# ASSIGNMENT-6.5

## HT.NO:2303A510I4

## Batch.No:30

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

## Prompt:

Generate a Python script that validates voter eligibility based on minimum age and citizenship status.

## Code:

assignment 6.5 > Task1.py > ...

```python
#2303A510I4
def check_voting_eligibility(age,is_citizen):
    if age >= 18 and is_citizen:
        return "You are eligible to vote."
    else:
        return "You are not eligible to vote."
age = int(input("Enter your age: "))
citizenship_input = input("Are you a citizen? (yes/no): ").lower()
is_citizen = citizenship_input == "yes"
result = check_voting_eligibility(age, is_citizen)
print(result)
```

## Output:

```
gnment 6.5/Task1.py"
Enter your age: 20
Are you a citizen? (yes/no): yes
You are eligible to vote.
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

## Observation:

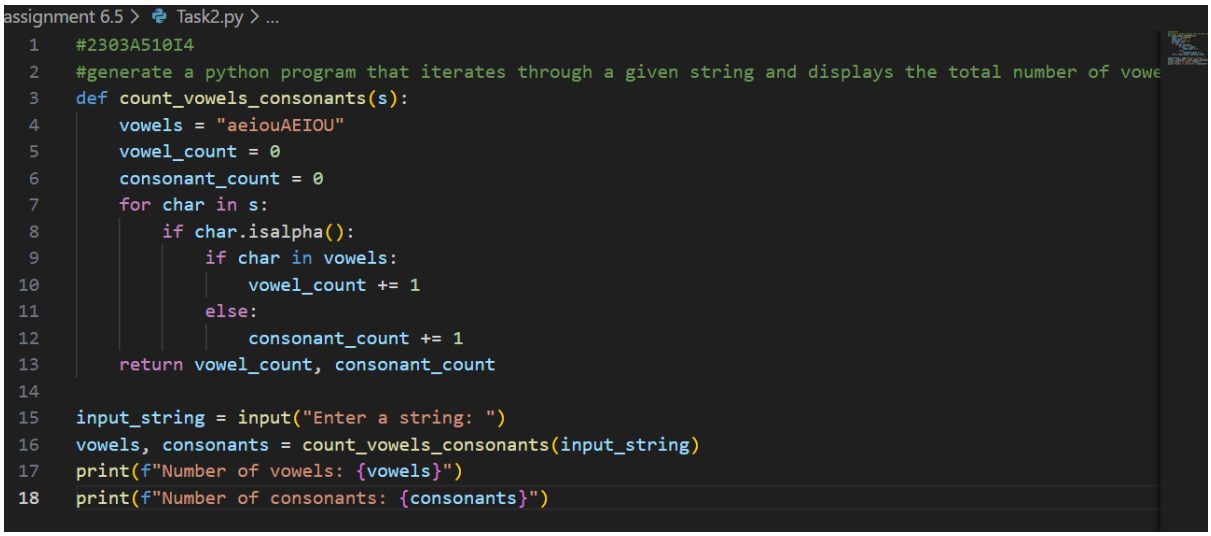
The conditional logic correctly determines voting eligibility based on age and citizenship.

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

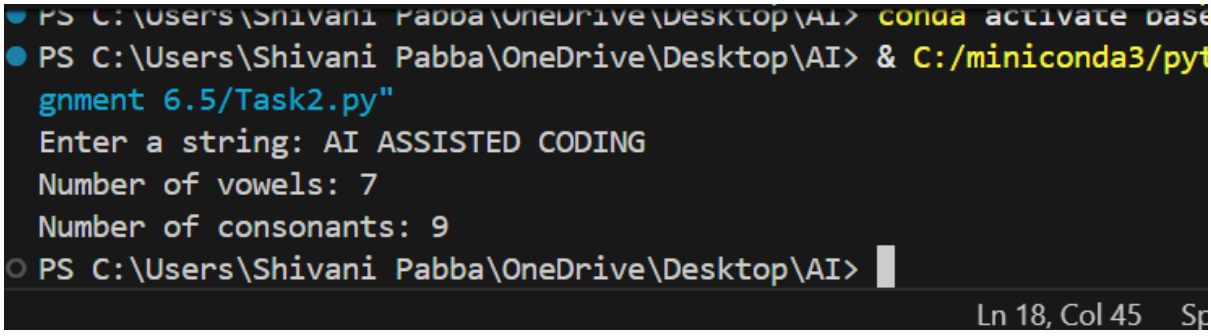
## Prompt:

Generate a Python program that iterates through a given string and displays the total number of vowels and consonants.

## Code:




## Output:




## Observation:

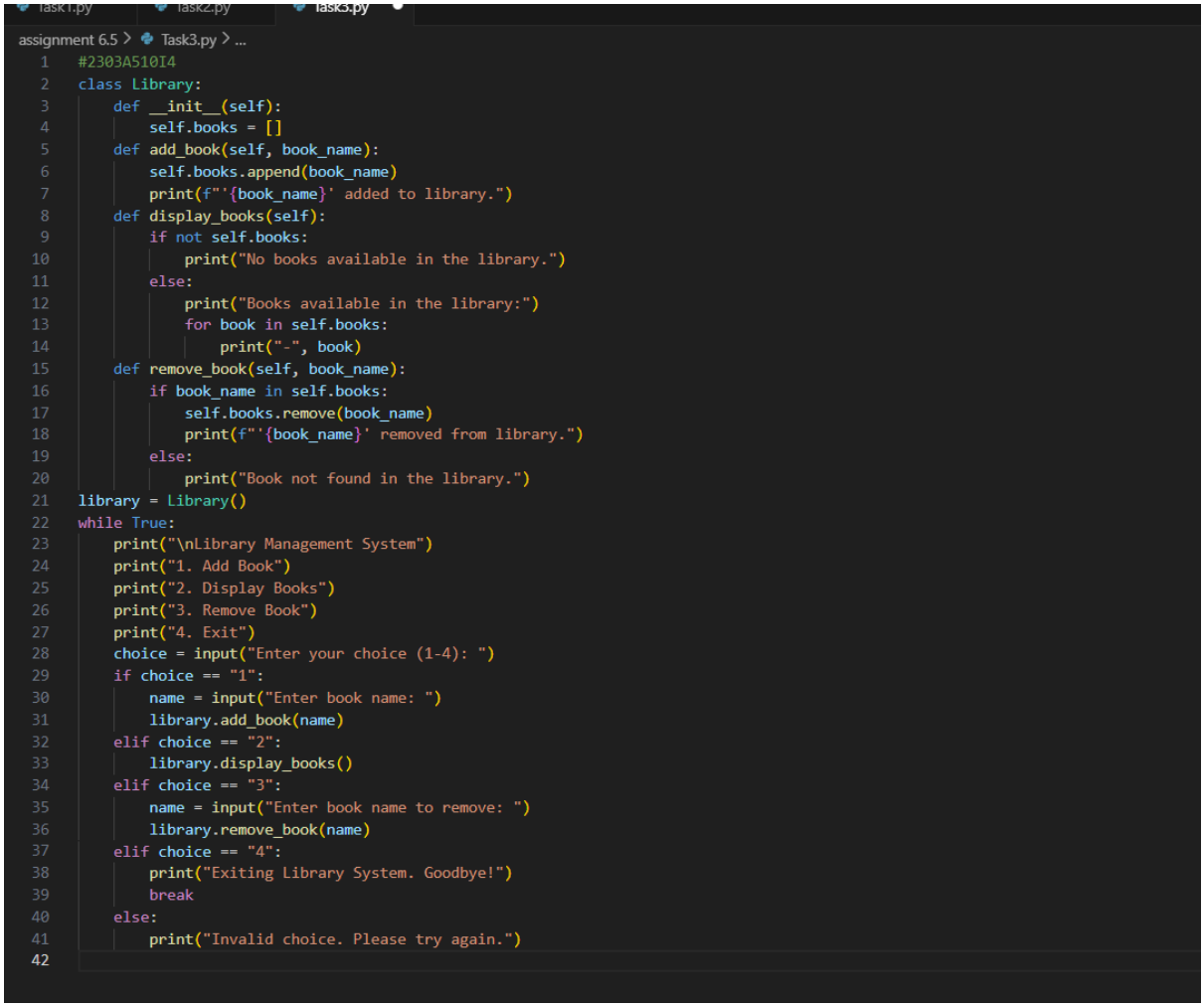
The program accurately processes the string and provides correct vowel and consonant counts.

# Task 3: Use an AI tool to generate a complete program using classes, loops, and conditionals.

## Prompt:

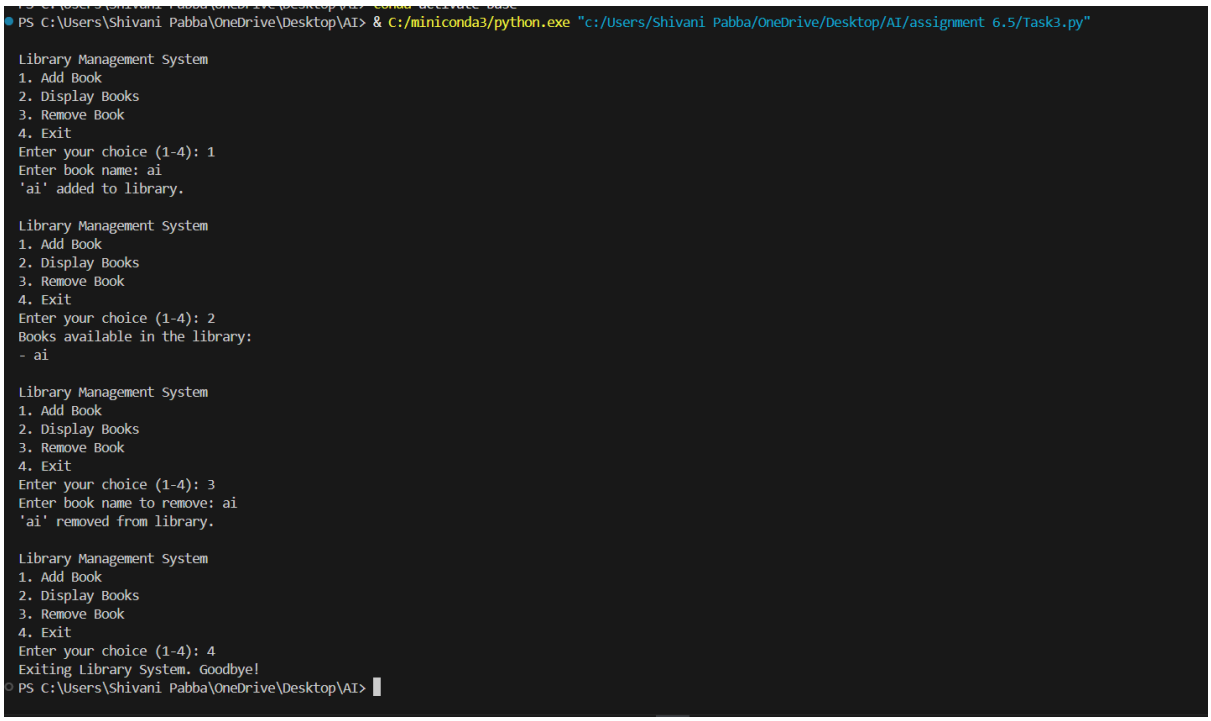
Create a complete Python program for a library management system using classes, loops, and conditional statements.

## Code:



```python
#2303A510I4
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book_name):
        self.books.append(book_name)
        print(f"'{book_name}' added to library.")
    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)
    def remove_book(self, book_name):
        if book_name in self.books:
            self.books.remove(book_name)
            print(f"'{book_name}' removed from library.")
        else:
            print("Book not found in the library.")
library = Library()
while True:
    print("\nLibrary Management System")
    print("1. Add Book")
    print("2. Display Books")
    print("3. Remove Book")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")
    if choice == "1":
        name = input("Enter book name: ")
        library.add_book(name)
    elif choice == "2":
        library.display_books()
    elif choice == "3":
        name = input("Enter book name to remove: ")
        library.remove_book(name)
    elif choice == "4":
        print("Exiting Library System. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

## Output:




## Observation:

The program uses a class to organize library operations such as adding, displaying, and removing books. A menu-driven loop with conditional statements allows continuous interaction with the user. Overall, the code clearly demonstrates the practical use of classes, loops, and conditionals.

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

## Prompt:

Generate a Python class that records and displays student attendance using loops.

## Code:

assignment 6.5 > Task4.py > ...

```python
#2303A510I4
#Generate a Python class that records and displays student attendance using loops
class Attendance:
    def __init__(self):
        self.attendance_record = {}
    def mark_attendance(self, student_name):
        self.attendance_record[student_name] = "Present"
        print(f"{student_name} marked as Present.")
    def display_attendance(self):
        if not self.attendance_record:
            print("No attendance records available.")
        else:
            print("Attendance Records:")
            for student, status in self.attendance_record.items():
                print(f"- {student}: {status}")
attendance = Attendance()
while True:
    print("\nStudent Attendance System")
    print("1. Mark Attendance")
    print("2. Display Attendance")
    print("3. Exit")
    choice = input("Enter your choice (1-3): ")
    if choice == "1":
        name = input("Enter student name: ")
        attendance.mark_attendance(name)
    elif choice == "2":
        attendance.display_attendance()
    elif choice == "3":
        print("Exiting Attendance System. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

# Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pa
gnment 6.5/Task4.py"

Student Attendance System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 1
Enter student name: siri
siri marked as Present.

Student Attendance System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 2
Attendance Records:
- siri: Present

Student Attendance System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 3
Exiting Attendance System. Goodbye!
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

## Observation:

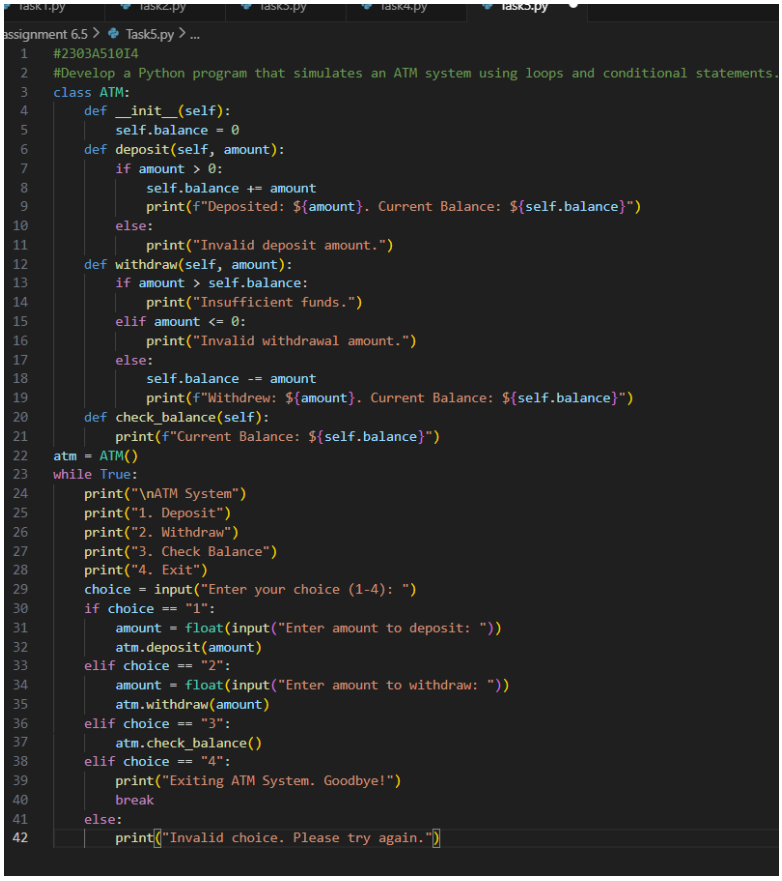
The program uses a class to store and manage student attendance records efficiently. A menu-driven loop with conditional statements allows marking and displaying attendance. The attendance details are displayed correctly using dictionary traversal.

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

## Prompt:

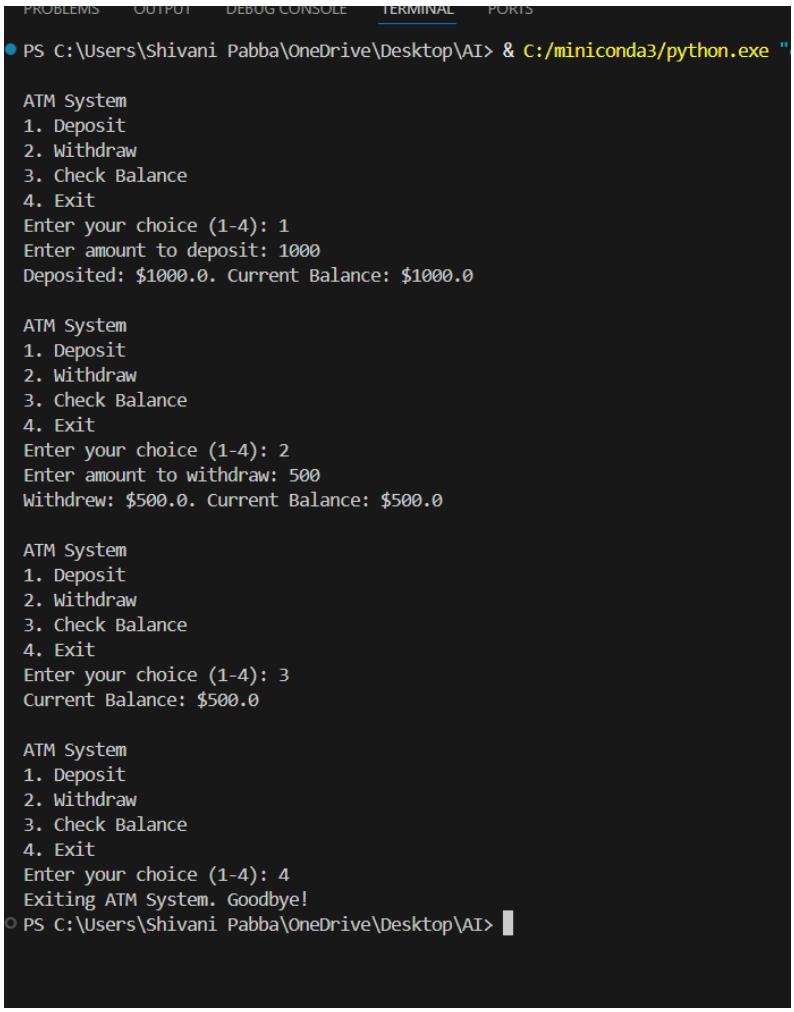
Develop a Python program that simulates an ATM system using loops and conditional statements.

## Code:



```
#2303A510I4
#Develop a Python program that simulates an ATM system using loops and conditional statements.
class ATM:
    def __init__(self):
        self.balance = 0
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited: ${amount}. Current Balance: ${self.balance}")
        else:
            print("Invalid deposit amount.")
    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds.")
        elif amount <= 0:
            print("Invalid withdrawal amount.")
        else:
            self.balance -= amount
            print(f"Withdrew: ${amount}. Current Balance: ${self.balance}")
    def check_balance(self):
        print(f"Current Balance: ${self.balance}")
atm = ATM()
while True:
    print("\nATM System")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Check Balance")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")
    if choice == "1":
        amount = float(input("Enter amount to deposit: "))
        atm.deposit(amount)
    elif choice == "2":
        amount = float(input("Enter amount to withdraw: "))
        atm.withdraw(amount)
    elif choice == "3":
        atm.check_balance()
    elif choice == "4":
        print("Exiting ATM System. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

# Output:




# Observation:

The program uses a class to manage ATM operations such as deposit, withdrawal, and balance enquiry. A menu-driven loop with conditional statements allows continuous user interaction. Input validation is handled properly to prevent invalid transactions and insufficient balance errors.