

AI ASSIGNMENT-6.5

Name : D.Varshitha

Hall No. : 2303A52268

Batch : 36

Task Description #1 (AI-Based Code Completion for Conditional

Eligibility Check)

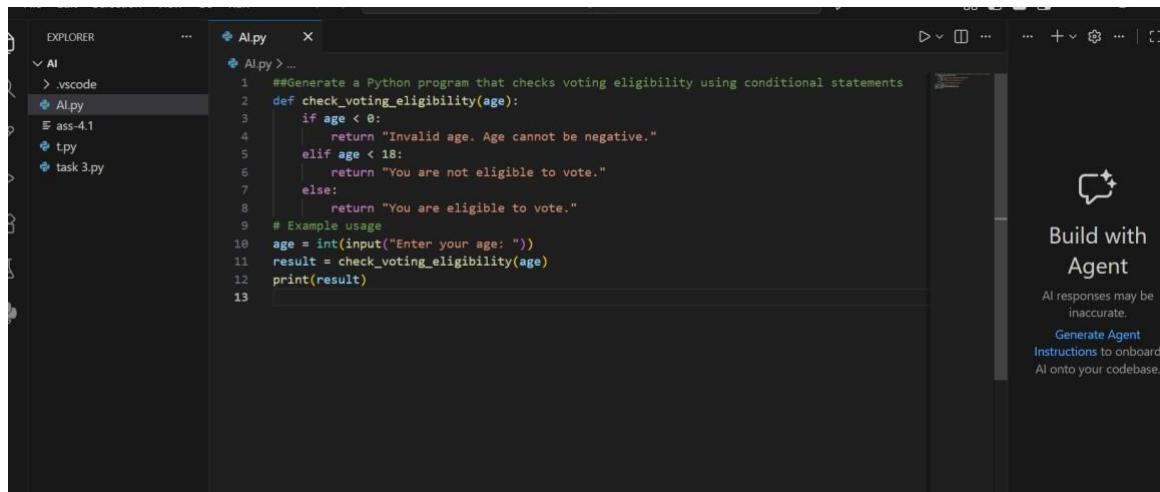
Task: Use an AI tool to generate eligibility logic.

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.



The screenshot shows a dark-themed instance of Visual Studio Code. In the left sidebar, the 'EXPLORER' view shows a folder named 'AI' containing files like '.vscode', 'AI.py', 'ass-4.1', 'tpy', and 'task 3.py'. The main 'CODE' editor window displays the following Python code:

```
##Generate a Python program that checks voting eligibility using conditional statements
def check_voting_eligibility(age):
    if age < 0:
        return "Invalid age. Age cannot be negative."
    elif age < 18:
        return "You are not eligible to vote."
    else:
        return "You are eligible to vote."
# Example usage
age = int(input("Enter your age: "))
result = check_voting_eligibility(age)
print(result)
```

To the right of the editor, there's a 'Build with Agent' panel. It features a speech bubble icon, the text 'Build with Agent', and a note: 'AI responses may be inaccurate.' Below that, it says 'Generate Agent Instructions to onboard AI onto your codebase.'

Output:

The screenshot shows the VS Code interface with the AI extension active. The Explorer sidebar shows files like AI.py, ass-4.1, tpy, and task 3.py. The AI.py file is open in the editor, displaying the following Python code:

```
#Generate a Python program that checks voting eligibility using conditional statements
def check_voting_eligibility(age):
    if age < 0:
        return "Invalid age. Age cannot be negative."
    elif age < 18:
        You are eligible to vote.
```

The terminal below shows the output of running the code with age 19:

```
PS C:\Users\reddy\OneDrive\Desktop\AI> & c:\Users\reddy\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/reddy/OneDrive/Desktop/AI/AI.py
Enter your age: 19
You are eligible to vote.
PS C:\Users\reddy\OneDrive\Desktop\AI>
```

A sidebar on the right titled "Build with Agent" contains instructions to onboard AI onto the codebase.

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.

The screenshot shows the VS Code interface with the AI extension active. The Explorer sidebar shows files like AI.py, ass-4.1, tpy, and task 3.py. The AI.py file is open in the editor, displaying the following Python code:

```
##Generate Python code to count the number of vowels and consonants in a given string using loops
#The program should take a string as input from the user and display the total count of vowels and consonants.
def count_vowels_and_consonants(input_string):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in input_string:
        if char.isalpha(): # Check if the character is a letter
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count

# Example usage
input_string = input("Enter a string: ")
vowels, consonants = count_vowels_and_consonants(input_string)
print(f"Number of vowels: {vowels}")
print(f"Number of consonants: {consonants}")
```

The terminal below shows the output of running the code with the input "Hello, World!"

```
Number of consonants: 8
PS C:\Users\reddy\OneDrive\Desktop\AI>
```

A sidebar on the right titled "Build with Agent" contains instructions to onboard AI onto the codebase.

Output:

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows files: AI, .vscode, Al.py, ass-4.1, tpy, task 3.py.
- Terminal:** Displays Python code for counting vowels and consonants in a string. The code includes a docstring, variable declarations, and a loop. It also shows the execution of the program and its output for the string "Ashwitha".
- Output:** Shows the terminal output: "thon3.13.exe c:/Users/reddy/OneDrive/Desktop/AI/AI.py", "thon3.13.exe c:/Users/reddy/OneDrive/Desktop/AI/AI.py", "Enter a string: Ashwitha", "Enter a string: Ashwitha", "Number of vowels: 3", "Number of consonants: 5", "PS C:\Users\reddy\OneDrive\Desktop\AI>".
- AI Assistant Panel:** On the right, there's an AI panel with a speech bubble icon. It says "Build with Agent" and "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." Below it is a "Describe what to build next" input field and some icons.

Task Description #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.

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

- File Explorer:** Shows files: AI, .vscode, ass-4.1, tpy, task 3.py.
- Code Editor:** Displays Python code for a library management system. The code defines classes Book and Library, and methods like __init__, __str__, add_book, display_books, and check_out_book.
- Right Panel:** Features a "Build with Agent" sidebar. It includes a speech bubble icon, a message "Build with Agent", and a note "AI responses may be inaccurate". Buttons for "Generate Agent", "Instructions to onboard AI onto your codebase", and a "Describe what to build next" input field.
- Bottom Bar:** Shows file status (Ln 113, Col 5), spaces (Spaces: 4), encoding (UTF-8), CRLF, Python interpreter (Python 3.13.9.final.0), and system status (ENG IN, 11:37, 23-01-2026).

This screenshot is nearly identical to the first one, showing the same code editor content and right-panel "Build with Agent" interface. The only difference is the timestamp at the bottom bar, which is now 11:37 on 23-01-2026.

The image shows two side-by-side instances of the Visual Studio Code (VS Code) code editor. Both instances are displaying the same Python script, `Al.py`, which implements a simple Library Management System.

Script Content:

```
81     print("Book '{title}' not found in the library.")
82 def main():
83     library = Library()
84     while True:
85         print("\nLibrary Management System")
86         print("1. Add Book")
87         print("2. Display Books")
88         print("3. Check Out Book")
89         print("4. Return Book")
90         print("5. Exit")
91         choice = input("Enter your choice (1-5): ")
92
93         if choice == '1':
94             title = input("Enter book title: ")
95             author = input("Enter book author: ")
96             book = Book(title, author)
97             library.add_book(book)
98         elif choice == '2':
99             library.display_books()
100        elif choice == '3':
101            title = input("Enter the title of the book to check out: ")
102            library.check_out_book(title)
103        elif choice == '4':
104            title = input("Enter the title of the book to return: ")
105            library.return_book(title)
106        elif choice == '5':
107            print("Exiting the Library Management System. Goodbye!")
108            break
109        else:
110            print("Invalid choice. Please try again.")
111    if __name__ == "__main__":
112        main()
```

VS Code Interface Details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Left Sidebar (Explorer):** Shows files in the workspace, including `.vscode`, `Al.py`, `ass-4.1`, `tpy`, and `task 3.py`.
- Right Sidebar:** "Build with Agent" panel with instructions for AI integration.
- Bottom Status Bar:** Labeled "Al.py", "defaultInterpreterPath: 3.13.9.final.0", "ENG IN", and the date "23-01-2026".

The two instances of the code editor are identical, showing the same code and interface, suggesting they are running on the same machine or connected to the same environment.

Output:

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

- File Explorer:** Shows files in the AI folder, including AI.py, ass-4.1, tpy, and task 3.py.
- Code Editor:** Displays the AI.py script:

```
100     print("Invalid choice. Please try again.")  
101 if __name__ == "__main__":  
102     main()  
103  
104
```
- Terminal:** Shows the output of running the script:

```
Enter book author: c:\Users\reddy\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/reddy/OneDrive/Desktop/AI/AI.py  
Book 'python' added to the library.  
  
Library Management System  
1. Add Book  
2. Display Books  
3. Check Out Book  
4. Return Book  
5. Exit  
Enter your choice (1-5): 1  
Enter book title: Python Basics  
Enter book author: john  
Book 'Python Basics' added to the library.  
  
Library Management System  
1. Add Book  
2. Display Books  
3. Check Out Book  
4. Return Book  
5. Exit  
Enter your choice (1-5):
```
- AI Sidebar:** Provides AI assistance with the following sections:
 - Build with Agent
 - AI responses may be inaccurate.
 - Generate Agent
 - Instructions to onboard AI onto your codebase.
- Bottom Status Bar:** Shows file path (AI.py), line (Ln 113), column (Col 5), spaces (Spaces: 4), encoding (UTF-8), interpreter (Python), default interpreter path (3.13.9.final.0), and date/time (23-01-2026).

Task Description #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.

The image displays two side-by-side instances of the Microsoft Visual Studio Code (VS Code) code editor, both showing Python code related to student attendance.

Top Window:

- File Explorer:** Shows files: `.vscode`, `AI.py`, `ass-4.1`, `tpy`, and `task 3.py`.
- Code Editor:** Displays the `AI.py` file content:113 ##Generate a Python class to mark and display student attendance using loops.
114 ##The program should allow marking students as present or absent and display the attendance
115 class Student:
116 def __init__(self, name):
117 self.name = name
118 self.attendance = []
119
120 def mark_present(self):
121 self.attendance.append("Present")
122
123 def mark_absent(self):
124 self.attendance.append("Absent")
125
126 def display_attendance(self):
127 print(f"Attendance for {self.name}:")
128 for i, status in enumerate(self.attendance, start=1):
129 print(f"Day {i}: {status}")
130
131 class AttendanceSystem:
132 def __init__(self):
133 self.students = []
134
135 def add_student(self, student):
136 self.students.append(student)
137
138 def display_all_attendance(self):
139 for student in self.students:
140 student.display_attendance()
141
- Bottom Right Panel:** Shows the "Build with Agent" feature, with a message: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."
- Bottom Status Bar:** Shows file path: `defaultInterpretorPath: 3.13.9.final.0`, and system status: ENG IN 11:50 23-01-2026.

Bottom Window:

- File Explorer:** Shows files: `.vscode`, `AI.py`, `ass-4.1`, `tpy`, and `task 3.py`.
- Code Editor:** Displays the `AI.py` file content:143
144
145 def main():
146 attendance_system = AttendanceSystem()
147
148 # Input validation
149 num_students = int(input("Enter the number of students: "))
150
151 for _ in range(num_students):
152 name = input("Enter student name: ")
153 student = Student(name)
154 attendance_system.add_student(student)
155
156 num_days = int(input("Enter the number of days to mark attendance: "))
157
158 for day in range(1, num_days + 1):
159 print(f"\nMarking attendance for Day {day}")
160 for student in attendance_system.students:
161 while True:
162 status = input(f"Is {student.name} present? (y/n): ").strip().lower()
163 if status == 'y':
164 student.mark_present()
165 break
166 elif status == 'n':
167 student.mark_absent()
168 break
169 else:
170 print("Invalid input! Please enter 'y' or 'n'.")
171
- Bottom Right Panel:** Shows the "Build with Agent" feature, with a message: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."
- Bottom Status Bar:** Shows file path: `defaultInterpretorPath: 3.13.9.final.0`, and system status: ENG IN 11:50 23-01-2026.

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

- File Explorer:** Shows a folder named "AI" containing files: ".vscode", "AI.py", "ass-4.1", "tpy", and "task 3.py".
- Code Editor:** The active file is "AI.py". The code defines a function "main" that handles user input for marking student attendance. It includes logic for marking present or absent status and displaying final attendance records.
- Right Panel:** An AI assistant interface titled "Build with Agent" is visible. It includes a message bubble icon, a note about AI responses being inaccurate, a "Generate Agent" button, and instructions to onboard AI onto the codebase.
- Bottom Bar:** Shows the current file path ("AI/AI.py"), line and column numbers (Ln 179, Col 1), character encoding (UTF-8), and other settings like interpreter path ("defaultInterpreterPath: 3.13.9.final.0").

Output:

The screenshot shows two instances of the Visual Studio Code (VS Code) interface, both displaying the same Python script named `AI.py`. The script contains a function `main()` that prompts the user for student names and their attendance status over four days. The output window shows the interaction between the user and the script.

```
145 def main():
PS C:\Users\reddy\OneDrive\Desktop\AI> & c:\Users\reddy\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users\reddy/OneDrive/Desktop/AI/AI.py
Enter the number of students: 4
Enter student name: Ashwutha
Enter student name: Harshini
Enter student name: Akshitha
Enter student name: varshitha
Enter the number of days to mark attendance: 4

Marking attendance for Day 1
Is Ashwutha present? (y/n): y
Is Harshini present? (y/n): y
Is Akshitha present? (y/n): y
Is varshitha present? (y/n): y

Marking attendance for Day 2
Is Ashwutha present? (y/n): n
Is Harshini present? (y/n): y
Is Akshitha present? (y/n): y
Is varshitha present? (y/n): y

Marking attendance for Day 3
Is Ashwutha present? (y/n): y
Is Harshini present? (y/n): n
Is Akshitha present? (y/n): n
Is varshitha present? (y/n): y

Marking attendance for Day 4
Is Ashwutha present? (y/n): y
Is Harshini present? (y/n): y
Is Akshitha present? (y/n): y
Is varshitha present? (y/n): y

Final Attendance Records
=====
Attendance For Ashwutha:
Day 1: Present
Day 2: Absent
Day 3: Present
Day 4: Present
-----
Attendance For Harshini:
Day 1: Present
Day 2: Present
Day 3: Absent
Day 4: Present
-----
Attendance for Akshitha:
Day 1: Present
Day 2: Present
Day 3: Absent
Day 4: Present
```

The right side of the interface features the "Build with Agent" panel, which includes a message input field, a "Generate Agent" button, and instructions for onboard AI onto the codebase. The bottom status bar shows the date and time (23-01-2026, 11:51), the language (ENG IN), and the Python interpreter path (3.13.9.final.0).

The screenshot shows a dark-themed code editor interface. In the Explorer sidebar, there are files: AI, .vscode, AI.py, ass-4.1, tpy, and task 3.py. The AI.py file is open in the main editor area, displaying Python code for generating attendance reports for three students: Akshitha, Varshitha, and Varshitha. The code uses loops and conditionals to print attendance status (Present or Absent) for each student over four days. A sidebar on the right is titled "Build with Agent" and includes a note about AI responses being inaccurate, a "Generate Agent" button, and instructions to onboard AI onto the codebase. The bottom status bar shows the path C:\Users\reddy\OneDrive\Desktop\AI>, line 179, column 1, and other system information.

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.

The screenshot shows two instances of the Visual Studio Code (VS Code) interface. Both instances have the same workspace open, featuring an Explorer sidebar on the left containing files like `.vscode`, `AI.py`, `ass-4.1`, `t.py`, and `task 3.py`. The main editor area displays Python code for an ATM menu system. In the top instance, the cursor is at line 187, and the AI panel on the right suggests the completion of the `atm_menu()` function. The bottom instance shows the completed code after the suggestion was accepted. The terminal below the editor shows the execution of the script, displaying the ATM menu options and a successful deposit of \$1000.00.

```
#WithDraw
#Exit
def atm_menu():
    balance = 1000 # Initial balance

    while True:
        print("\nATM Menu:")
        print("1. Check Balance")
        print("2. Deposit")
        print("3. Withdraw")
        print("4. Exit")

        choice = input("Please select an option (1-4): ")

        if choice == '1':
            print(f"Your current balance is: ${balance:.2f}")
        elif choice == '2':
            amount = float(input("Enter amount to deposit: $"))
            if amount > 0:
                balance += amount
                print(f"${amount:.2f} deposited successfully.")
            else:
                print("Invalid amount. Please enter a positive value.")
        elif choice == '3':
            amount = float(input("Enter amount to withdraw: $"))
            if 0 < amount <= balance:
                balance -= amount
                print(f"${amount:.2f} withdrawn successfully.")
            else:
```

Please select an option (1-4): 1
Your current balance is: \$1000.00
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Please select an option (1-4): 2
Enter amount to deposit: \$1000
\$1000.00 deposited successfully.

A screenshot of the Visual Studio Code (VS Code) interface. The main area shows a Python file named `AI.py` with the following code:

```
187     def atm_menu():
188         balance = 1000
189         print(f"${balance:.2f} available in your account")
190         choice = input("Enter choice (1-4): ")
191         if choice == '1':
192             amount = float(input("Enter amount to deposit: "))
193             balance += amount
194             print(f"${amount:.2f} deposited successfully.")
195         elif choice == '2':
196             print("Invalid amount. Please enter a positive value.")
197         elif choice == '3':
198             amount = float(input("Enter amount to withdraw: "))
199             if 0 < amount <= balance:
200                 balance -= amount
201                 print(f"${amount:.2f} withdrawn successfully.")
202             else:
203                 print("Invalid amount. Please enter a positive value within your balance.")
204         elif choice == '4':
205             print("Thank you for using the ATM. Goodbye!")
206             break
207         else:
208             print("Invalid choice. Please select a valid option.")
209     if __name__ == "__main__":
210         atm_menu()
```

The right side of the interface features an "AI" sidebar with the following content:

- Build with Agent**
- AI responses may be inaccurate.
- Generate Agent
- Instructions to onboard AI onto your codebase.

The bottom status bar shows the following information: Line 222, Col 5, Spaces: 4, UTF-8, CRLF, Python, defaultInterpreterPath: 3.13.9.final.0, Go Live, Go Live, ENG IN, 11:55, 23-01-2026.