

AI ASSIGNMENT-6.5

Name : L.Srinivas

Hall No. : 2303A52274

Batch : 36

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

Eligibility Check)

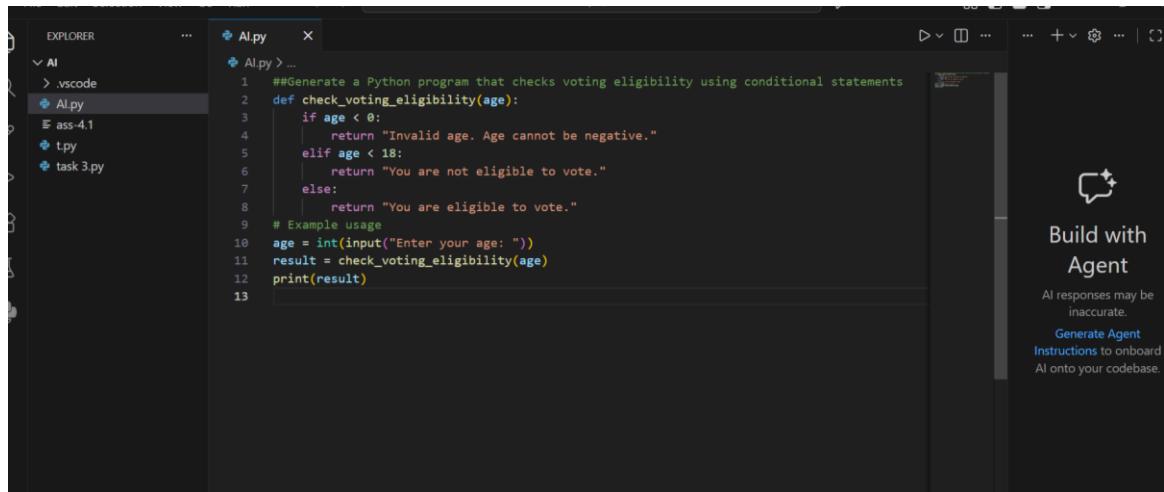
Task: Use an AI tool to generate eligibility logic.

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 version of the Visual Studio Code interface. In the center is a code editor window titled 'AI.py' containing 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 code editor, there is a sidebar with the heading 'Build with Agent'. It includes a note that 'AI responses may be inaccurate.' and a button labeled '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 `.vscode`, `AI.py`, `ass-4.1`, `tpy`, and `task 3.py`. The `AI.py` file is open in the editor, containing the following code:

```

1  ##Generate a Python program that checks voting eligibility using conditional statements
2  def check_voting_eligibility(age):
3      if age < 0:
4          return "Invalid age. Age cannot be negative."
5      elif age < 18:

```

The terminal window shows the output of running the code:

```

You are eligible to vote.
PS C:\Users\reddy\OneDrive\Desktop\AI> & c:\Users\reddy\AppData\Local\Microsoft\WindowsApps\py
thon3.13.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 `.vscode`, `AI.py`, `ass-4.1`, `tpy`, and `task 3.py`. The `AI.py` file is open in the editor, containing the following code:

```

13  ##Generate Python code to count the number of vowels and consonants in a given string using
14  ##The program should take a string as input from the user and display the total count of vo
15  def count_vowels_and_consonants(input_string):
16      vowels = "aeiouAEIOU"
17      vowel_count = 0
18      consonant_count = 0
19
20      for char in input_string:
21          if char.isalpha(): # Check if the character is a letter
22              if char in vowels:
23                  vowel_count += 1
24              else:
25                  consonant_count += 1
26
27      return vowel_count, consonant_count
28  # Example usage
29  input_string = input("Enter a string: ")
30  vowels, consonants = count_vowels_and_consonants(input_string)
31  print(f"Number of vowels: {vowels}")
32  print(f"Number of consonants: {consonants}")
33

```

The terminal window shows the output of running the code:

```

Number of consonants: 5
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 Visual Studio Code interface with the AI extension active. The Explorer sidebar on the left lists files: .vscode, Al.py, ass-4.1, tpy, and task 3.py. The Al.py file is selected and open in the main editor area. The code is a Python program to count vowels and consonants in a string. The terminal below the editor shows the program's execution and output. A sidebar on the right titled 'Build with Agent' provides instructions for AI integration.

```
13  ##Generate Python code to count the number of vowels and consonants in a given string using
14  ##The program should take a string as input from the user and display the total count of vo
15  def count_vowels_and_consonants(input_string):
16      vowels = "aeiouAEIOU"
17      vowel_count = 0
18      consonant_count = 0
19
20      for char in input_string:
21          if char in vowels:
22              vowel_count += 1
23          else:
24              consonant_count += 1
25
26      print("Number of vowels:", vowel_count)
27      print("Number of consonants:", consonant_count)
28
29  if __name__ == "__main__":
30      input_string = input("Enter a string: ")
31      count_vowels_and_consonants(input_string)
```

thon3.13.exe c:/Users/reddy/OneDrive/Desktop/AI/Al.py
thon3.13.exe c:/Users/reddy/OneDrive/Desktop/AI/Al.py
Enter a string: Ashwatha
Enter a string: Ashwatha
Number of vowels: 3
Number of consonants: 5
PS C:\Users\reddy\OneDrive\Desktop\AI>

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

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 in the .vscode folder, including Al.py, ass-4.1, tpy, and task 3.py.
- Code Editor:** Displays the Al.py file content. The code defines a Book class with methods for initialization, string representation, adding books to a library, displaying books, and checking out books. It also defines a Library class with methods for adding books, displaying books, and checking out books.
- Right Panel:** Features an "AI" sidebar with a "Build with Agent" section. It includes a message bubble icon, a "Generate Agent" button, and instructions: "Instructions to onboard AI onto your codebase". A "Describe what to build next" input field is present.
- Bottom Bar:** Shows standard VS Code icons and status information like "Ln 113, Col 5", "Spaces: 4", "UTF-8", "CRLF", "Python", and "defaultInterpretorPath: 3.13.9.final.0".

This screenshot shows the same VS Code environment as the first one, but with additional code added to the Al.py file:

```
43     class Library:
44         def check_out_book(self, title):
45             if book.is_available:
46                 book.is_available = False
47                 print(f"You have checked out '{book.title}'")
48                 return
49             else:
50                 print(f"Sorry, '{book.title}' is currently checked out.")
51                 return
52             print(f"Book '{title}' not found in the library.")
53
54         def return_book(self, title):
55             for book in self.books:
56                 if book.title == title:
57                     if not book.is_available:
58                         book.is_available = True
59                         print(f"You have returned '{book.title}'")
60                         return
61                     else:
62                         print(f"'{book.title}' was not checked out.")
63                         return
64             print(f"Book '{title}' not found in the library.")
65
66         def main():
67             library = Library()
68             while True:
69                 print("\nLibrary Management System")
70                 print("1. Add Book")
71                 print("2. Display Books")
72                 print("3. Check-Out Book")
```

The new code implements a Library class with methods for checking out and returning books. It also adds a main() function that creates a Library instance and provides a menu for the user to interact with the system.

The screenshot shows two instances of the Visual Studio Code (VS Code) interface, both displaying the same Python code for a Library Management System. The code is named `Al.py` and is located in a folder structure that includes `.vscode`, `ass-4.1`, `tpy`, and `task 3.py`.

```
81     print(f"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()
```

The interface includes a top bar with File, Edit, Selection, View, Go, Run, etc., and a bottom status bar showing Ln 113, Col 5, Spaces: 4, UTF-8, CRLF, Python, defaultInterpretorPath: 3.13.9.final.0, ENG IN, and 23-01-2026. A sidebar on the right features a "Build with Agent" panel with options like "Generate Agent", "Instructions to onboard AI onto your codebase", and a "Describe what to build next" input field.

Output:

```
AI.py
109     print("Invalid choice. Please try again.")
110
111     if __name__ == "__main__":
112         main()
113
114
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
python3.13
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
AI responses may be inaccurate.
Describe what to build next
A. v A. v Go Live Go Live
11:39 ENG IN 23-01-2026
```

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):

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 screenshot shows two instances of the Visual Studio Code (VS Code) interface, both displaying the same Python file named `Al.py`. The code implements a class `AttendanceSystem` to manage student attendance.

```
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
```

The second instance of VS Code shows the addition of a main function and user interaction logic:

```
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'.")
```

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar has icons for Explorer, Search, Open, Task, and Python. The main area displays a Python file named 'AI.py' with the following code:

```
145     def main():
146         ...
147         break
148     elif status == 'n':
149         student.mark_absent()
150         break
151     else:
152         print("Invalid input! Please enter 'y' or 'n'.")
153
154     print("\nFinal Attendance Records")
155     print("-" * 30)
156     attendance_system.display_all_attendance()
157
158 if __name__ == "__main__":
159     main()
```

The status bar at the bottom shows: Ln 179, Col 1 | Spaces: 4 | UTF-8 | CRLF | Python | defaultInterpreterPath: 3.13.9.final.0 | Go Live | Go Live | ENG IN | 11:51 | 23-01-2026.

Output:

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 AI, .vscode, Al.py, ass-4.1, tpy, and task 3.py. The main editor area displays a Python script named Al.py. The terminal tab at the bottom shows the execution of the script, which asks for the number of students and their names, then iterates through four days, querying the user about each student's presence (y/n). The output also includes a summary of attendance records for each student across the four days.

```
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 screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The Explorer sidebar on the left lists files: AI, .vscode, Al.py, ass-4.1, t.py, and task 3.py. The Al.py file is open in the main editor area, showing Python code for generating attendance reports. The terminal at the bottom shows the command PS C:\Users\reddy\OneDrive\Desktop\AI>. The status bar indicates the file has 179 lines, 4 spaces, and is saved in UTF-8 format. A sidebar on the right titled "Build with Agent" includes a note about AI responses being inaccurate, a "Generate Agent" button, and an "Instructions to onboard AI onto your codebase" link. The bottom right corner shows the date and time as 23-01-2026 and 11:52.

```
def main():
    print("Attendance for Akshitha:")
    print("Day 1: Present")
    print("Day 2: Present")
    print("Day 3: Absent")
    print("Day 4: Present")
    print("-----")
    print("Attendance for varshitha:")
    print("Day 1: Present")
    print("Day 2: Present")
    print("Day 3: Present")
    print("Day 4: Present")
    print("Day 1: Present")
    print("Day 2: Present")
    print("Day 3: Present")
    print("Day 4: Present")
    print("Day 1: Present")
    print("Day 2: Present")
    print("Day 3: Present")
    print("Day 4: Present")
    print("-----")
```

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, demonstrating an AI integration feature. The top instance shows the code for an ATM menu system in Python:

```
185     #Withdraw
186     #Exit
187     def atm_menu():
188         balance = 1000 # Initial balance
189
190         while True:
191             print("\nATM Menu:")
192             print("1. Check Balance")
193             print("2. Deposit")
194             print("3. Withdraw")
195             print("4. Exit")
196
197             choice = input("Please select an option (1-4): ")
198
199             if choice == '1':
200                 print(f"Your current balance is: ${balance:.2f}")
201             elif choice == '2':
202                 amount = float(input("Enter amount to deposit: $"))
203                 if amount > 0:
204                     balance += amount
205                     print(f"${amount:.2f} deposited successfully.")
206                 else:
207                     print("Invalid amount. Please enter a positive value.")
208             elif choice == '3':
209                 amount = float(input("Enter amount to withdraw: $"))
210                 if 0 < amount <= balance:
211                     balance -= amount
212                     print(f"${amount:.2f} withdrawn successfully.")
213                 else:
```

The bottom instance shows the same code running in a terminal window. The AI has added a check for the main module and provided a user interaction loop:

```
187     def atm_menu():
188         print("Invalid choice. Please select a valid option.")
189         if __name__ == "__main__":
190             atm_menu()
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
```

Output from the terminal:

```
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.
```

```
187 def atm_menu():
188     balance = 1000
189     choice = input("1. Deposit\n2. Withdraw\n3. Exit\nEnter your choice: ")
190     if choice == '1':
191         amount = float(input("Enter amount to deposit: $"))
192         balance += amount
193         print(f"${amount:.2f} deposited successfully.")
194     elif choice == '2':
195         amount = float(input("Enter amount to withdraw: $"))
196         if 0 < amount <= balance:
197             balance -= amount
198             print(f"${amount:.2f} withdrawn successfully.")
199         else:
200             print("Invalid amount. Please enter a positive value within your balance.")
201     elif choice == '3':
202         print("Thank you for using the ATM. Goodbye!")
203         break
204     else:
205         print("Invalid choice. Please select a valid option.")
206 if __name__ == "__main__":
207     atm_menu()
```

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

Ln 222, Col 5 Spaces: 4 UTF-8 CRLF {} Python defaultInterpreterPath: 3.13.9.final.0 Go Live Go Live 11:55 ENG IN 23-01-2026