ASSIGNMENT-6.5

NAME:T.SASYA

BATCH:20

HTNO:2303A51346

TASK1- AI-Generated Python Code (Eligibility Logic)

Prompt :Generate an python code for voting

```
def check_voting_eligibility(age, citizenship):
    """
    Checks whether a person is eligible to vote based on age and citizenship.

    Parameters:
    age (int): Age of the person
    citizenship (str): Citizenship status ("Yes" or "No")

    Returns:
    str: Eligibility result
    """

    if age >= 18 and citizenship.lower() == "yes":
        return "Eligible to vote"
    else:
        return "Not eligible to vote"
```

```
# Example Usage
age = int(input("Enter your age: "))
citizenship = input("Are you a citizen? (Yes/No): ")

result = check_voting_eligibility(age, citizenship)
print(result)
```

```
Enter your age: 34
Are you a citizen? (Yes/No): yes
Eligible to vote
```

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
20	Yes	Eligible to vote
17	Yes	Not eligible
25	No	Not eligible
16	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.”

```
def count_vowels_consonants(text):
    """
    Counts vowels and consonants in a given string using a loop.

    Parameters:
    text (str): Input string

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

    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in text:
        # Check only alphabetic characters
        if char.isalpha():
            if char in vowels:
                vowel_count += 1
```

```
            else:
                consonant_count += 1

    return vowel_count, consonant_count


# Example Usage
string_input = input("Enter a string: ")

vowels, consonants = count_vowels_consonants(string_input)

print("Number of vowels:", vowels)
print("Number of consonants:", consonants)
```

```
Enter a string: codingusingai
Number of vowels: 6
Number of consonants: 7
```

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.

Output Verification (Example Runs)

Example 1

Input:

codeusingai

Output:

Number of vowels: 6

Number of consonants: 7

Explanation:

- Vowels: i,i,i,u,a, o → 6
- Consonants: c,d,n,g,s,n,g→ 7

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 system

using classes, loops, and conditional statements.”

```
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()
```

```
===== Library Management System =====
1. Add Book
2. Display Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice (1-5): 1
Enter Book ID: 23
Enter Title: aiac
Enter Author: myself
Book added successfully!
```

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**.

Reflection

Using AI tools for coding assistance is highly beneficial, especially for beginners and intermediate programmers. The AI helps in quickly generating structured programs that follow best practices such as modular design, object-oriented programming, and proper control flow.

In this task, the AI effectively combined classes, loops, and conditional statements to create a functional library management system. It reduced development time and provided a strong foundation that can be further improved.

However, AI-generated code should not be used blindly. Programmers must review and understand the logic to identify limitations, optimize performance, and ensure security. AI serves best as a learning assistant and productivity tool rather than a complete replacement for human problem-solving.

Overall, AI-assisted coding enhances learning, improves efficiency, and supports better software development when used responsibly.

TASK-4 AI-Assisted Code Completion for Class-Based Attendance System

Prompt: “Generate a Python class to mark and display student

attendance 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.

Sample Output (Verification)

Rahul (P/A): P

Ananya (P/A): A

Attendance Report

Rahul : Present

Ananya : Absent

4. Test Cases

Input	Expected Output
P	Present
A	Absent
X	Absent

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

PROMPT: “Generate a Python program using loops and conditionals

to simulate an ATM menu.”

```python
balance = 5000   # Initial balance

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

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

    if choice == "1":
        print("Your Balance is:", balance)

    elif choice == "2":
        amount = int(input("Enter deposit amount: "))
        balance += amount
        print("Amount Deposited Successfully.")

    elif choice == "3":
        amount = int(input("Enter withdrawal amount: "))

        if amount <= balance:
            balance -= amount
            print("Please collect your cash.")
        else:
            print("Insufficient balance.")

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

```
===== ATM MENU =====
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice (1-4): 2
Enter deposit amount: 2000
Amount Deposited Successfully.
```

Menu Logic

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

Sample Output

Example Run

===== ATM MENU =====

1. Check Balance

2. Deposit

3. Withdraw

4. Exit

Enter choice: 1

Your Balance is: 5000

Enter choice: 2

Enter deposit amount: 1000

Amount Deposited Successfully.

Enter choice: 1

Your Balance is: 6000

Enter choice: 3

Enter withdrawal amount: 2000

Please collect your cash.

Enter choice: 4

Thank you for using ATM.

Test Cases

Choice	Input	Output
1	—	Shows balance
2	1000	Balance increases
3	>Balance	Error message
4	—	Program exits
Other	X	Invalid option