

# ASSIGNMENT-6.5

BATCH-29

ROLL-NO:2303A51600

## TASK-1

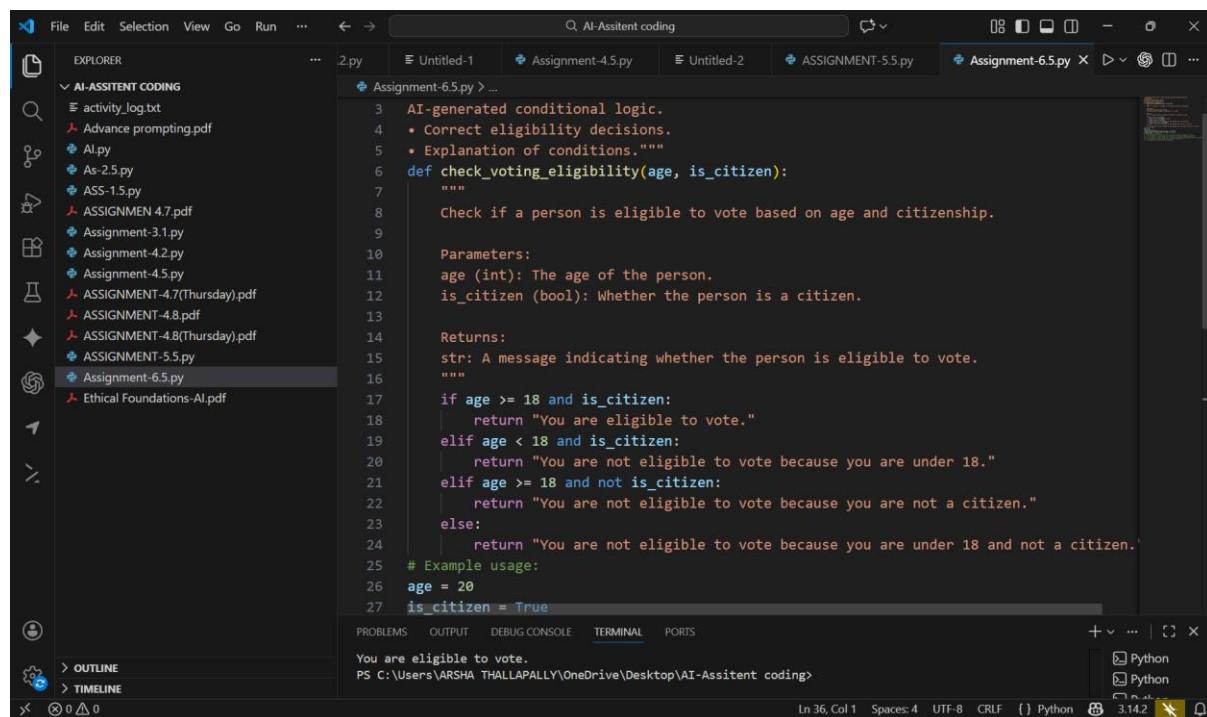
### PROMPT:

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

AI-generated conditional logic.

- Correct eligibility decisions.
- Explanation of conditions.

### CODE:



The screenshot shows a code editor interface with multiple tabs open. The active tab is 'Assignment-6.5.py'. The code is as follows:

```
3  AI-generated conditional logic.
4  • Correct eligibility decisions.
5  • Explanation of conditions."""
6  def check_voting_eligibility(age, is_citizen):
7      """
8          Check if a person is eligible to vote based on age and citizenship.
9
10         Parameters:
11             age (int): The age of the person.
12             is_citizen (bool): Whether the person is a citizen.
13
14         Returns:
15             str: A message indicating whether the person is eligible to vote.
16
17         if age >= 18 and is_citizen:
18             return "You are eligible to vote."
19         elif age < 18 and is_citizen:
20             return "You are not eligible to vote because you are under 18."
21         elif age >= 18 and not is_citizen:
22             return "You are not eligible to vote because you are not a citizen."
23         else:
24             return "You are not eligible to vote because you are under 18 and not a citizen."
25
26     # Example usage:
27     age = 20
28     is_citizen = True
```

The terminal tab at the bottom shows the output: 'You are eligible to vote.'

## OBSERVATIONS:

When the prompt given to explain about the condition – it specified

Logic was correctly given

Specifications given in order of the prompt

## TASK-2

### 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:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar (EXPLORER) lists files and folders related to AI-assistant coding, including 'AI-ASSISTANT CODING', 'activity.log.txt', 'Advance prompting.pdf', 'AI.py', 'As-2.5.py', 'ASS-1.5.py', 'ASSIGNMEN 4.7.pdf', 'Assignment-3.1.py', 'Assignment-4.2.py', 'Assignment-4.5.py', 'ASSIGNMENT-4.7(Thursday).pdf', 'ASSIGNMENT-4.8.pdf', 'ASSIGNMENT-4.8(Thursday).pdf', 'Assignment-5.5.py', 'Assignment-6.5.py', and 'Ethical Foundations-AI.pdf'. The right pane displays a Python script named 'Assignment-6.5.py' with the following code:

```
37     """Generate Python code to count vowels and consonants in a string
38     using a loop."""
39     Expected Output:
40     • AI-generated string processing logic.
41     • Correct counts.
42     • Output verification."""
43     def count_vowels_and_consonants(input_string):
44         """
45             Count the number of vowels and consonants in a given string.
46
47             Parameters:
48                 input_string (str): The string to be analyzed.
49
50             Returns:
51                 tuple: A tuple containing the count of vowels and consonants.
52         """
53         vowels = "aeiouAEIOU"
54         vowel_count = 0
55         consonant_count = 0
56
57         for char in input_string:
58             if char.isalpha(): # Check if the character is a letter
```

The 'TERMINAL' tab at the bottom shows the output of running the script:

```
You are eligible to vote.
Vowels: 3, Consonants: 7
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assistent coding>
```

The 'PROBLEMS' tab shows no errors or warnings.

## OBSERVATIONS:

Checking output verification-correct

No explanation as not mentioned in the prompt

Few Test considered like checking alphabets

## TASK-3

### PROMPT:

Generate a Python program for a library management system

using classes, loops, and conditional statements.”

### Expected Output:

- Complete AI-generated program.

### CODE:

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a folder named "AI-ASSISTANT CODING" containing files like "activity.log.txt", "AI.py", "As-2.5.py", "ASS-1.5.py", "Assignment-4.7.pdf", "Assignment-4.8.pdf", "Assignment-4.8(Thursday).pdf", "Assignment-4.9.py", "Assignment-5.5.py", "Assignment-6.5.py", and "Ethical Foundations-AI.pdf".
- Code Editor:** The main pane displays a Python script titled "Assignment-6.5.py".

```
76     """Generate a Python program for a library management system
77     using classes, loops, and conditional statements."""
78
79     Expected Output:
80     • Complete AI-generated program.
81     • Review of AI suggestions quality.
82     • Short reflection on AI-assisted coding experience."""
83
84     class Book:
85         def __init__(self, title, author):
86             self.title = title
87             self.author = author
88             self.is_available = True
89
90         def borrow(self):
91             if self.is_available:
92                 self.is_available = False
93                 return True
94             else:
95                 return False
96
97         def return_book(self):
98             self.is_available = True
99
100    class Library:
```
- Terminal:** The bottom terminal window shows the output of running the script:

```
Title: 1984, Author: George Orwell, Status: Not Available
Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
You have returned '1984'.
Title: 1984, Author: George Orwell, Status: Available
Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assistent coding>
```
- Python Interpreter:** A sidebar on the right lists multiple Python environments.

### OBSERVATIONS:

Output given with respect to the prompting stating loop

Class and conditions

Output is clear and neat

Correct and solves few test cases

## TASK-4

PROMPT:

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

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

CODE:

The screenshot shows a Visual Studio Code (VS Code) interface. The left sidebar has a tree view titled 'EXPLORER' with several files listed under 'AI-ASSISTANT CODING'. The main editor area contains a Python file named 'Assignment-6.5.py'. The code defines a class 'StudentAttendance' with methods for marking attendance and displaying it. A terminal tab at the bottom shows the output of running the code, which prints an attendance record for a student named Alice. The terminal also shows the path 'C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>'.

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

OBSERVATIONS:

AI-generated attendance logic-correct

Correct display of attendance-done

Example usage also given

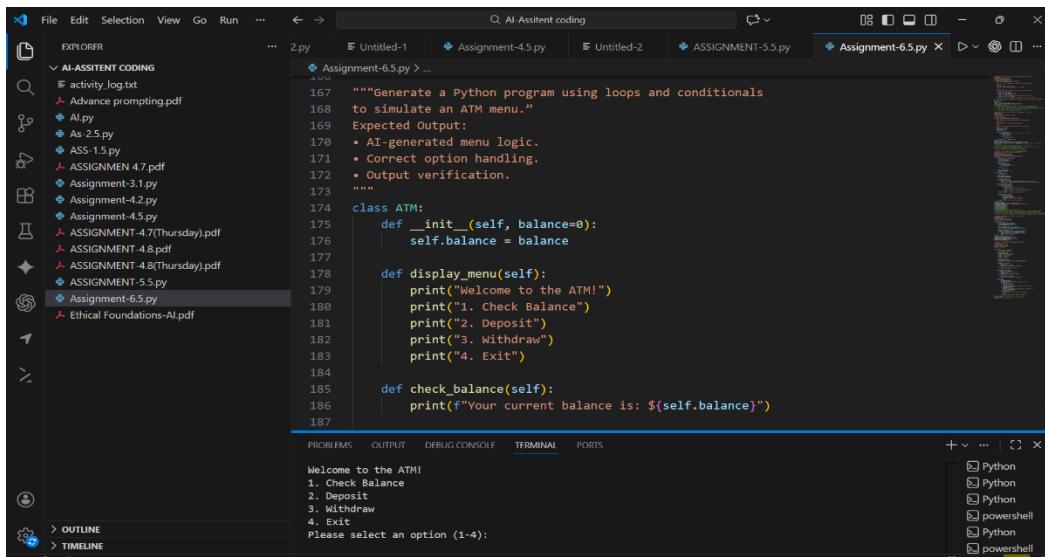
Many variables are considered to solve

Task-5:

PROMPT:

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

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



The screenshot shows a code editor window titled "Assignment-6.5.py". The code is a Python program for an ATM menu. It includes a docstring, a class definition for ATM, and several methods like \_\_init\_\_, display\_menu, and check\_balance. The code is annotated with comments and expected output. The code editor has a sidebar with file navigation and a bottom bar with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. A terminal window on the right shows the execution of the program.

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

Expected Output:
• AI-generated menu logic.
• Correct option handling.
• Output verification.

"""

class ATM:
    def __init__(self, balance=0):
        self.balance = balance

    def display_menu(self):
        print("Welcome to the ATM!")
        print("1. Check Balance")
        print("2. Deposit")
        print("3. Withdraw")
        print("4. Exit")

    def check_balance(self):
        print(f"Your current balance is: ${self.balance}")

    def withdraw(self):
        amount = float(input("Enter amount to withdraw: "))
        if amount > self.balance:
            print("Insufficient funds")
        else:
            self.balance -= amount
            print(f"Withdrawal successful. Your new balance is: ${self.balance}")

    def deposit(self):
        amount = float(input("Enter amount to deposit: "))
        self.balance += amount
        print(f"Deposit successful. Your new balance is: ${self.balance}")

    def exit(self):
        print("Thank you for using the ATM. Goodbye!")

# Main program loop
while True:
    ATM().display_menu()
    choice = input("Please select an option (1-4): ")
    if choice == "1":
        ATM().check_balance()
    elif choice == "2":
        ATM().deposit()
    elif choice == "3":
        ATM().withdraw()
    elif choice == "4":
        ATM().exit()
        break
    else:
        print("Invalid choice. Please enter 1, 2, 3, or 4.")
```

CO

OBSERVATION:

Logic is optimized one

Option handling is on point

Output is correct

Goes well with the prompt

No violations of prompt seen