School of Computer Science and Artificial Intelligence

Lab Assignment # 6.5

Program : B. Tech (CSE)

Specialization :AIML

Course Title : AI Assisted Coding

Course Code : 23CS002PC304

Semester : VI

Academic Session : 2025-2026

Name of Student : k.sathwika

Enrollment No. : 2303A52096

Batch No. : 33

Date :30/01/26

Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals LO1. Use AI-based code completion tools to generate Python code involving classes, loops, and conditionals. LO2. Interpret and explain AI-generated code line-by-line. LO3. Identify errors, inefficiencies, or logical flaws in AI-suggested implementations. LO4. Optimize AI-generated code for better readability and performance. LO5. Demonstrate ethical and responsible use of AI tools in coding tasks.

Task Description #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.” Expected Output: • AI-generated conditional logic. • Correct eligibility decisions. • Explanation of conditions.

Task Description #2(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.” Expected Output: • AI-generated string processing logic. • Correct counts. • Output verification.

Task Description #3 (AI-Assisted Code Completion Reflection Task) 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.

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.” Expected Output: • AI-generated attendance logic. • Correct display of attendance. • Test cases. 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.” Expected Output: • AI-generated menu logic. • Correct option handling. • Output verification.

Task1:

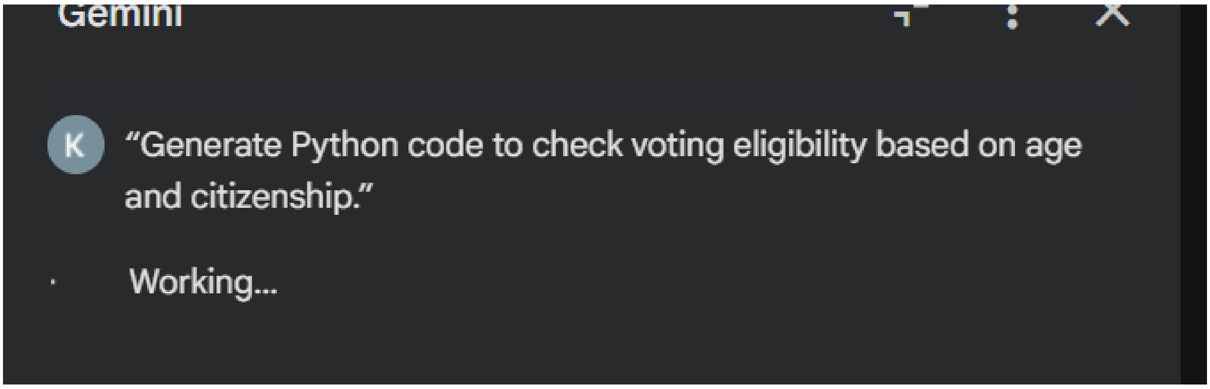
Aim

To use AI-based code completion tools to generate, analyze, optimize, and ethically evaluate Python programs involving classes, loops, and conditional statements.

Learning Outcomes Addressed

- LO1: Generate Python code using AI
- LO2: Explain AI-generated code line-by-line
- LO3: Identify logical flaws or inefficiencies
- LO4: Optimize AI-generated code

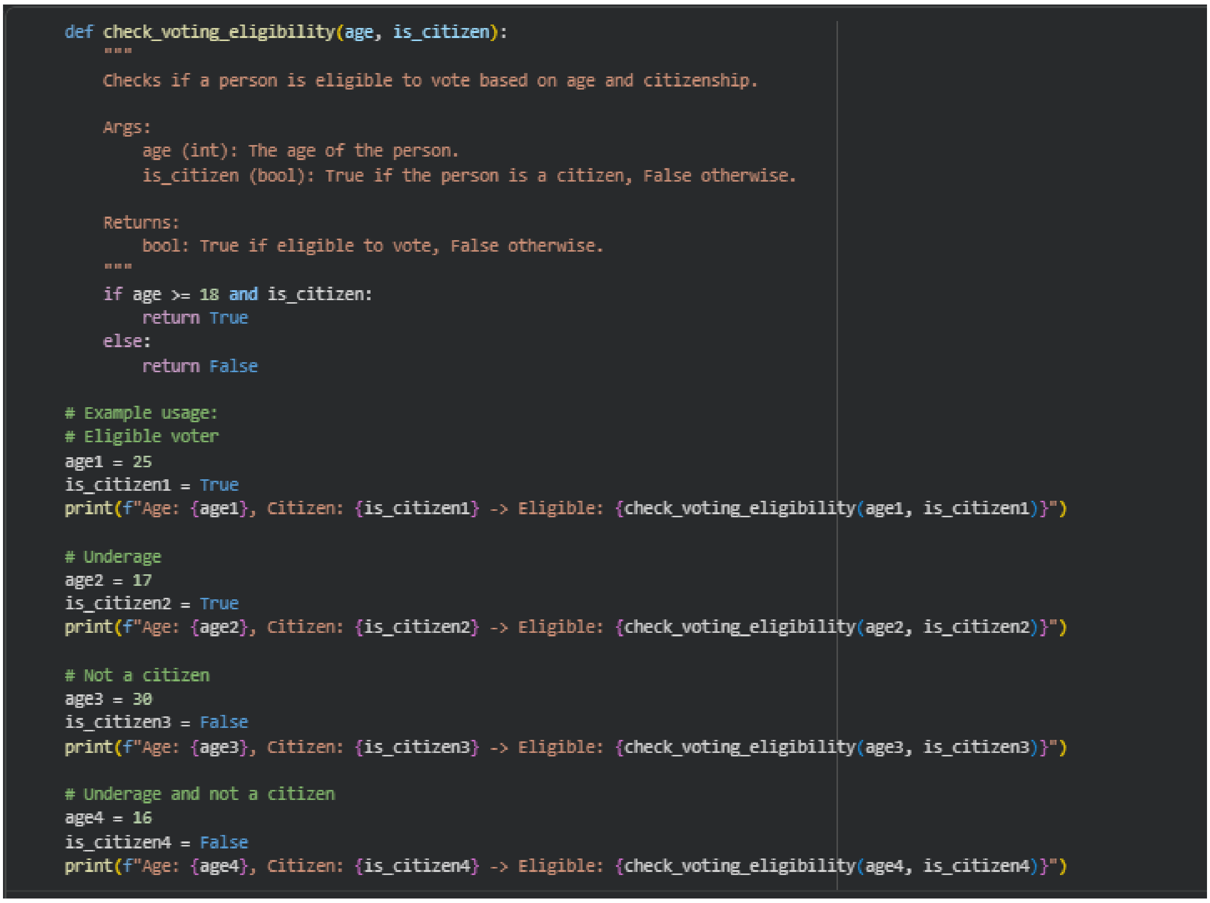
- LO5: Demonstrate ethical use of AI tools



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

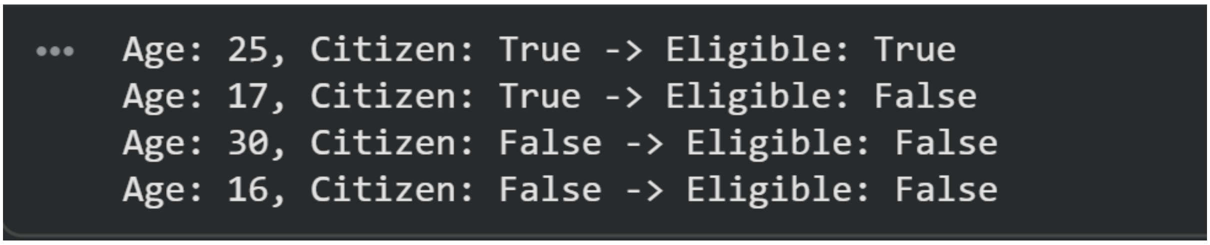
Working...

Code:



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

    Args:
        age (int): The age of the person.
        is_citizen (bool): True if the person is a citizen, False otherwise.

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

# Example usage:
# Eligible voter
age1 = 25
is_citizen1 = True
print(f"Age: {age1}, Citizen: {is_citizen1} -> Eligible: {check_voting_eligibility(age1, is_citizen1)}")

# Underage
age2 = 17
is_citizen2 = True
print(f"Age: {age2}, Citizen: {is_citizen2} -> Eligible: {check_voting_eligibility(age2, is_citizen2)}")

# Not a citizen
age3 = 30
is_citizen3 = False
print(f"Age: {age3}, Citizen: {is_citizen3} -> Eligible: {check_voting_eligibility(age3, is_citizen3)}")

# Underage and not a citizen
age4 = 16
is_citizen4 = False
print(f"Age: {age4}, Citizen: {is_citizen4} -> Eligible: {check_voting_eligibility(age4, is_citizen4)}")
```

Output:



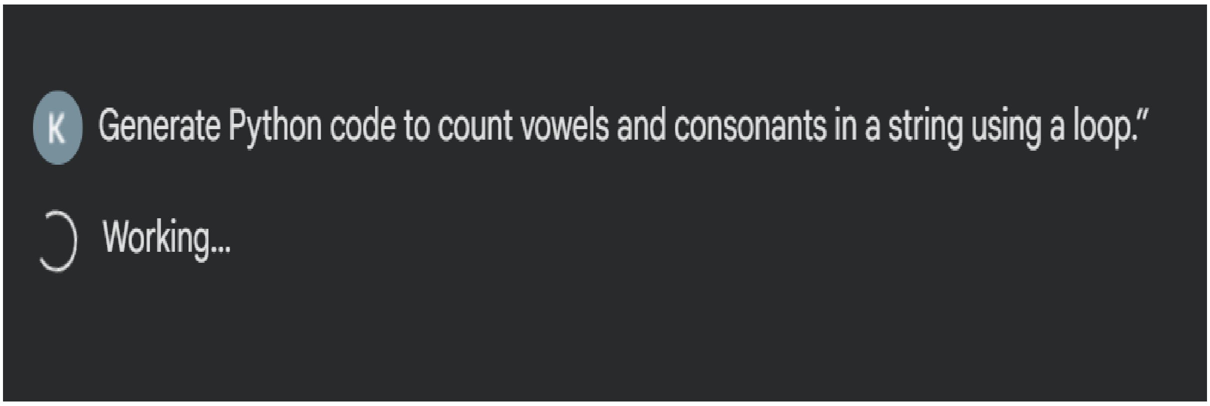
```
Age: 25, Citizen: True -> Eligible: True
Age: 17, Citizen: True -> Eligible: False
Age: 30, Citizen: False -> Eligible: False
Age: 16, Citizen: False -> Eligible: False
```

Explanation (Line-by-Line)

- Takes age as integer input
- Takes citizenship status as string
- Uses AND condition:
    - Age must be 18 or above
    - Citizenship must be yes
- Prints eligibility result

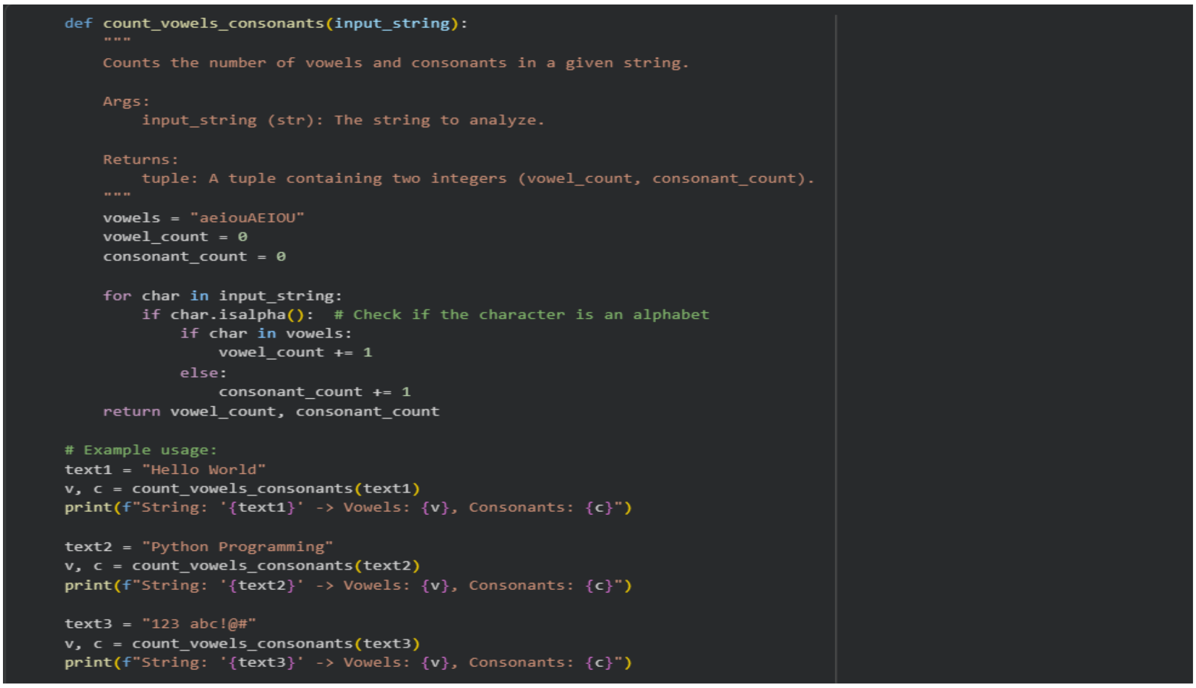
Task2:

Loop-Based String Processing (Vowels & Consonants)



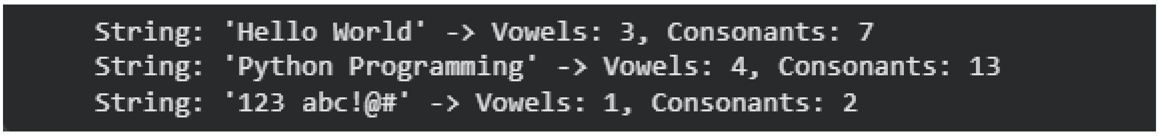
Generate Python code to count vowels and consonants in a string using a loop.”

Code:



```python
def count_vowels_consonants(input_string):
    """
    Counts the number of vowels and consonants in a given string.

    Args:
        input_string (str): The string to analyze.

    Returns:
        tuple: A tuple containing two integers (vowel_count, consonant_count).
    """
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in input_string:
        if char.isalpha():  # Check if the character is an alphabet
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1
    return vowel_count, consonant_count

# Example usage:
text1 = "Hello World"
v, c = count_vowels_consonants(text1)
print(f"String: '{text1}' -> Vowels: {v}, Consonants: {c}")

text2 = "Python Programming"
v, c = count_vowels_consonants(text2)
print(f"String: '{text2}' -> Vowels: {v}, Consonants: {c}")

text3 = "123 abc!@#"
v, c = count_vowels_consonants(text3)
print(f"String: '{text3}' -> Vowels: {v}, Consonants: {c}")
```

Output:



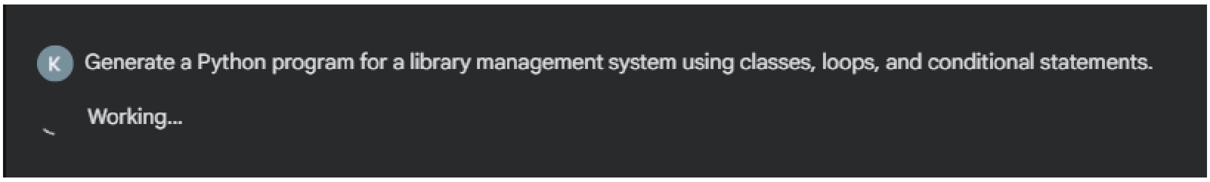
```
String: 'Hello World' -> Vowels: 3, Consonants: 7
String: 'Python Programming' -> Vowels: 4, Consonants: 13
String: '123 abc!@#' -> Vowels: 1, Consonants: 2
```

Explanation:

- Converts string to lowercase
- Loops through each character
- Checks:
    - Alphabet or not
    - Vowel or consonant
- Counts correctly

Task 3:

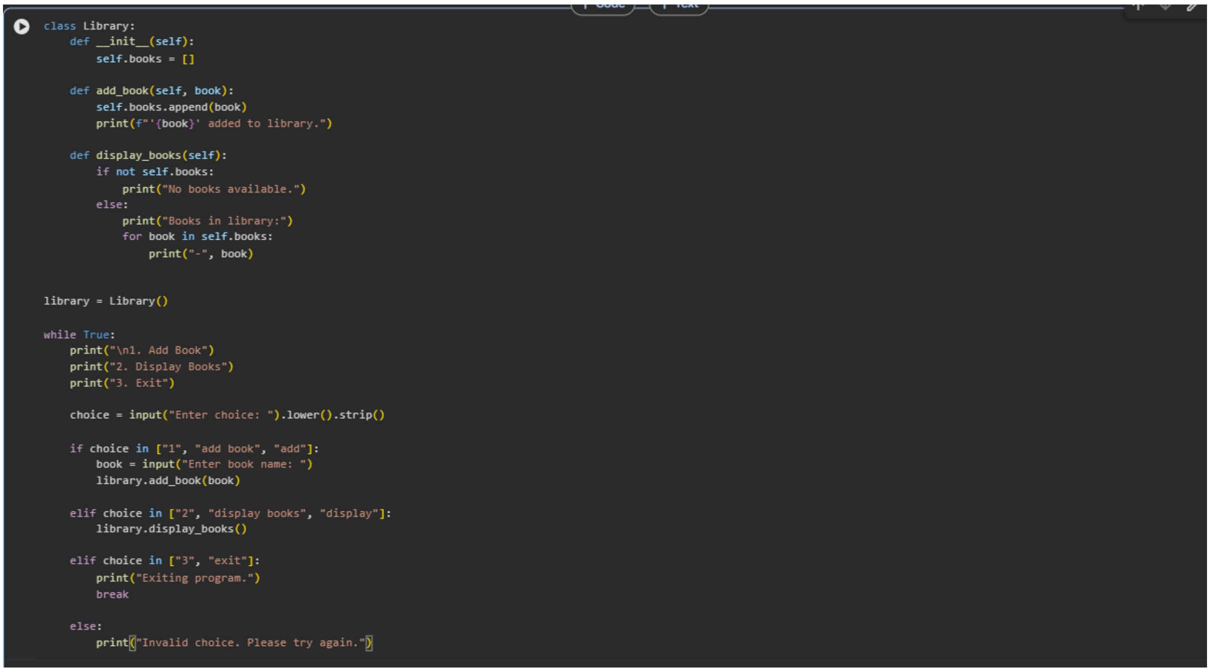
AI-Generated Library Management System



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

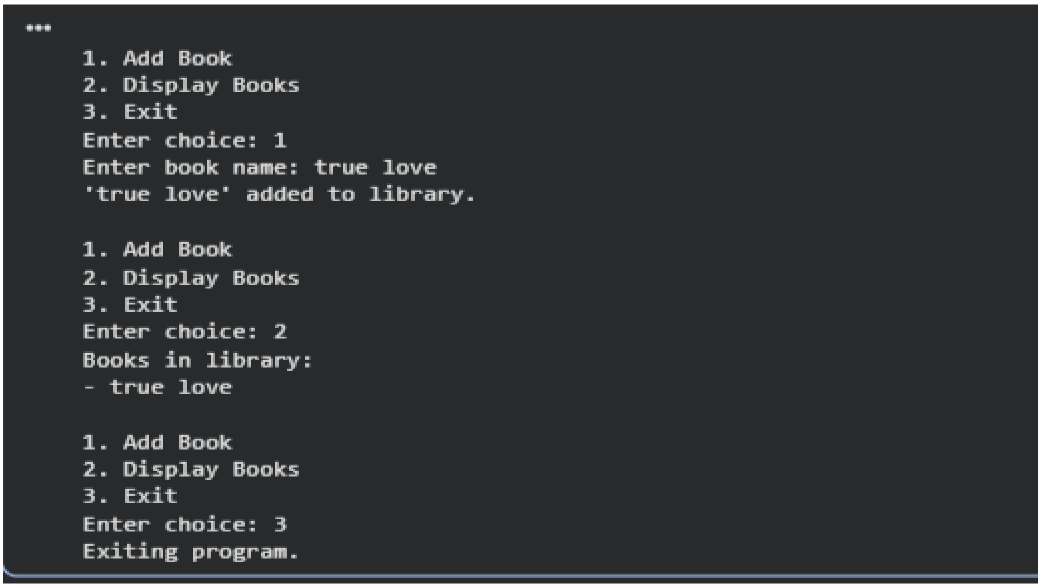
Working...

Code:



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

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

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


library = Library()

while True:
    print("\n1. Add Book")
    print("2. Display Books")
    print("3. Exit")

    choice = input("Enter choice: ").lower().strip()

    if choice in ["1", "add book", "add"]:
        book = input("Enter book name: ")
        library.add_book(book)

    elif choice in ["2", "display books", "display"]:
        library.display_books()

    elif choice in ["3", "exit"]:
        print("Exiting program.")
        break

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

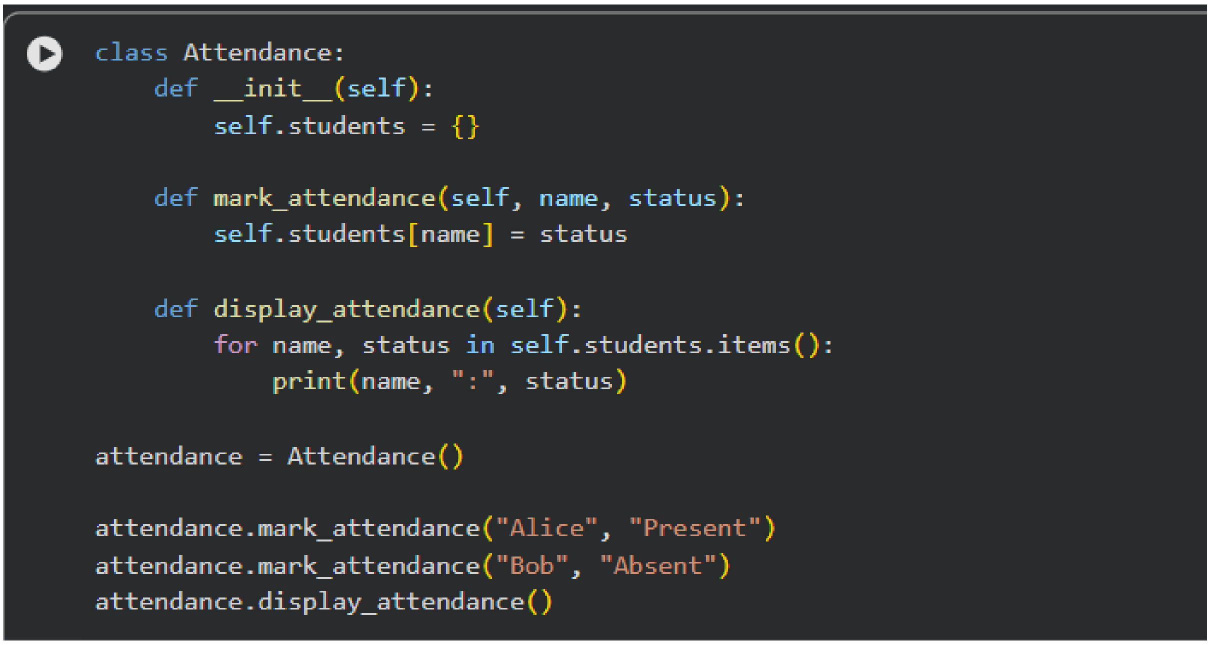
Ouput:



```

1. Add Book
2. Display Books
3. Exit
Enter choice: 1
Enter book name: true love
'true love' added to library.

1. Add Book
2. Display Books
3. Exit
Enter choice: 2
Books in library:
- true love

1. Add Book
2. Display Books
3. Exit
Enter choice: 3
Exiting program.
```

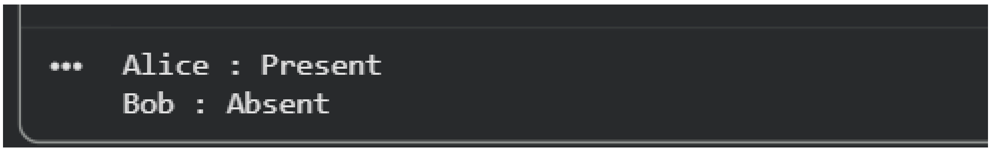
Task 4: Class-Based Attendance System

Code:



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

    def mark_attendance(self, name, status):
        self.students[name] = status

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

attendance = Attendance()

attendance.mark_attendance("Alice", "Present")
attendance.mark_attendance("Bob", "Absent")
attendance.display_attendance()
```

Output:

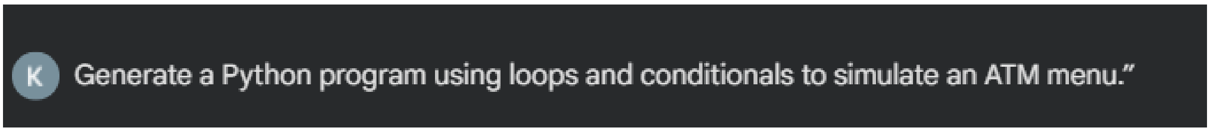


```
Alice : Present
Bob : Absent
```

Explanation

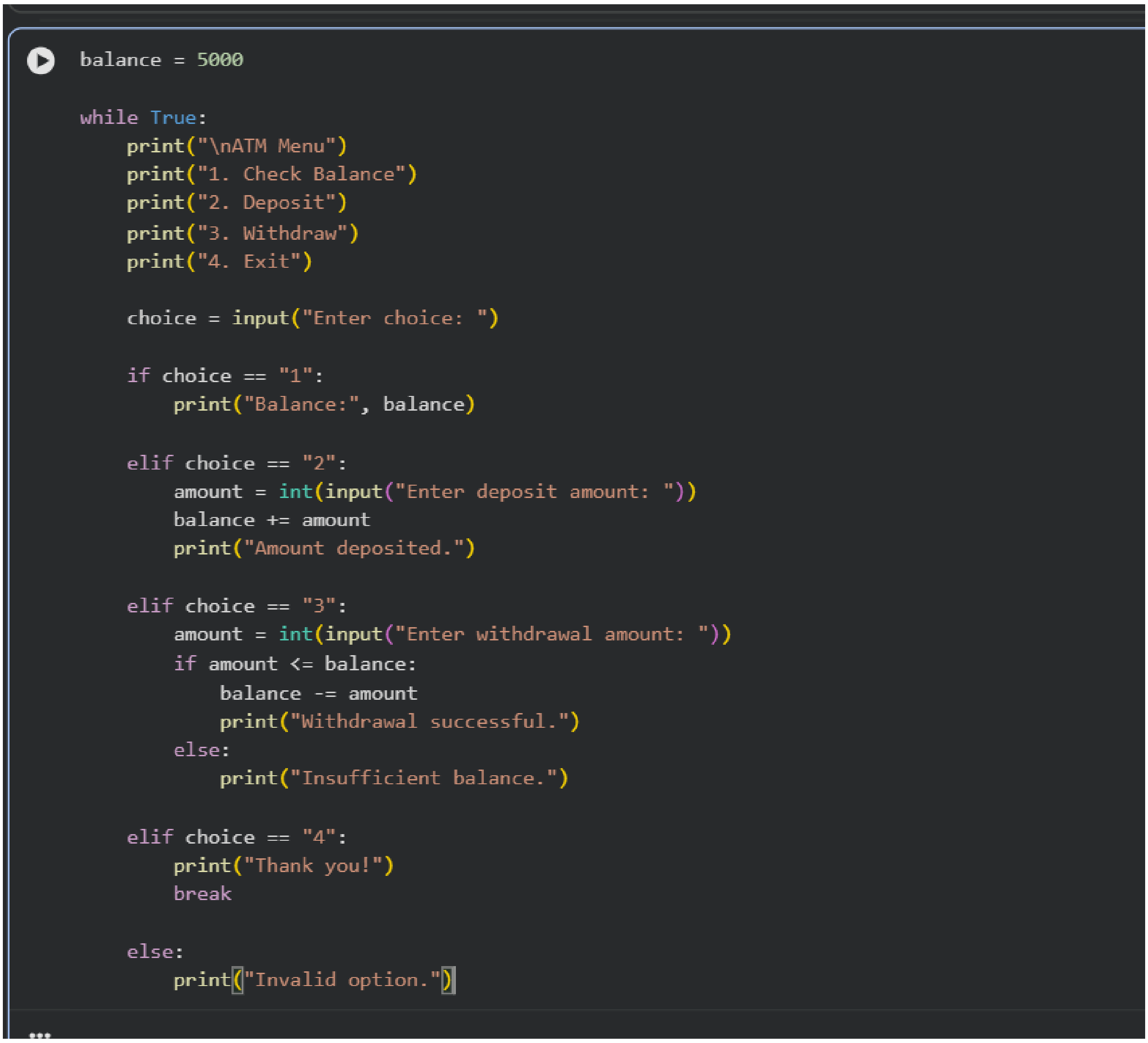
- Dictionary stores student attendance
- Loop prints all records
- Class improves data organization

Task_5:



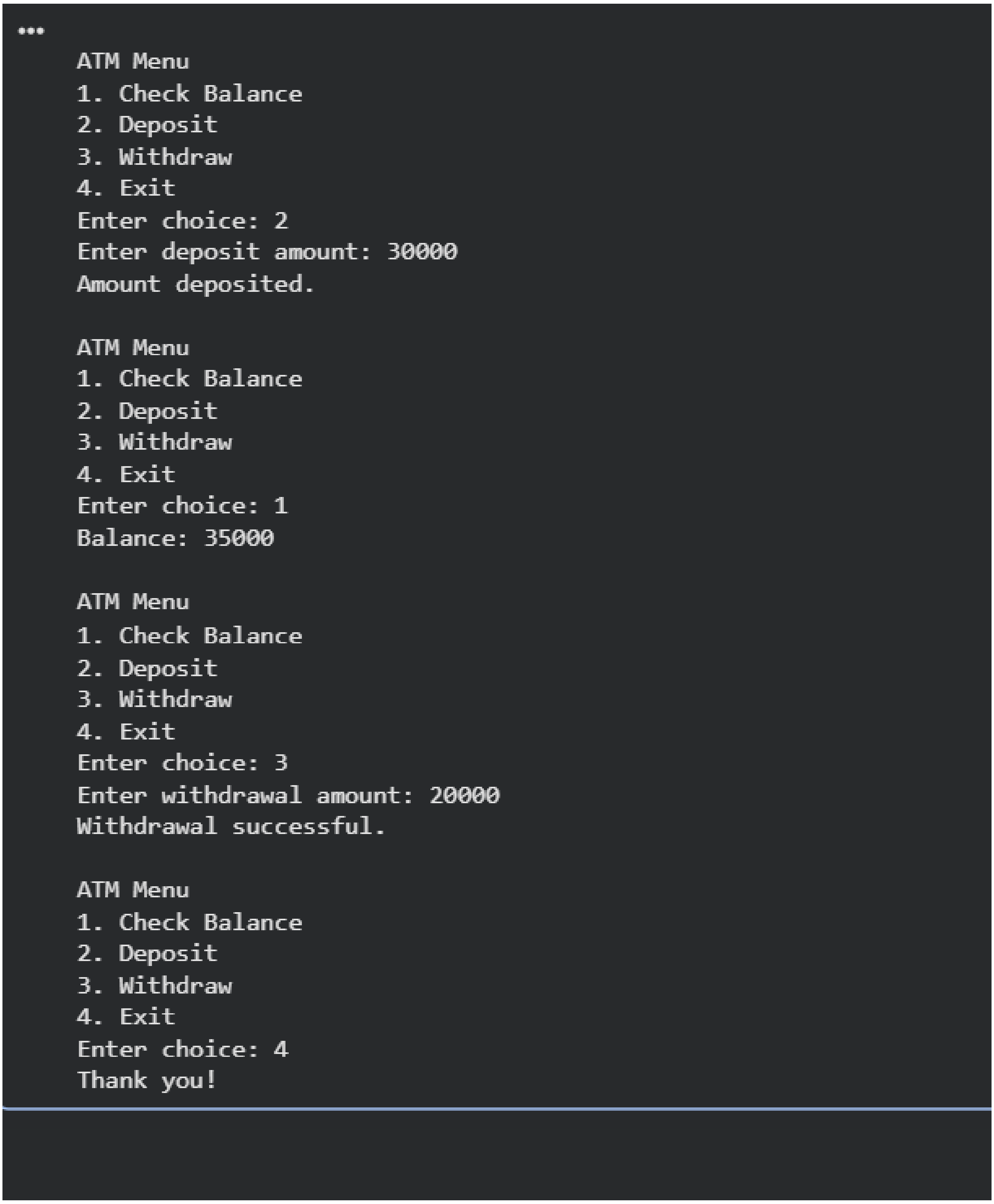
Generate a Python program using loops and conditionals to simulate an ATM menu.”

Code:



```python
balance = 5000

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

    choice = input("Enter choice: ")

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

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

    elif choice == "3":
        amount = int(input("Enter withdrawal amount: "))
        if amount <= balance:
            balance -= amount
            print("Withdrawal successful.")
        else:
            print("Insufficient balance.")

    elif choice == "4":
        print("Thank you!")
        break

    else:
        print("Invalid option.")
```

Output:



```
ATM Menu
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 2
Enter deposit amount: 30000
Amount deposited.

ATM Menu
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Balance: 35000

ATM Menu
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 3
Enter withdrawal amount: 20000
Withdrawal successful.

ATM Menu
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 4
Thank you!
```

Explanation:

This program simulates an ATM system using loops and conditional statements. A variable balance is initialized with a fixed amount. The while True loop is used to repeatedly display the ATM menu until the user chooses to exit. The program shows options such as checking balance, depositing money, withdrawing money, and exiting