Lab Assignment: 6.5

T HRUSHIKESH

2303A51788

Batch – 12

Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

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

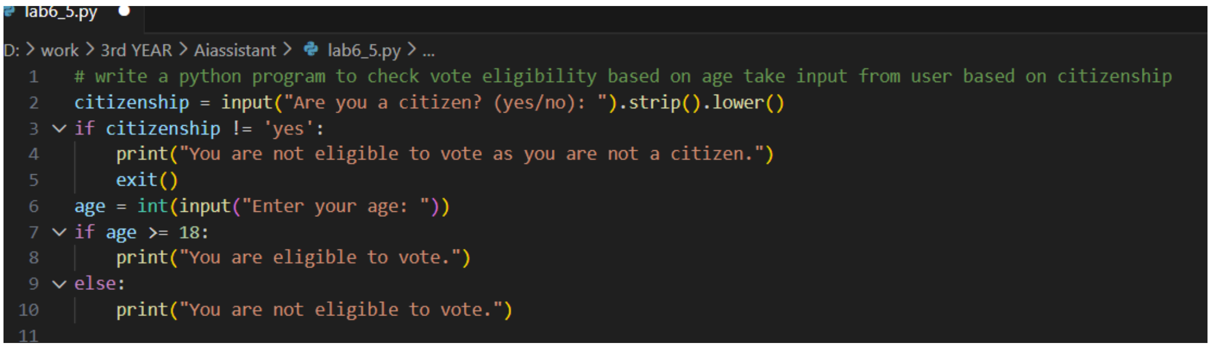
Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

Prompt:

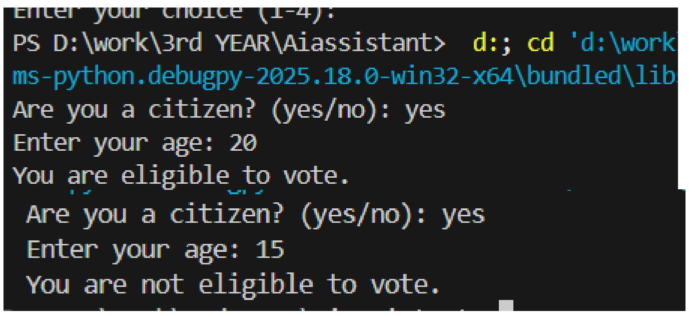
"write a python program to check vote eligibility based on age take input from user based on citizenship"

Code:



```
lab6_5.py
D: > work > 3rd YEAR > Aiassistant > lab6_5.py > ...
# write a python program to check vote eligibility based on age take input from user based on citizenship
citizenship = input("Are you a citizen? (yes/no): ").strip().lower()
if citizenship != 'yes':
    print("You are not eligible to vote as you are not a citizen.")
    exit()
age = int(input("Enter your age: "))
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

Output:



```
Enter your choice (1-4):
PS D:\work\3rd YEAR\Aiassistant>  d:; cd 'd:\work
ms-python.debugpy-2025.18.0-win32-x64\bundled\lib
Are you a citizen? (yes/no): yes
Enter your age: 20
You are eligible to vote.
```

```
Are you a citizen? (yes/no): yes
Enter your age: 15
You are not eligible to vote.
```

Explanation: The program checks voting eligibility by first asking if you're a citizen—if you answer anything other than "yes", it immediately stops and prevents you from proceeding further using exit(). Once citizenship is confirmed, it prompts you to enter your age and converts it to an integer for comparison. The program then checks if your age is 18 or older, which is the legal voting age in most democracies. Finally, based on this age check, it provides your verdict: either confirming that you are eligible to vote, or informing you that you must wait until you turn 18 to be eligible.

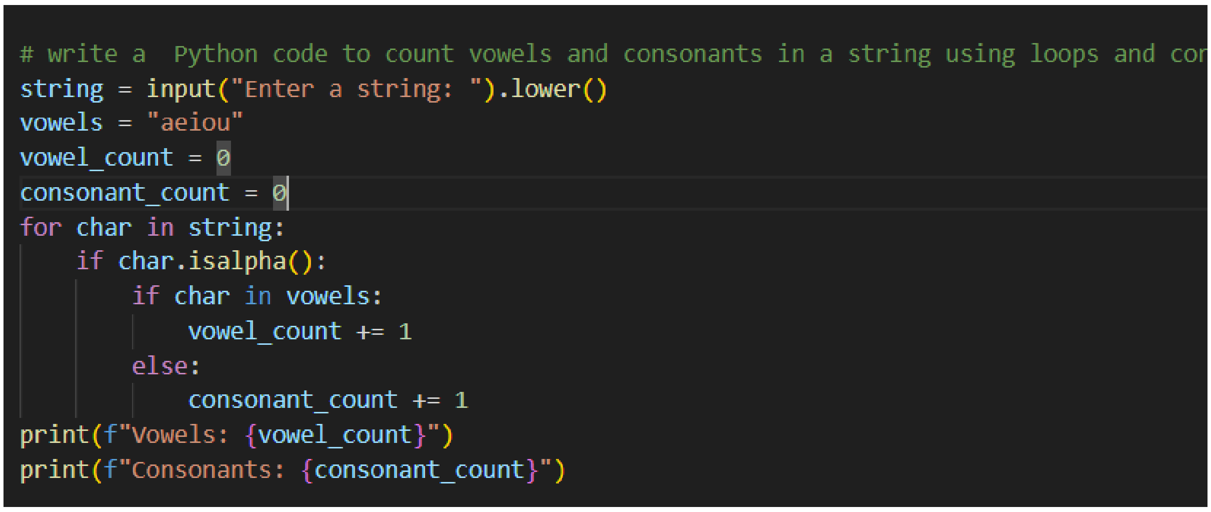
Task Description #2(AI-Based Code Completion for Loop-Based String Processing)

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

Prompt:

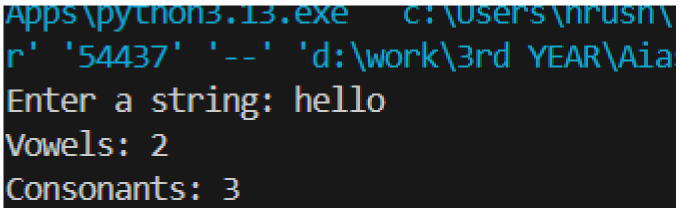
"# write a Python code to count vowels and consonants in a string using loops and conditional statements."

Code:

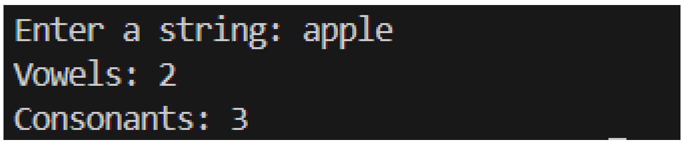


```
# write a  Python code to count vowels and consonants in a string using loops and con
string = input("Enter a string: ").lower()
vowels = "aeiou"
vowel_count = 0
consonant_count = 0
for char in string:
    if char.isalpha():
        if char in vowels:
            vowel_count += 1
        else:
            consonant_count += 1
print(f"Vowels: {vowel_count}")
print(f"Consonants: {consonant_count}")
```

Output:



```
Apps\python3.13.exe   c:\Users\hrush\
r' '54437' '--' 'd:\work\3rd YEAR\Aia
Enter a string: hello
Vowels: 2
Consonants: 3
```



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

Explanation: The program takes a string input, converts it to lowercase, and counts vowels and consonants by looping through each character. It uses char.isalpha() to identify only letters, then checks if each letter is a vowel (a, e, i, o, u) or a consonant, incrementing the respective counters, and finally displays the totals. Two counters (vowel_count and consonant_count) are initialized at zero and incremented each time a vowel or consonant is found. The program ignores spaces, numbers, and special characters, focusing only on alphabetic characters.

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

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

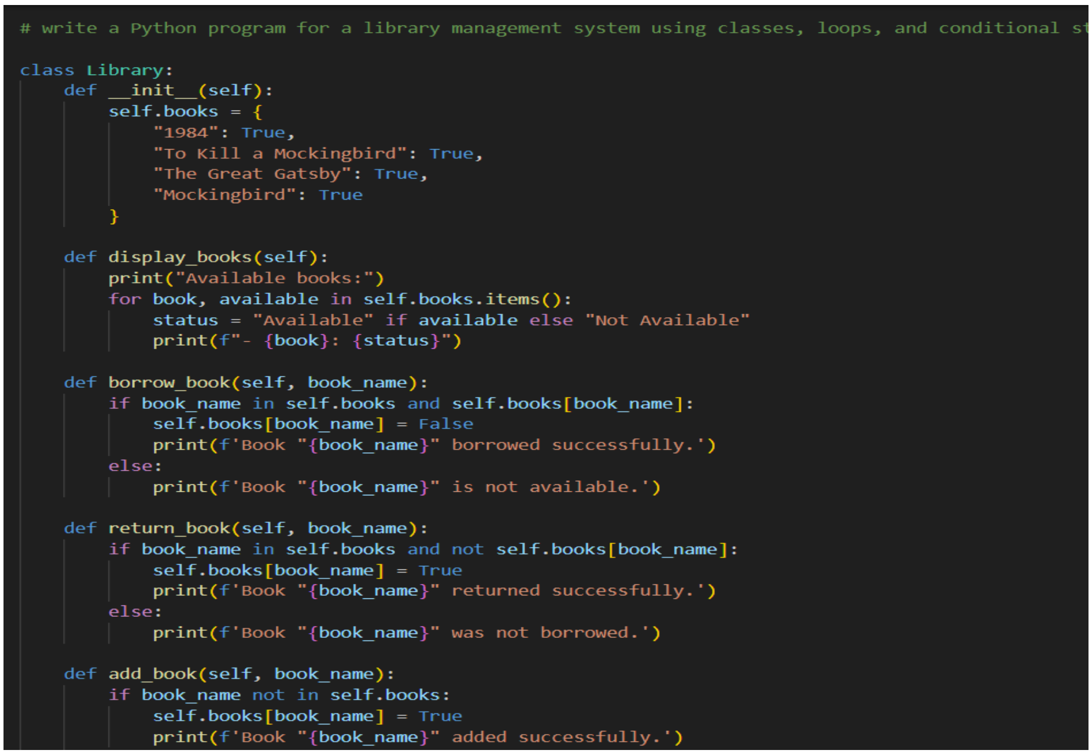
loops, and conditionals.

Prompt:

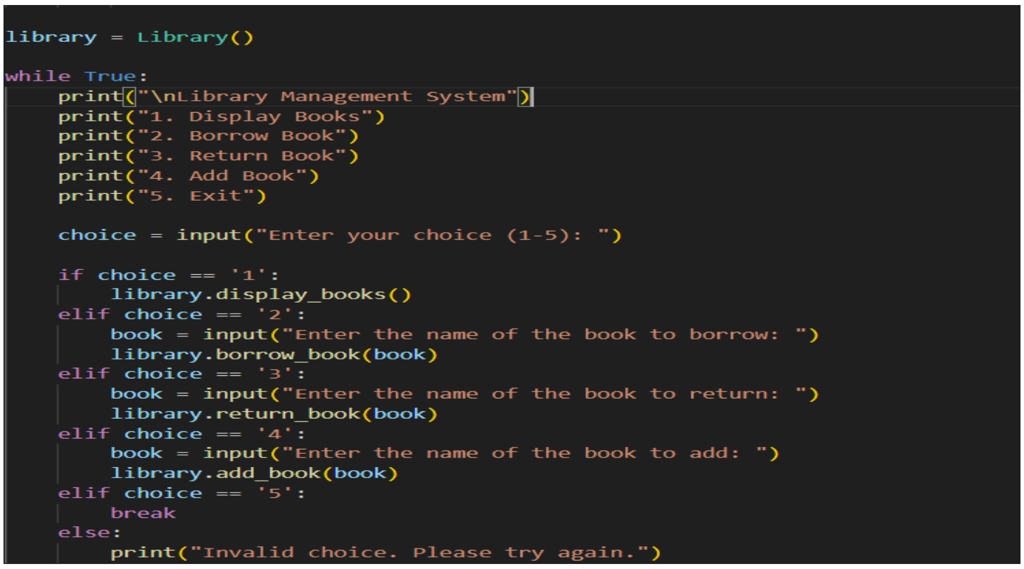
“Generate a Python program for a library management system

using classes, loops, and conditional statements.”

Code:

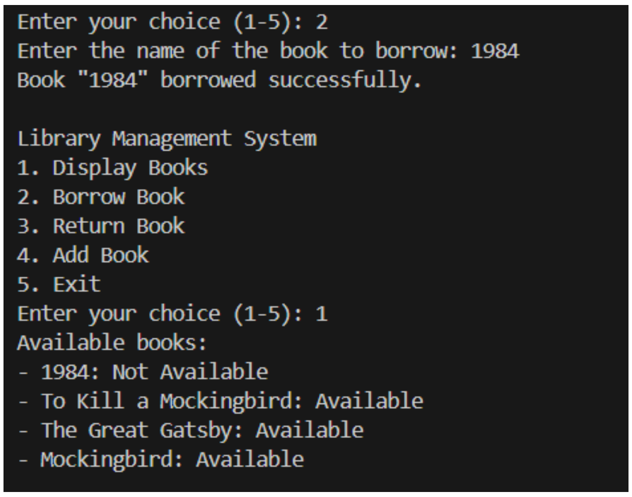


```python
# write a Python program for a library management system using classes, loops, and conditional st

class Library:
    def __init__(self):
        self.books = {
            "1984": True,
            "To Kill a Mockingbird": True,
            "The Great Gatsby": True,
            "Mockingbird": True
        }

    def display_books(self):
        print("Available books:")
        for book, available in self.books.items():
            status = "Available" if available else "Not Available"
            print(f"- {book}: {status}")

    def borrow_book(self, book_name):
        if book_name in self.books and self.books[book_name]:
            self.books[book_name] = False
            print(f'Book "{book_name}" borrowed successfully.')
        else:
            print(f'Book "{book_name}" is not available.')

    def return_book(self, book_name):
        if book_name in self.books and not self.books[book_name]:
            self.books[book_name] = True
            print(f'Book "{book_name}" returned successfully.')
        else:
            print(f'Book "{book_name}" was not borrowed.')

    def add_book(self, book_name):
        if book_name not in self.books:
            self.books[book_name] = True
            print(f'Book "{book_name}" added successfully.')
```



```python
library = Library()

while True:
    print("\nLibrary Management System")
    print("1. Display Books")
    print("2. Borrow Book")
    print("3. Return Book")
    print("4. Add Book")
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")

    if choice == '1':
        library.display_books()
    elif choice == '2':
        book = input("Enter the name of the book to borrow: ")
        library.borrow_book(book)
    elif choice == '3':
        book = input("Enter the name of the book to return: ")
        library.return_book(book)
    elif choice == '4':
        book = input("Enter the name of the book to add: ")
        library.add_book(book)
    elif choice == '5':
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:



```
Enter your choice (1-5): 2
Enter the name of the book to borrow: 1984
Book "1984" borrowed successfully.

Library Management System
1. Display Books
2. Borrow Book
3. Return Book
4. Add Book
5. Exit
Enter your choice (1-5): 1
Available books:
- 1984: Not Available
- To Kill a Mockingbird: Available
- The Great Gatsby: Available
- Mockingbird: Available
```

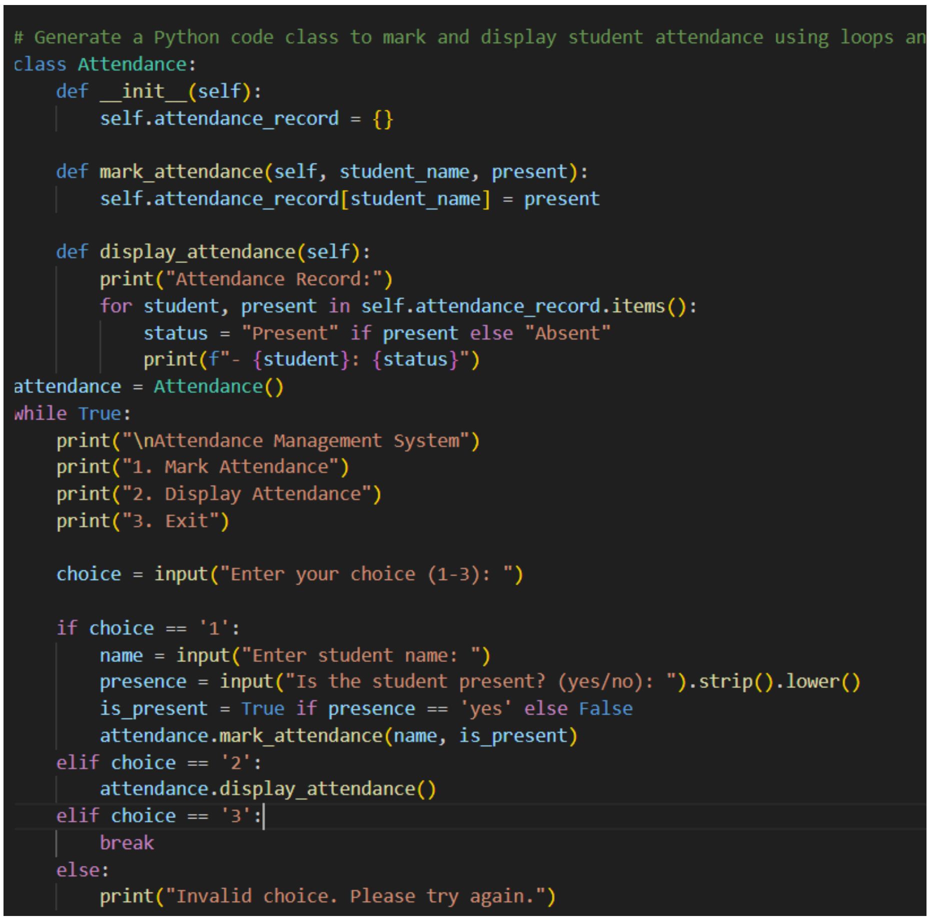
Explanation: This program creates a Library class that manages a collection of books with a dictionary tracking their availability status (True for available, False for borrowed).
The display_books() method lists all books with their current status,
while borrow_book() changes a book's status to unavailable if it's in the library,
and return_book() marks it as available again. The add_book() method adds new books to the collection. A menu-driven while loop continuously prompts users to choose operations (display, borrow, return, add books, or exit), calling the appropriate class methods based on their selection until they choose to exit.

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

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

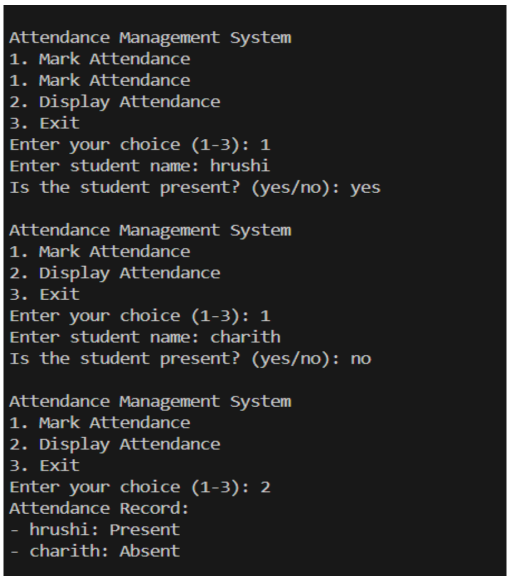
Prompt: “Generate a Python class to mark and display student attendance using loops.”

Code:



```python
# Generate a Python code class to mark and display student attendance using loops an
class Attendance:
    def __init__(self):
        self.attendance_record = {}

    def mark_attendance(self, student_name, present):
        self.attendance_record[student_name] = present

    def display_attendance(self):
        print("Attendance Record:")
        for student, present in self.attendance_record.items():
            status = "Present" if present else "Absent"
            print(f"- {student}: {status}")
attendance = Attendance()
while True:
    print("\nAttendance Management 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: ")
        presence = input("Is the student present? (yes/no): ").strip().lower()
        is_present = True if presence == 'yes' else False
        attendance.mark_attendance(name, is_present)
    elif choice == '2':
        attendance.display_attendance()
    elif choice == '3':
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:



```
Attendance Management System
1. Mark Attendance
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 1
Enter student name: hrushi
Is the student present? (yes/no): yes

Attendance Management System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 1
Enter student name: charith
Is the student present? (yes/no): no

Attendance Management System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 2
Attendance Record:
- hrushi: Present
- charith: Absent
```

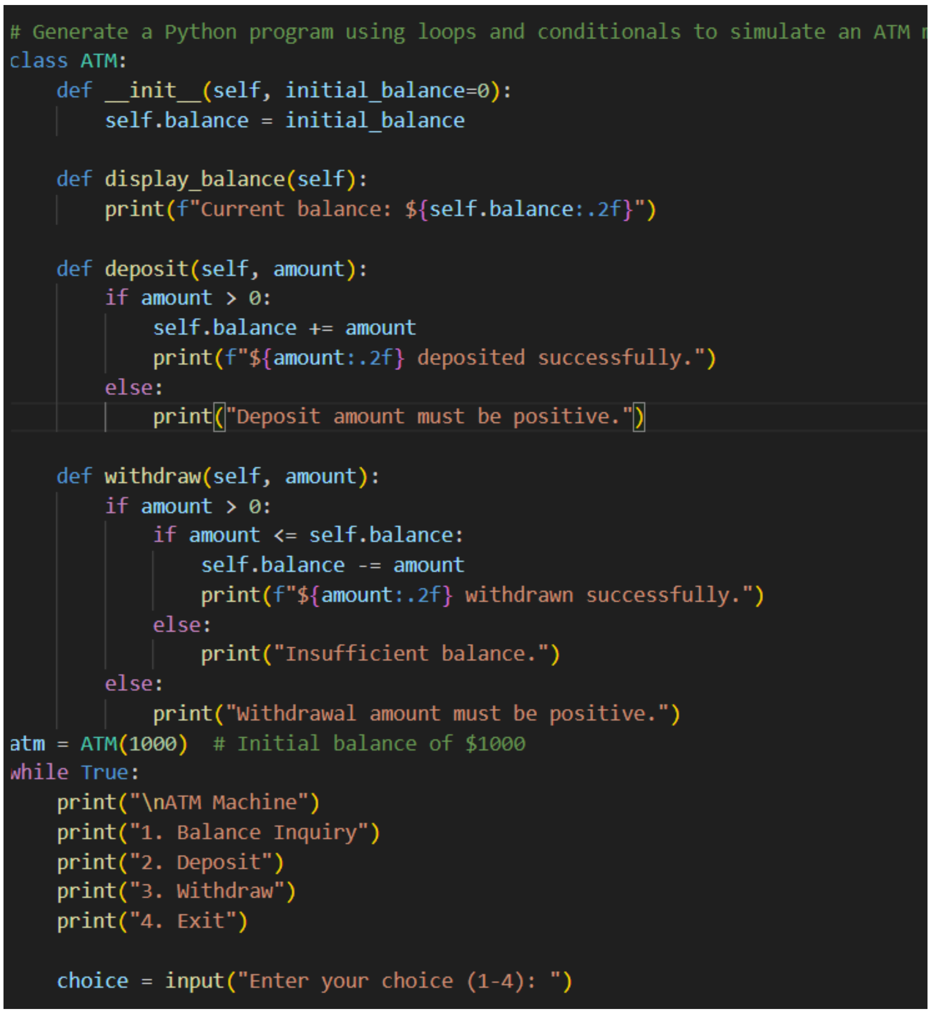
Explanation: This program creates an Attendance class that stores student attendance records in a dictionary where the student name is the key and their presence status (True/False) is the value. The mark_attendance() method records whether a student is present or absent, while display_attendance() loops through all records and shows each student's status as "Present" or "Absent". An instance of the class is created, and a menu-driven while loop allows users to mark attendance by entering a student's name and yes/no response (which converts to a boolean), display all attendance records, or exit the system until they choose option 3.

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

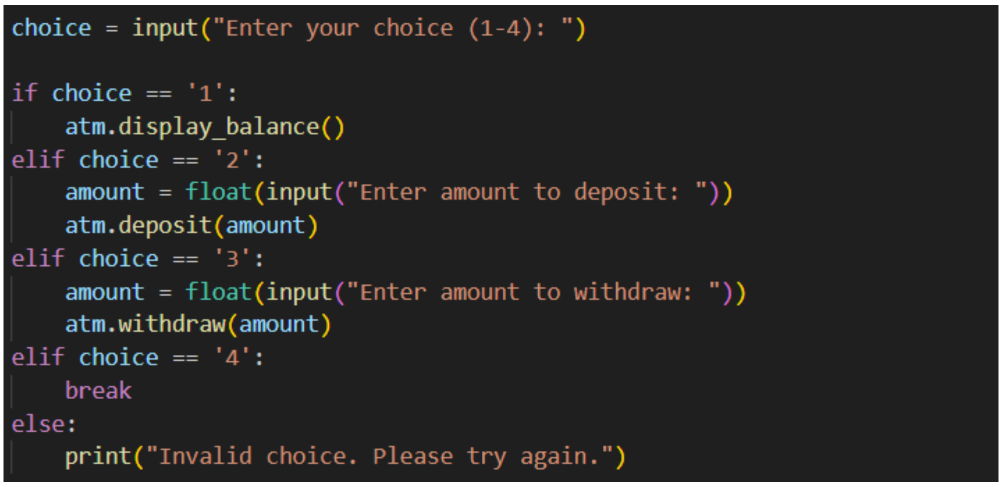
Task: Use an AI tool to complete a navigation menu.

Prompt: “Generate a Python program using loops and conditionals to simulate an ATM machine with options for balance inquiry, withdrawal, and deposit..”

Code:

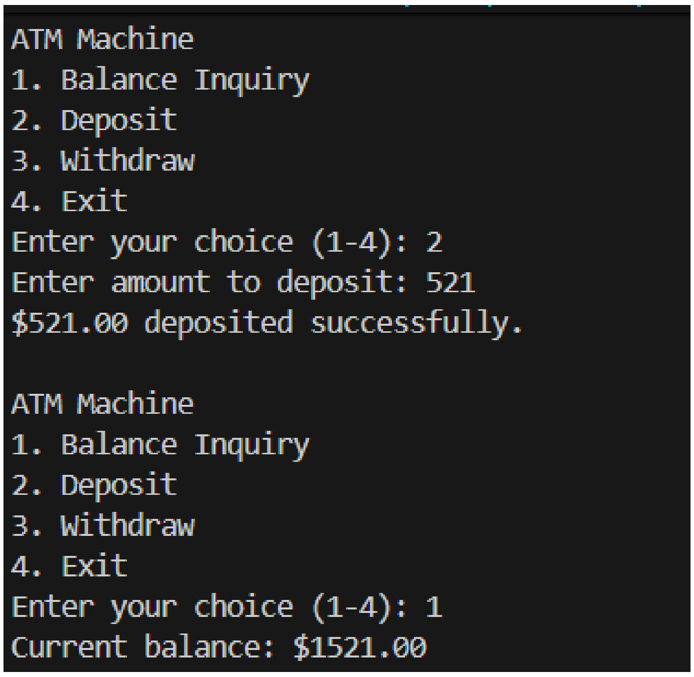


```python
# Generate a Python program using loops and conditionals to simulate an ATM
class ATM:
    def __init__(self, initial_balance=0):
        self.balance = initial_balance

    def display_balance(self):
        print(f"Current balance: ${self.balance:.2f}")

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"${amount:.2f} deposited successfully.")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"${amount:.2f} withdrawn successfully.")
            else:
                print("Insufficient balance.")
        else:
            print("Withdrawal amount must be positive.")
atm = ATM(1000)  # Initial balance of $1000
while True:
    print("\nATM Machine")
    print("1. Balance Inquiry")
    print("2. Deposit")
    print("3. Withdraw")
    print("4. Exit")

    choice = input("Enter your choice (1-4): ")
```



```python
choice = input("Enter your choice (1-4): ")

if choice == '1':
    atm.display_balance()
elif choice == '2':
    amount = float(input("Enter amount to deposit: "))
    atm.deposit(amount)
elif choice == '3':
    amount = float(input("Enter amount to withdraw: "))
    atm.withdraw(amount)
elif choice == '4':
    break
else:
    print("Invalid choice. Please try again.")
```

Output:



```
ATM Machine
1. Balance Inquiry
2. Deposit
3. Withdraw
4. Exit
Enter your choice (1-4): 2
Enter amount to deposit: 521
$521.00 deposited successfully.

ATM Machine
1. Balance Inquiry
2. Deposit
3. Withdraw
4. Exit
Enter your choice (1-4): 1
Current balance: $1521.00
```

Explanation: This program creates an ATM class that simulates a real ATM machine with a balance attribute initialized to 1000 dollars. The display_balance() method shows the current balance in formatted currency, while deposit() adds money only if the amount is positive. The withdraw() method removes money with nested conditionals—first checking if the amount is positive, then verifying sufficient balance exists, and displaying error messages for invalid transactions. An instance is created with an initial $1000 balance, and a menu-driven while loop allows users to check balance, deposit funds, withdraw cash, or exit the system based on their choice selection.