2303A51062

BATCH 29

ASSIGNMENT 6.5

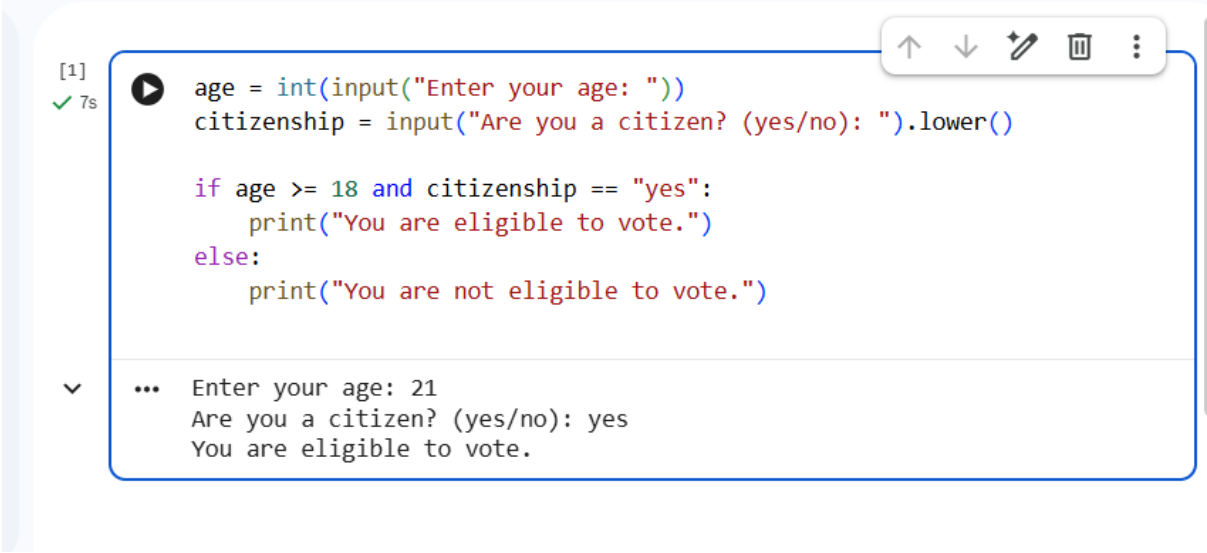
Task Description #1 (AI-Based Code Completion for Conditional Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

Prompt:

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

CODE:



JUSTIFICATION:

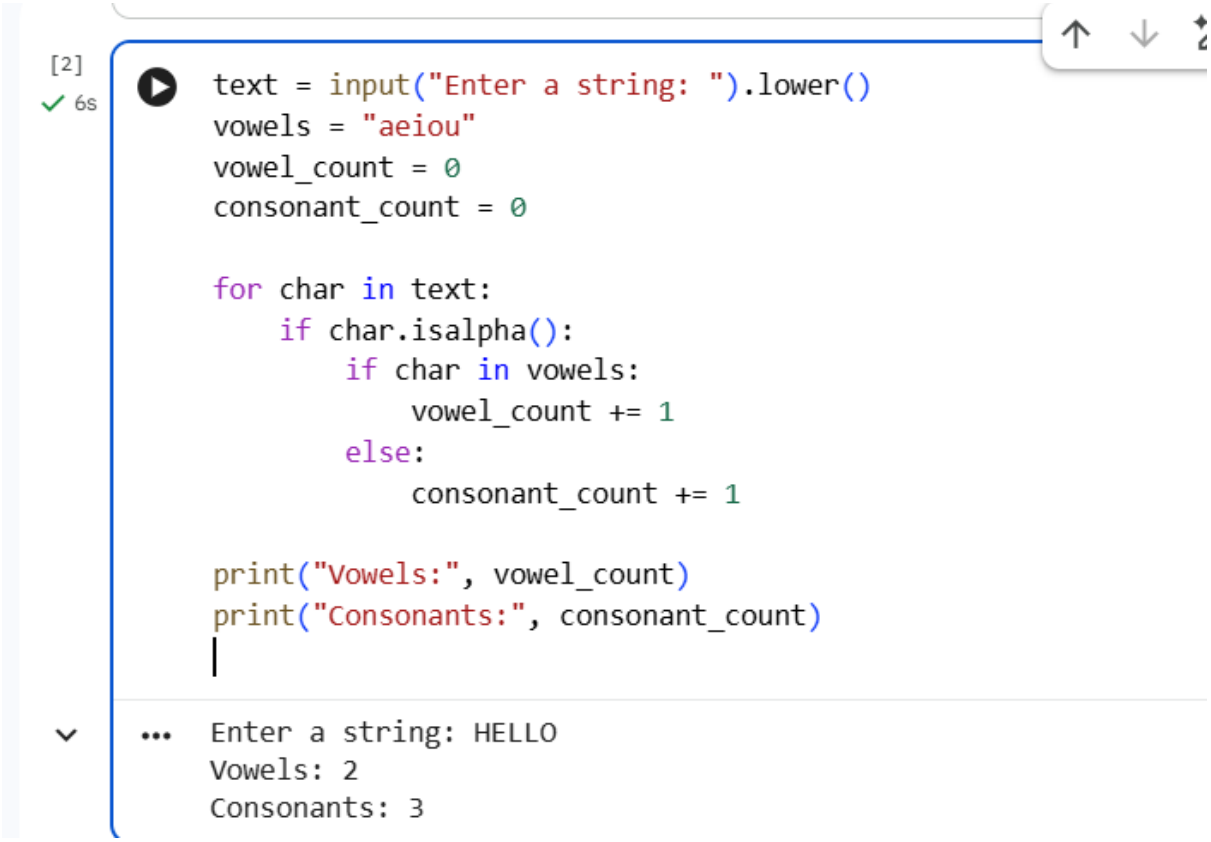
This program checks voting eligibility using conditional statements. First, it takes the user’s age and citizenship status as input. The if condition ensures that both requirements are satisfied: the person must be at least 18 years old and must be a citizen. If both conditions are true, the program prints that the user is eligible to vote; otherwise, it clearly states that the user is not eligible. This logic is simple, readable, and correctly follows real-world voting rules.

Task Description #2: Loop-Based String Processing

Prompt:

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

CODE:



```
[2]
✓ 6s
text = input("Enter a string: ").lower()
vowels = "aeiou"
vowel_count = 0
consonant_count = 0

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

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

...
Enter a string: HELLO
Vowels: 2
Consonants: 3
```

JUSTIFICATION:

This program processes a string character by character using a loop. It first converts the string to lowercase for uniform comparison. The program checks whether each character is an alphabet and then determines whether it is a vowel or a consonant. Separate counters are maintained for vowels and consonants, ensuring accurate results. This approach avoids counting spaces or special characters and demonstrates effective use of loops and conditionals.

Task Description #3: AI-Assisted Code Completion Reflection Task

Prompt :

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

CODE:

```
[3]
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book):
        self.books.append(book)
        print(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\n2. Display Books\n3. Exit")
    choice = input("Enter choice: ")

    if choice == "1":
        book_name = input("Enter book name: ")
        library.add_book(book_name)
    elif choice == "2":
        library.display_books()
    elif choice == "3":
        print("Exiting Library System.")
        break
    else:
        print("Invalid choice.")
```

JUSTIFICATION:

This AI-generated program effectively combines classes, loops, and conditional statements to simulate a simple library management system. The Library class manages book data, while a loop allows continuous user interaction. Conditional statements handle menu choices and validate user input. The AI suggestions were clear and logically structured, though the system could be optimized further by adding book removal or search

features. Overall, AI assistance helped in quickly generating a functional and readable program while still allowing room for human improvements.

Task Description #4: Class-Based Attendance System

Prompt:

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

CODE:

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

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

    def display_attendance(self):
        print("Attendance Records:")
        for name, status in self.records.items():
            print(name, ":", status)


attendance = Attendance()

attendance.mark_attendance("Alice", "Present")
attendance.mark_attendance("Bob", "Absent")
attendance.mark_attendance("Charlie", "Present")

attendance.display_attendance()
```

```
Attendance Records:
Alice : Present
Bob : Absent
Charlie : Present
```

JUSTIFICATION:

This attendance management system uses a class to store and manage student attendance records. The mark_attendance method records each

student's attendance status, while the display_attendance method uses a loop to print all records clearly. This structure keeps the code organized and easy to maintain. The AI-generated logic is efficient and demonstrates practical usage of classes and loops in real-world academic systems.

Task Description #5: Conditional Menu Navigation (ATM Menu)

Prompt:

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

CODE:

[5]

```
balance = 10000
while True:
    print("\nATM Menu")
    print("1. Check Balance")
    print("2. Deposit")
    print("3. Withdraw")
    print("4. Exit")
    choice = input("Enter your choice: ")
    if choice == "1":
        print("Current Balance:", 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 the ATM.")
        break
    else:
        print("Invalid option. Try again.")
```

JUSTIFICATION:

This ATM simulation program uses a loop to continuously display menu options until the user exits. Conditional statements handle different banking operations such as checking balance, depositing money, and withdrawing funds. A balance check prevents invalid withdrawals, ensuring logical correctness. The AI-generated code is user-friendly, efficient, and accurately represents a real ATM system, making it a strong example of conditional menu navigation.