Lab Assignment-6.5

A.Viveksai

2303A51710

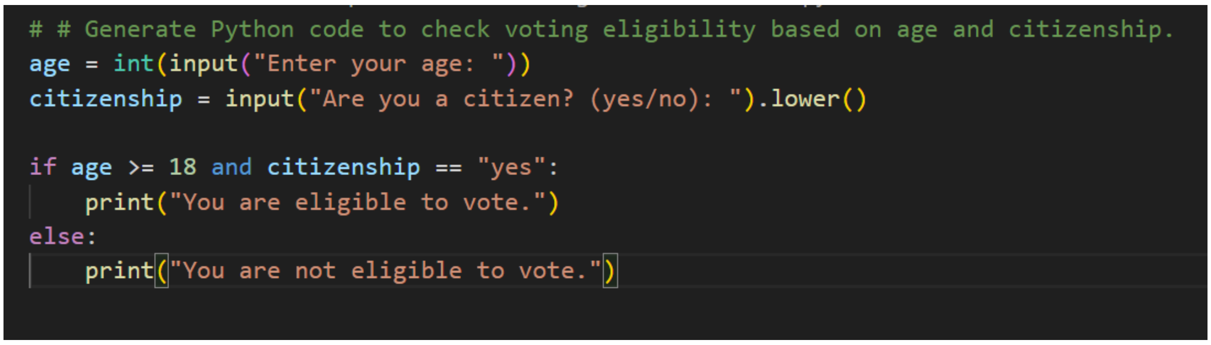
Batch-11

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

Task: Use an AI tool to generate eligibility logic.

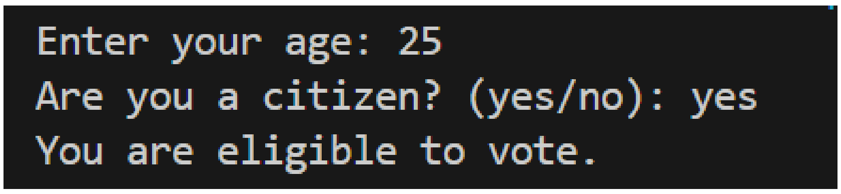
Prompt: Generate Python code to check voting eligibility based on age and citizenship.

Code:

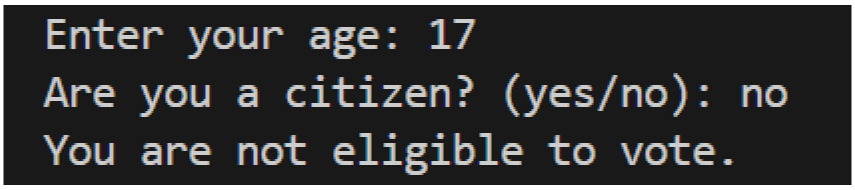


```python
# # Generate Python code to check voting eligibility based on age and citizenship.
age = int(input("Enter your age: "))
citizenship = input("Are you a citizen? (yes/no): ").lower()

if age >= 18 and citizenship == "yes":
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

Output:



```
Enter your age: 25
Are you a citizen? (yes/no): yes
You are eligible to vote.
```



```
Enter your age: 17
Are you a citizen? (yes/no): no
You are not eligible to vote.
```

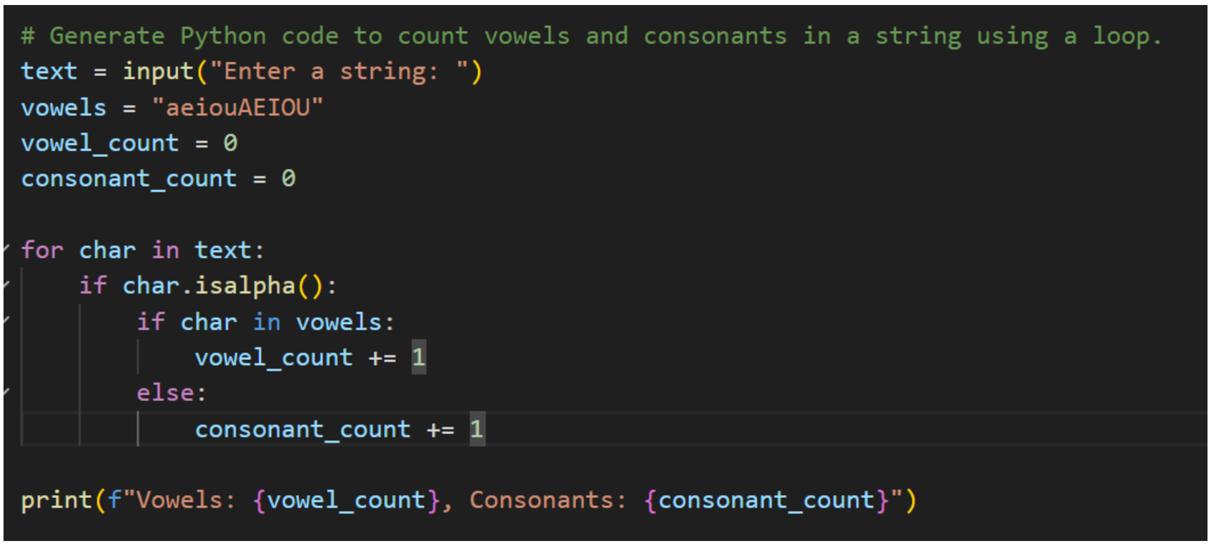
Explaination: This code checks if a person is eligible to vote. It first asks the user to enter their age and whether they are a citizen. If the user is 18 or older and answers "yes" to being a citizen, the program prints that they are eligible to vote. Otherwise, it tells them they are not eligible. The code uses simple input prompts and a conditional statement to make the decision.

Task Description #2(AI-Based Code Completion for Loop-Based String Processing)

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

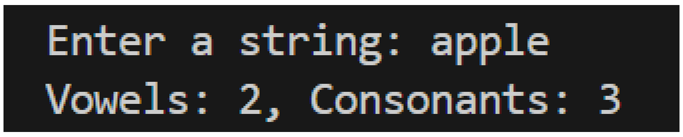
Prompt: Generate Python code to count vowels and consonants in a string using a loop.

Code:

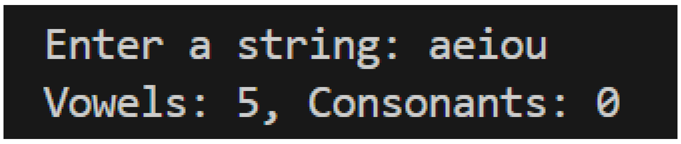


```python
# Generate Python code to count vowels and consonants in a string using a loop.
text = input("Enter a string: ")
vowels = "aeiouAEIOU"
vowel_count = 0
consonant_count = 0

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

print(f"Vowels: {vowel_count}, Consonants: {consonant_count}")
```

Output:



```
Enter a string: apple
Vowels: 2, Consonants: 3
```



```
Enter a string: aeiou
Vowels: 5, Consonants: 0
```

Explaination: This code counts the number of vowels and consonants in a user-provided string. It first asks the user to enter any text. Then, for each character in the string, it checks if the character is a letter. If it is, the code determines whether it is a vowel or a consonant and updates the respective count. Finally, it prints out the total number of vowels and consonants found in the input.

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

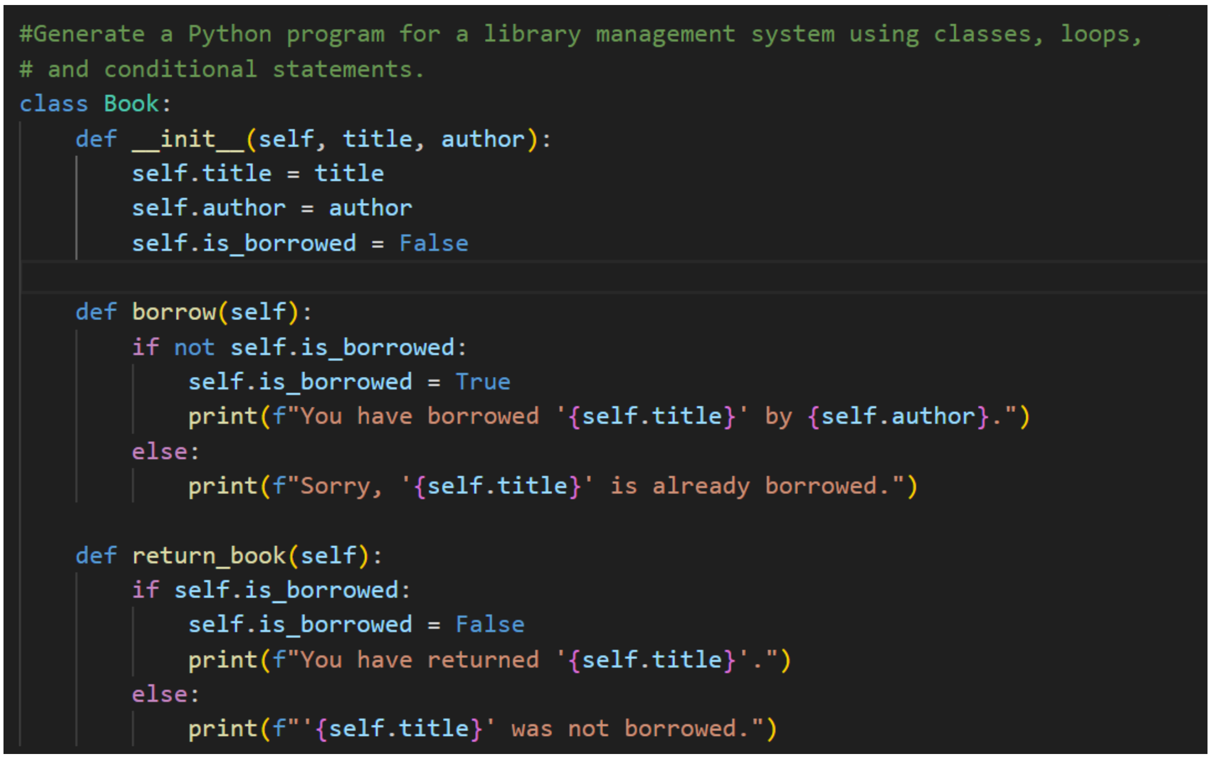
Task: Use an AI tool to generate a complete program using classes,

loops, and conditionals.

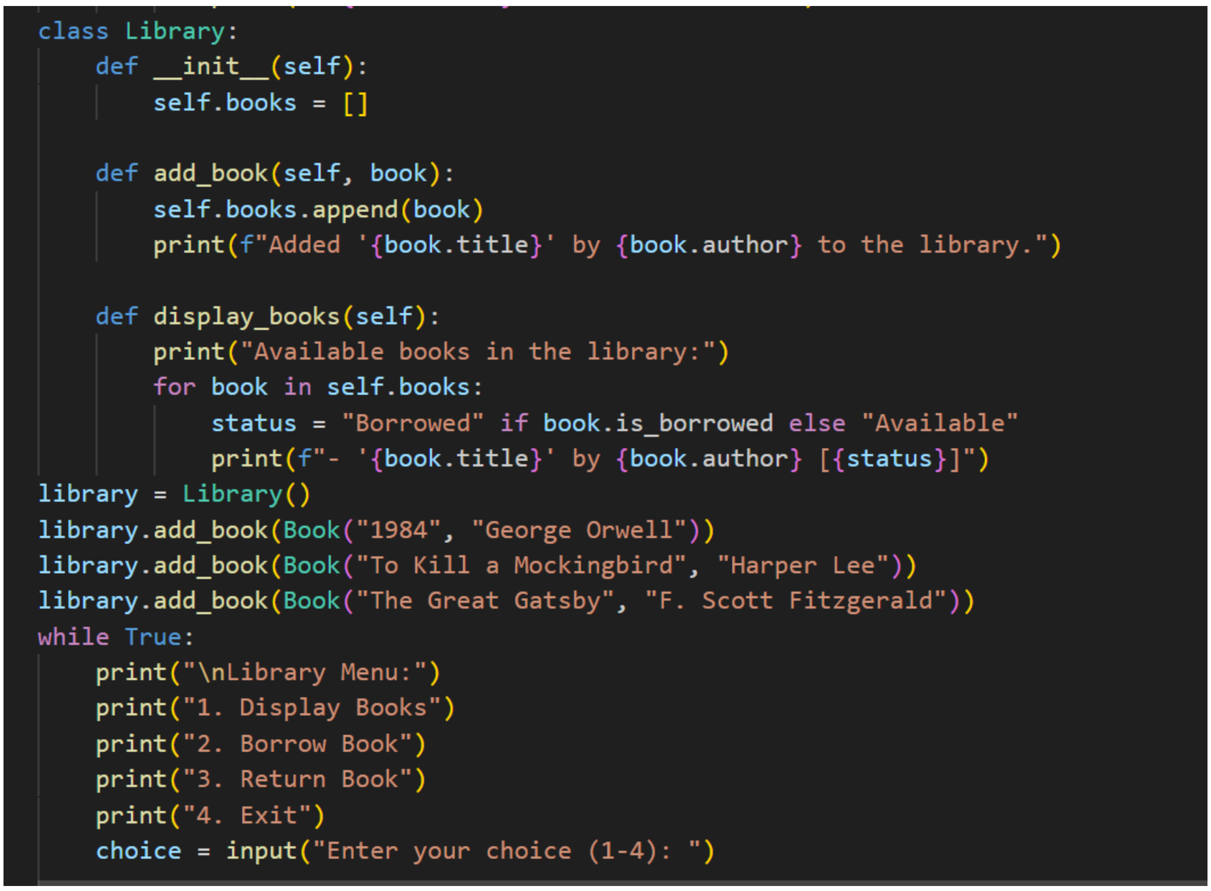
Prompt: Generate a Python program for a library management system

using classes, loops, and conditional statements.

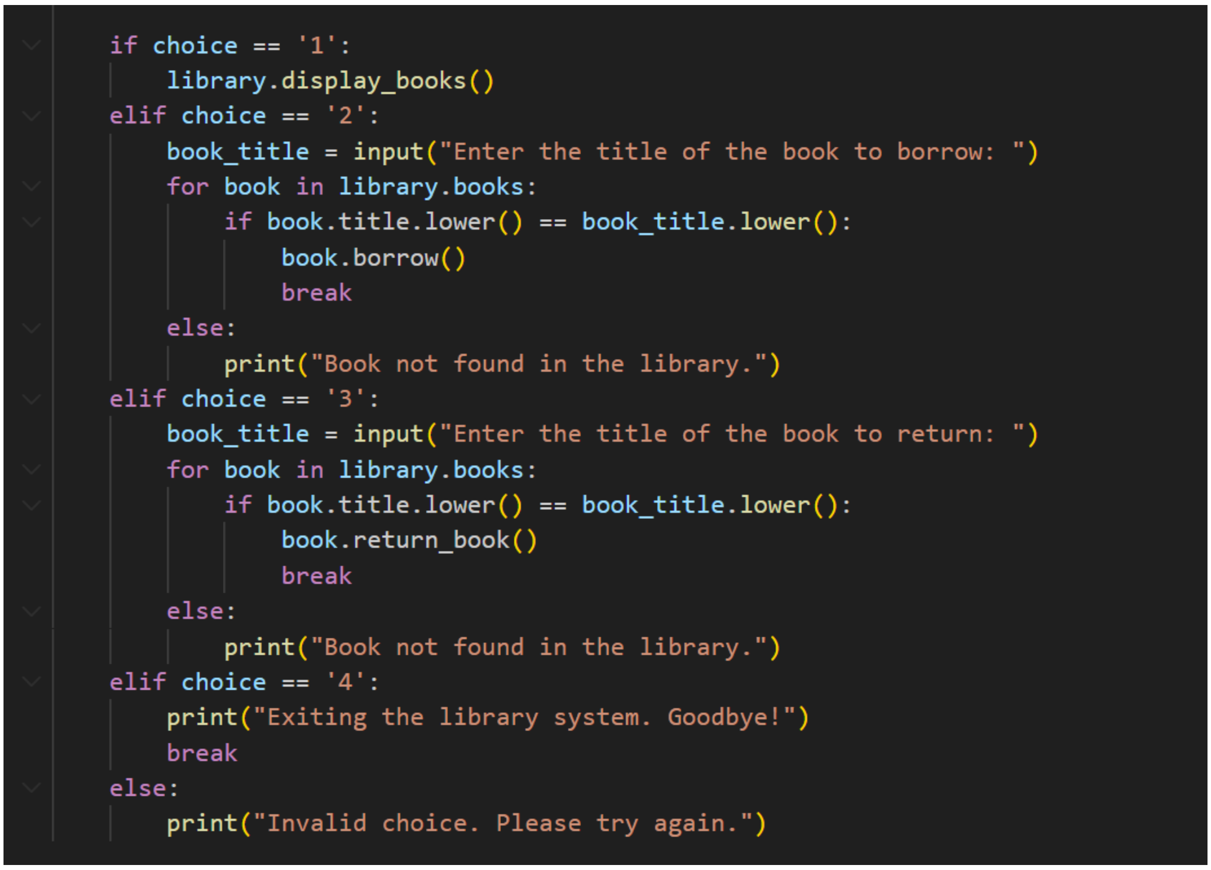
Code:



```python
#Generate a Python program for a library management system using classes, loops,
# and conditional statements.
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_borrowed = False

    def borrow(self):
        if not self.is_borrowed:
            self.is_borrowed = True
            print(f"You have borrowed '{self.title}' by {self.author}.")
        else:
            print(f"Sorry, '{self.title}' is already borrowed.")

    def return_book(self):
        if self.is_borrowed:
            self.is_borrowed = False
            print(f"You have returned '{self.title}'.")
        else:
            print(f"'{self.title}' was not borrowed.")
```

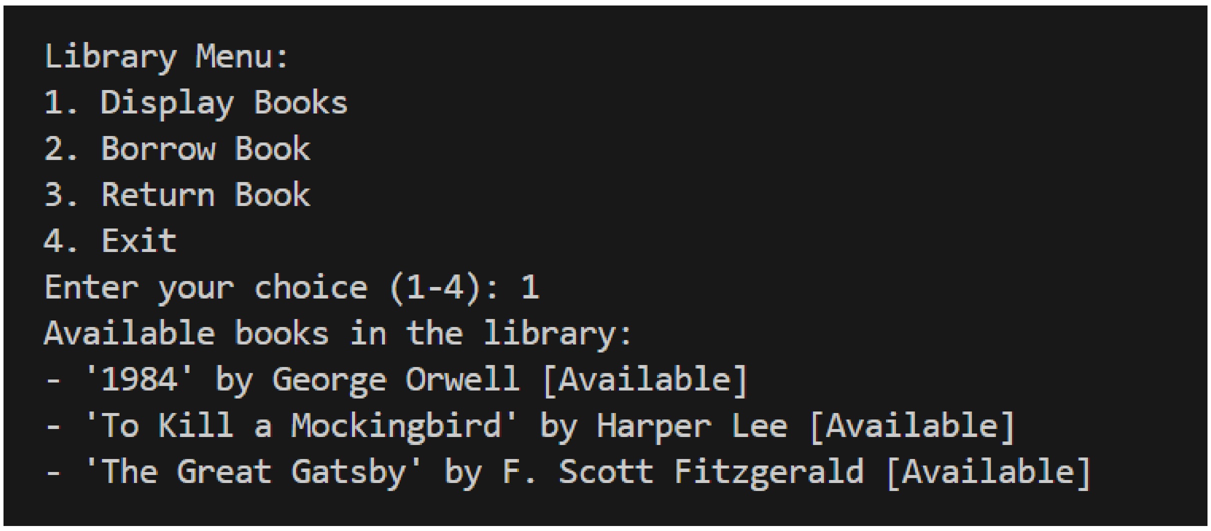


```python
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)
        print(f"Added '{book.title}' by {book.author} to the library.")

    def display_books(self):
        print("Available books in the library:")
        for book in self.books:
            status = "Borrowed" if book.is_borrowed else "Available"
            print(f"- '{book.title}' by {book.author} [{status}]")
library = Library()
library.add_book(Book("1984", "George Orwell"))
library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
while True:
    print("\nLibrary Menu:")
    print("1. Display Books")
    print("2. Borrow Book")
    print("3. Return Book")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")
```



```python
    if choice == '1':
        library.display_books()
    elif choice == '2':
        book_title = input("Enter the title of the book to borrow: ")
        for book in library.books:
            if book.title.lower() == book_title.lower():
                book.borrow()
                break
        else:
            print("Book not found in the library.")
    elif choice == '3':
        book_title = input("Enter the title of the book to return: ")
        for book in library.books:
            if book.title.lower() == book_title.lower():
                book.return_book()
                break
        else:
            print("Book not found in the library.")
    elif choice == '4':
        print("Exiting the library system. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:



```
Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice (1-4): 1
Available books in the library:
- '1984' by George Orwell [Available]
- 'To Kill a Mockingbird' by Harper Lee [Available]
- 'The Great Gatsby' by F. Scott Fitzgerald [Available]
```

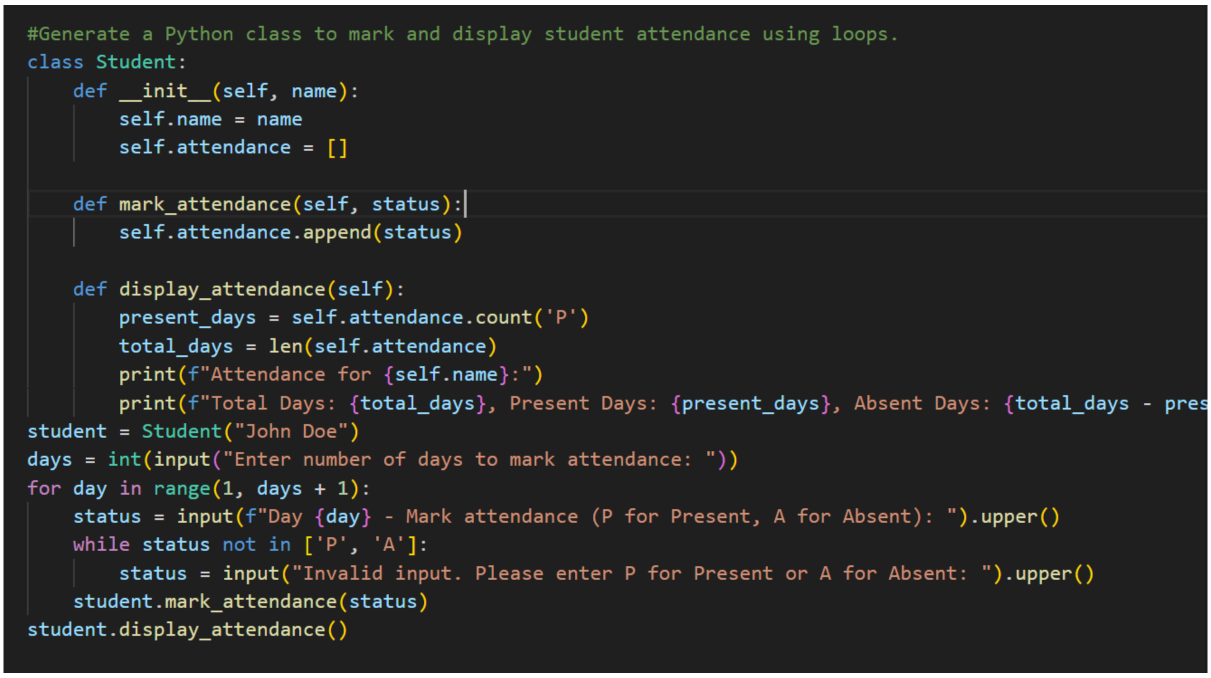
Explaination: This section uses classes to represent books and a library. Users can add books, display available books, borrow, and return books through a menu-driven loop.
Each part uses input prompts, conditionals, and loops to interact with the user and process their responses. The code demonstrates basic Python programming concepts in practical scenarios.

Task Description #4 (AI-Assisted Code Completion for Class- Based Attendance System)

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

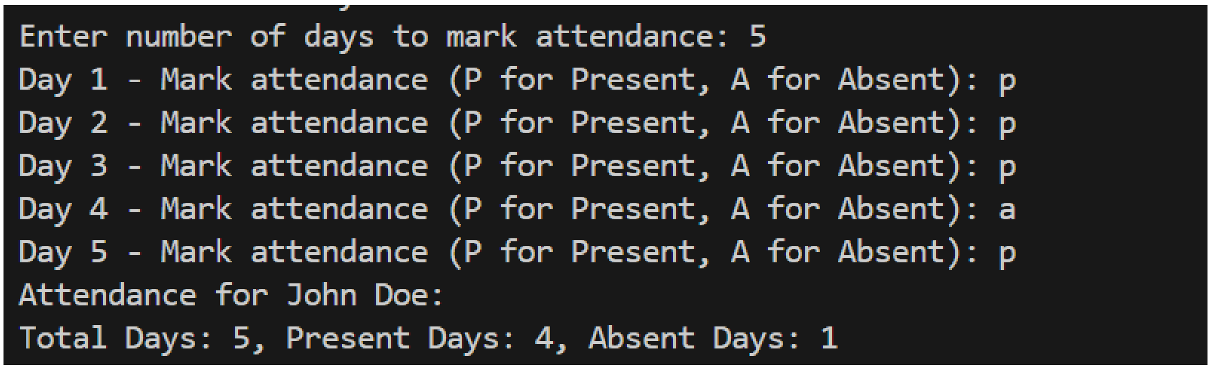
Prompt: Generate a Python class to mark and display student attendance using loops.

Code:



```python
#Generate a Python class to mark and display student attendance using loops.
class Student:
    def __init__(self, name):
        self.name = name
        self.attendance = []

    def mark_attendance(self, status):
        self.attendance.append(status)

    def display_attendance(self):
        present_days = self.attendance.count('P')
        total_days = len(self.attendance)
        print(f"Attendance for {self.name}:")
        print(f"Total Days: {total_days}, Present Days: {present_days}, Absent Days: {total_days - pres
student = Student("John Doe")
days = int(input("Enter number of days to mark attendance: "))
for day in range(1, days + 1):
    status = input(f"Day {day} - Mark attendance (P for Present, A for Absent): ").upper()
    while status not in ['P', 'A']:
        status = input("Invalid input. Please enter P for Present or A for Absent: ").upper()
    student.mark_attendance(status)
student.display_attendance()
```

Output:



```
Enter number of days to mark attendance: 5
Day 1 - Mark attendance (P for Present, A for Absent): p
Day 2 - Mark attendance (P for Present, A for Absent): p
Day 3 - Mark attendance (P for Present, A for Absent): p
Day 4 - Mark attendance (P for Present, A for Absent): a
Day 5 - Mark attendance (P for Present, A for Absent): p
Attendance for John Doe:
Total Days: 5, Present Days: 4, Absent Days: 1
```

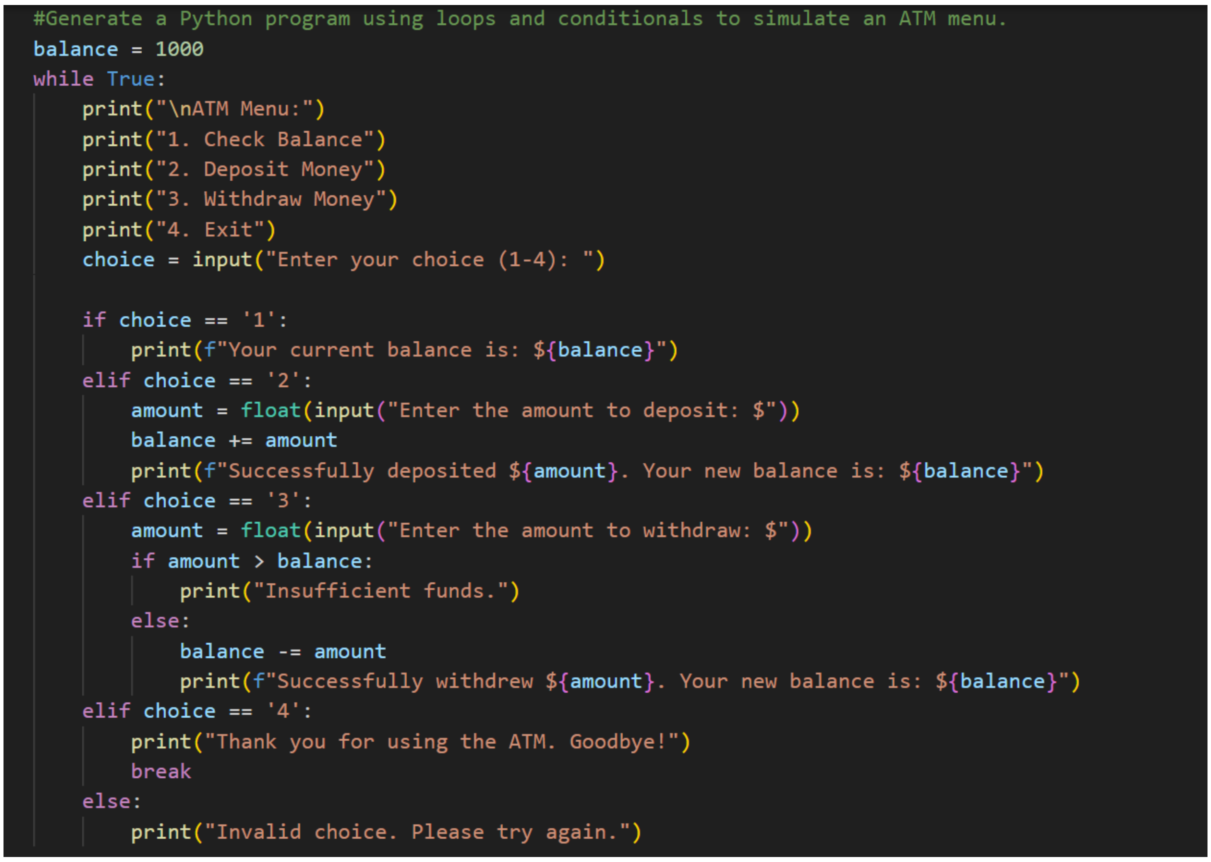
Explaination: This code defines a simple student attendance tracker. It creates a Student class that stores the student's name and a list of their attendance records. The user is asked how many days to mark attendance, then for each day, they enter 'P' for present or 'A' for absent. The program ensures only valid inputs are accepted. After all days are marked, it displays the total days, number of presents, and absents for the student.

Task Description #5 (AI-Based Code Completion for Conditional Menu Navigation)

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

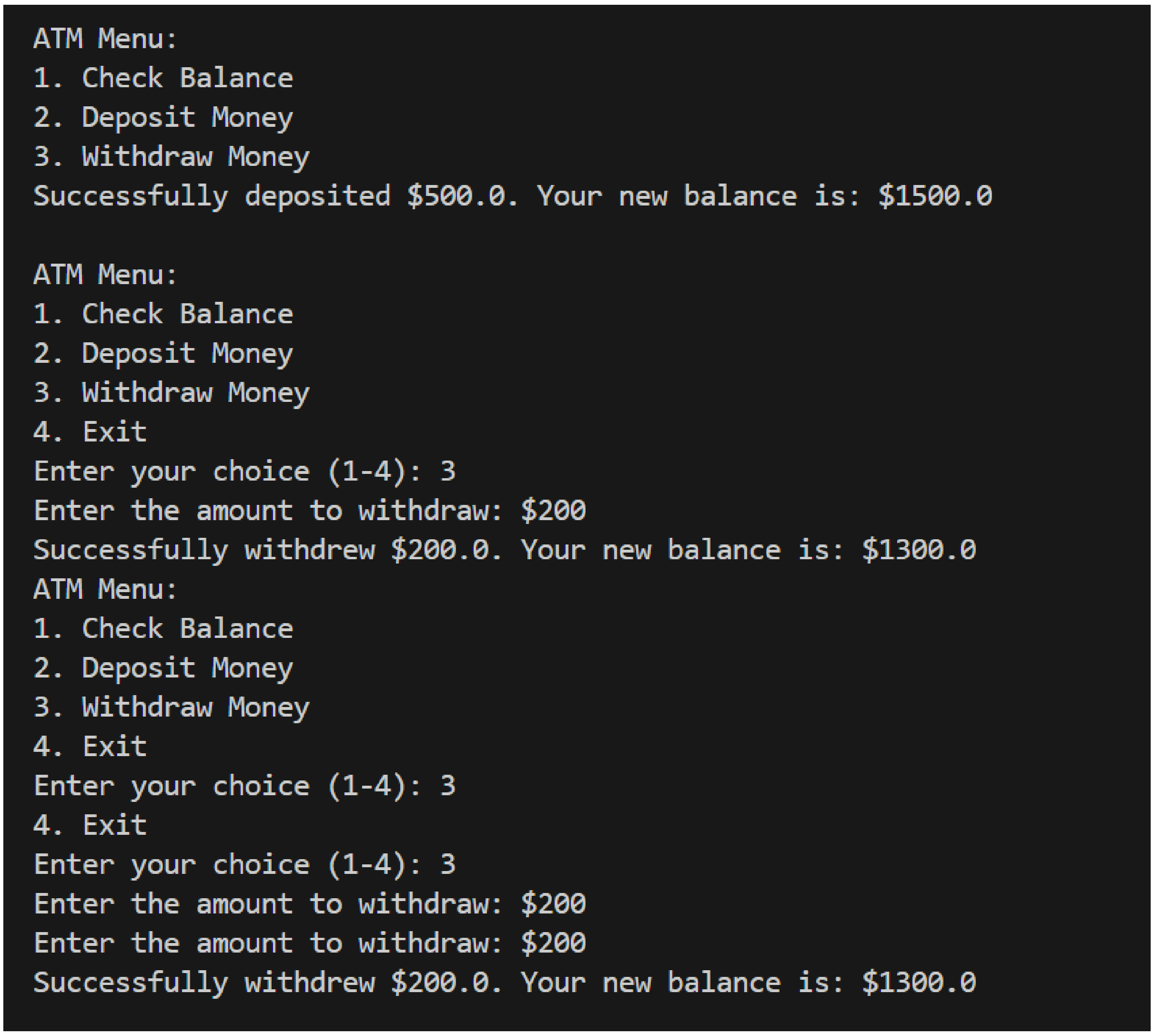
Prompt: Generate a Python program using loops and conditionals to simulate an ATM menu.

Code:



```python
#Generate a Python program using loops and conditionals to simulate an ATM menu.
balance = 1000
while True:
    print("\nATM Menu:")
    print("1. Check Balance")
    print("2. Deposit Money")
    print("3. Withdraw Money")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")

    if choice == '1':
        print(f"Your current balance is: ${balance}")
    elif choice == '2':
        amount = float(input("Enter the amount to deposit: $"))
        balance += amount
        print(f"Successfully deposited ${amount}. Your new balance is: ${balance}")
    elif choice == '3':
        amount = float(input("Enter the amount to withdraw: $"))
        if amount > balance:
            print("Insufficient funds.")
        else:
            balance -= amount
            print(f"Successfully withdrew ${amount}. Your new balance is: ${balance}")
    elif choice == '4':
        print("Thank you for using the ATM. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:



```
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
Successfully deposited $500.0. Your new balance is: $1500.0

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 3
Enter the amount to withdraw: $200
Successfully withdrew $200.0. Your new balance is: $1300.0
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 3
4. Exit
Enter your choice (1-4): 3
Enter the amount to withdraw: $200
Enter the amount to withdraw: $200
Successfully withdrew $200.0. Your new balance is: $1300.0
```

Explaination: This code simulates a simple ATM system. It starts with a balance of $1000 and repeatedly shows a menu with options to check balance, deposit money, withdraw money, or exit. The user selects an option by entering a number. Deposits increase the balance, and withdrawals decrease it if there are enough funds; otherwise, it warns about insufficient funds. The loop continues until the user chooses to exit.