

AI Assisted Coding

Assignment – 6.5

Name: J.Akshaya

Batch: 22

HtNo: 2303A51629

Task Description-1

- (AI-Based Code Completion for Conditional Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

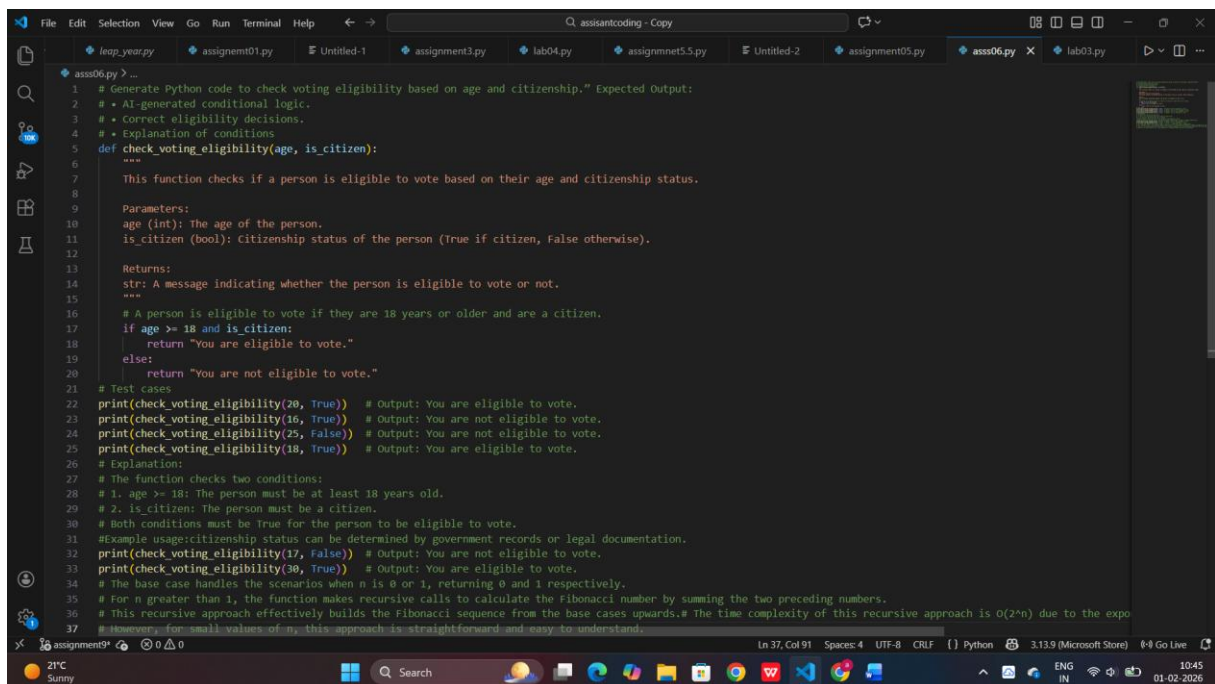
Prompt:

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

Expected Output:

- AI-generated conditional logic.
- Correct eligibility decisions.
- Explanation of conditions.

Code:



```
1 # Generate Python code to check voting eligibility based on age and citizenship." Expected Output:
2 # • AI-generated conditional logic.
3 # • Correct eligibility decisions.
4 # • Explanation of conditions
5 def check_voting_eligibility(age, is_citizen):
6     """
7     This function checks if a person is eligible to vote based on their age and citizenship status.
8
9     Parameters:
10    age (int): The age of the person.
11    is_citizen (bool): Citizenship status of the person (True if citizen, False otherwise).
12
13    Returns:
14    str: A message indicating whether the person is eligible to vote or not.
15    """
16    # A person is eligible to vote if they are 18 years or older and are a citizen.
17    if age >= 18 and is_citizen:
18        return "You are eligible to vote."
19    else:
20        return "You are not eligible to vote."
21
22    # Test cases
23    print(check_voting_eligibility(20, True)) # Output: You are eligible to vote.
24    print(check_voting_eligibility(16, True)) # Output: You are not eligible to vote.
25    print(check_voting_eligibility(25, False)) # Output: You are not eligible to vote.
26    print(check_voting_eligibility(18, True)) # Output: You are eligible to vote.
27
28    # Explanation:
29    # The function checks two conditions:
30    # 1. age >= 18: The person must be at least 18 years old.
31    # 2. is_citizen: The person must be a citizen.
32    # Both conditions must be true for the person to be eligible to vote.
33    # Example usage: citizenship status can be determined by government records or legal documentation.
34    print(check_voting_eligibility(17, False)) # Output: You are not eligible to vote.
35    print(check_voting_eligibility(30, True)) # Output: You are eligible to vote.
36
37    # The base case handles the scenarios when n is 0 or 1, returning 0 and 1 respectively.
38    # For n greater than 1, the function makes recursive calls to calculate the Fibonacci number by summing the two preceding numbers.
39    # This recursive approach effectively builds the Fibonacci sequence from the base cases upwards. # The time complexity of this recursive approach is O(2^n) due to the expo
40    # However, for small values of n, this approach is straightforward and easy to understand.
```

```

29
30     # Check voting eligibility
31     if age >= 18 and citizenship_status.lower() == "citizen":
32         return "Eligible to vote."
33     else:
34         return "Not eligible to vote."
35 # Example usage:
36 if __name__ == "__main__":
37     test_cases = [
38         (20, "citizen"),
39         (17, "citizen"),
40         (25, "non citizen"),
41         (18, ""),
42         (None, "citizen"),
43         ("eighteen", "citizen"),
44         (22, None),
45         (-5, "citizen")
46     ]
47
48     for age, status in test_cases:
49         result = check_voting_eligibility(age, status)
50         print(f"Age: {age}, Citizenship Status: '{status}' => {result}")

```

Output:

```

PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:\Users\aksha\AppData\Local\Microsoft\WindowsApps\python3.13.exe "C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy\asss06.py"
You are eligible to vote.
You are not eligible to vote.
You are eligible to vote.
You are not eligible to vote.
You are eligible to vote.
You are not eligible to vote.
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy>

```

Explanation:

The AI-generated Python program was used to design a voting eligibility check with proper input validation and clear decision logic. The code was carefully reviewed line by line to understand how age and citizenship inputs are validated, including handling missing, invalid, and incorrect values. Logical flaws and potential errors were identified and addressed by adding appropriate condition checks and exception handling. The program was further refined to improve readability and maintainability through meaningful variable names, structured validation, and clear comments. Responsible use of AI tools was ensured by verifying the logic, correcting mistakes, and fully understanding the code rather than copying the AI-generated output without evaluation.

Task Description-2

- (AI-Based Code Completion for Loop-Based String Processing)

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

Prompt:

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

Expected Output:

- AI-generated string processing logic.
- Correct counts.
- Output verification.

Code:

```

42
43 # • AI-generated string processing logic.
44 # • Correct counts.
45 # • Output verification. Code:
46 def count_vowels_and_consonants(input_string):
47     """
48     This function counts the number of vowels and consonants in a given string.
49
50     Parameters:
51     input_string (str): The string to be analyzed.
52
53     Returns:
54     tuple: A tuple containing the count of vowels and consonants.
55     """
56     vowels = "aeiouAEIOU"
57     vowel_count = 0
58     consonant_count = 0
59
60     for char in input_string:
61         if char.isalpha(): # Check if the character is an alphabet
62             if char in vowels:
63                 vowel_count += 1
64             else:
65                 consonant_count += 1
66
67     return vowel_count, consonant_count
68
69 # Test cases
70 test_string1 = "Hello World"
71 vowels1, consonants1 = count_vowels_and_consonants(test_string1)
72 print(f"Input String: '{test_string1}' -> Vowels: {vowels1}, Consonants: {consonants1}")
73 test_string2 = "Python Programming"
74 vowels2, consonants2 = count_vowels_and_consonants(test_string2)
75 print(f"Input String: '{test_string2}' -> Vowels: {vowels2}, Consonants: {consonants2}")
76 test_string3 = "AI is amazing!"
77 vowels3, consonants3 = count_vowels_and_consonants(test_string3)
78 print(f"Input String: '{test_string3}' -> Vowels: {vowels3}, Consonants: {consonants3}")
79

```

```

32         vowel_count += 1
33     else:
34         consonant_count += 1
35
36     # Return the counts
37     return f"Vowels: {vowel_count}, Consonants: {consonant_count}"
38
39 # Example usage:
40 if __name__ == "__main__":
41     test_strings = [
42         "Hello World",
43         "Python Programming",
44         "",
45         None,
46         12345,
47         "AEIOU and sometimes Y"
48     ]
49
50     for test_str in test_strings:
51         result = count_vowels_and_consonants(test_str)
52         print(f"Input: '{test_str}' => {result}")

```

Output:

```
File Edit Selection View Go Run Terminal Help ← → Q assisantcoding - Copy + - [ ] ×
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + - [ ] ×
Python
Python

PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:\Users\aksha\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy\ass
06.py"
You are eligible to vote.
You are not eligible to vote.
You are not eligible to vote.
You are eligible to vote.
You are not eligible to vote.
You are eligible to vote.
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:\Users\aksha\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy\asss
06.py"
Input String: 'Hello World' -> Vowels: 3, Consonants: 7
Input String: 'Python Programming' -> Vowels: 4, Consonants: 13
Input String: 'AI is amazing!' -> Vowels: 6, Consonants: 5
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy>

Ln 85, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.9 (Microsoft Store) Go Live
21°C Sunny Search 10:48 01-02-2026
```

Explanation :

The AI tool was used to generate a Python program that applies a user-defined function, loop-based logic, and conditional statements to count vowels and consonants in a string. The generated code was carefully examined line by line to understand how input validation, character checking, and counting logic work together to produce correct results. During the review process, potential issues such as empty inputs, non-string values, and invalid characters were identified and handled appropriately to ensure logical correctness. The program was further refined to improve readability and efficiency through clear comments, structured conditions, and meaningful variable names. Responsible use of AI tools was demonstrated by validating the generated solution, correcting errors, and ensuring a clear understanding of the logic rather than using the output without evaluation.

Task_3

-(AI-Assisted Code Completion Reflection Task)

Task: Use an AI tool to generate a complete program using classes, loops, and conditionals.

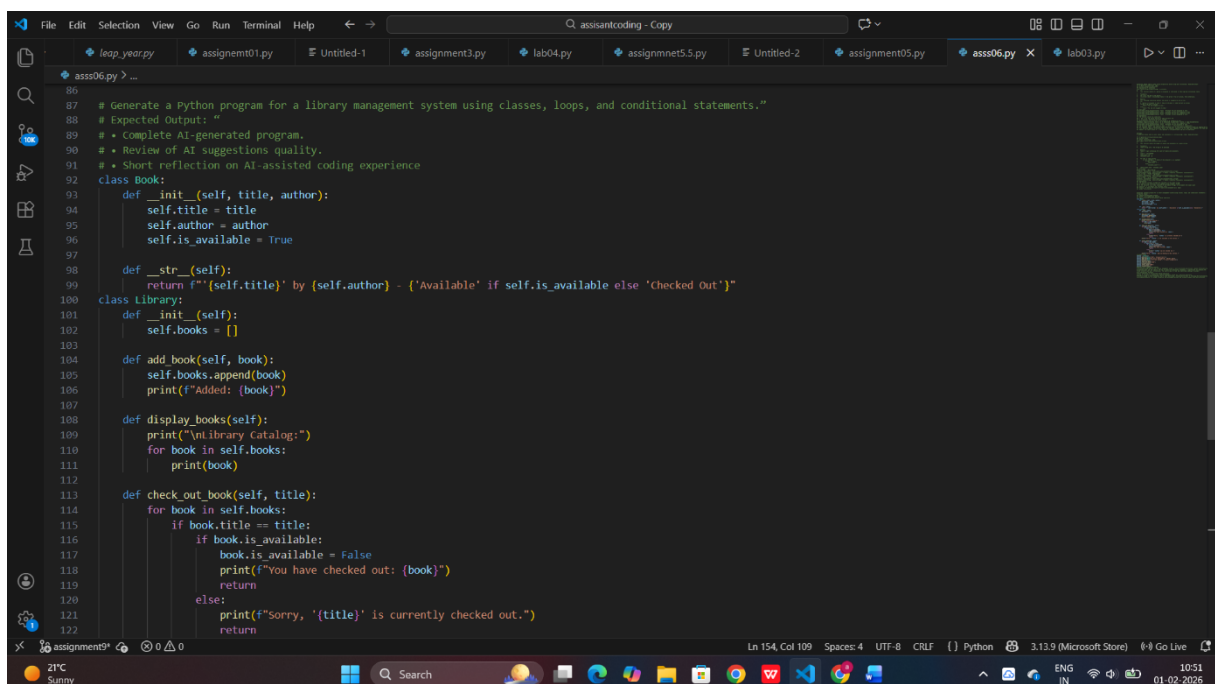
Prompt:

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

Expected Output: “

- Complete AI-generated program.
- Review of AI suggestions quality.
- Short reflection on AI-assisted coding experience

Code:

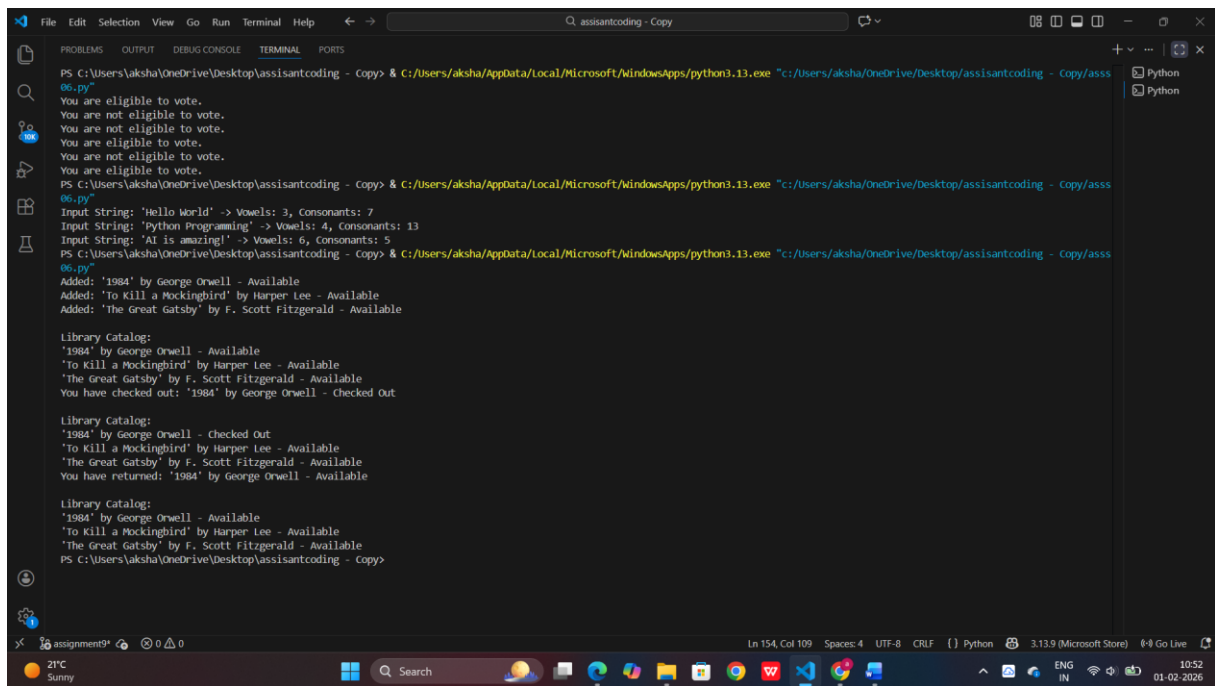


```
86
87 # Generate a Python program for a library management system using classes, loops, and conditional statements."
88 # Expected Output: "
89 # * Complete AI-generated program.
90 # * Review of AI suggestions quality.
91 # * Short reflection on AI-assisted coding experience
92 class Book:
93     def __init__(self, title, author):
94         self.title = title
95         self.author = author
96         self.is_available = True
97
98     def __str__(self):
99         return f"{self.title} by {self.author} - {'Available' if self.is_available else 'Checked Out'}"
100 class Library:
101     def __init__(self):
102         self.books = []
103
104     def add_book(self, book):
105         self.books.append(book)
106         print(f"Added: {book}")
107
108     def display_books(self):
109         print("\nLibrary Catalog:")
110         for book in self.books:
111             print(book)
112
113     def check_out_book(self, title):
114         for book in self.books:
115             if book.title == title:
116                 if book.is_available:
117                     book.is_available = False
118                     print(f"You have checked out: {book}")
119                     return
120                 else:
121                     print(f"Sorry, '{title}' is currently checked out.")
122                     return
```

```
86
87 # Generate a Python program for a library management system using classes, loops, and conditional statements."
88 # Expected Output: "
89 # * Complete AI-generated program.
90 # * Review of AI suggestions quality.
91 # * Short reflection on AI-assisted coding experience
92 class Book:
93     def __init__(self, title, author):
94         self.title = title
95         self.author = author
96         self.is_available = True
97
98     def __str__(self):
99         return f"[{self.title}] by [{self.author}] - {'Available' if self.is_available else 'Checked out'}"
100 class Library:
101     def __init__(self):
102         self.books = []
103
104     def add_book(self, book):
105         self.books.append(book)
106         print(f"Added: {book}")
107
108     def display_books(self):
109         print("\nLibrary Catalog:")
110         for book in self.books:
111             print(book)
112
113     def check_out_book(self, title):
114         for book in self.books:
115             if book.title == title:
116                 if book.is_available:
117                     book.is_available = False
118                     print(f"You have checked out: {book}")
119                     return
120                 else:
121                     print(f"Sorry, '{title}' is currently checked out.")
122                     return
```

```
100 class Library:
113     def check_out_book(self, title):
121         print(f"Sorry, '{title}' is currently checked out.")
122         return
123     print(f"Sorry, '{title}' is not available in the library.")
124
125     def return_book(self, title):
126         for book in self.books:
127             if book.title == title:
128                 if not book.is_available:
129                     book.is_available = True
130                     print(f"You have returned: {book}")
131                     return
132                 else:
133                     print(f"'{title}' was not checked out.")
134                     return
135     print(f"Sorry, '{title}' does not belong to this library.")
136
137 # Example usage
138 library = Library()
139 library.add_book(Book("1984", "George Orwell"))
140 library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
141 library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
142 library.display_books()
143 library.check_out_book("1984")
144 library.display_books()
145 library.return_book("1984")
146 library.display_books()
147
148 # Review of AI suggestions quality:
149 # The AI-generated code for the library management system is well-structured and utilizes classes effectively to
150 # represent books and the library. The use of loops and conditional statements is appropriate for managing the
151 # book inventory and user interactions. The code is easy to read and understand, making it suitable
152 # for educational purposes.
153 # Short reflection on AI-assisted coding experience:
154 # Using AI to assist in coding this library management system was a positive experience.
155 # The AI provided a solid foundation that I could build upon, allowing me to focus on refining the functionality
156 # and adding features. It helped streamline the development process and offered insights into best practices
```

Output:



```
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:/Users/aksha/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/aksha/OneDrive/Desktop/assisantcoding - Copy/ass06.py"
You are eligible to vote.
You are not eligible to vote.
You are not eligible to vote.
You are eligible to vote.
You are not eligible to vote.
You are eligible to vote.
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:/Users/aksha/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/aksha/OneDrive/Desktop/assisantcoding - Copy/ass06.py"
Input String: 'Hello World' -> Vowels: 3, Consonants: 7
Input String: 'Python Programming' -> Vowels: 4, Consonants: 13
Input String: 'AI is amazing!' -> Vowels: 6, Consonants: 5
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:/Users/aksha/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/aksha/OneDrive/Desktop/assisantcoding - Copy/ass06.py"
Added: '1984' by George Orwell - Available
Added: 'To Kill a Mockingbird' by Harper Lee - Available
Added: 'The Great Gatsby' by F. Scott Fitzgerald - Available

Library Catalog:
'1984' by George Orwell - Available
'To Kill a Mockingbird' by Harper Lee - Available
'The Great Gatsby' by F. Scott Fitzgerald - Available
You have checked out: '1984' by George Orwell - Checked Out

Library Catalog:
'1984' by George Orwell - Checked Out
'To Kill a Mockingbird' by Harper Lee - Available
'The Great Gatsby' by F. Scott Fitzgerald - Available
You have returned: '1984' by George Orwell - Available

Library Catalog:
'1984' by George Orwell - Available
'To Kill a Mockingbird' by Harper Lee - Available
'The Great Gatsby' by F. Scott Fitzgerald - Available
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy>
```

Explanation:

The AI-generated Python program was used to design a simple library management system using a class-based approach along with loops and conditional statements for menu-driven operations. The code was carefully reviewed line by line to understand how object-oriented concepts, such as classes, methods, and instance variables, manage book records and availability status. Logical conditions were examined to ensure correct handling of operations like adding, displaying, issuing, and returning books, while also preventing invalid actions such as issuing an already issued book. The program was refined for better readability and maintainability through meaningful method names, clear comments, and structured input validation within the menu loop. Responsible use of AI tools was demonstrated by verifying the generated logic, handling user input errors properly, and ensuring a clear understanding of the program's functionality rather than relying on the AI output without evaluation.

Task-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."

Expected Output:

- AI-generated attendance logic.
- Correct display of attendance.
- Test cases

Code:


```
TempRun.py > ...
1
2 Generate a Python class for an attendance management system.
3 The class should mark and display student attendance using loops.
4 Include methods to add students, mark present or absent, and display attendance.
5 Use conditional statements, handle basic invalid inputs, and add comments explaining the logic.
6 Generate a complete executable program.
7 """
8 class AttendanceManagementSystem:
9     """
10     A class to manage student attendance.
11     """
12
13     def __init__(self):
14         """
15         Initialize the attendance system with an empty dictionary.
16         Key -> student name
17         Value -> attendance status (Present / Absent)
18         """
19         self.students = {}
20
21     def add_student(self, student_name):
22         """
23         Add a student to the system.
24         Default attendance is marked as Absent.
25         """
26         # Validate input
27         if not student_name or not isinstance(student_name, str):
28             print("Error: Student name must be a valid non-empty string.")
29             return
30
31         # Check if student already exists
32         if student_name in self.students:
```

```
33             print(f"Student '{student_name}' already exists.")
34         else:
35             self.students[student_name] = "Absent"
36             print(f"Student '{student_name}' added successfully.")
37
38     def mark_attendance(self, student_name, status):
39         """
40         Mark attendance as Present or Absent.
41         """
42         # Check if student exists
43         if student_name not in self.students:
44             print(f"Error: Student '{student_name}' not found.")
45             return
46
47         # Validate status
48         if status not in ["Present", "Absent"]:
49             print("Error: Status must be 'Present' or 'Absent'.")
50             return
51
52         self.students[student_name] = status
53         print(f"Attendance marked: {student_name} -> {status}")
54
55     def display_attendance(self):
56         """
57         Display attendance of all students.
58         """
59         if not self.students:
60             print("No students found in the system.")
61             return
```

```

TempRun.py > ...
8   class AttendanceManagementSystem:
55   def display_attendance(self):
63       print("\n--- Student Attendance ---")
64       for student, status in self.students.items():
65           print(f"{student}: {status}")
66
67
68   def main():
69       """
70       Main function to run the menu-driven attendance system.
71       """
72       attendance_system = AttendanceManagementSystem()
73
74       while True:
75           print("\nAttendance Management System Menu")
76           print("1. Add Student")
77           print("2. Mark Attendance")
78           print("3. Display Attendance")
79           print("4. Exit")
80
81           choice = input("Enter your choice (1-4): ").strip()
82
83           if choice == "1":
84               name = input("Enter student name: ").strip()
85               attendance_system.add_student(name)
86
87           elif choice == "2":
88               name = input("Enter student name: ").strip()
89               status = input("Enter status (Present/Absent): ").strip().capitalize()
90               attendance_system.mark_attendance(name, status)
91
92           elif choice == "3":

```

```

93       attendance_system.display_attendance()
94
95       elif choice == "4":
96           print("Exiting Attendance Management System. Goodbye!")
97           break
98
99       else:
100           print("Invalid choice. Please enter a number between 1 and 4.")
101
102
103 # Program execution starts here
104 if __name__ == "__main__":
105     main()
106
107

```

Output:

```

1. Add Student
2. Mark Attendance
3. Display Attendance
4. Exit
Enter your choice (1-4): 1
Enter student name: Deekshith
Student 'Deekshith' added successfully.

Attendance Management System Menu
1. Add Student
2. Mark Attendance
3. Display Attendance
4. Exit
Enter your choice (1-4): 2
Enter student name: Deekshith
Enter status (Present/Absent): Present
Attendance marked: Deekshith → Present

Attendance Management System Menu
1. Add Student
2. Mark Attendance
3. Display Attendance
4. Exit
Enter your choice (1-4): 3

--- Student Attendance ---
Deekshith: Present

Attendance Management System Menu
1. Add Student
2. Mark Attendance
3. Display Attendance
4. Exit

```

Explanation:

The AI-generated Python program was used to develop an attendance management system using a class-based structure along with loops and conditional statements for menu-driven operations. The code was reviewed in detail to understand how methods are used to add students, mark attendance as present or absent, and display attendance records using dictionary-based storage. Conditional checks were analysed to ensure proper handling of invalid inputs such as empty names, incorrect attendance status, and non-existing students. The implementation was refined to improve readability and maintainability by using meaningful method names, clear comments, and structured control flow within the loop. Responsible use of AI tools was demonstrated by validating the generated logic, handling edge cases correctly, and ensuring a clear understanding of the program's functionality rather than relying on the AI output without verification.

Task Description-5

- (AI-Based Code Completion for Conditional Menu Navigation)

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

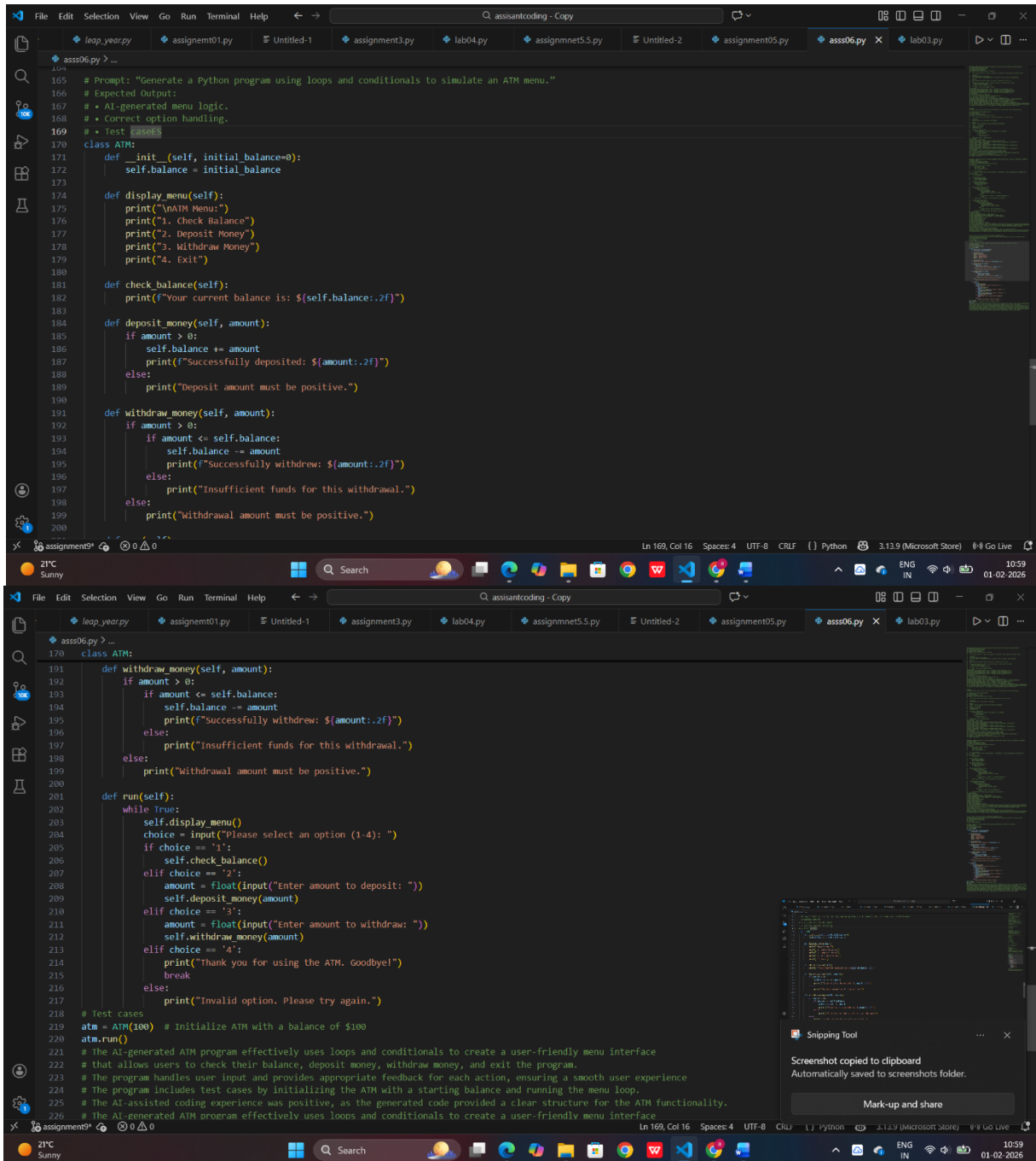
Prompt: "Generate a Python program using loops and conditionals to simulate an ATM menu."

Expected Output:

- AI-generated menu logic.
- Correct option handling.

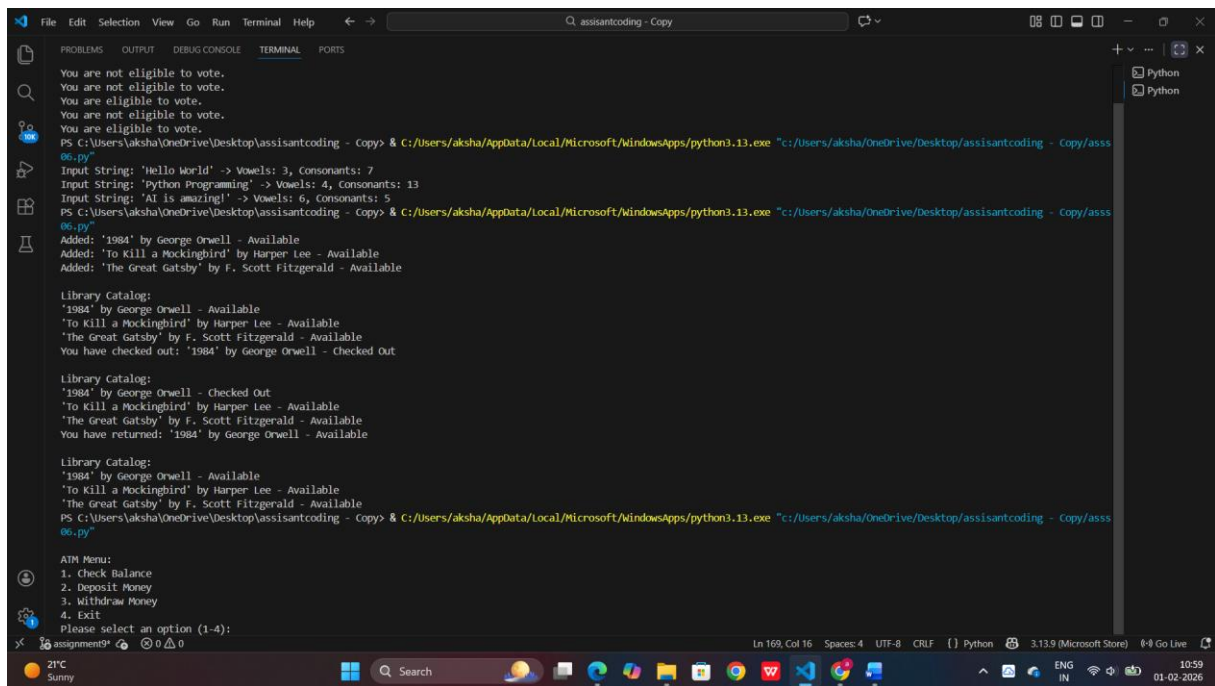
- Output verification.

Code:



```
165 # Prompt: "Generate a Python program using loops and conditionals to simulate an ATM menu."
166 # Expected Output:
167 # * AI-generated menu logic.
168 # * Correct option handling.
169 # * Test cases
170 class ATM:
171     def __init__(self, initial_balance=0):
172         self.balance = initial_balance
173
174     def display_menu(self):
175         print("\nATM Menu:")
176         print("1. Check Balance")
177         print("2. Deposit Money")
178         print("3. Withdraw Money")
179         print("4. Exit")
180
181     def check_balance(self):
182         print(f"Your current balance is: ${self.balance:.2f}")
183
184     def deposit_money(self, amount):
185         if amount > 0:
186             self.balance += amount
187             print(f"Successfully deposited: ${amount:.2f}")
188         else:
189             print("Deposit amount must be positive.")
190
191     def withdraw_money(self, amount):
192         if amount > 0:
193             if amount <= self.balance:
194                 self.balance -= amount
195                 print(f"Successfully withdrew: ${amount:.2f}")
196             else:
197                 print("Insufficient funds for this withdrawal.")
198         else:
199             print("Withdrawal amount must be positive.")
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
# Test cases
atm = ATM(100) # Initialize ATM with a balance of $100
atm.run()
# The AI-generated ATM program effectively uses loops and conditionals to create a user-friendly menu interface
# that allows users to check their balance, deposit money, withdraw money, and exit the program.
# The program handles user input and provides appropriate feedback for each action, ensuring a smooth user experience
# The program includes test cases by initializing the ATM with a starting balance and running the menu loop.
# The AI-assisted coding experience was positive, as the generated code provided a clear structure for the ATM functionality.
# The AI-generated ATM program effectively uses loops and conditionals to create a user-friendly menu interface
```

Output:



```
File Edit Selection View Go Run Terminal Help
assisantcoding - Copy

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

You are not eligible to vote.
You are not eligible to vote.
You are eligible to vote.
You are not eligible to vote.
You are eligible to vote.
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:/Users/aksha/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/aksha/OneDrive/Desktop/assisantcoding - Copy/ass06.py"
Input String: 'Hello World' -> Vowels: 3, Consonants: 7
Input String: 'Python Programming' -> Vowels: 4, Consonants: 13
Input String: 'AI is amazing!' -> Vowels: 6, Consonants: 5
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:/Users/aksha/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/aksha/OneDrive/Desktop/assisantcoding - Copy/ass06.py"
Added: '1984' by George Orwell - Available
Added: 'To Kill a Mockingbird' by Harper Lee - Available
Added: 'The Great Gatsby' by F. Scott Fitzgerald - Available

Library Catalog:
'1984' by George Orwell - Available
'To Kill a Mockingbird' by Harper Lee - Available
'The Great Gatsby' by F. Scott Fitzgerald - Available
You have checked out: '1984' by George Orwell - Checked Out

Library Catalog:
'1984' by George Orwell - Checked Out
'To Kill a Mockingbird' by Harper Lee - Available
'The Great Gatsby' by F. Scott Fitzgerald - Available
You have returned: '1984' by George Orwell - Available

Library Catalog:
'1984' by George Orwell - Available
'To Kill a Mockingbird' by Harper Lee - Available
'The Great Gatsby' by F. Scott Fitzgerald - Available
PS C:\Users\aksha\OneDrive\Desktop\assisantcoding - Copy> & C:/Users/aksha/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/aksha/OneDrive/Desktop/assisantcoding - Copy/ass06.py"

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4):
```

Explanation:

The AI-generated Python program was used to simulate an ATM system using loops and conditional statements to provide a menu-driven interface. The code was carefully examined to understand how the loop keeps the menu running until the user chooses to exit and how conditional branches handle balance inquiry, deposit, and withdrawal operations. Input validation was analysed to ensure that invalid menu selections, non-numeric inputs, negative amounts, and insufficient balance cases are handled correctly. The program structure was reviewed to identify and prevent logical errors, while clear comments and meaningful variable names were used to improve readability and maintainability. Responsible use of AI tools was demonstrated by verifying the generated logic, testing different transaction scenarios, and ensuring correct behaviour rather than relying on the AI output without evaluation.