

# AI Assisted Coding

## Assignment 6.5

Name: P. Vineeth Kumar

Hall ticket no: 2303A51256

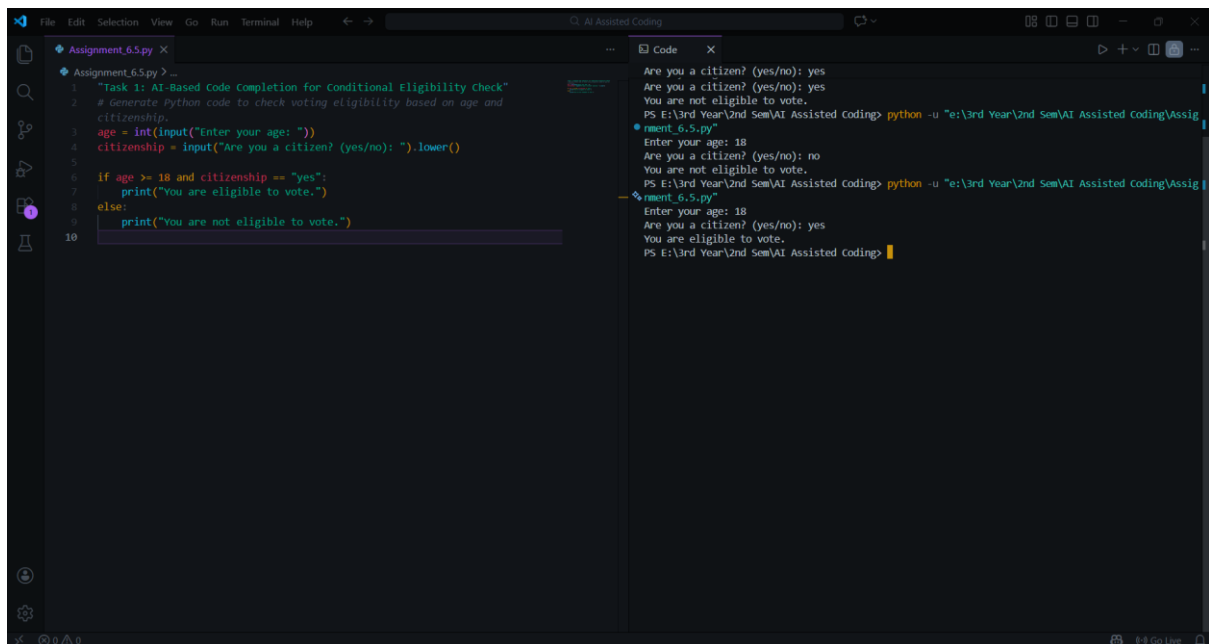
Batch no: 19

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

#### Prompt:

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

#### Code & Output:



```
Assignment_6.5.py X
Task 1: AI-Based Code Completion for Conditional Eligibility Check
# Generate Python code to check voting eligibility based on age and citizenship.
1 age = int(input("Enter your age: "))
2 citizenship = input("Are you a citizen? (yes/no): ").lower()
3
4 if age >= 18 and citizenship == "yes":
5     print("You are eligible to vote.")
6 else:
7     print("You are not eligible to vote.")
8
9
10

Code X
Are you a citizen? (yes/no): yes
Are you a citizen? (yes/no): yes
You are not eligible to vote.
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_6.5.py"
Enter your age: 18
Are you a citizen? (yes/no): no
You are not eligible to vote.
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_6.5.py"
Enter your age: 18
Are you a citizen? (yes/no): yes
You are eligible to vote.
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

#### Explanation:

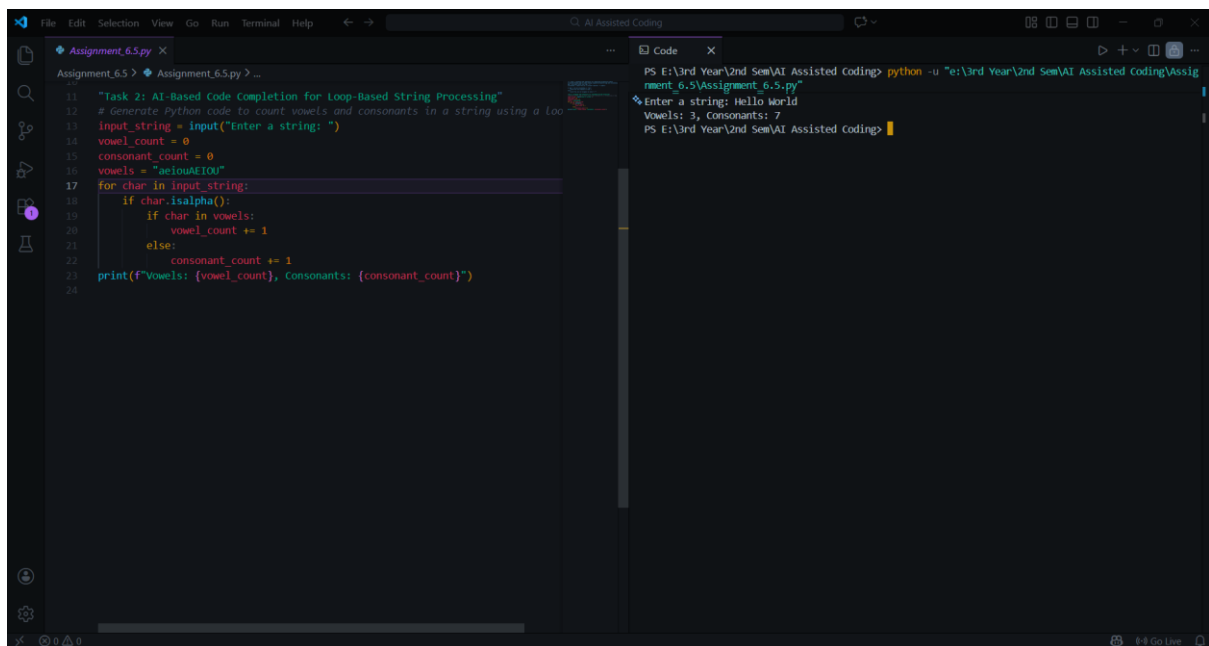
The AI-generated code uses conditional statements to check voting eligibility. It verifies whether the age is 18 or above and whether the user is a citizen. Both conditions must be true for eligibility. This demonstrates correct use of conditional logic generated through AI-based code completion.

### Task 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.

## Code & Output:



```
Assignment_6.5.py
11 "Task 2: AI-Based Code Completion for Loop-Based String Processing"
12 # Generate Python code to count vowels and consonants in a string using a loop
13 input_string = input("Enter a string: ")
14 vowel_count = 0
15 consonant_count = 0
16 vowels = "aeiouAEIOU"
17 for char in input_string:
18     if char.isalpha():
19         if char in vowels:
20             vowel_count += 1
21         else:
22             consonant_count += 1
23 print(f"Vowels: {vowel_count}, Consonants: {consonant_count}")
24
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_6.5\Assignment_6.5.py"
Enter a string: Hello world
Vowels: 3, Consonants: 7
PS E:\3rd Year\2nd Sem\AI Assisted coding>
```

## Explanation:

The AI-generated code processes the input string using a loop. Each character is checked to determine whether it is a vowel or a consonant. Alphabetic characters are counted correctly, while non-letter characters are ignored. The output verifies that the logic works as expected.

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

### Prompt:

Generate a Python program for a library management system using classes, loops, and conditional statements.

## Code & Output:

```
File Edit Selection View Go Run Terminal Help AI Assisted Coding
Assignment_6.5.py X
Assignment_6.5 > Assignment_6.5.py > ...
25 "Task 3: AI-Assisted Code Completion Reflection Task"
26 # Generate a python program for a library management system using classes,
    loops, and conditional statements.
27 class Book:
28     def __init__(self, title, author):
29         self.title = title
30         self.author = author
31         self.is_borrowed = False
32
33     def borrow(self):
34         if not self.is_borrowed:
35             self.is_borrowed = True
36             return True
37         return False
38
39     def return_book(self):
40         if self.is_borrowed:
41             self.is_borrowed = False
42             return True
43         return False
44
45 class Library:
46     def __init__(self):
47         self.books = []
48
49     def add_book(self, book):
50         self.books.append(book)
51
52     def display_books(self):
53         for idx, book in enumerate(self.books):
54             status = "Borrowed" if book.is_borrowed else "Available"
55             print(f'{idx + 1}. {book.title} by {book.author} - {status}')
56
57     def borrow_book(self, index):
58         if 0 <= index < len(self.books):
59             if self.books[index].borrow():
60                 print(f'You have borrowed '{self.books[index].title}'.')
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_6.5\Assignment_6.5.py"
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 2
Enter the book index to borrow: 3
You have borrowed 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book index to return: 3
You have returned 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed
```

```
File Edit Selection View Go Run Terminal Help AI Assisted Coding
Assignment_6.5.py X
Assignment_6.5 > Assignment_6.5.py > main
44 class Library:
45     def borrow_book(self, index):
46         else:
47             print(f'{self.books[index].title}' is already borrowed.")
48         else:
49             print("Invalid book index.")
50
51     def return_book(self, index):
52         if 0 <= index < len(self.books):
53             if self.books[index].return_book():
54                 print(f'You have returned '{self.books[index].title}'.')
```

```
PS E:\3rd Year\2nd Sem\AI Assisted coding> python -u "e:\3rd Year\2nd Sem\AI Assisted coding\Assignment_6.5\Assignment_6.5.py"
Enter the book index to borrow: 3
You have borrowed 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book index to return: 3
You have returned 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Available

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
```

The screenshot shows a VS Code editor with a Python file named `Assignment_6.5.py`. The code defines a `Library` class and a `main` function. The `main` function initializes a `Library` object with three books: "1984" by George Orwell, "To Kill a Mockingbird" by Harper Lee, and "The Great Gatsby" by F. Scott Fitzgerald. It then enters a `while True` loop that displays a menu with four options: 1. Display Books, 2. Borrow Book, 3. Return Book, and 4. Exit. The user enters '3' to return a book. The program prompts for the book index to return (3) and displays the returned book: "The Great Gatsby". The terminal output on the right shows the execution of the program, including the menu display and the successful return of the book.

```
def main():
    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: ")

        if choice == '1':
            library.display_books()
        elif choice == '2':
            index = int(input("Enter the book index to borrow: ")) - 1
            library.borrow_book(index)
        elif choice == '3':
            index = int(input("Enter the book index to return: ")) - 1
            library.return_book(index)
        elif choice == '4':
            print("Exiting the library system.")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Codings> python -u "e:\3rd Year\2nd Sem\AI Assisted coding\Assignment_6.5\Assignment_6.5.py"
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book index to return: 3
You have returned "The Great Gatsby".

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Available

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 4
Exiting the library system.
PS E:\3rd Year\2nd Sem\AI Assisted Codings>
```

## Explanation:

The AI-generated program uses a class to represent a library and includes loops and conditional statements for menu-driven interaction. The loop allows continuous user input, and conditionals control program flow. The program correctly demonstrates AI-assisted use of object-oriented programming concepts.

## Reflection on AI-Assisted Coding:

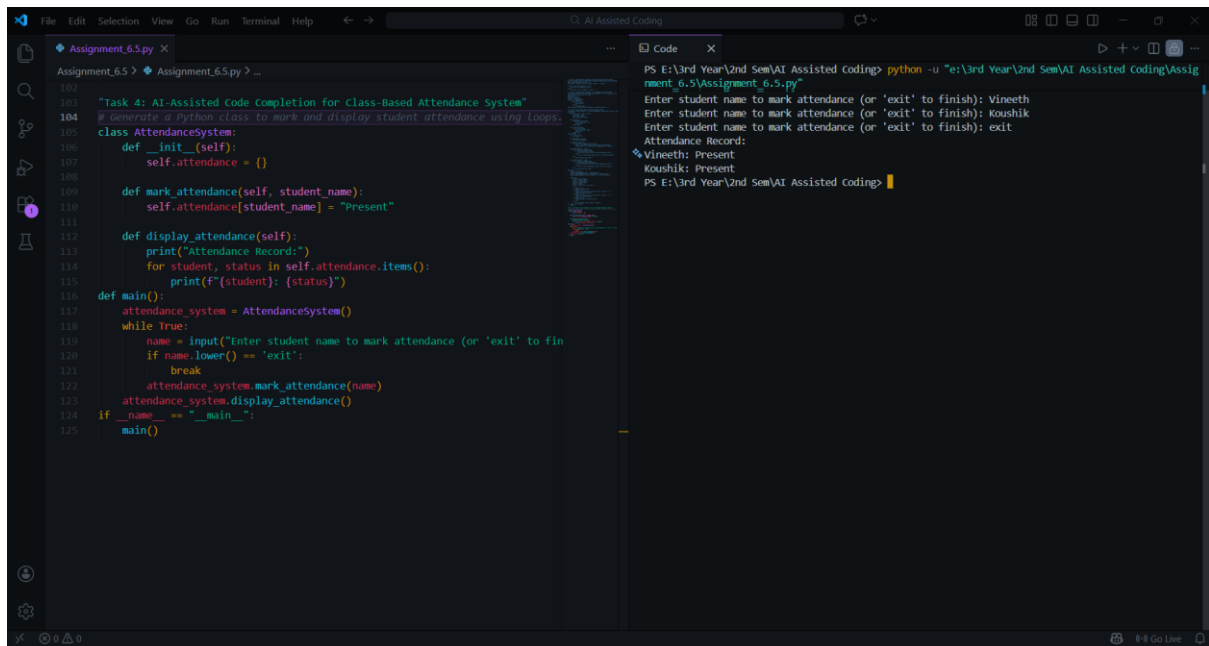
The AI tool generated a complete and functional program quickly. While the logic is correct, the code can be further improved with input validation and advanced features. This task shows that AI is useful for speeding up development but still requires human review and optimization.

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

### Prompt:

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

### Code & Output:



```
103 # Task 4: AI-Assisted Code Completion for Class-Based Attendance System
104 # Generate a Python class to mark and display student attendance using loops.
105 class AttendanceSystem:
106     def __init__(self):
107         self.attendance = {}
108
109     def mark_attendance(self, student_name):
110         self.attendance[student_name] = "Present"
111
112     def display_attendance(self):
113         print("Attendance Record:")
114         for student, status in self.attendance.items():
115             print(f"{student}: {status}")
116
117 def main():
118     attendance_system = AttendanceSystem()
119     while True:
120         name = input("Enter student name to mark attendance (or 'exit' to finish): ")
121         if name.lower() == 'exit':
122             break
123         attendance_system.mark_attendance(name)
124     attendance_system.display_attendance()
125 if __name__ == "__main__":
126     main()
```

PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment\_6.5\Assignment\_6.5.py"

Enter student name to mark attendance (or 'exit' to finish): Vineeth  
Enter student name to mark attendance (or 'exit' to finish): Koushik  
Enter student name to mark attendance (or 'exit' to finish): exit  
Attendance Record:  
Vineeth: Present  
Koushik: Present  
PS E:\3rd Year\2nd Sem\AI Assisted Coding>

## Explanation:

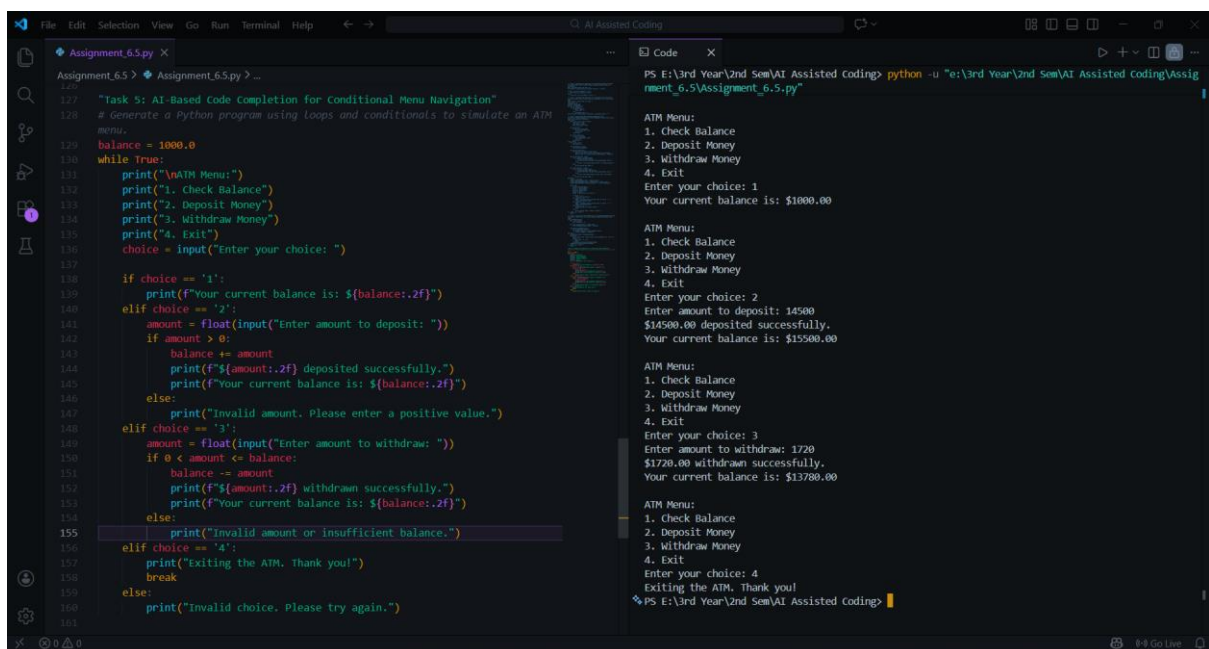
The AI-generated attendance system uses a class to store attendance data. A loop is used to take multiple student entries, and another loop displays the attendance records. The code works correctly and demonstrates class-based AI code completion.

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

### Prompt:

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

### Code & Output:



```
127 # Task 5: AI-Based Code Completion for Conditional Menu Navigation
128 # Generate a Python program using loops and conditionals to simulate an ATM menu.
129 balance = 1000.0
130 while True:
131     print("\nATM Menu:")
132     print("1. Check Balance")
133     print("2. Deposit Money")
134     print("3. Withdraw Money")
135     print("4. Exit")
136     choice = input("Enter your choice: ")
137
138     if choice == '1':
139         print(f"Your current balance is: ${balance:.2f}")
140     elif choice == '2':
141         amount = float(input("Enter amount to deposit: "))
142         if amount > 0:
143             balance += amount
144             print(f"${amount:.2f} deposited successfully.")
145             print(f"Your current balance is: ${balance:.2f}")
146         else:
147             print("Invalid amount. Please enter a positive value.")
148     elif choice == '3':
149         amount = float(input("Enter amount to withdraw: "))
150         if 0 < amount <= balance:
151             balance -= amount
152             print(f"${amount:.2f} withdrawn successfully.")
153             print(f"Your current balance is: ${balance:.2f}")
154         else:
155             print("Invalid amount or insufficient balance.")
156     elif choice == '4':
157         print("Exiting the ATM. Thank you!")
158         break
159     else:
160         print("Invalid choice. Please try again.")
161
```

ATM Menu:  
1. Check Balance  
2. Deposit Money  
3. Withdraw Money  
4. Exit  
Enter your choice: 1  
Your current balance is: \$1000.00

ATM Menu:  
1. Check Balance  
2. Deposit Money  
3. Withdraw Money  
4. Exit  
Enter your choice: 2  
Enter amount to deposit: 14500  
\$14500.00 deposited successfully.  
Your current balance is: \$15500.00

ATM Menu:  
1. Check Balance  
2. Deposit Money  
3. Withdraw Money  
4. Exit  
Enter your choice: 3  
Enter amount to withdraw: 1720  
\$1720.00 withdrawn successfully.  
Your current balance is: \$13780.00

ATM Menu:  
1. Check Balance  
2. Deposit Money  
3. Withdraw Money  
4. Exit  
Enter your choice: 4  
Exiting the ATM. Thank you!  
PS E:\3rd Year\2nd Sem\AI Assisted Coding>

**Explanation:**

The AI-generated ATM program uses a loop to display the menu repeatedly and conditional statements to handle user choices. The logic correctly updates the balance and prevents invalid withdrawals. This task demonstrates effective AI-based code completion for menu-driven programs.

**Final Conclusion:**

This experiment shows how AI-based code completion tools can generate useful Python code involving classes, loops, and conditionals. While AI speeds up development, developers must still review logic, handle edge cases, and ensure ethical and responsible use of AI-generated code.