SR UNIVERSITY

School of Computer Science and Artificial Intelligence

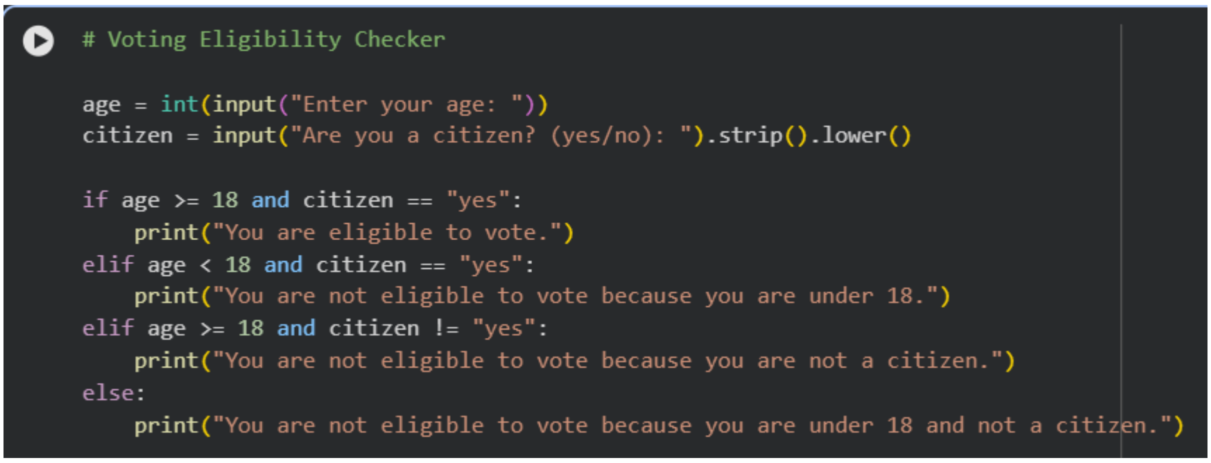
Lab Assignment # 6.5

Program	***: B. Tech (CSE)***
Specialization	***:***
Course Title	***: AI Assisted coding***
Course Code	***:***
Semester	***: II***
Academic Session	***: 2025-2026***
Name of Student	***: P Abhinav***
Enrollment No.	***: 2403A51L38***
Batch No.	***: 52***
Date	***:23-01-2026***

Task 1: Use an AI tool to generate eligibility logic.

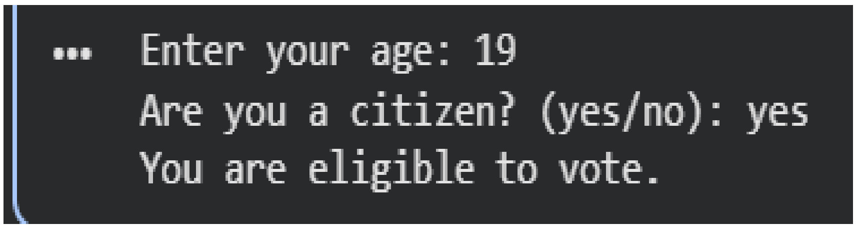
Prompt: Generate Python code to check voting eligibility based on age and citizenship.

Code Screenshot:



```python
# Voting Eligibility Checker

age = int(input("Enter your age: "))
citizen = input("Are you a citizen? (yes/no): ").strip().lower()

if age >= 18 and citizen == "yes":
    print("You are eligible to vote.")
elif age < 18 and citizen == "yes":
    print("You are not eligible to vote because you are under 18.")
elif age >= 18 and citizen != "yes":
    print("You are not eligible to vote because you are not a citizen.")
else:
    print("You are not eligible to vote because you are under 18 and not a citizen.")
```

Code Output:



```
Enter your age: 19
Are you a citizen? (yes/no): yes
You are eligible to vote.
```

Code Explanation:

· This program checks whether a person is eligible to vote based on given conditions.

· It takes the input age from the user and converts it into an integer.

· It takes citizenship status as input and converts it into lowercase for uniform comparison.

· The main decision is made using an if-else conditional statement.

· The condition age >= 18 ensures the person is at least 18 years old.

· The condition citizen == "yes" ensures the person is a citizen.

· The logical operator and is used because both conditions must be true.

· If both conditions are true, the program prints that the person can vote.

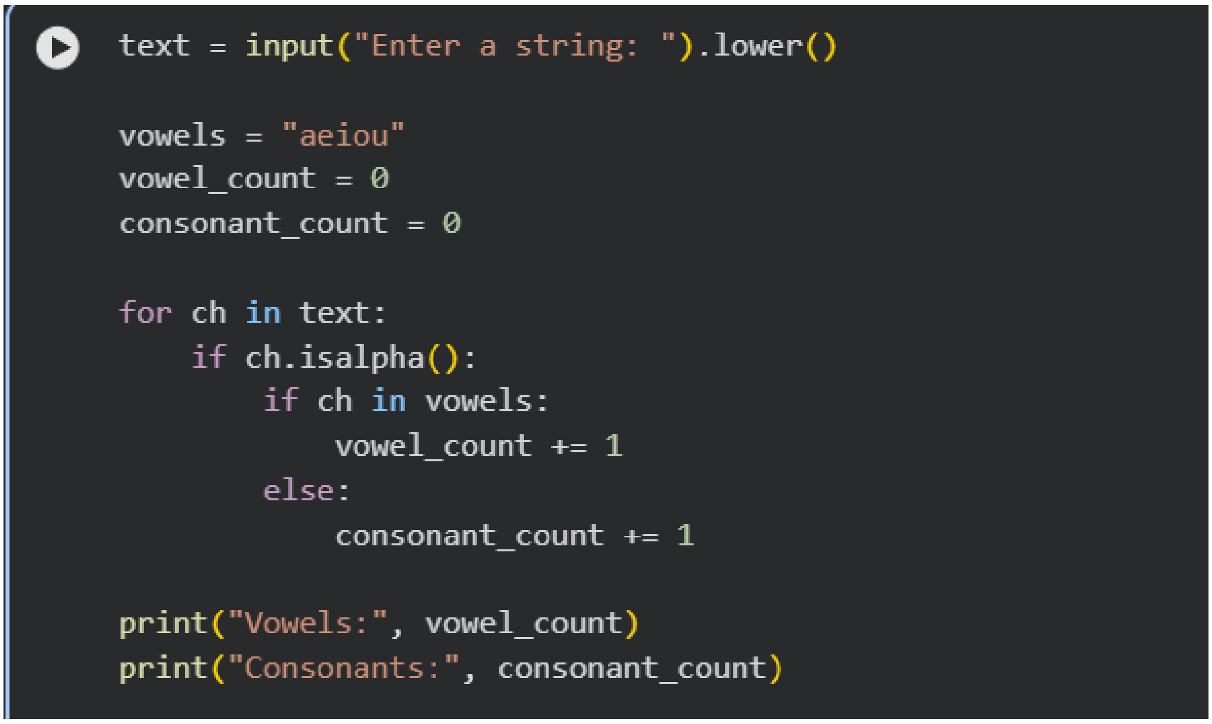
· If either age is below 18 or citizenship is not yes, the else block executes.

· This produces a correct eligibility decision based on the combined conditions

Task 2: **Use an AI tool to process strings using loops.**

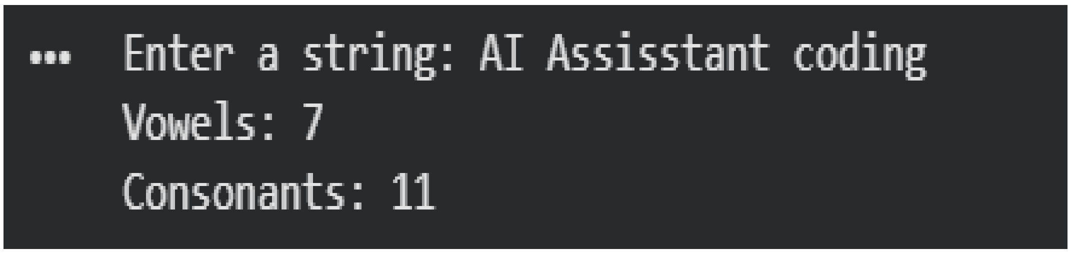
Prompt: Generate Python code to count vowels and consonants in a string using a loop.

Code Screenshot:



```python
text = input("Enter a string: ").lower()

vowels = "aeiou"
vowel_count = 0
consonant_count = 0

for ch in text:
    if ch.isalpha():
        if ch in vowels:
            vowel_count += 1
        else:
            consonant_count += 1

print("Vowels:", vowel_count)
print("Consonants:", consonant_count)
```

Output Screenshot:



```
Enter a string: AI Assisstant coding
Vowels: 7
Consonants: 11
```

Explanation:

· This program counts the number of vowels and consonants in a given string.

· The input string is converted to lowercase to simplify vowel checking.

· A variable vowels stores all vowel characters (a, e, i, o, u).

· Two counters vowel_count and consonant_count are initialized to zero.

· A for loop is used to traverse each character in the string.

· The condition ch.isalpha() ensures only alphabetic letters are counted.

· If the character is a letter and exists in the vowel set, vowel count is increased.

· Otherwise the character is treated as a consonant and consonant count increases.

· Numbers, spaces, and symbols are ignored because they are not alphabets.

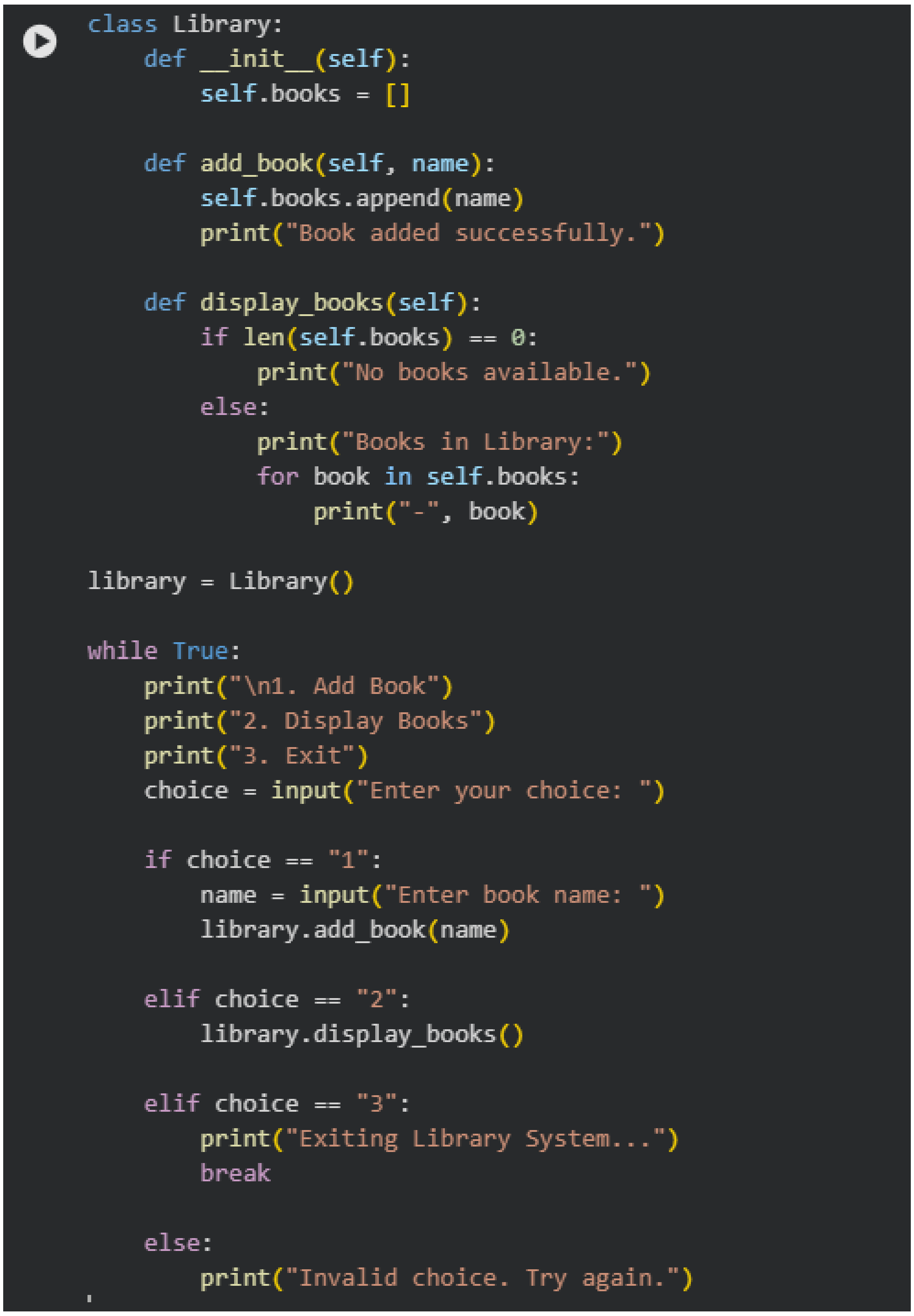
· Finally, the program prints the total vowel and consonant counts correctly.

Task 3: Use an AI tool to generate a complete program using classes,

loops, and conditionals.

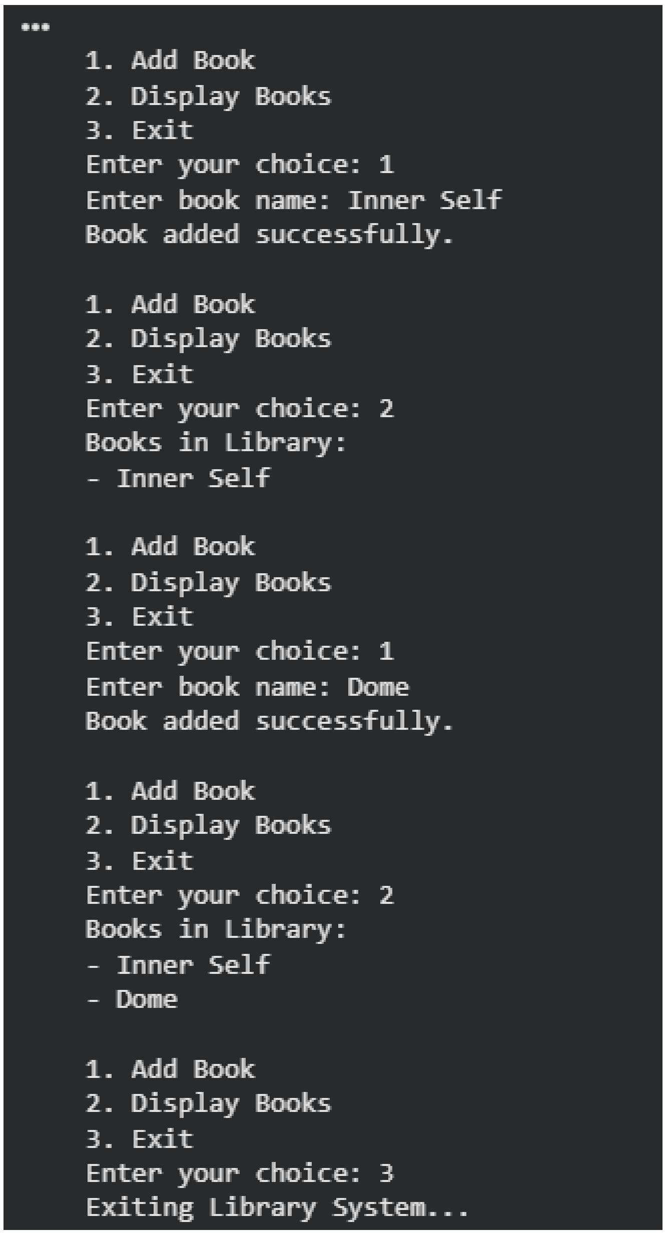
Prompt: Generate a Python program for a library management system using classes, loops, and conditional statements.

Code Screenshot:



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

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

    def display_books(self):
        if len(self.books) == 0:
            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 your choice: ")

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

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

    elif choice == "3":
        print("Exiting Library System...")
        break

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

Output Screenshot:



```
...
    1. Add Book
    2. Display Books
    3. Exit
    Enter your choice: 1
    Enter book name: Inner Self
    Book added successfully.

    1. Add Book
    2. Display Books
    3. Exit
    Enter your choice: 2
    Books in Library:
    - Inner Self

    1. Add Book
    2. Display Books
    3. Exit
    Enter your choice: 1
    Enter book name: Dome
    Book added successfully.

    1. Add Book
    2. Display Books
    3. Exit
    Enter your choice: 2
    Books in Library:
    - Inner Self
    - Dome

    1. Add Book
    2. Display Books
    3. Exit
    Enter your choice: 3
    Exiting Library System...
```

Explanation:

- This program simulates a simple library management system using Python.
- The Library class is created to store and manage books in the library.
- The constructor (__init__) initializes an empty list to store book names.
- The method add_book() adds a new book name into the list of books.
- The method display_books() prints all books currently available in the library.

- A condition checks whether the book list is empty before displaying books.

· A loop (while True) is used to repeatedly show the menu until the user exits.

· Conditional statements (if-elif-else) choose the correct operation from menu input.

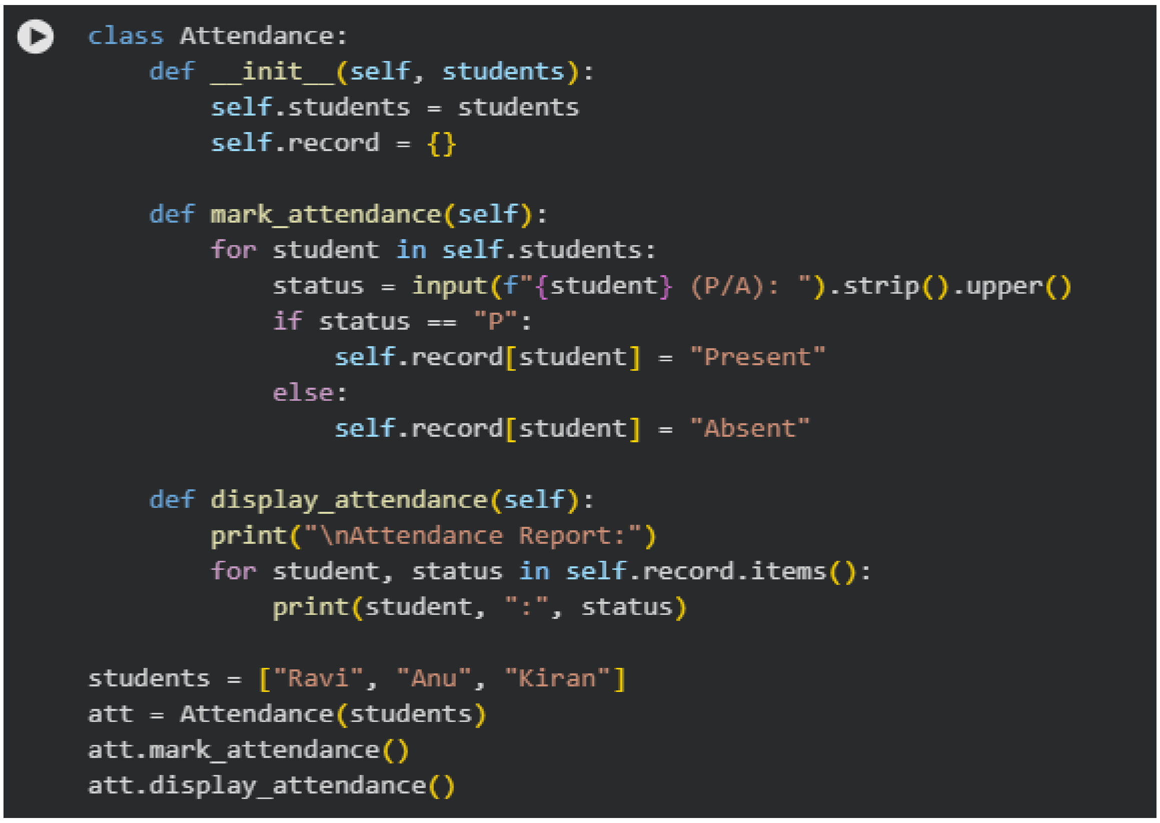
· The program is interactive and allows multiple operations without restarting.

· This demonstrates usage of classes, loops, and conditions in one complete program.

Task 4: Use an AI tool to generate an attendance management class.

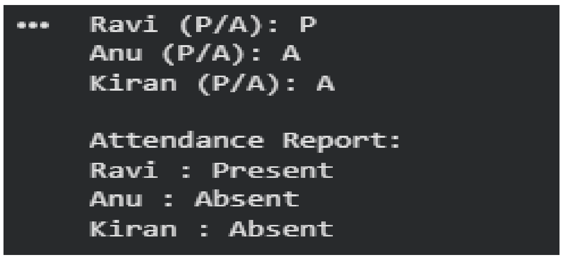
Prompt: Generate a Python class to mark and display student attendance using loops.

Code Screenshot:



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

    def mark_attendance(self):
        for student in self.students:
            status = input(f"{student} (P/A): ").strip().upper()
            if status == "P":
                self.record[student] = "Present"
            else:
                self.record[student] = "Absent"

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

students = ["Ravi", "Anu", "Kiran"]
att = Attendance(students)
att.mark_attendance()
att.display_attendance()
```

Code Output:



```
Ravi (P/A): P
Anu (P/A): A
Kiran (P/A): A

Attendance Report:
Ravi : Present
Anu : Absent
Kiran : Absent
```

Explanation:

· This program builds an attendance management system using a Python class.

· The Attendance class accepts a list of students and stores it in self.students.

· Attendance status is stored in a dictionary self.record with student names as keys.

· The function mark_attendance() loops through each student one by one.

· For every student, it asks the user to enter P for Present or A for Absent.

· The input is converted to uppercase for correct comparison.

· A condition checks whether the entered status is "P" or not.

· If input is "P", it marks the student as Present, otherwise Absent.

· The display_attendance() function prints the final attendance list.

· This task shows AI-generated class logic with loop processing and conditions.

Task 5: Use an AI tool to complete a navigation menu.

Prompt: Generate a Python program using loops and conditionals

to simulate an ATM menu.

Explanation:

· This program simulates an ATM system using a menu-driven approach.

· The program starts with an initial account balance stored in balance.

· A while True loop keeps displaying the ATM menu until the user exits.

· The program provides options like check balance, deposit money, withdraw money, and exit.

· Conditionals (if-elif-else) are used to execute the correct option based on user choice.

· Option 1 displays the current balance directly.

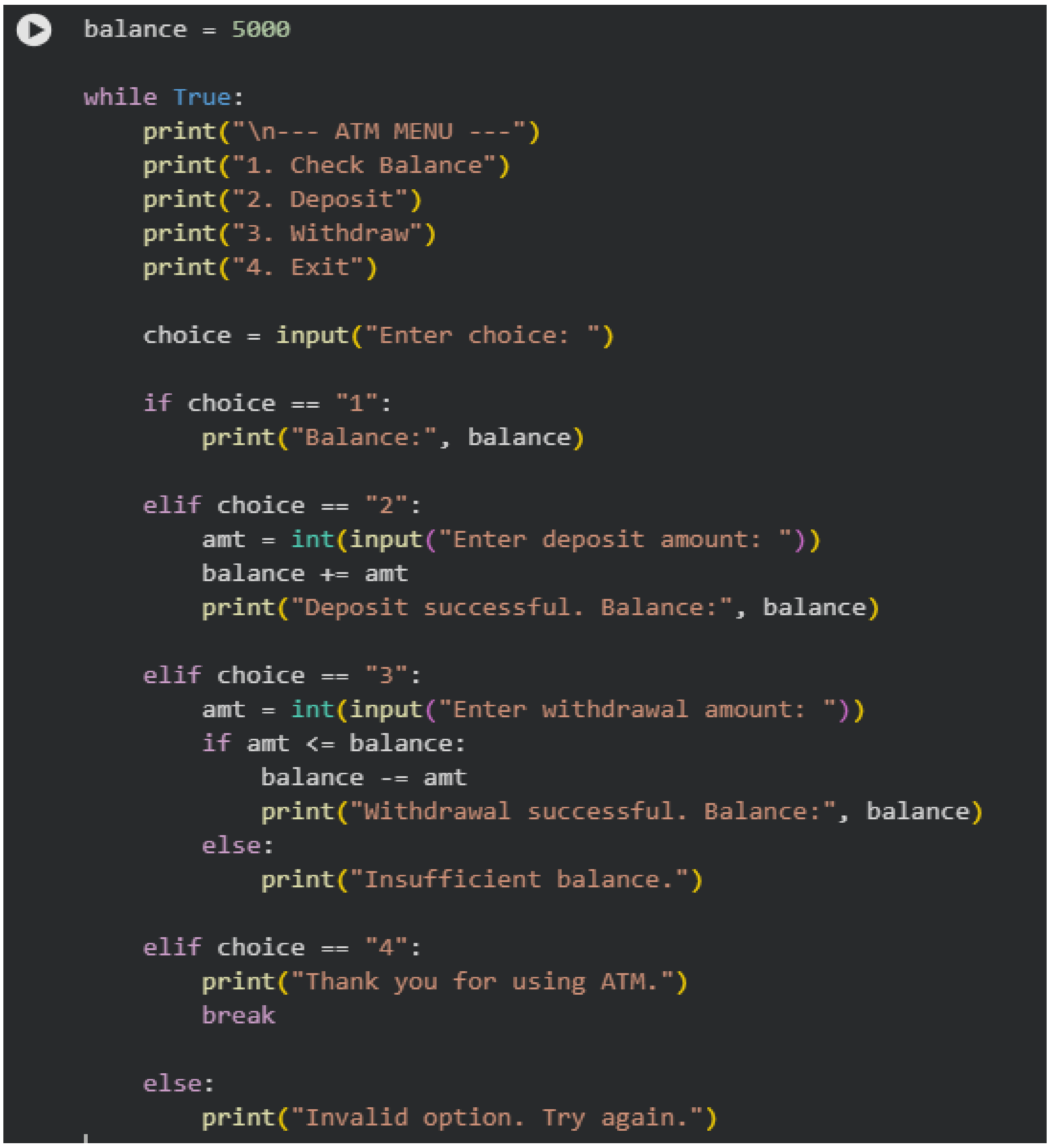
· Option 2 accepts a deposit amount and adds it to the balance.

· Option 3 checks withdrawal amount using the condition amt <= balance.

· If sufficient balance exists, it deducts amount, otherwise prints insufficient balance.

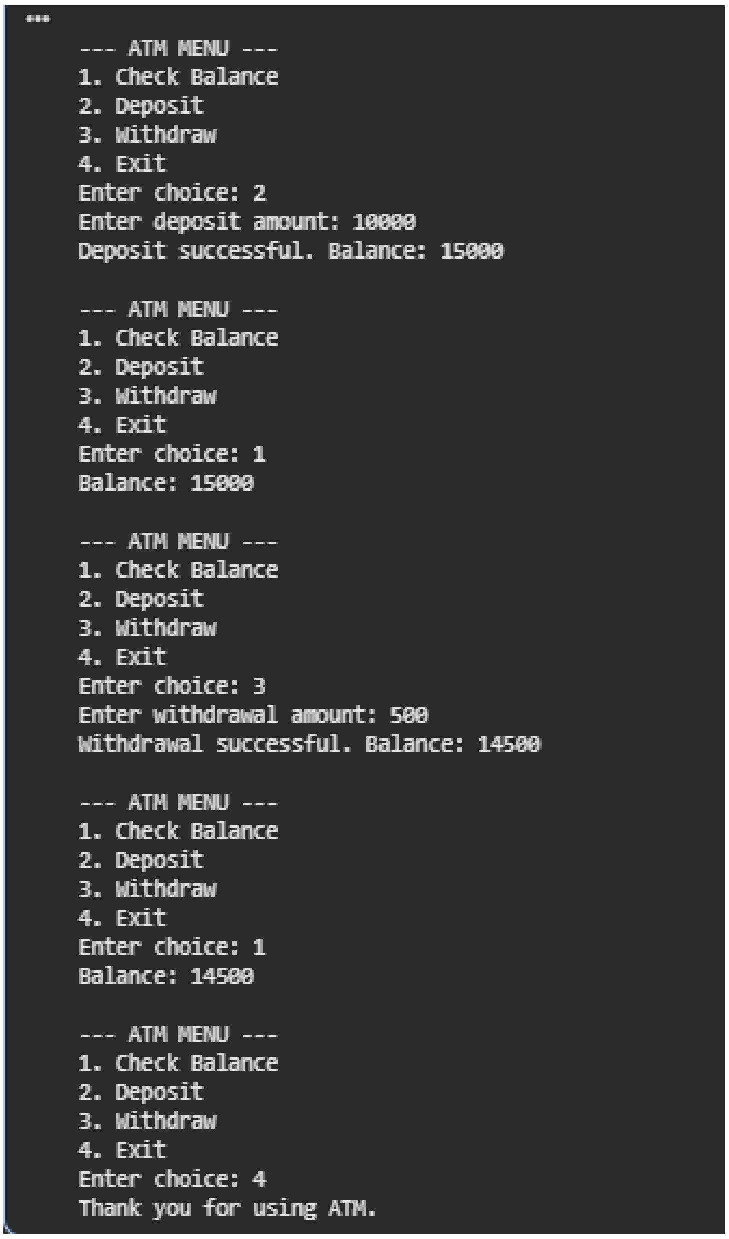
· Option 4 exits the program using break, and invalid inputs are handled properly.

Code Screenshot:



```python
balance = 5000

while True:
    print("\n--- ATM 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":
        amt = int(input("Enter deposit amount: "))
        balance += amt
        print("Deposit successful. Balance:", balance)

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

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

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

Code Output:



```
...
--- ATM MENU ---
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 2
Enter deposit amount: 10000
Deposit successful. Balance: 15000

--- ATM MENU ---
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Balance: 15000

--- ATM MENU ---
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 3
Enter withdrawal amount: 500
Withdrawal successful. Balance: 14500

--- ATM MENU ---
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Balance: 14500

--- ATM MENU ---
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 4
Thank you for using ATM.
```