ASSIGNMENT-6.5

NAME:PRANEETH BANDA
BATCH:20
HTNO:2303A51711

TASK1- AI-Generated Python Code (Eligibility Logic)
Prompt :Generate an python code for voting

```python
def check_voting_eligibility(age, is_citizen):
    """
    Check if a person is eligible to vote.

    Args:
        age (int): The person's age
        is_citizen (bool): Whether the person is a citizen

    Returns:
        bool: True if eligible to vote, False otherwise
    """
    if age >= 18 and is_citizen:
        return True
    return False


# Example usage
if __name__ == "__main__":
    age = int(input("Enter your age: "))
    is_citizen = input("Are you a citizen? (yes/no): ").lower() == "yes"

    if check_voting_eligibility(age, is_citizen):
        print("You are eligible to vote!")
    else:
        print("You are not eligible to vote.")
```

```
PS C:\Users\Banda Praneeth\OneDrive\Desktop\AI ASSIST> & "C:/Program Files/Python313/python.exe" "c:/Users/Banda Praneeth
assignment.py"
Enter your age: 19
Are you a citizen? (yes/no): yes
You are eligible to vote!
PS C:\Users\Banda Praneeth\OneDrive\Desktop\AI ASSIST>
```

Conditional Logic Explanation

The eligibility is checked using an if condition with two requirements:

Condition 1: Age Requirement

age >= 18

- A person must be at least 18 years old to be eligible to vote.

Condition 2: Citizenship Requirement

citizenship.lower() == "yes"

- The person must be a citizen.
- .lower() is used to handle inputs like "YES", "yes", "Yes".

Combined Condition (Logical AND)

if age >= 18 and citizenship.lower() == "yes":

- The and operator ensures that both conditions must be true.
- If either condition fails → Not eligible.

3. Correct Eligibility Decisions (Examples)

Age	Citizenship	Result
29	Yes	Eligible to vote
17	no	Not eligible
24	No	Not eligible
13	No	Not eligible

How the Program Works (Step-by-Step)

1. Takes user input for:
    - Age
    - Citizenship status
2. Sends values to check_voting_eligibility() function.
3. Applies conditional logic.
4. Returns result.

5. Prints eligibility status.

TASK-2-Vowels and Consonants

Prompt-“Generate Python code to count vowels and consonants in a string using a loop.”

```python
#TASK_2
def count_vowels_and_consonants(text):
    """
    Count vowels and consonants in a string.

    Args:
        text (str): The input string

    Returns:
        tuple: (vowel_count, consonant_count)
    """
    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

    return vowel_count, consonant_count


# Example usage
if __name__ == "__main__":
    text = input("Enter a string: ")
    vowels, consonants = count_vowels_and_consonants(text)
    print(f"Vowels: {vowels}")
    print(f"Consonants: {consonants}")
```

```
PS C:\Users\Banda Praneeth\OneDrive\Desktop\AI ASSIST> & "C:/Program Files/Python313/python.exe" "c:/Users/Banda Praneeth/On
assignment.py"
Enter a string: praneeth
Vowels: 3
Consonants: 5
PS C:\Users\Banda Praneeth\OneDrive\Desktop\AI ASSIST>
```

The program uses a for loop to scan each character in the string.

Step 1: Define Vowels

vowels = "aeiouAEIOU"

- Contains both lowercase and uppercase vowels.

Step 2: Initialize Counters

vowel_count = 0

consonant_count = 0

- Stores total vowels and consonants.

Step 3: Loop Through String

for char in text:

- Iterates over each character.

Step 4: Check Alphabetic Characters

if char.isalpha():

- Ignores digits, spaces, and symbols.

Step 5: Identify Vowel or Consonant

```
if char in vowels:
    vowel_count += 1
else:
    consonant_count += 1
```

- Updates respective counters.

TASK-3 Use an AI tool to generate a complete program using classes, loops, and conditionals.

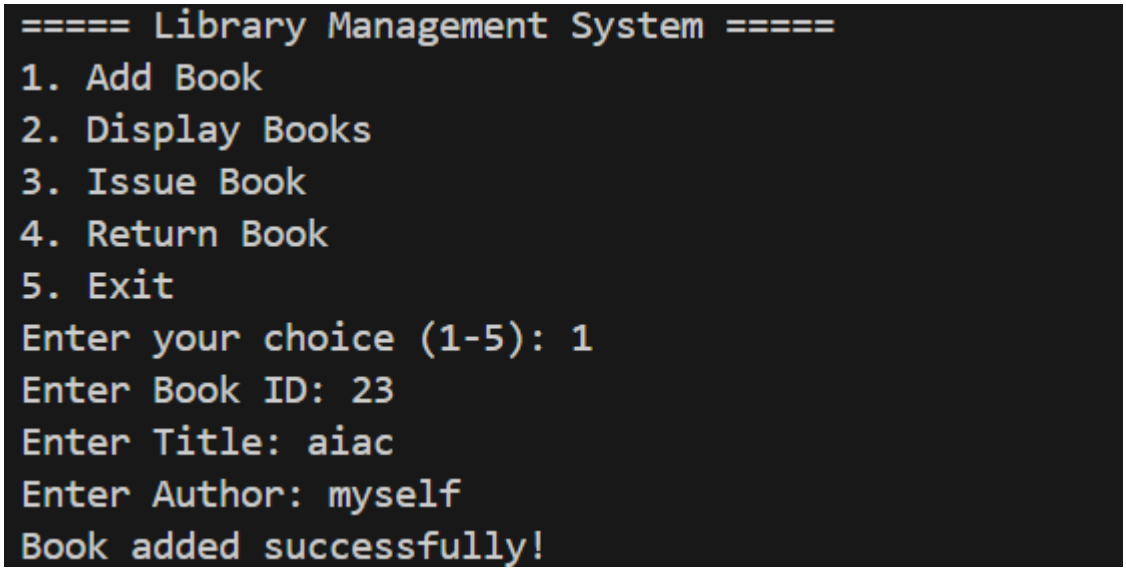
Prompt:

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

```python
class Book:
    def __init__(self, book_id, title, author):
        self.book_id = book_id
        self.title = title
        self.author = author
        self.is_issued = False


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

    def add_book(self, book_id, title, author):
        book = Book(book_id, title, author)
        self.books.append(book)
        print("Book added successfully!")

    def display_books(self):
        if not self.books:
            print("No books available in the library.")
            return

        print("\nAvailable Books:")
        print("-" * 40)

        for book in self.books:
            status = "Issued" if book.is_issued else "Available"
            print(f"ID: {book.book_id}, Title: {book.title}, "
                  f"Author: {book.author}, Status: {status}")
```

```
    def issue_book(self, book_id):
        for book in self.books:
            if book.book_id == book_id:
                if not book.is_issued:
                    book.is_issued = True
                    print("Book issued successfully!")
                else:
                    print("Book is already issued.")
                return

        print("Book not found.")

    def return_book(self, book_id):
        for book in self.books:
            if book.book_id == book_id:
                if book.is_issued:
                    book.is_issued = False
                    print("Book returned successfully!")
                else:
                    print("This book was not issued.")
                return

        print("Book not found.")


f main():
    library = Library()

    while True:
        print("\n===== Library Management System =====")
        print("1. Add Book")
        print("2. Display Books")
```

```
        choice = input("Enter your choice (1-5): ")

        if choice == "1":
            book_id = input("Enter Book ID: ")
            title = input("Enter Title: ")
            author = input("Enter Author: ")
            library.add_book(book_id, title, author)

        elif choice == "2":
            library.display_books()

        elif choice == "3":
            book_id = input("Enter Book ID to Issue: ")
            library.issue_book(book_id)

        elif choice == "4":
            book_id = input("Enter Book ID to Return: ")
            library.return_book(book_id)

        elif choice == "5":
            print("Exiting Library System. Goodbye!")
            break

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


f __name__ == "__main__":
    main()
```



Review of AI Suggestions Quality

The AI-generated program demonstrates good quality in the following ways:

a) Proper Use of Object-Oriented Programming

- Uses two classes:
  - Book → Represents individual books
  - Library → Manages collection of books
- Shows clear separation of responsibilities.

b) Effective Use of Loops

- Uses while True loop for menu-driven execution.
- Uses for loops to search books.

c) Correct Use of Conditionals

- if-elif-else for menu handling.
- Conditional checks for:
  - Book availability
  - Valid user input
  - Issued/returned status

d) Readability and Structure

- Clear function names.

- Meaningful variable names.
- Well-organized program flow.
- Easy to understand for beginners.

e) Limitations

Some minor limitations include:

- Data is not stored permanently (no file/database storage).
- No advanced validation (duplicate IDs allowed).
- No user authentication system.
- No error handling for wrong data types.

Overall, the AI suggestions are **logically correct, structured, and suitable for academic purposes and basic applications**.

TASK-4 AI-Assisted Code Completion for Class-Based Attendance System
Prompt: “Generate a Python class to mark and display studentattendance using loops.”

```python
class Attendance:
    def __init__(self):
        self.students = {}

    def add_student(self, name):
        self.students[name] = "Absent"

    def mark_attendance(self):
        for name in self.students:
            status = input(f"{name} (P/A): ").upper()

            if status == "P":
                self.students[name] = "Present"
            else:
                self.students[name] = "Absent"

    def display_attendance(self):
        print("\nAttendance Report")
        for name, status in self.students.items():
            print(name, ":", status)


# Test Program
attendance = Attendance()

attendance.add_student("Rahul")
attendance.add_student("Ananya")

attendance.mark_attendance()
attendance.display_attendance()
```

```
Rahul (P/A): A
Ananya (P/A): P

Attendance Report
Rahul : Absent
Ananya : Present
```

Logic

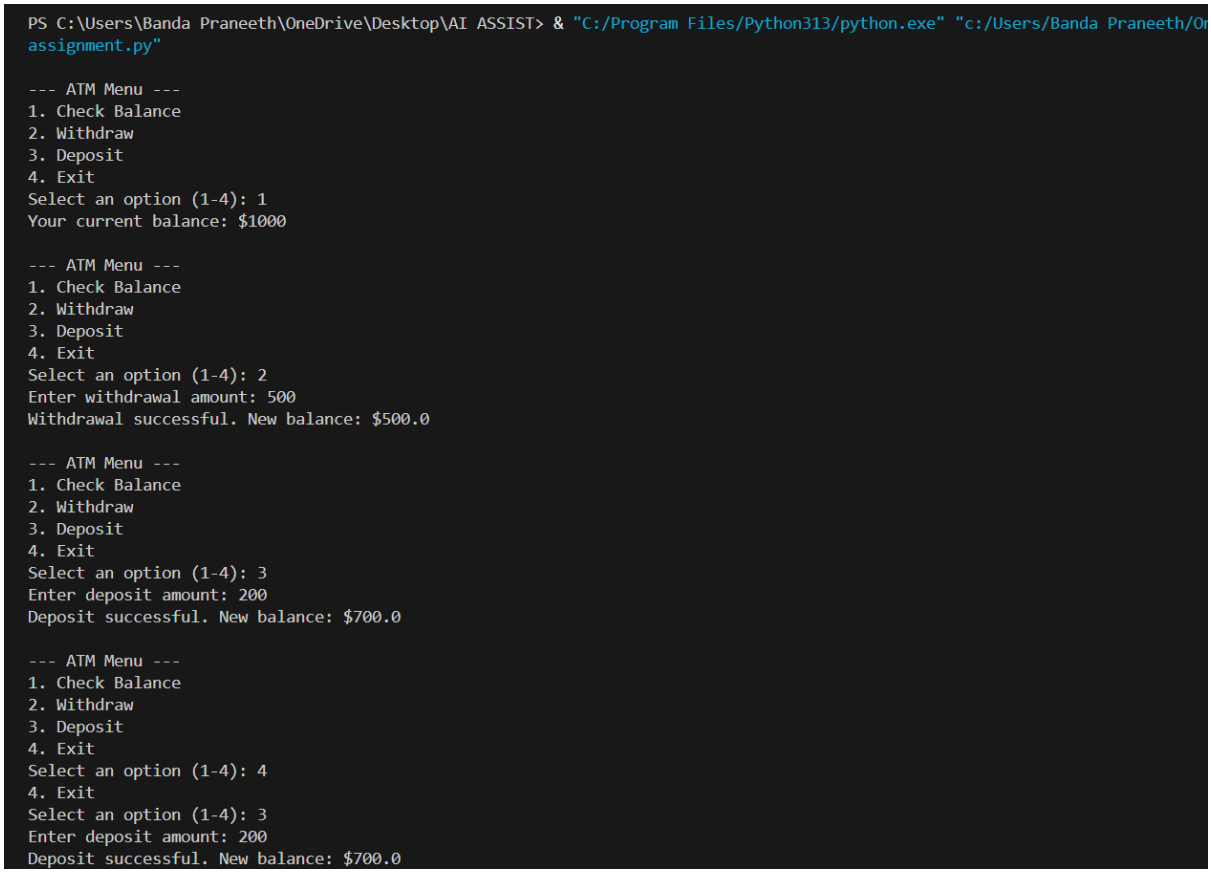
- Uses a **dictionary** to store names and attendance.
- for loop is used to mark attendance.
- if-else checks Present or Absent.
- Another loop displays records.

Task 5- AI-Based Code Completion for Conditional Menu Navigation

PROMPT: “Generate a Python program using loops and conditionalsto simulate an ATM menu.”

```python
#task-5

class ATM:

    def __init__(self, balance=1000):
        self.balance = balance

    def check_balance(self):
        print(f"Your current balance: ${self.balance}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds.")
        elif amount <= 0:
            print("Invalid amount.")
        else:
            self.balance -= amount
            print(f"Withdrawal successful. New balance: ${self.balance}")

    def deposit(self, amount):
        if amount <= 0:
            print("Invalid amount.")
        else:
            self.balance += amount
            print(f"Deposit successful. New balance: ${self.balance}")

    def run(self):
        while True:
            print("\n--- ATM Menu ---")
            print("1. Check Balance")
            print("2. Withdraw")
            print("3. Deposit")
            print("4. Exit")
```

```python
class ATM:
    def run(self):
            choice = input("Select an option (1-4): ")

            if choice == "1":
                self.check_balance()
            elif choice == "2":
                amount = float(input("Enter withdrawal amount: "))
                self.withdraw(amount)
            elif choice == "3":
                amount = float(input("Enter deposit amount: "))
                self.deposit(amount)
            elif choice == "4":
                print("Thank you for using ATM. Goodbye!")
                break
            else:
                print("Invalid option. Please try again.")


if __name__ == "__main__":
    atm = ATM()
    atm.run()
```



Menu Logic

- while True → Runs menu continuously.
- if-elif-else → Handles user choices.
- Balance is updated using conditions.
- Prevents withdrawal if balance is low.