

ASSIGNMENT-6.5

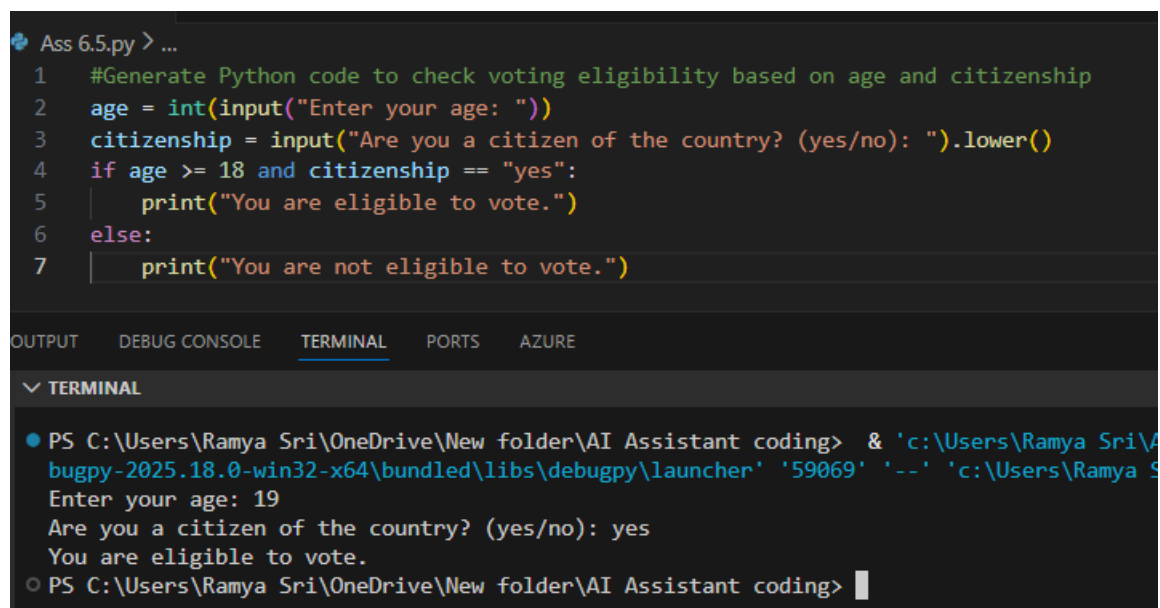
2303A510I9

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:



```
Ass 6.5.py > ...
1 #Generate Python code to check voting eligibility based on age and citizenship
2 age = int(input("Enter your age: "))
3 citizenship = input("Are you a citizen of the country? (yes/no): ").lower()
4 if age >= 18 and citizenship == "yes":
5     print("You are eligible to vote.")
6 else:
7     print("You are not eligible to vote.")

OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

▼ TERMINAL
● PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & 'c:\Users\Ramya Sri\A
bugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59069' '--' 'c:\Users\Ramya S
Enter your age: 19
Are you a citizen of the country? (yes/no): yes
You are eligible to vote.
○ PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> |
```

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:

```
8
9 #Generate Python code to count vowels and consonants in a string using a loop.
10 string = input("Enter a string: ")
11 vowels = "aeiouAEIOU"
12 vowel_count = 0
13 consonant_count = 0
14 for char in string:
15     if char.isalpha():
16         if char in vowels:
17             vowel_count += 1
18         else:
19             consonant_count += 1
20 print(f"Vowels: {vowel_count}, Consonants: {consonant_count}")
```

```
ppData\Local\Programs\Python\Python313\python.exe" "C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding\AI Assistant coding.py"
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:

```
21
22 #Generate a Python program for a library management system using classes, loops, and conditional statements.
23 class Book:
24     def __init__(self, title, author):
25         self.title = title
26         self.author = author
27         self.is_available = True
28 class Library:
29     def __init__(self):
30         self.books = []
31     def add_book(self, book):
32         self.books.append(book)
33     def display_books(self):
34         for book in self.books:
35             status = "Available" if book.is_available else "Checked out"
36             print(f"{book.title} by {book.author} - {status}")
37     def check_out_book(self, title):
38         for book in self.books:
39             if book.title == title and book.is_available:
40                 book.is_available = False
41                 print(f"You have checked out '{book.title}'.")
42                 return
43         print("Sorry, that book is not available.")
44 library = Library()
45 library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
46 library.add_book(Book("1984", "George Orwell"))
47 library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
48 library.display_books()
49 library.check_out_book("1984")
50 library.display_books()
51
```

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';
.vscod\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62406' '---' 'c:\Users\Ramya Sri\OneDrive\New
.vscod\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:

```
Ass 6.5.py > ...
53 #Generate a Python class to mark and display student attendance using loops.
54 class StudentAttendance:
55     def __init__(self, student_name):
56         self.student_name = student_name
57         self.attendance = []
58     def mark_attendance(self, date, status):
59         self.attendance.append((date, status))
60     def display_attendance(self):
61         print(f"Attendance for {self.student_name}:")
62         for date, status in self.attendance:
63             print(f>Date: {date}, Status: {status}")
64
65 # Example usage:
66 student1 = StudentAttendance("Alice")
67 student1.mark_attendance("2024-05-01", "Present")
68 student1.mark_attendance("2024-05-02", "Absent")
69 student1.display_attendance()
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

▼ TERMINAL

```
● PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & 'c:\Users\Ramya Sri\AppData\Local\Microsoft\WindowsApps\Code.exe' -c 'python c:\Users\Ramya Sri\AppData\Local\Microsoft\WindowsApps\Code.exe .\Ass 6.5.py'
Attendance for Alice:
Date: 2024-05-01, Status: Present
Date: 2024-05-02, Status: Absent
○ PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

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:

```

70
71 #Generate a Python program using loops and conditionals to simulate an ATM menu.
72 balance = 1000
73 while True:
74     print("\nATM Menu:")
75     print("1. Check Balance")
76     print("2. Deposit Money")
77     print("3. Withdraw Money")
78     print("4. Exit")
79     choice = input("Enter your choice: ")
80     if choice == "1":
81         print(f"Your balance is: ${balance}")
82     elif choice == "2":
83         amount = float(input("Enter amount to deposit: "))
84         balance += amount
85         print(f"Deposited ${amount}. New balance is ${balance}")
86     elif choice == "3":
87         amount = float(input("Enter amount to withdraw: "))
88         if amount <= balance:
89             balance -= amount
90             print(f"Withdrew ${amount}. New balance is ${balance}")
91         else:
92             print("Insufficient funds.")
93     elif choice == "4":
94         print("Thank you for using the ATM.")
95         break
96     else:
97         print("Invalid choice. Please try again.")
98

```

```

▼ 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\bund
n' '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.