ASSIGNMENT-06

Name: Rishitha Kunchala

HT.No: 2303A51333

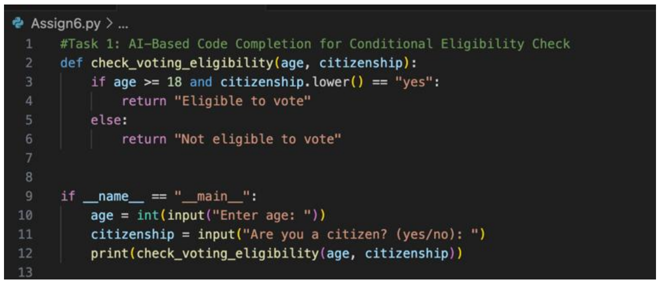
Batch: 20

Task 1: AI-Based Code Completion for Conditional Eligibility Check

The task is to generate Python code that checks voting eligibility based on a person's age and citizenship status using conditional statements.

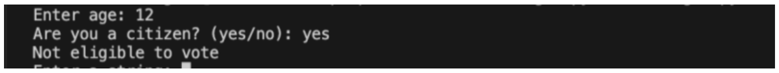
Prompt: # Generate Python code to check voting eligibility based on age and citizenship.

Code:

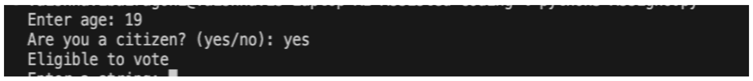


```
Assign6.py > ...
#Task 1: AI-Based Code Completion for Conditional Eligibility Check
def check_voting_eligibility(age, citizenship):
    if age >= 18 and citizenship.lower() == "yes":
        return "Eligible to vote"
    else:
        return "Not eligible to vote"


if __name__ == "__main__":
    age = int(input("Enter age: "))
    citizenship = input("Are you a citizen? (yes/no): ")
    print(check_voting_eligibility(age, citizenship))
```

Result:



```
Enter age: 12
Are you a citizen? (yes/no): yes
Not eligible to vote
```



```
Enter age: 19
Are you a citizen? (yes/no): yes
Eligible to vote
```

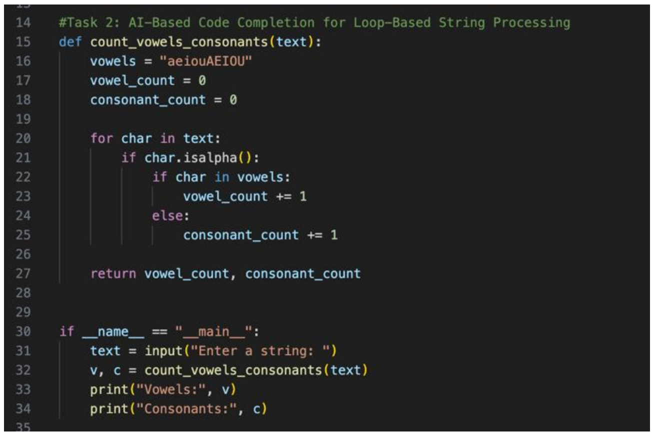
Observation:
The AI-generated code correctly applies conditional logic using logical operators. The conditions are clearly defined and easy to understand. Minor optimization was done by handling case sensitivity using the `lower()` method. The logic is accurate and suitable for real-world eligibility checks.

Task 2: AI-Based Code Completion for Loop-Based String Processing
The task is to generate Python code that counts vowels and consonants in a given string using loops.

Prompt: # Generate Python code to count vowels

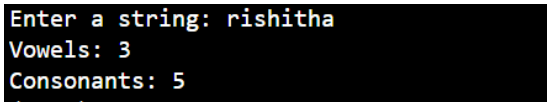
and consonants in a string using a loop.

Code:



```python
#Task 2: AI-Based Code Completion for Loop-Based String Processing
def count_vowels_consonants(text):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

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

    return vowel_count, consonant_count


if __name__ == "__main__":
    text = input("Enter a string: ")
    v, c = count_vowels_consonants(text)
    print("Vowels:", v)
    print("Consonants:", c)
```

Result:



```
Enter a string: rishitha
Vowels: 3
Consonants: 5
```

Observation:
The AI-generated loop efficiently iterates through each character and uses conditional checks to classify vowels and consonants. The use of `isalpha()` avoids counting spaces and symbols. The logic is correct, readable, and optimized for accuracy.

Task 3: AI-Assisted Code Completion Reflection Task

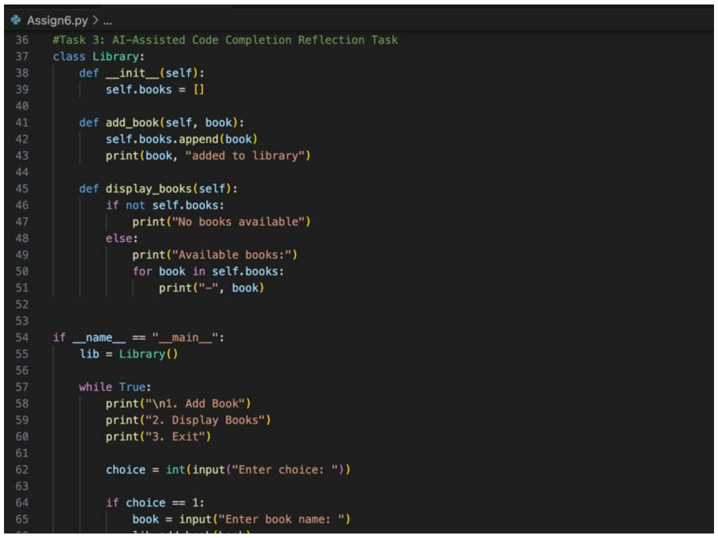
The task is to generate a complete Python program using classes, loops, and conditionals for

a library management system.

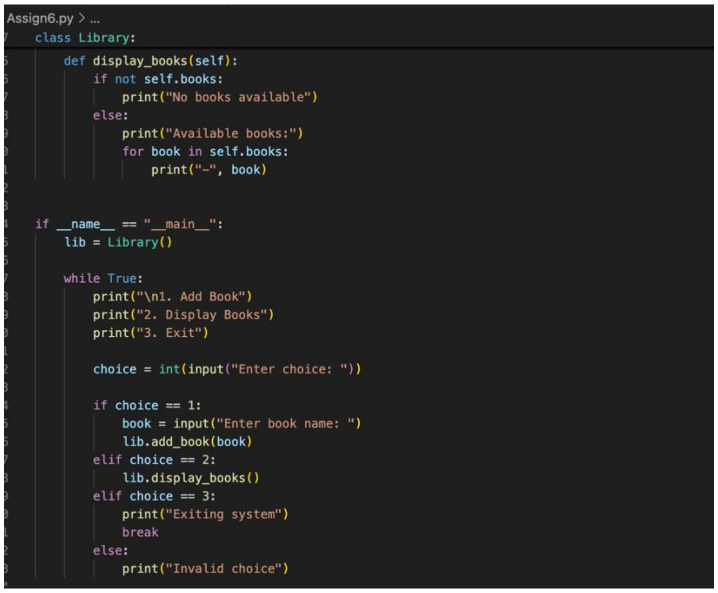
Prompt: # Generate a Python program for a library management system using classes, loops, and

conditional statements.

Code:

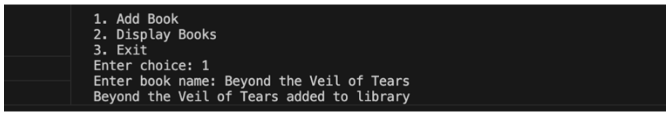


```
Assign6.py > ...
#Task 3: AI-Assisted Code Completion Reflection Task
class Library:
    def __init__(self):
        self.books = []

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


if __name__ == "__main__":
    lib = Library()

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

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

        if choice == 1:
            book = input("Enter book name: ")
```

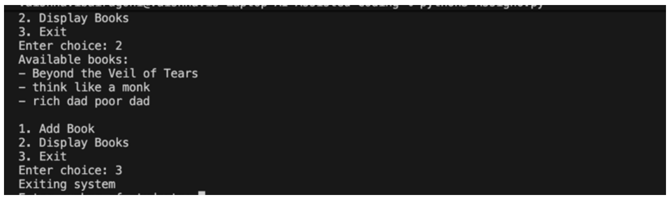


```
Assign6.py > ...
class Library:
    def display_books(self):
        if not self.books:
            print("No books available")
        else:
            print("Available books:")
            for book in self.books:
                print("-", book)


if __name__ == "__main__":
    lib = Library()

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

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

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

Result:



```
1. Add Book
2. Display Books
3. Exit
Enter choice: 1
Enter book name: Beyond the Veil of Tears
Beyond the Veil of Tears added to library
```



```
2. Display Books
3. Exit
Enter choice: 2
Available books:
- Beyond the Veil of Tears
- think like a monk
- rich dad poor dad

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

Observation:

The program allows adding books, displaying all books, and exiting the system through a menu-driven interface.

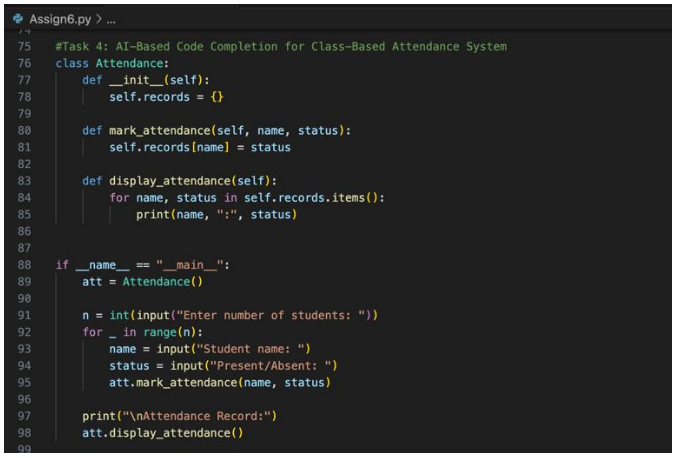
The AI-generated program effectively combines classes, loops, and conditionals. The logic is simple and functional. The menu-driven loop ensures continuous interaction. The structure is readable, and the program reflects a basic real-world library system.

Task 4: AI-Based Code Completion for Class-Based Attendance System

The task is to generate a Python class that marks and displays student attendance using loops.

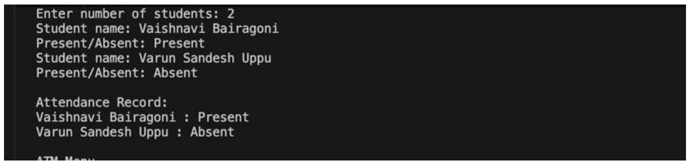
Prompt: #Generate a Python class to mark and display student attendance using loops.

Code:



```python
#Task 4: AI-Based Code Completion for Class-Based Attendance System
class Attendance:
    def __init__(self):
        self.records = {}

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

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


if __name__ == "__main__":
    att = Attendance()

    n = int(input("Enter number of students: "))
    for _ in range(n):
        name = input("Student name: ")
        status = input("Present/Absent: ")
        att.mark_attendance(name, status)

    print("\nAttendance Record:")
    att.display_attendance()
```

Result:



```
Enter number of students: 2
Student name: Vaishnavi Bairagoni
Present/Absent: Present
Student name: Varun Sandesh Uppu
Present/Absent: Absent

Attendance Record:
Vaishnavi Bairagoni : Present
Varun Sandesh Uppu : Absent

ATM Menu
```

Observation

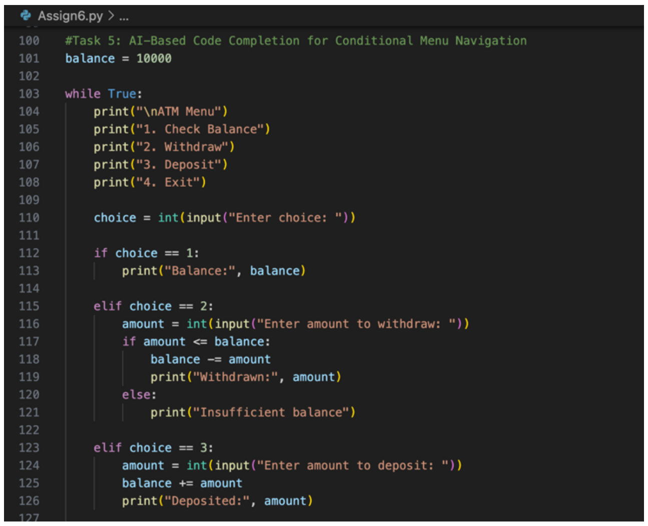
The program correctly stores and displays attendance for all students entered by the user. The AI-generated class uses a dictionary for efficient storage. Loop-based input ensures scalability. The design is simple, logical, and easy to extend for real attendance systems.

Task 5: AI-Based Code Completion for Conditional Menu Navigation

The task is to generate a Python program using loops and conditionals to simulate an ATM menu

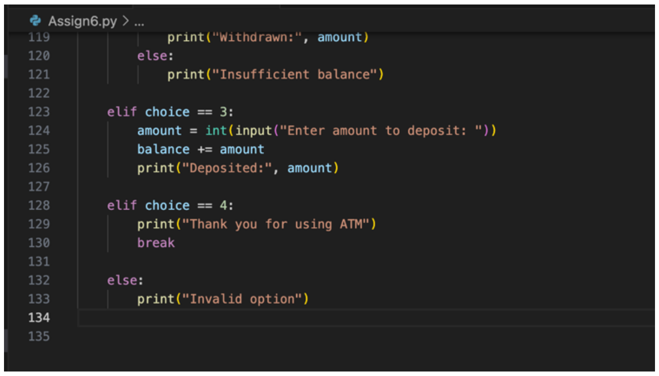
Prompt: #Generate a Python program using loops and conditionals to simulate an ATM menu.

Code:



Assign6.py > ...

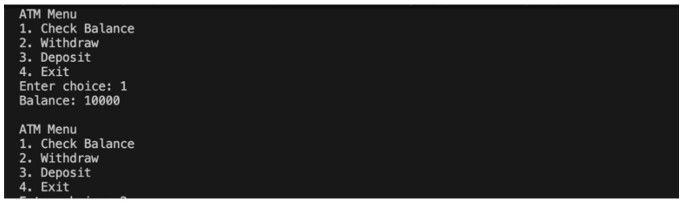
```python
#Task 5: AI-Based Code Completion for Conditional Menu Navigation
balance = 10000

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

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

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

    elif choice == 2:
        amount = int(input("Enter amount to withdraw: "))
        if amount <= balance:
            balance -= amount
            print("Withdrawn:", amount)
        else:
            print("Insufficient balance")

    elif choice == 3:
        amount = int(input("Enter amount to deposit: "))
        balance += amount
        print("Deposited:", amount)
```



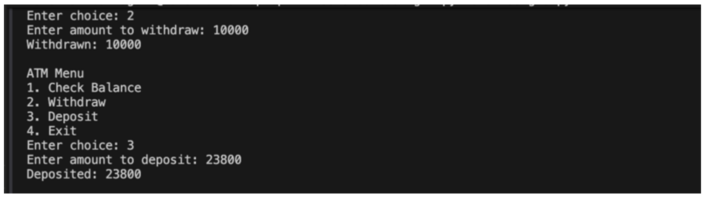
Assign6.py > ...

```python
            print("Withdrawn:", amount)
        else:
            print("Insufficient balance")

    elif choice == 3:
        amount = int(input("Enter amount to deposit: "))
        balance += amount
        print("Deposited:", amount)

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

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

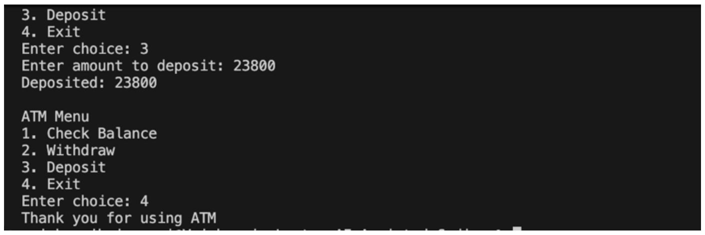
Result:



```
ATM Menu
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
Enter choice: 1
Balance: 10000

ATM Menu
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
```



```
Enter choice: 2
Enter amount to withdraw: 10000
Withdrawn: 10000

ATM Menu
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
Enter choice: 3
Enter amount to deposit: 23800
Deposited: 23800
```



```
3. Deposit
4. Exit
Enter choice: 3
Enter amount to deposit: 23800
Deposited: 23800

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

Observation:
The ATM menu allows checking balance, withdrawing, depositing, and exiting correctly.
The AI-generated logic correctly uses loops for repeated menu display and conditionals for option handling. The program is efficient and user-friendly. Minor improvements such as input validation could further enhance robustness