

Vyshnavi Parisha

2403a51l34

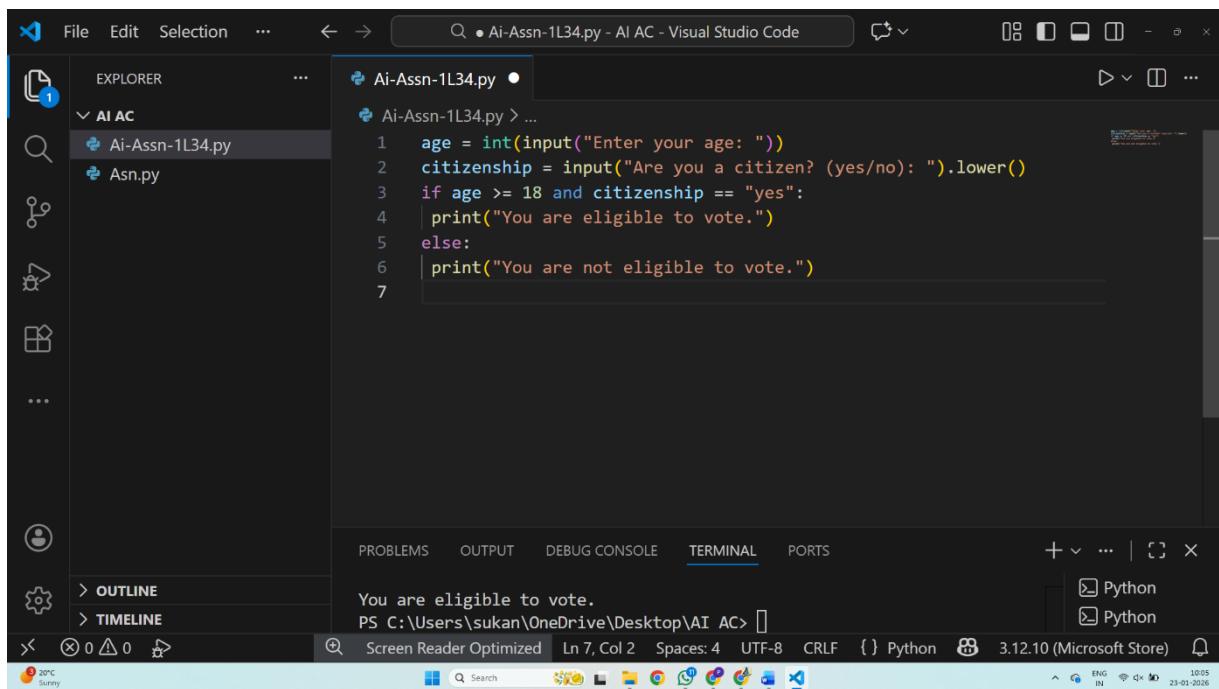
B-52

## Lab 5: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

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

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

#### Code

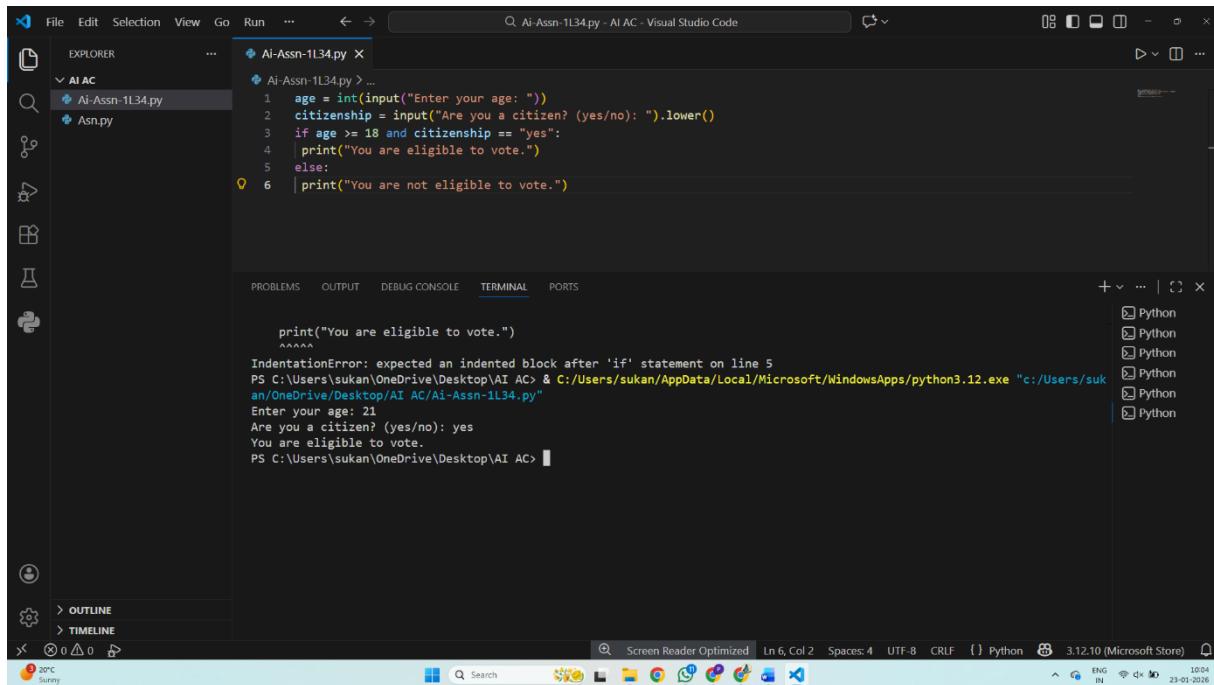


A screenshot of the Visual Studio Code interface. The title bar shows "Ai-Assn-1L34.py - AI AC - Visual Studio Code". The Explorer sidebar on the left shows a folder named "AI AC" containing files "Ai-Assn-1L34.py" and "Asn.py". The main editor area displays the following Python code:

```
1 age = int(input("Enter your age: "))
2 citizenship = input("Are you a citizen? (yes/no): ").lower()
3 if age >= 18 and citizenship == "yes":
4     print("You are eligible to vote.")
5 else:
6     print("You are not eligible to vote.)
```

The terminal tab at the bottom shows the output: "You are eligible to vote." and "PS C:\Users\sukan\Desktop\AI AC>". The status bar at the bottom right shows "3.12.10 (Microsoft Store)" and the date "23-01-2026".

## OUTPUT:



```
File Edit Selection View Go Run ... ⏪ Ai-Assn-1L34.py - AI AC - Visual Studio Code
EXPLORER
AI AC
Ai-Assn-1L34.py
Asn.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
print("You are eligible to vote.")
~~~~~
IndentationError: expected an indented block after 'if' statement on line 5
PS C:\Users\sukan\OneDrive\Desktop\AI AC & C:/Users/sukan/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/sukan/OneDrive/Desktop/AI AC/Ai-Assn-1L34.py"
Enter your age: 21
Are you a citizen? (yes/no): yes
You are eligible to vote.
PS C:\Users\sukan\OneDrive\Desktop\AI AC>
```

## Explanation :

- The program accepts the user's age and citizenship status.
- The conditional statement checks if age is 18 or above and citizenship is "yes".
- If both conditions are satisfied, eligibility is confirmed; otherwise, the user is not eligible.

## Verification:

Age = 20, Citizenship = yes → Eligible

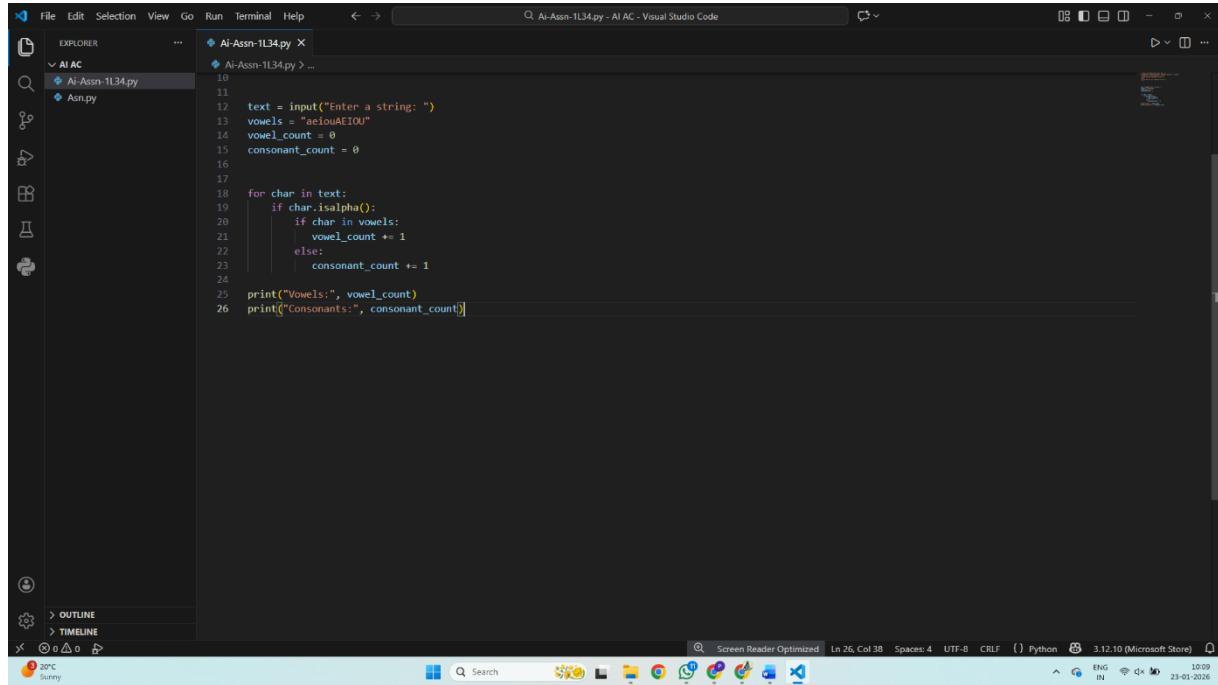
Age = 16, Citizenship = yes → Not eligible

**Observations:** The initial API code is insecure because it uses a hardcoded API key and does not protect user data. The corrected version improves security by validating inputs, hashing passwords, and using token-based authentication for safer access control.

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

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

### AI-Generated Code:



```
File Edit Selection View Go Run Terminal Help < > Q Ai-Assn-1L34.py - AI AC - Visual Studio Code
EXPLORER
  AI AC
    Ai-Assn-1L34.py
    Asn.py
    ...
10
11
12 text = input("Enter a string: ")
13 vowels = "aeiouAEIOU"
14 vowel_count = 0
15 consonant_count = 0
16
17
18 for char in text:
19     if char.isalpha():
20         if char in vowels:
21             vowel_count += 1
22         else:
23             consonant_count += 1
24
25 print("Vowels:", vowel_count)
26 print("Consonants:", consonant_count)

Screen Reader Optimized Ln 26, Col 38 Spaces: 4 UTF-8 CRLF {} Python 3.12.10 (Microsoft Store) Q
Outline Timeline
20°C
10:09
ENG IN 23-01-2026
```

### Observations:

1. The logic unfairly favors urban students
2. Rural or semi-urban students are excluded
3. No flexibility or weighted scoring approach

### Improved Version:

## OUTPUT:

The screenshot shows a Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files: 'AI AC' and 'Ai-Assn-1L34.py'. The main editor window displays the following Python code:

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

The terminal below the editor shows the output of running the script with the input 'Vishnavi':

```
Enter a string: Vishnavi
Vowels: 2
Consonants: 6
PS C:\Users\sukan\OneDrive\Desktop\AI AC>
```

The status bar at the bottom indicates the file is 'Screen Reader Optimized' with 'Ln 26, Col 38' and other system information.

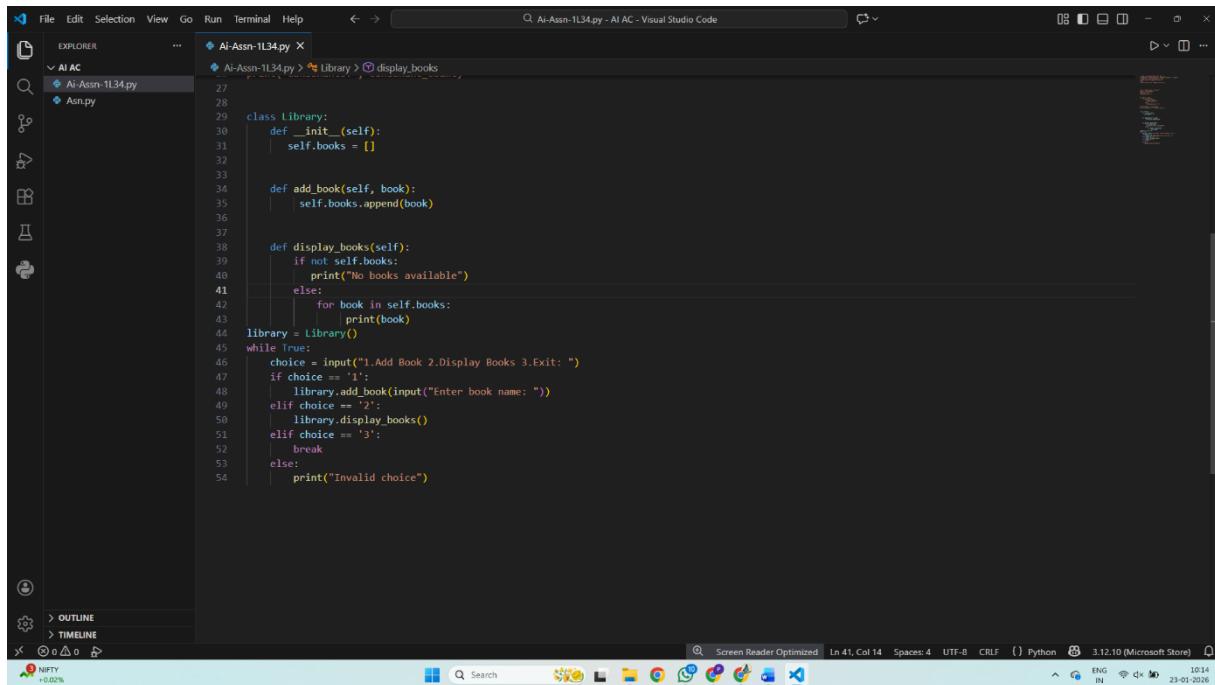
## Explanation:

- A loop iterates through each character in the string.
- Alphabetic characters are checked.
- Vowels and consonants are counted separately

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

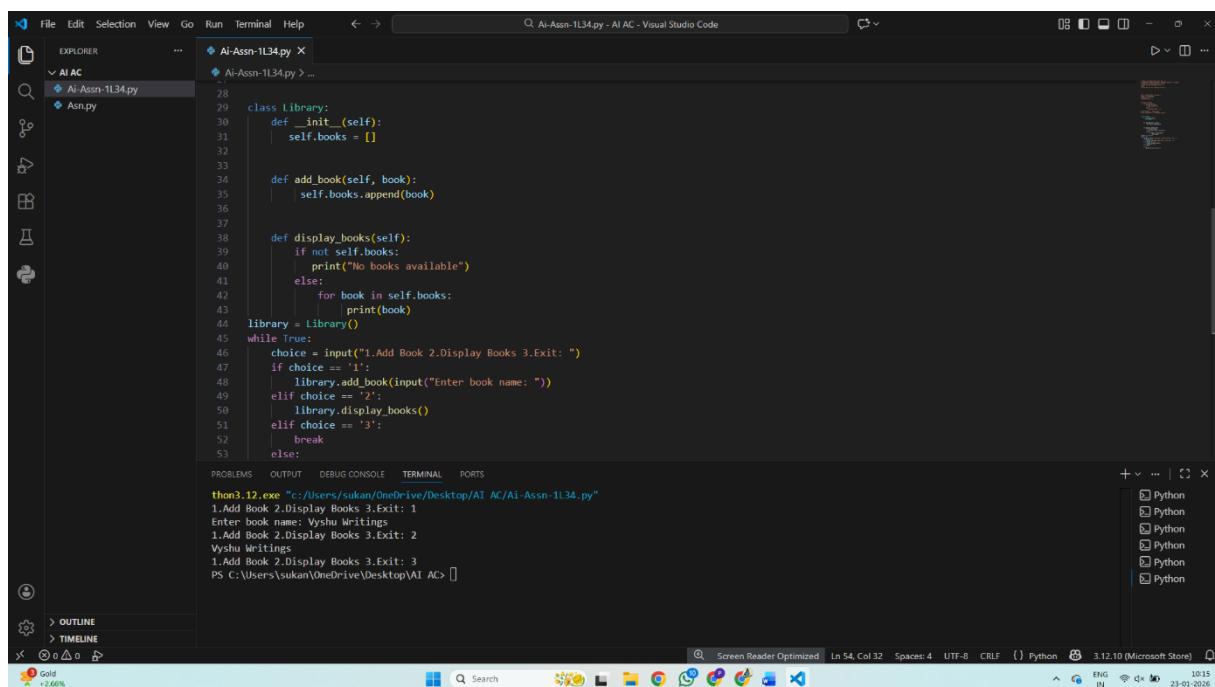
**Prompt:** “Generate a Python program for a library management system using classes, loops, and conditional statements.”

## Code:



```
File Edit Selection View Go Run Terminal Help <- > Ai-Assn-1L34.py - AI AC - Visual Studio Code
EXPLORER
AI AC
Ai-Assn-1L34.py
Asn.py
27
28
29  class Library:
30      def __init__(self):
31          self.books = []
32
33
34      def add_book(self, book):
35          self.books.append(book)
36
37
38      def display_books(self):
39          if not self.books:
40              print("No books available")
41          else:
42              for book in self.books:
43                  print(book)
44
library = Library()
while True:
    choice = input("1.Add Book 2.Display Books 3.Exit: ")
    if choice == '1':
        library.add_book(input("Enter book name: "))
    elif choice == '2':
        library.display_books()
    elif choice == '3':
        break
    else:
        print("Invalid choice")
```

## OUTPUT:



```
File Edit Selection View Go Run Terminal Help <- > Ai-Assn-1L34.py - AI AC - Visual Studio Code
EXPLORER
AI AC
Ai-Assn-1L34.py
Asn.py
28
29  class Library:
30      def __init__(self):
31          self.books = []
32
33
34      def add_book(self, book):
35          self.books.append(book)
36
37
38      def display_books(self):
39          if not self.books:
40              print("No books available")
41          else:
42              for book in self.books:
43                  print(book)
44
library = Library()
while True:
    choice = input("1.Add Book 2.Display Books 3.Exit: ")
    if choice == '1':
        library.add_book(input("Enter book name: "))
    elif choice == '2':
        library.display_books()
    elif choice == '3':
        break
    else:
        print("Invalid choice")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
thon3.12.exe "c:/Users/sukan/OneDrive/Desktop/AI AC/Ai-Assn-1L34.py"
1.Add Book 2.Display Books 3.Exit: 1
Enter book name: Vyshu Writings
1.Add Book 2.Display Books 3.Exit: 2
Vyshu Writings
1.Add Book 2.Display Books 3.Exit: 3
PS C:\Users\sukan\OneDrive\Desktop\AI AC:
```

## Explanation:

- The AI generated a logically correct and readable program.
- It effectively used classes, loops, and conditionals.

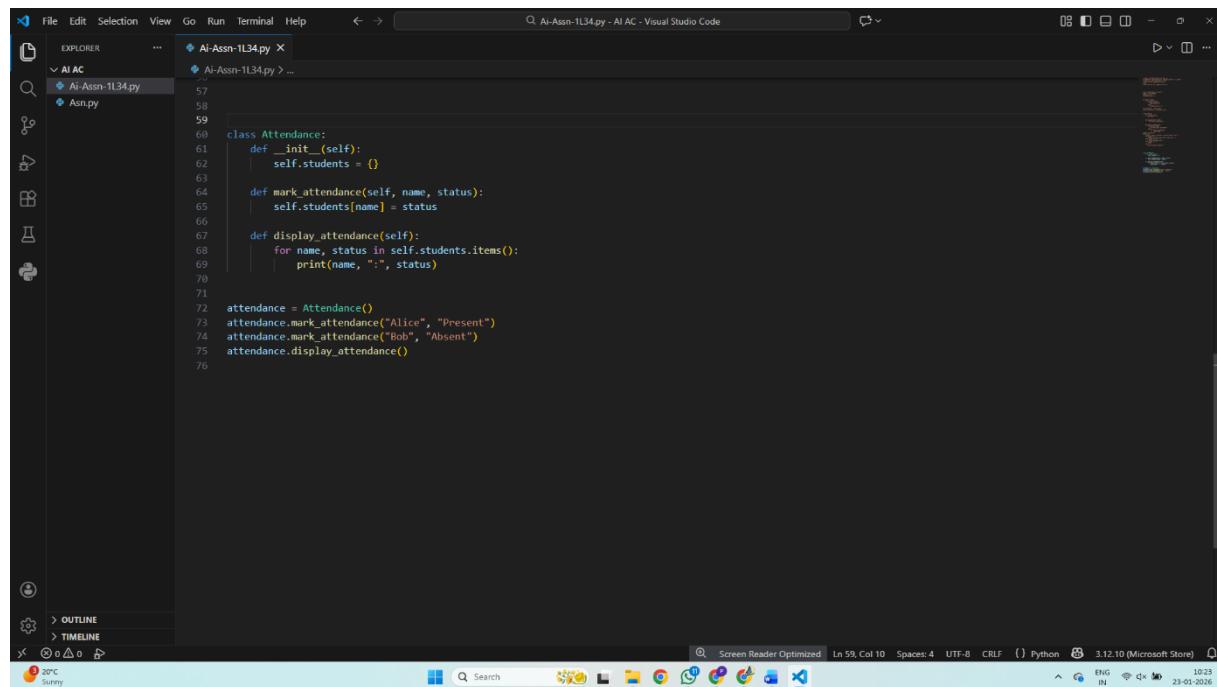
## Observations:

AI-assisted coding helps speed up development and provides structured solutions. However, human review is essential to ensure efficiency, security, and correctness.

## Task Description – 4: AI-Assisted Code Completion for Class-Based Attendance System

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

### Code:



The screenshot shows a dark-themed instance of Visual Studio Code. The left sidebar has icons for Explorer, Search, and Outline. The main editor area contains the following Python code:

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

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

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

attendance = Attendance()
attendance.mark_attendance("Alice", "Present")
attendance.mark_attendance("Bob", "Absent")
attendance.display_attendance()
```

The code defines a class named `Attendance` with methods for marking attendance and displaying it. It uses a dictionary to store student names and their status. The code is run at the bottom, resulting in the output:

```
Alice : Present
Bob : Absent
```

## OUTPUT:

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: `Ai-Assn-1L34.py` and `Asn.py`.
- Code Editor:** Displays Python code for an `Attendance` class with methods `__init__`, `mark_attendance`, and `display_attendance`. It also shows the creation of an `Attendance` object, marking attendance for "Alice" as "Present" and "Bob" as "Absent", and then displaying the attendance status.
- Terminal:** Shows the command-line output of the program running in Python 3.12.10. The output indicates that Bob is Absent and Alice is Present.
- Python Environment:** A sidebar on the right lists multiple Python environments, all named "Python".
- System Tray:** At the bottom, it shows the date and time as 23-01-2026, and icons for search, file explorer, and other system functions.

## Observations:

- The class-based design improves code organization and readability.
- Dictionary usage ensures efficient storage and retrieval of attendance data.
- The program produces accurate output with minimal code.

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

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

## Code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: `Ai-Assn-1L34.py` and `Asn.py`.
- Code Editor:** Displays the contents of `Ai-Assn-1L34.py`. The code is a Python script for a bank account management system. It initializes a balance of 50000. It then enters a loop where it prints menu options (1.Check Balance, 2.Deposit, 3.Withdraw, 4.Exit), takes user input, and performs the corresponding action. If deposit or withdrawal is attempted, it checks if the amount is less than or equal to the current balance before updating it.
- Terminal:** Shows the command `PS C:\Users\sukan\OneDrive\Desktop\AI AC>` followed by the user's input: `Enter choice: 4`.
- Status Bar:** Shows the Python extension version (3.12.10 Microsoft Store), system information (ENG IN), and date (23-01-2026).

## OUTPUT:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the same files as the previous screenshot.
- Code Editor:** Displays the same Python code as before.
- Terminal:** Shows the full execution of the script. The user inputs `4` to exit the loop. The terminal also displays the command used to run the script (`C:/Users/sukan/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/sukan/Desktop/AI AC/Ai-Assn-1L34.py"`) and the resulting output of the program's logic.
- Status Bar:** Shows the Python extension version (3.12.10 Microsoft Store), system information (ENG IN), and date (23-01-2026).

## **Observations:**

- The program effectively simulates real-world ATM operations.
- Conditional statements and loops are used correctly.
- The system handles invalid inputs and balance conditions properly.