

## **LAB\_ASSIGNMENT\_6.5.**

NAME:AKSHITHA

HT NO:2303A51360

BATCH NO:29

### **TASK-1:**

Use an AI tool to generate eligibility logic.

## PROMPT:

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

## CODE:

## OBSERVATION:

AI-based code completion efficiently generates conditional logic.

The generated code is **readable, correct, and easy to optimize**.

Human oversight is necessary to detect edge cases and improve robustness.

AI tools enhance productivity when used responsibly and ethically.

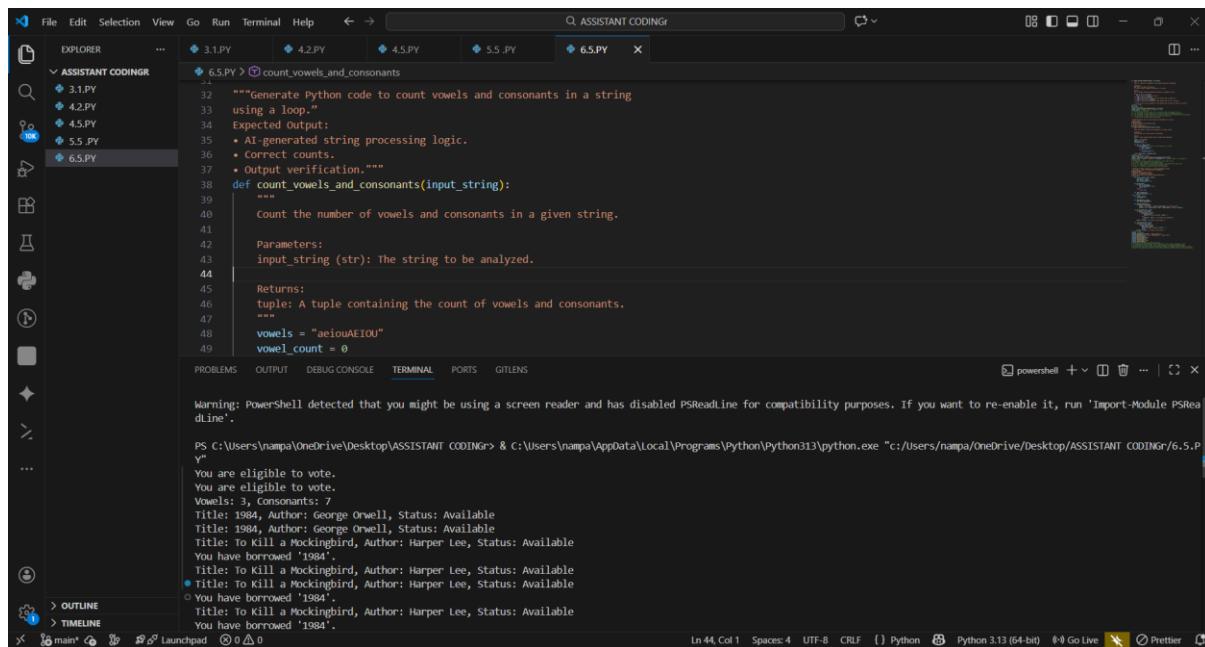
## TASK-2:

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:



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

- File Explorer:** Shows files 3.1.PY, 4.2.PY, 4.5.PY, 5.5.PY, and 6.5.PY. 6.5.PY is the active file.
- Code Editor:** Displays the following Python code:

```
32 """Generate Python code to count vowels and consonants in a string
33     using a loop."""
34
35     Expected output:
36     • AI-generated string processing logic.
37     • Correct counts.
38     • Output verification."""
39
40     def count_vowels_and_consonants(input_string):
41         """
42             Count the number of vowels and consonants in a given string.
43
44             Parameters:
45                 input_string (str): The string to be analyzed.
46
47             Returns:
48                 tuple: A tuple containing the count of vowels and consonants.
49
50             vowel_count = 0
```
- Terminal:** Shows the command PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/6.5.PY". The output includes:

```
You are eligible to vote.
You are eligible to vote.
Vowels: 3, Consonants: 7
Title: 1984, Author: George Orwell, Status: Available
Title: 1984, Author: George Orwell, Status: Available
Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
You have borrowed '1984'.
Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
● Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
○ You have borrowed '1984'.
Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
You have borrowed '1984'.
```
- Status Bar:** Shows the current file is main\*, line 44, column 1, spaces: 4, encoding: UTF-8, CRLF, Python 3.13 (64-bit), Go Live, and Prettier.

## OBSERVATION:

AI-based code completion efficiently generates loop-based string processing logic.

The code correctly counts vowels and consonants.

Manual review helps identify edge cases and improve efficiency.

AI tools are effective when used responsibly with human validation.

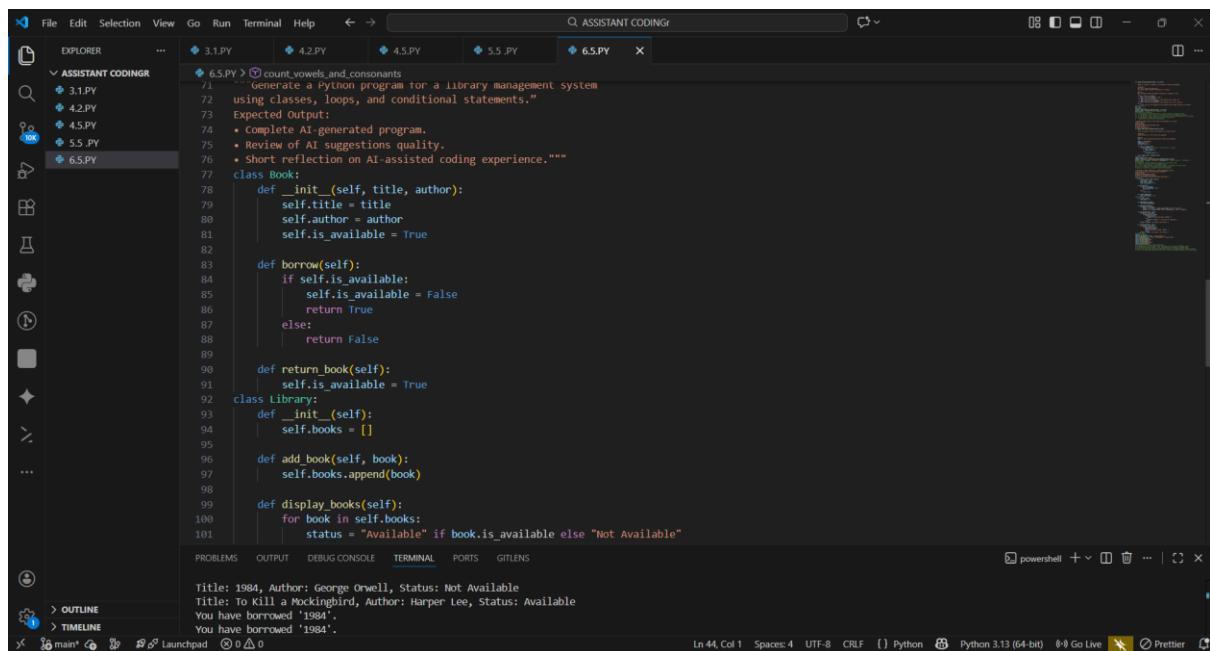
### TASK-3:

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:



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The left sidebar has an Explorer view showing files: 3.1.PY, 4.2.PY, 4.5.PY, 5.5.PY, and 6.5.PY. The main editor area contains the following Python code:

```
3.1.PY 4.2.PY 4.5.PY 5.5.PY 6.5.PY

6.5.PY > count.vowels_and.consonants
71     Generate a Python program for a library management system
72     using classes, loops, and conditional statements."
73
74     Expected output:
75     • Complete AI-generated program.
76     • Review of AI suggestions quality.
77     • Short reflection on AI-assisted coding experience."
78
79     class Book:
80         def __init__(self, title, author):
81             self.title = title
82             self.author = author
83             self.is_available = True
84
85         def borrow(self):
86             if self.is_available:
87                 self.is_available = False
88                 return True
89             else:
90                 return False
91
92     class Library:
93         def __init__(self):
94             self.books = []
95
96         def add_book(self, book):
97             self.books.append(book)
98
99         def display_books(self):
100             for book in self.books:
101                 status = "Available" if book.is_available else "Not Available"
102
103             print(f"Title: {book.title}, Author: {book.author}, Status: {status}")
104
105             if book.is_available:
106                 print("You have borrowed '1984'.")
107             else:
108                 print("You have borrowed 'To Kill a Mockingbird'.")


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
```

The bottom status bar indicates the code is in main\*, using Python 3.13 (64-bit), and the terminal shows the generated code output:

```
Title: 1984, Author: George Orwell, Status: Not Available
Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
You have borrowed '1984'.
You have borrowed 'To Kill a Mockingbird.'
```

#### OBSERVATION:

AI-assisted code completion significantly reduces development time.

The generated program provides a solid starting structure.

Human intervention is required to handle edge cases and improve robustness.

Using AI responsibly enhances learning and coding productivity.

## 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:

The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and several icons. The left sidebar has sections for Explorer, Assistant Coding (with files 3.1.PY, 4.2.PY, 4.5.PY, 5.5.PY, and 6.5.PY listed), and a terminal section showing book titles and their status. The main code editor area contains the following Python code:

```
136     """Generate a Python class to mark and display student
137     attendance using loops."""
138     Expected output:
139     • AI-generated attendance logic.
140     • Correct display of attendance.
141     • Test cases."""
142     class StudentAttendance:
143         def __init__(self, student_name):
144             self.student_name = student_name
145             self.attendance_record = []
146
147         def mark_attendance(self, day, is_present):
148             self.attendance_record.append((day, is_present))
149
150         def display_attendance(self):
151             print(f"Attendance record for {self.student_name}:")
152             for day, is_present in self.attendance_record:
153                 status = "Present" if is_present else "Absent"
154                 print(f"\t{day}: {status}")
155
156     # Example usage:
157     student = StudentAttendance("Alice")
158     student.mark_attendance("Monday", True)
159     student.mark_attendance("Tuesday", False)
160     student.display_attendance()
```

The bottom status bar indicates the code is in Python mode, with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and GITLENS. It also shows file paths like 'main\*' and 'Launchpad', and a terminal history with entries for '1984' and 'To Kill a Mockingbird'. The status bar also shows the current line (Ln 218), column (Col 32), and various configuration details.

OBSERVATION:

AI-assisted code completion efficiently generates class-based systems.

The attendance logic works correctly using loops and conditionals.

Human review is essential to handle invalid inputs and edge cases.

AI tools improve productivity when used responsibly.

## 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:

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing files 3.1.PY, 4.2.PY, 4.5.PY, 5.5.PY, and 6.5.PY. The main editor area contains the following Python code:

```
162     """Generate a Python program using loops and conditionals
163     to simulate an ATM menu."""
164     Expected Output:
165     • AI-generated menu logic.
166     • Correct option handling.
167     • Output verification.
168 """
169 class ATM:
170     def __init__(self, balance=0):
171         self.balance = balance
172
173     def display_menu(self):
174         print("Welcome to the ATM!")
175         print("1. Check Balance")
176         print("2. Deposit")
177         print("3. Withdraw")
178         print("4. Exit")
179
180     def check_balance(self):
181         print(f"Your current balance is: ${self.balance}")
182
183     def deposit(self, amount):
184         if amount > 0:
185             self.balance += amount
186             print(f"You have deposited: ${amount}")
187         else:
188             print("Please select an option (1-4): 1
189             Your current balance is: $80.0
190             Welcome to the ATM!
191             1. Check Balance
192             2. Deposit
193             3. Withdraw
194             4. Exit
195             Please select an option (1-4): 4
196             Thank you for using the ATM. Goodbye!
```

The terminal tab at the bottom shows the execution of the script and its output. The status bar at the bottom right indicates the file is 218 lines long, 32 columns wide, using spaces, in UTF-8 encoding, with CRLF line endings, and is run in Python 3.13 (64-bit) mode.

## OBSERVATION:

The ATM menu is displayed continuously using a loop until the user selects the exit option.

Conditional statements correctly handle user choices such as checking balance, deposit, and withdrawal.

The balance is updated accurately after each deposit and withdrawal operation.

Invalid inputs and insufficient balance conditions are handled properly.

The program demonstrates effective use of classes, loops, and conditionals.

AI-generated code provides a structured and functional solution with minimal manual modification.