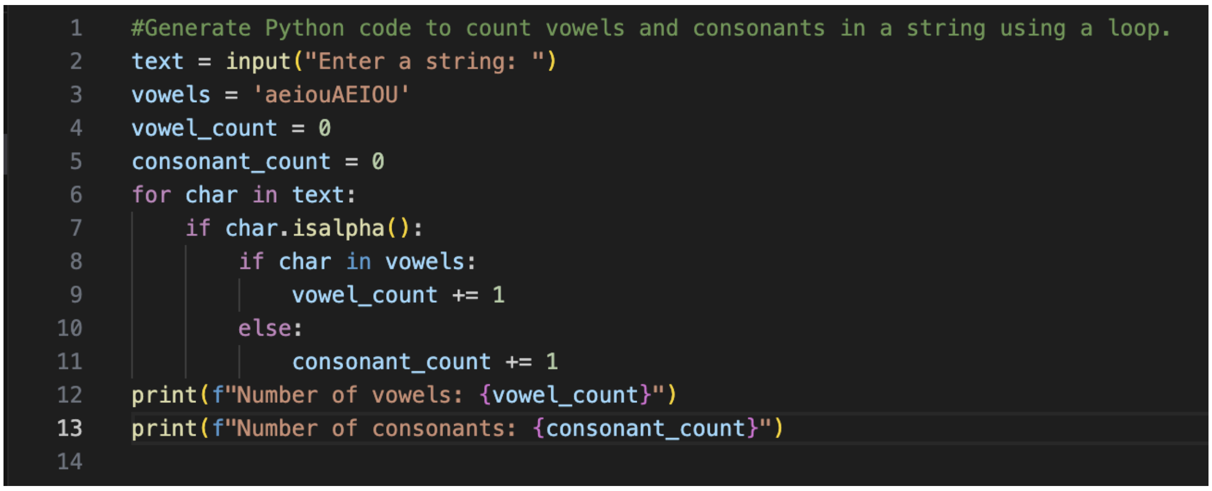
OBSERVATION :

The code correctly verifies voting eligibility by checking both age and citizenship.
It uses conditional logic to ensure the user is at least 18 years old and an

Indian citizen.
Case-insensitive input handling improves accuracy and reliability.

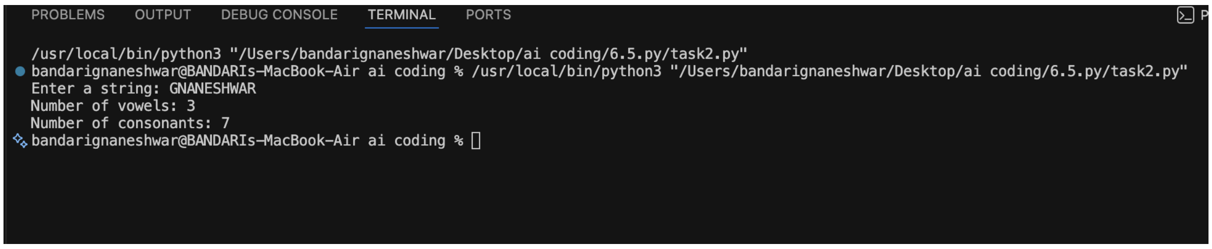
TASK 2 :
Task: Use an AI tool to process strings using loops.
Prompt:
“Generate Python code to count vowels and consonants in a string using a loop.”
Expected Output:
• AI-generated string processing logic.
• Correct counts.
• Output verification.

CODE :



```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/bandarignaneshwar/Desktop/ai coding/6.5.py/task2.py"
bandarignaneshwar@BANDARIs-MacBook-Air ai coding % /usr/local/bin/python3 "/Users/bandarignaneshwar/Desktop/ai coding/6.5.py/task2.py"
Enter a string: GNANESHWAR
Number of vowels: 3
Number of consonants: 7
bandarignaneshwar@BANDARIs-MacBook-Air ai coding %
```

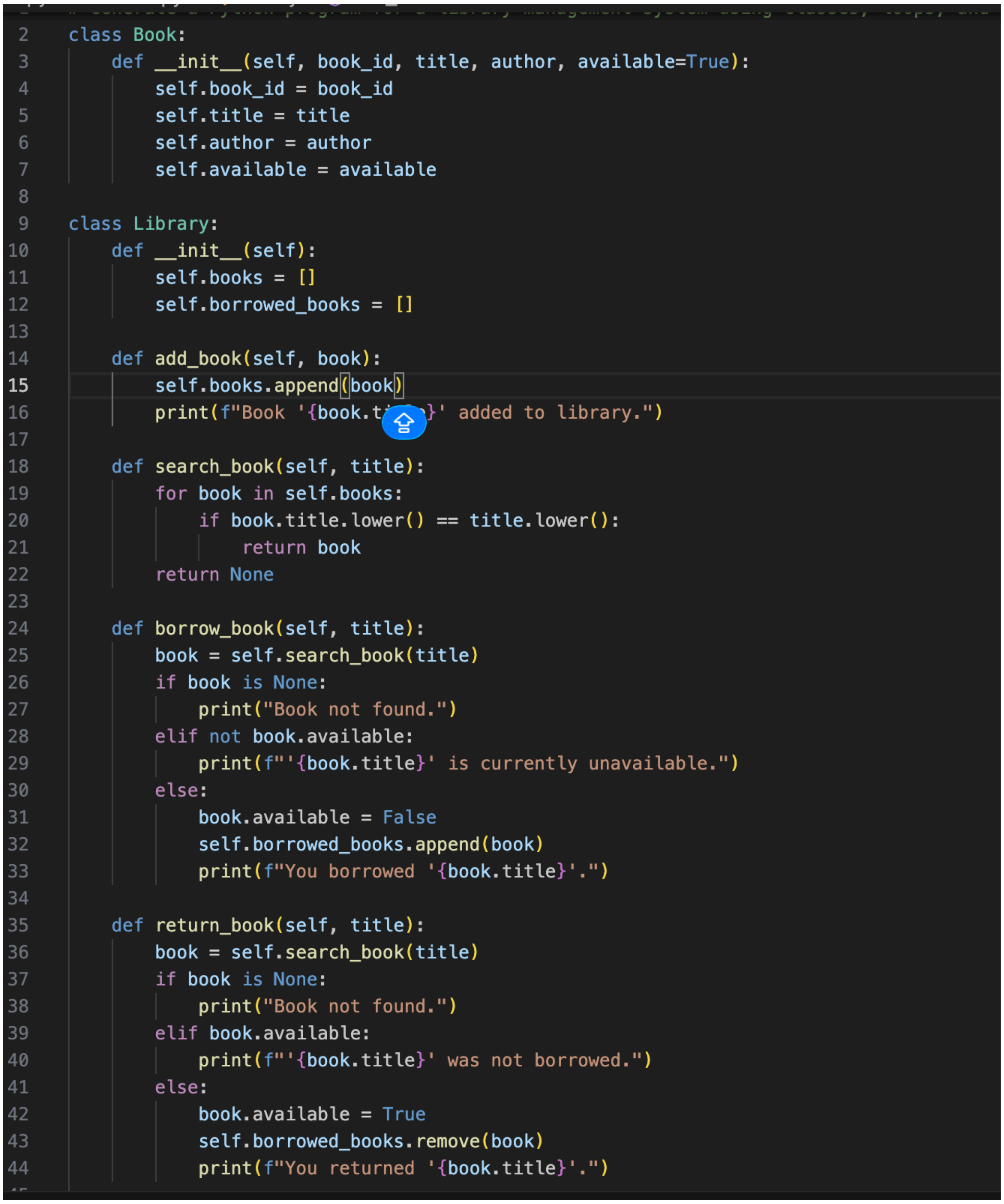
OBSERVATION :

The code efficiently uses a loop to iterate through each character in the string.
It correctly distinguishes vowels from consonants while ignoring non-alphabetic characters.
This approach ensures accurate counting and demonstrates clear use of conditional logic.

TASK 3 :

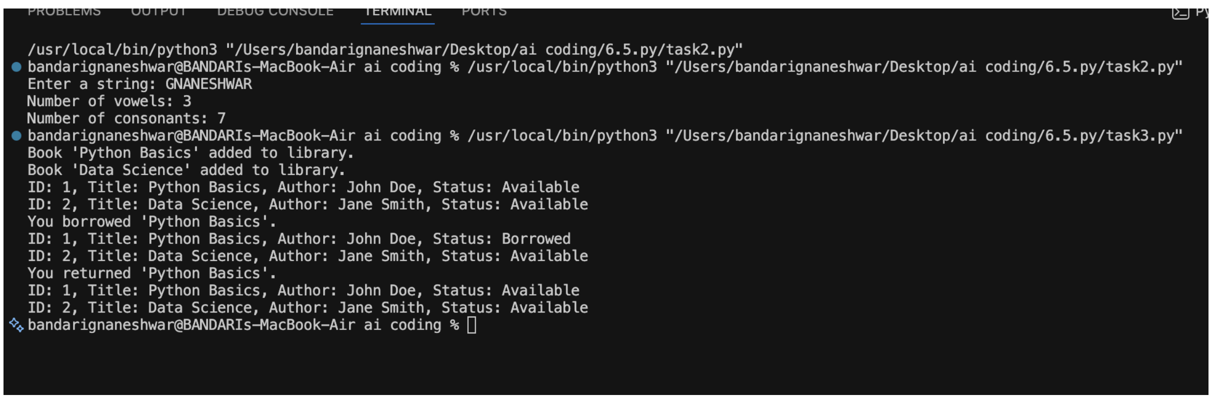
Task: Use an AI tool to generate a complete program using classes, loops, and conditionals.
Prompt:
“Generate a Python program for a library management system using classes, loops, and conditional statements.”
Expected Output:
• Complete AI-generated program.
• Review of AI suggestions quality.
• Short reflection on AI-assisted coding experience.

CODE :



```python
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.ti[illegible]}' 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}'.")
```

OUTPUT :


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

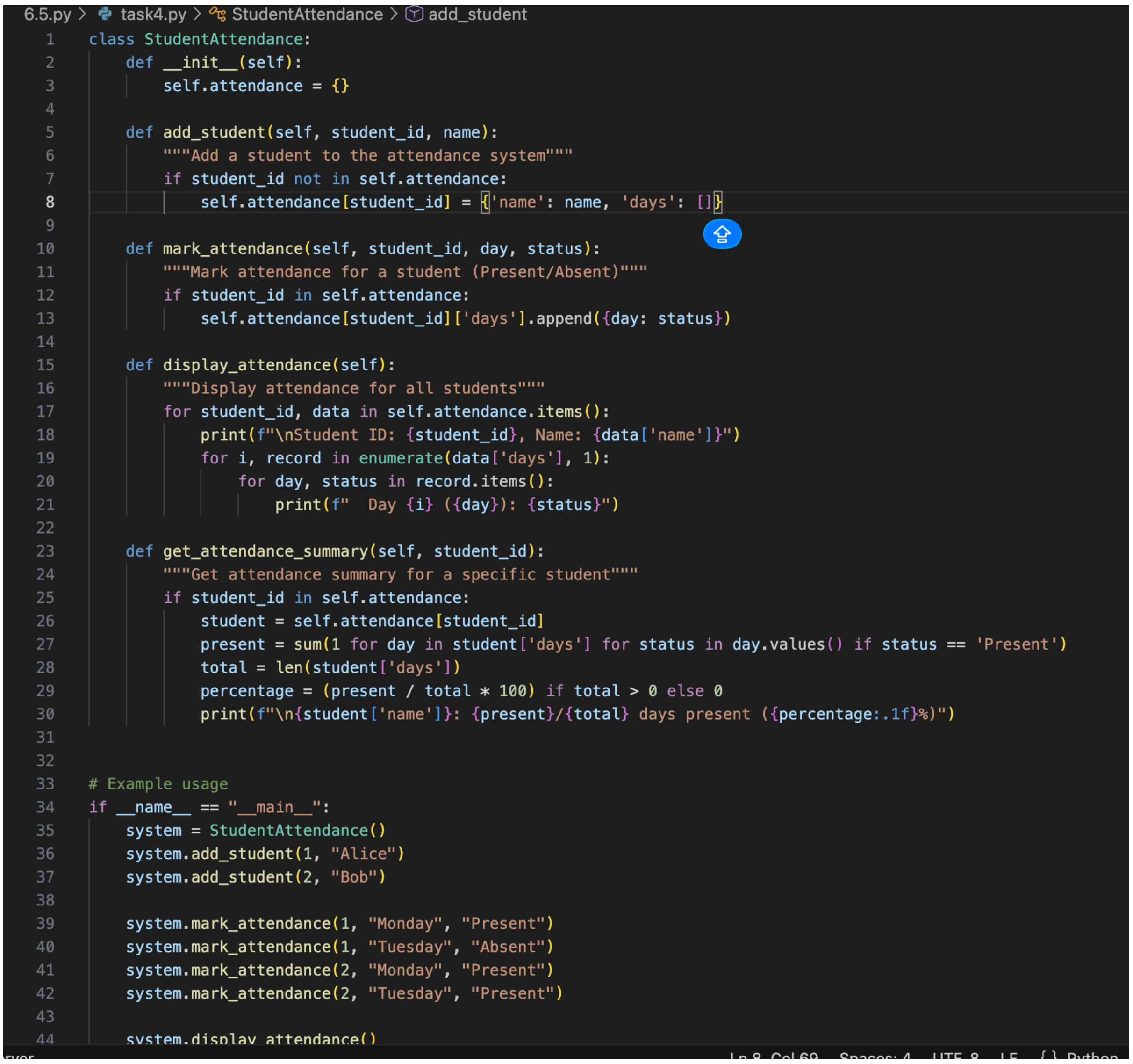
```
/usr/local/bin/python3 "/Users/bandarignaneshwar/Desktop/ai coding/6.5.py/task2.py"
bandarignaneshwar@BANDARIs-MacBook-Air ai coding % /usr/local/bin/python3 "/Users/bandarignaneshwar/Desktop/ai coding/6.5.py/task2.py"
Enter a string: GNANESHWAR
Number of vowels: 3
Number of consonants: 7
bandarignaneshwar@BANDARIs-MacBook-Air ai coding % /usr/local/bin/python3 "/Users/bandarignaneshwar/Desktop/ai coding/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
bandarignaneshwar@BANDARIs-MacBook-Air ai coding %
```

OBSERVATION : The code demonstrates effective use of classes to model a library management system.
It correctly handles book addition, searching, borrowing, and returning with proper status updates.
Overall, the logic is clear, modular, and suitable for understanding object-oriented programming concepts.

TASK 4 :

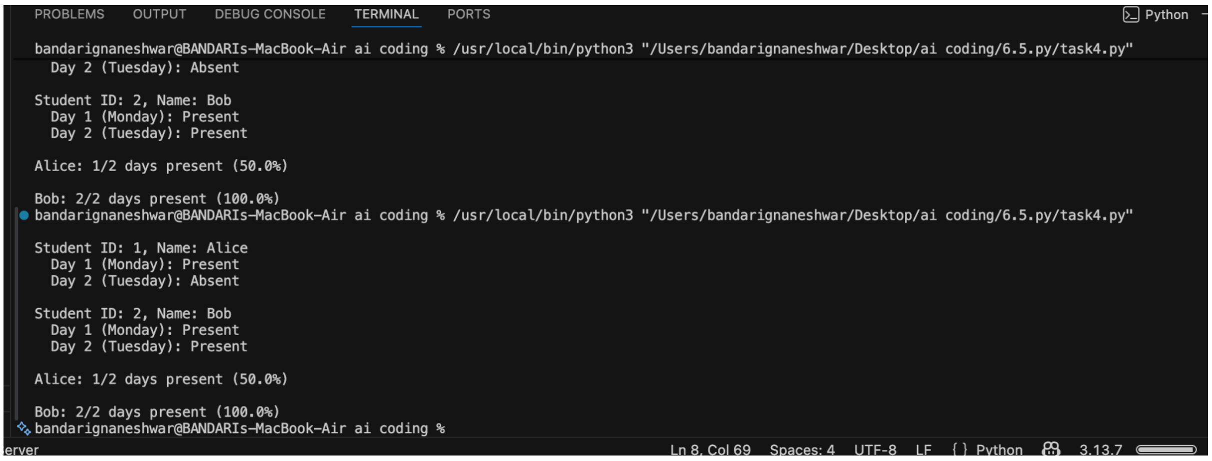
Task: Use an AI tool to generate an attendance management class.
Prompt: “Generate a Python class to mark and display student attendance using loops.”
Expected Output:
• AI-generated attendance logic.
• Correct display of attendance.
• Test cases.

CODE :



```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 status == 'Present')
            total = len(student['days'])
            percentage = (present / total * 100) if total > 0 else 0
            print(f"\n{student['name']}: {present}/{total} days present ({percentage:.1f}%)")


# 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()
```

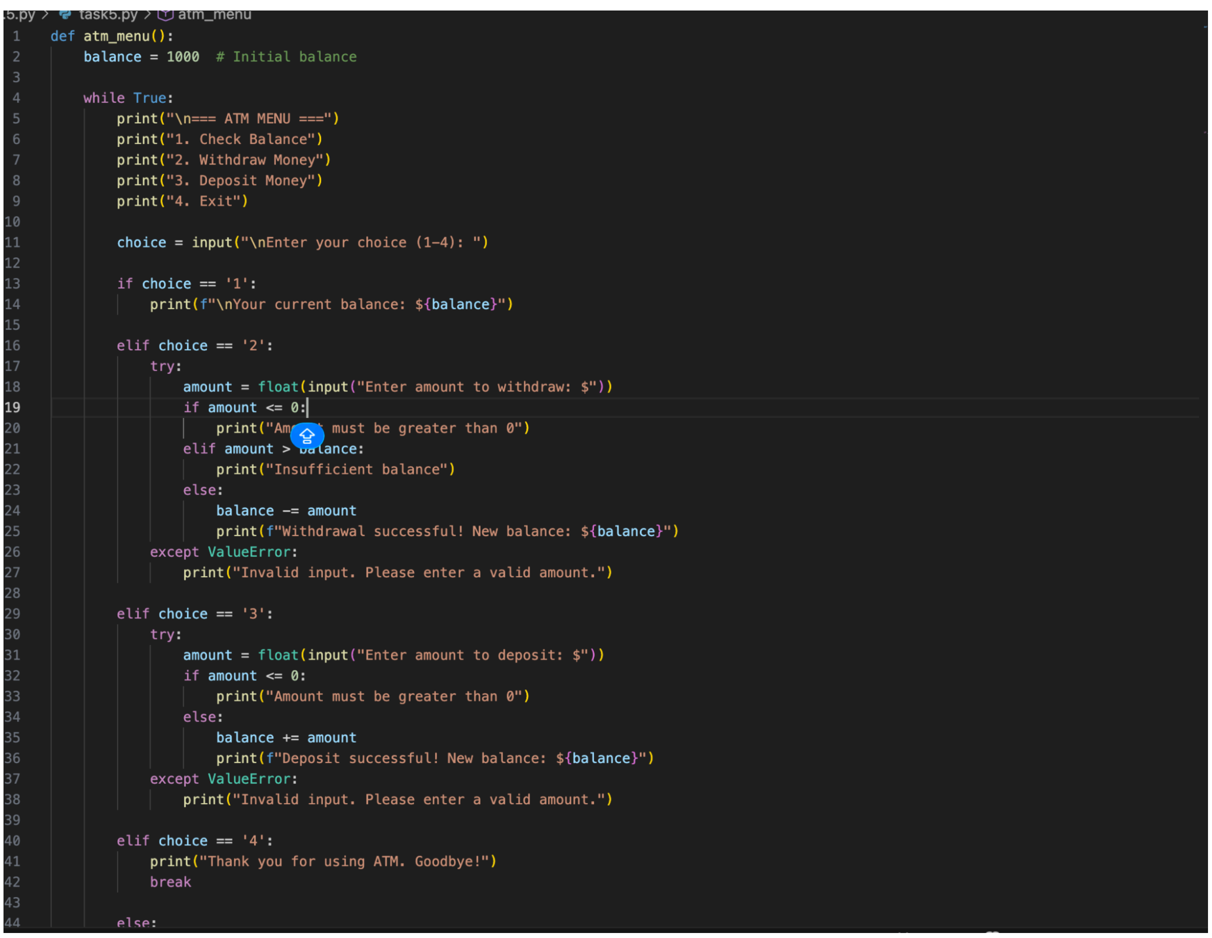
OUTPUT :


PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   Python

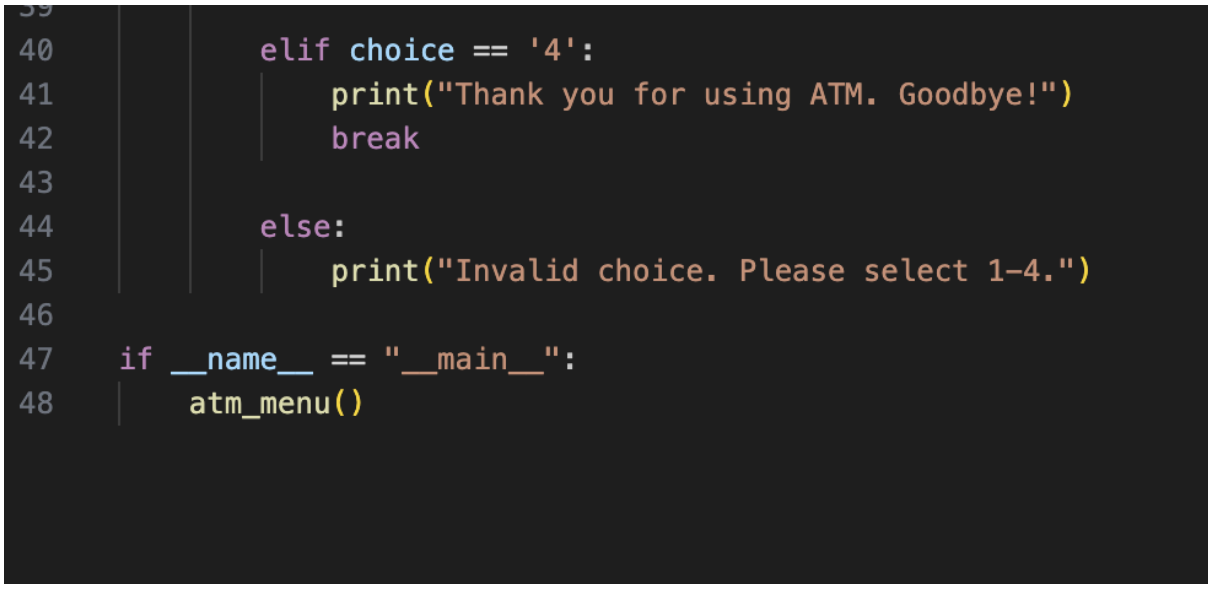
```
bandarignaneshwar@BANDARIs-MacBook-Air ai coding % /usr/local/bin/python3 "/Users/bandarignaneshwar/Desktop/ai coding/6.5.py/task4.py"
  Day 2 (Tuesday): Absent

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%)
bandarignaneshwar@BANDARIs-MacBook-Air ai coding % /usr/local/bin/python3 "/Users/bandarignaneshwar/Desktop/ai coding/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%)
bandarignaneshwar@BANDARIs-MacBook-Air ai coding %
```

OBSERVATION :
The code effectively uses object-oriented programming to manage student attendance records.
It stores daily attendance in a structured way and supports detailed display and summary calculations.
Overall, the implementation is clear, modular, and demonstrates good use of loops and data structures.

TASK 5 :
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.

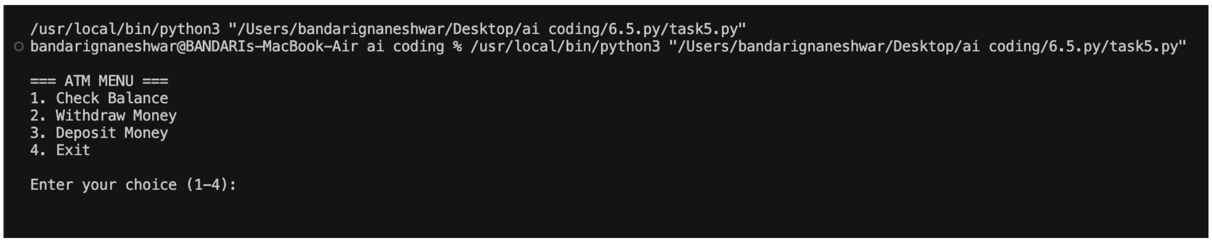


```python
def atm_menu():
    balance = 1000  # Initial balance

    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:
```



```python
        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/bandarignaneshwar/Desktop/ai coding/6.5.py/task5.py"
bandarignaneshwar@BANDARIs-MacBook-Air ai coding % /usr/local/bin/python3 "/Users/bandarignaneshwar/Desktop/ai coding/6.5.py/task5.py"

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

Enter your choice (1-4):
```

OBSERVATION :

The code simulates an ATM system using a loop and menu-driven approach.
It correctly handles balance checking, deposits, withdrawals, and input validation.
Overall, it demonstrates clear control flow, user interaction, and basic error handling.