SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 6.5

Program	**: B. Tech (CSE)**
Specialization	**:**
Course Title	**: AI Assisted coding**
Course Code	**:**
Semester	**: II**
Academic Session	**: 2025-2026**
Name of Student	**: Naniprasad Macharla**
Enrollment No.	**: 2403A51L11**
Batch No.	**: 51**
Date	**:23-01-2026**

Submission Start from here:

- **Task 1:** (AI-Based Code Completion for Conditional Eligibility Check)
- Task: Use an AI tool to generate eligibility logic.
- Prompt:
- "Generate Python code to check voting eligibility based on age and citizenship."

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

if age >= 18 and citizenship.lower() == "indian":
    print("You are eligible to vote")
else:
    print("You are not eligible to vote")
```

```
Enter your age: 20
Enter your citizenship: indian
You are eligible to vote
```

Explanation (Line by Line)

```
age = int(input("Enter your age: "))
```

- Takes the user's age as input.
- `input()` reads the value as a **string**.
- `int()` converts it into an **integer** so we can compare numbers.

```
citizenship = input("Enter your citizenship: ")
```

- Takes the user's citizenship as input.

- Stored as a **string**.

```
citizenship.lower()
```

- Converts the input to lowercase.
- This avoids problems with different cases like:
    - `Indian`
    - `INDIAN`
    - `indian`

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

- Checks **two conditions**:
    1. Age must be **18 or above**
    2. Citizenship must be **Indian**
- The `and` operator means **both conditions must be true**.

```
print("You are eligible to vote")
```

- Executes when both conditions are satisfied.

```
else:
    print("You are not eligible to vote")
```

- Executes when **any one** of the conditions is false.

- **Task2:**AI-Based Code Completion for Loop-Based String Processing)
- 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."

```
def vowels_consonants_count(string):
    vowels = 0
    consonants =0
    for i in string:
      if i in "AEIOUaeiou":
        vowels+=1
      else:
        consonants+=1
    return vowels,consonants
string=input()
print(vowels_consonants_count(string))
```

```
naniprasad
(4, 6)
```

Explanation

```
def vowels_consonants_count(string):
```

- Defines a function named `vowels_consonants_count`.
- It takes one parameter `string`.

```
vowels = 0
```

```
consonants = 0
```

- Two variables are initialized to **count vowels and consonants**.

```
for i in string:
```

- Iterates through **each character** in the input string.

```
if i in "AEIOUaeiou":
```

- Checks whether the character is a **vowel**.
- Both uppercase and lowercase vowels are included.

```
vowels += 1
```

- Increments the vowel count if the condition is true.

```
else:
    consonants += 1
```

- If the character is **not a vowel**, it is counted as a consonants

```
return vowels, consonants
```

- Returns both counts as a **tuple** `(vowels, consonants)`.

```
string = input()
```

- Takes a string input from the user.

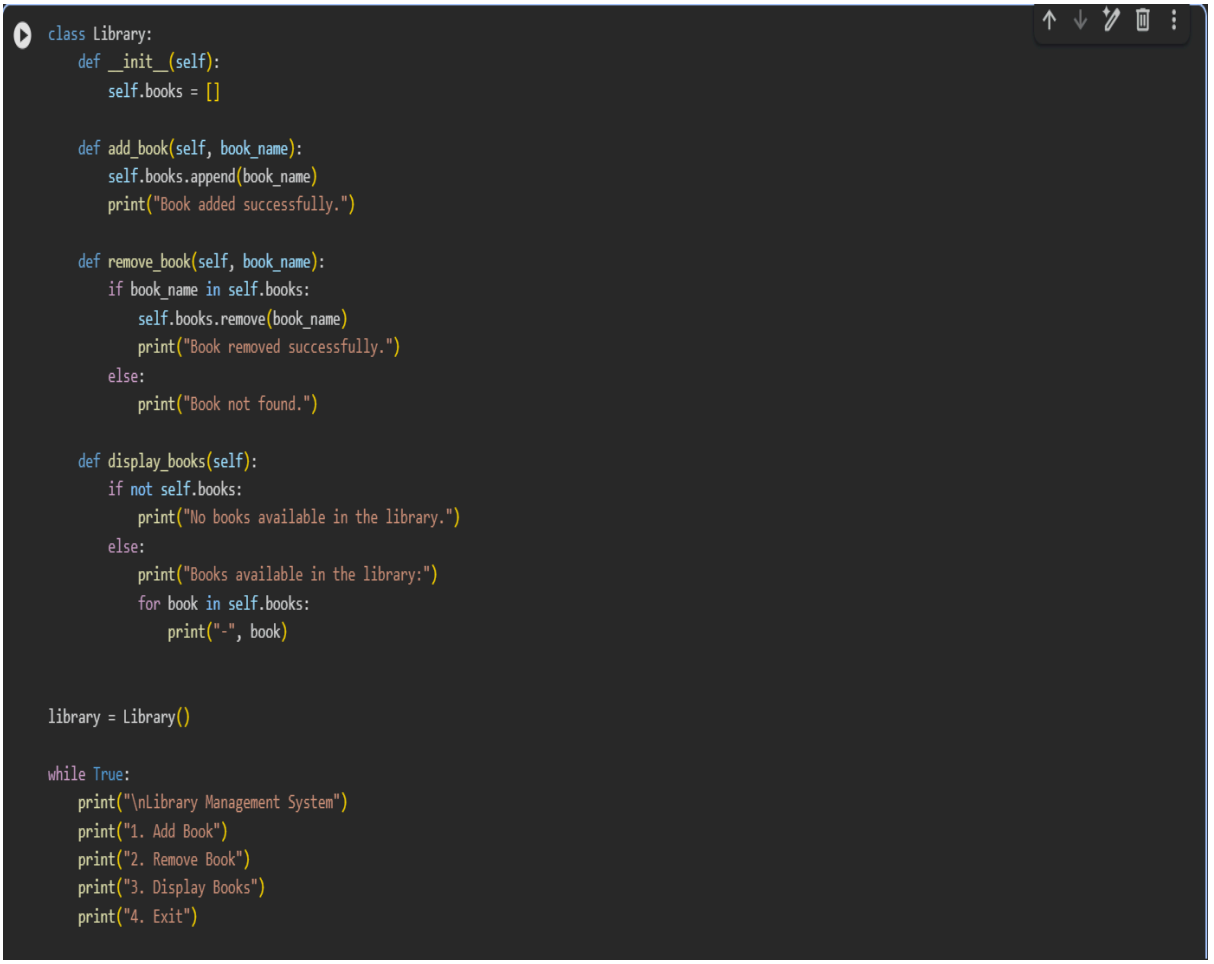
```
print(vowels_consonants_count(string))
```

- Calls the function and prints the result.

❖ **Task3:** 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."



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

    def add_book(self, book_name):
        self.books.append(book_name)
        print("Book added successfully.")

    def remove_book(self, book_name):
        if book_name in self.books:
            self.books.remove(book_name)
            print("Book removed successfully.")
        else:
            print("Book not found.")

    def display_books(self):
        if not self.books:
            print("No books available in the library.")
        else:
            print("Books available in the library:")
            for book in self.books:
                print("-", book)


library = Library()

while True:
    print("\nLibrary Management System")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Display Books")
    print("4. Exit")
```

```
library = Library()

while True:
    print("\nLibrary Management System")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Display Books")
    print("4. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        book = input("Enter book name: ")
        library.add_book(book)

    elif choice == 2:
        book = input("Enter book name to remove: ")
        library.remove_book(book)

    elif choice == 3:
        library.display_books()

    elif choice == 4:
        print("Exiting the system.")
        break

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

Library Management System
1. Add Book
```

```
Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice: 3
No books available in the library.

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice: 1
Enter book name: python
Book added successfully.

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice: 3
Books available in the library:
- python

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice: 4
Exiting the system.
```

Library Management System – Explanation

1. Class Definition

```
class Library:
```

- Defines a class named **Library**.
- It represents a library and its operations.

---

2. Constructor (`__init__` method)

```
def __init__(self):
    self.books = []
```

- Initializes an empty list `books`.
- This list stores the names of books available in the library.

---

3. Add Book Method

```
def add_book(self, book_name):
    self.books.append(book_name)
    print("Book added successfully."
```

- Adds a new book to the library.
- Uses `append()` to store the book name in the lists.

---

4. Remove Book Method

```
def remove_book(self, book_name):
    if book_name in self.books:
        self.books.remove(book_name)
        print("Book removed successfully.")
    else:
        print("Book not found.")
```

- Checks whether the book exists in the library.
- If found, it removes the book.
- Otherwise, it displays “Book not found”.

---

5. Display Books Method

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

```
else:
    for book in self.books:
    print("-", book)
```

- Checks if the library is empty.
- If not empty, uses a **for loop** to display all books.

---

6. Object Creation

```
library = Library()
```

- Creates an object of the `Library` class.
- This object is used to call library functions.

---

7. Menu-Driven Loop

```
while True:
```

- Keeps the program running until the user chooses to exit.

---

8. Conditional Statements

```
if choice == 1:
```

- Add a book.

```
elif choice == 2:
```

- Remove a book.

```
elif choice == 3:
```

- Display all books.

```
elif choice == 4:
```

- Exit the program.

```
else:
```

- Handles invalid input.

❖ **Task Description #4** (AI-Assisted Code Completion for Class-Based Attendance System)

Task: Use an AI tool to generate an attendance management class.

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

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

    def mark_attendance(self, name, status):
        self.students[name] = status
        print("Attendance marked.")

    def display_attendance(self):
        if not self.students:
            print("No attendance records found.")
        else:
            print("Attendance Report:")
            for name, status in self.students.items():
                print(name, ":", status)


attendance = Attendance()

while True:
    print("\nAttendance Management System")
    print("1. Mark Attendance")
    print("2. Display Attendance")
    print("3. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        name = input("Enter student name: ")
        status = input("Enter status (Present/Absent): ")
        attendance.mark_attendance(name, status)

    elif choice == 2:
```

```python
        if choice == 1:
            name = input("Enter student name: ")
            status = input("Enter status (Present/Absent): ")
            attendance.mark_attendance(name, status)

        elif choice == 2:
            attendance.display_attendance()

        elif choice == 3:
            print("Exiting the system.")
            break

        else:
            print("Invalid choice. Try again.")
```

```
...
Attendance Management System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice: 1
Enter student name: nani
Enter status (Present/Absent): Present
Attendance marked.

Attendance Management System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice: 3
Exiting the system.
```

Program Explanation

1. Class Definition

```
class Attendance:
```

- Defines a class named **Attendance**.
- This class is responsible for managing student attendance.

---

2. Constructor (`__init__` method)

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

- Initializes an empty dictionary `students`.
- The dictionary stores:
  - **Key** → Student name
  - **Value** → Attendance status (Present / Absent)

---

3. Mark Attendance Method

```
def mark_attendance(self, name, status):
    self.students[name] = status
    print("Attendance marked.")
```

- Takes the student's **name** and **attendance status** as input.
- Stores or updates the attendance in the dictionary.
- Prints a confirmation messages

---

4. Display Attendance Method

```
def display_attendance(self):
    if not self.students:
        print("No attendance records found.")
    else:
        print("Attendance Report:")
```

```
        for name, status in self.students.items():
            print(name, ":", status)
```

- Checks if attendance records exist.
- If empty, displays a message.
- Otherwise:
    - Uses a **for loop** to print each student's attendances

---

5. Object Creation

```
attendance = Attendance()
```

- Creates an object of the `Attendance` class.
- This object is used to access class methods.

---

6. Menu-Driven Loop

```
while True:
```

- Keeps the program running until the user exits.

---

7. User Choices with Conditionals

```
if choice == 1:
```

- Marks attendance.

```
elif choice == 2:
```

- Displays attendance.

```
elif choice == 3:
```

- Exits the program.

```
else:
```

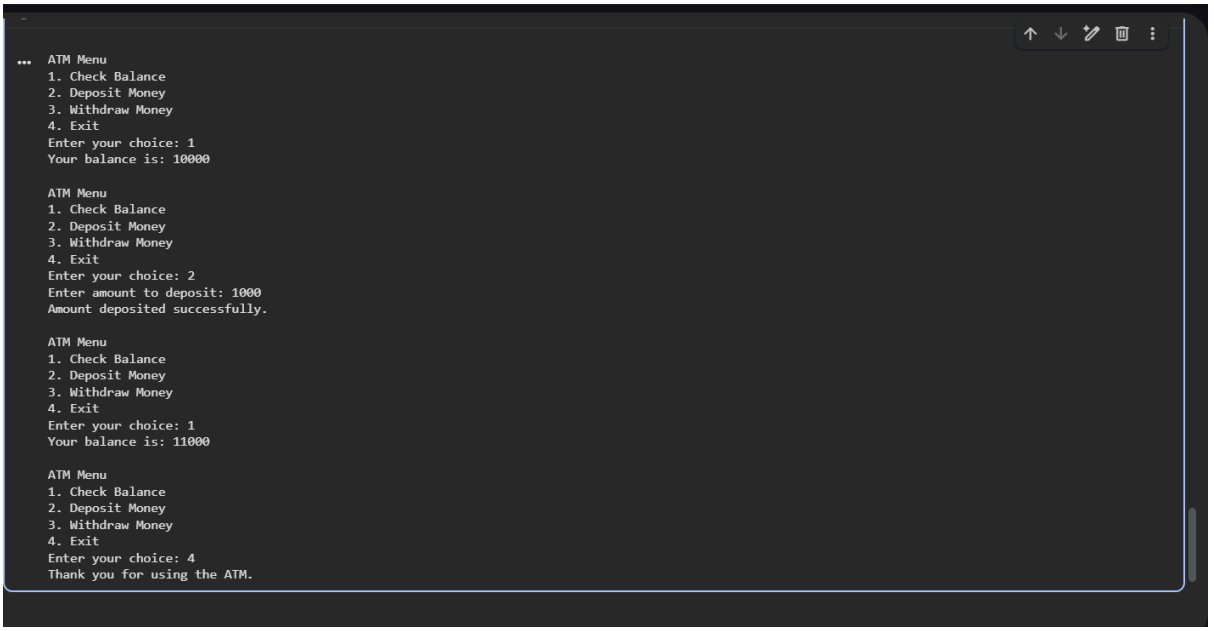
- Handles invalid input.

- Task Description #5 (AI-Based Code Completion for Conditional Menu Navigation)
- Task: Use an AI tool to complete a navigation menu.
- Prompt: “Generate a Python program using loops and conditionals to simulate an ATM menu.”

```python
balance = 10000  # initial balance

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

    choice = int(input("Enter your choice: "))

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

    elif choice == 2:
        amount = int(input("Enter amount to deposit: "))
        balance += amount
        print("Amount deposited successfully.")

    elif choice == 3:
        amount = int(input("Enter amount to withdraw: "))
        if amount <= balance:
            balance -= amount
            print("Please collect your cash.")
        else:
            print("Insufficient balance.")

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

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

```python
    elif choice == 2:
        amount = int(input("Enter amount to deposit: "))
        balance += amount
        print("Amount deposited successfully.")

    elif choice == 3:
        amount = int(input("Enter amount to withdraw: "))
        if amount <= balance:
            balance -= amount
            print("Please collect your cash.")
        else:
            print("Insufficient balance.")

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

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

```
ATM Menu
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 1
Your balance is: 10000

ATM Menu
1. Check Balance
2. Deposit Money
3. Withdraw Money
```



Explanation

1. Initial Balance

```
balance = 10000
```

- Stores the initial amount available in the account.

---

2. Loop Usage

```
while True:
```

- Keeps the ATM menu running continuously.
- Stops only when the user selects **Exit**.

---

3. Menu Display

```
print("\nATM Menu")
```

- Displays available ATM options to the user.

---

4. Conditional Statements

```
if choice == 1:
```

- Checks and displays current balance.

```
elif choice == 2:
```

- Deposits money and updates the balance.

```
elif choice == 3:
```

- Withdraws money.
- Uses a condition to check for **sufficient balance**.

```
elif choice == 4:
```

- Exits the program using `break`.

```
else:
```

- Handles invalid menu choices.