ASSIGNMENT-6.5

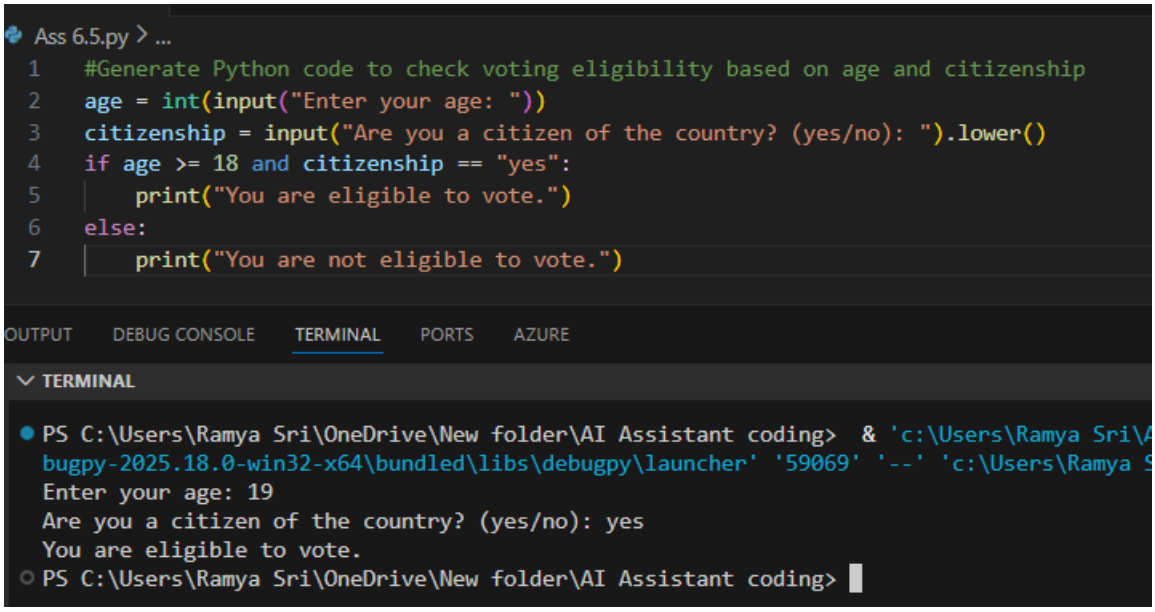
2303A510I9

BATCH:30

Task-1: Use an AI tool to generate eligibility logic.

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

Code & Output:



Justification:

AI tools simplify logic generation by quickly producing accurate code for common eligibility checks. Automating this task ensures consistency and reduces human error in applying rules like age and citizenship. It saves development time, allowing programmers to focus on more complex features instead of repetitive conditions.

Task 2: 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 & Output:

```python
#Generate Python code to count vowels and consonants in a string using a loop.
string = input("Enter a string: ")
vowels = "aeiouAEIOU"
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}, Consonants: {consonant_count}")
```

```
ppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Ra
r' '62887' '--' 'c:\Users\Ramya Sri\OneDrive\New folder\AI Assi
Enter a string: RamyaSri
Vowels: 3, Consonants: 5
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

Justification:

Using a loop allows systematic traversal of each character in the string. Conditional checks (is alpha, membership in vowels) ensure accurate classification. This approach demonstrates how AI tools can automate repetitive logic creation efficiently.

Task 3: 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 & Output:

```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_available = True
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book):
        self.books.append(book)
    def display_books(self):
        for book in self.books:
            status = "Available" if book.is_available else "Checked out"
            print(f"{book.title} by {book.author} - {status}")
    def check_out_book(self, title):
        for book in self.books:
            if book.title == title and book.is_available:
                book.is_available = False
                print(f"You have checked out '{book.title}'.")
                return
        print("Sorry, that book is not available.")
library = Library()
library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
library.add_book(Book("1984", "George Orwell"))
library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
library.display_books()
library.check_out_book("1984")
library.display_books()
```

```
OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE
TERMINAL
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>  c:; cd 'c:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding';
.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62406' '--' 'c:\Users\Ramya Sri\OneDrive\New
.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62406' '--' 'c:\Users\Ramya Sri\OneDrive\New
Enter a string: RamyaSri
Enter a string: RamyaSri
Vowels: 3, Consonants: 5
To Kill a Mockingbird by Harper Lee - Available
1984 by George Orwell - Available
The Great Gatsby by F. Scott Fitzgerald - Available
You have checked out '1984'.
To Kill a Mockingbird by Harper Lee - Available
1984 by George Orwell - Checked out
The Great Gatsby by F. Scott Fitzgerald - Available
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

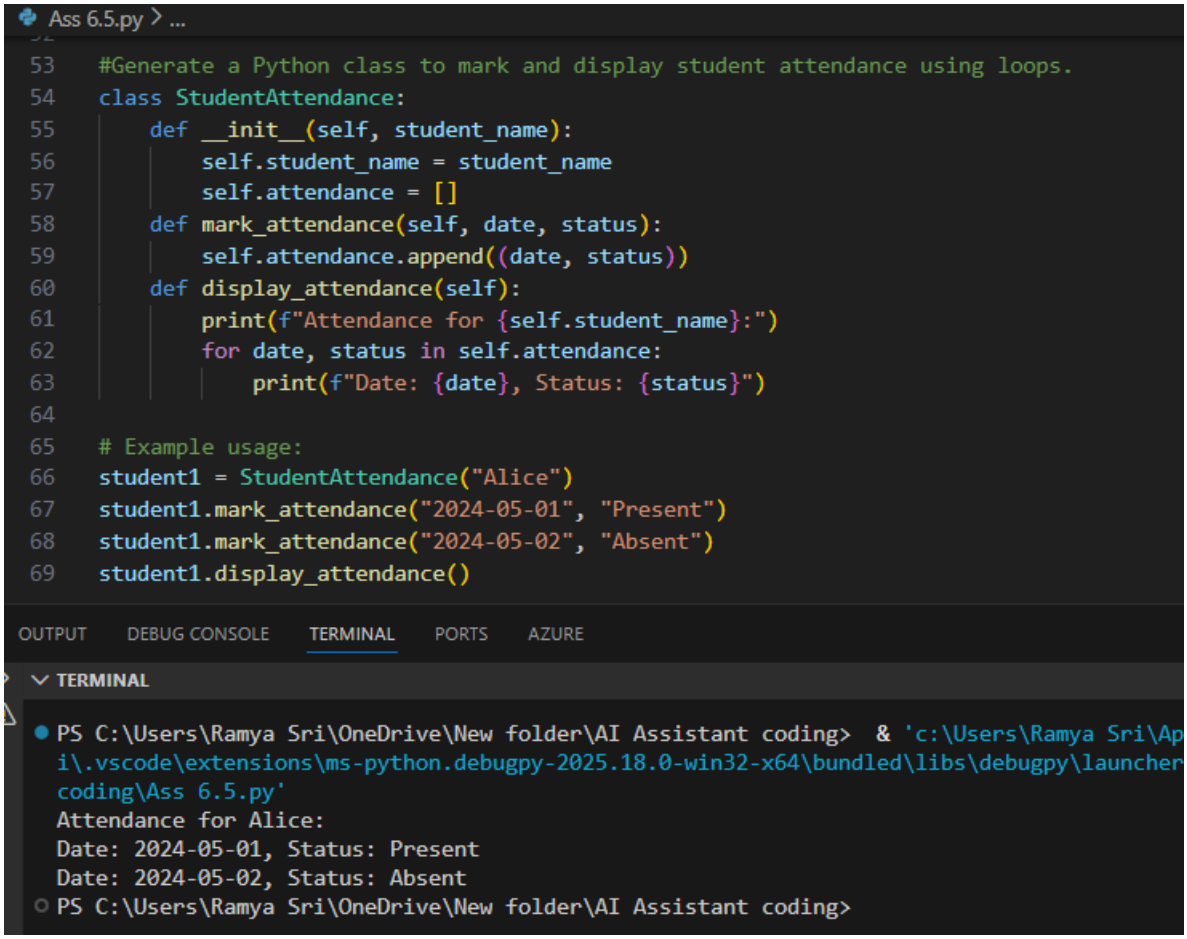
Justification:

The program uses **classes** (Book and Library) to model real-world entities and their behaviours. **Loops** provide continuous interaction through the menu until the user exits. **Conditionals** handle user choices and book availability, ensuring logical flow in the system.

Task 4: Use an AI tool to generate an attendance management class.

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

Code & Output:



Justification:

The class encapsulates attendance logic, making it reusable and organized. Loops are used to iterate through student lists and mark attendance efficiently. Conditional handling ensures flexibility in marking and displaying records clearly.

Task 5: Use an AI tool to complete a navigation menu.

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

Code & Output:

```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: ")
    if choice == "1":
        print(f"Your balance is: ${balance}")
    elif choice == "2":
        amount = float(input("Enter amount to deposit: "))
        balance += amount
        print(f"Deposited ${amount}. New balance is ${balance}")
    elif choice == "3":
        amount = float(input("Enter amount to withdraw: "))
        if amount <= balance:
            balance -= amount
            print(f"Withdrew ${amount}. New balance is ${balance}")
        else:
            print("Insufficient funds.")
    elif choice == "4":
        print("Thank you for using the ATM.")
        break
    else:
        print("Invalid choice. Please try again.")

```

TERMINAL

```
  ATM Menu:
  1. Check Balance …
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>  c:; cd 'c:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding';
  ppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Ramya Sri\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundi
  r' '58342' '--' 'c:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding\Ass 6.5.py'

  ATM Menu:
  1. Check Balance
  2. Deposit Money
  3. Withdraw Money
  4. Exit
  Enter your choice: 1
  Your balance is: $1000

  ATM Menu:
  1. Check Balance
  2. Deposit Money
  3. Withdraw Money
  4. Exit
  Enter your choice: 2
  Enter amount to deposit: 3000
  Deposited $3000.0. New balance is $4000.0

  ATM Menu:
  1. Check Balance
  2. Deposit Money
  3. Withdraw Money
  4. Exit
  Enter your choice: 3
  Enter amount to withdraw: 1000
  Withdrew $1000.0. New balance is $3000.0

  ATM Menu:
  1. Check Balance
  2. Deposit Money
  3. Withdraw Money
  4. Exit
  Enter your choice: 4
  Thank you for using the ATM.
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

Justification:

ATM Menu Simulation – Using loops and conditionals for menu navigation demonstrates interactive program design, strengthening logic flow and user input handling in Python.