

AI-ASSISTANT CODING ASSIGNMENT-6.3

B.HARSHAVARDHAN

2303A52T02

Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals

Lab Outcomes (LOs)

After completing this lab, students will be able to:

- Use AI tools to generate and complete Python class definitions and methods.
- Understand and assess AI-suggested loop constructs for iterative tasks.
- Generate and evaluate conditional statements using AI-driven prompts.
- Critically analyze AI-assisted code for correctness, clarity, and efficiency.

Task Description #1: Classes (Student Class)

Scenario

You are developing a simple student information management module.

Task

- Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
- The class should include attributes such as name, roll number, and branch.
- Add a method `display_details()` to print student information.
- Execute the code and verify the output.
- Analyze the code generated by the AI tool for correctness and clarity.

Expected Output #1

- A Python class with a constructor (`__init__`) and a `display_details()` method.
- Sample object creation and output displayed on the console.
- Brief analysis of AI-generated code.

AI Prompt Used

“Generate a Python class named `Student` with attributes `name`, `roll_number`, and `branch`. Include a constructor to initialize these attributes and a method `display_details()` to print the student information. Also show example object creation and method call.”

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

- File Explorer:** Shows a folder named "AI-CODING" containing a file "ASS-6.3.py".
- Code Editor:** Displays the following Python code:

```
1  class Student:
2      """A Class to represent a student and manage their information."""
3
4      def __init__(self, name, roll_number, branch):
5          """Initialize student attributes."""
6          self.name = name
7          self.roll_number = roll_number
8          self.branch = branch
9
10     def display_details(self):
11         """Display student information."""
12         print(f"Name: {self.name}")
13         print(f"Roll Number: {self.roll_number}")
14         print(f"Branch: {self.branch}")
15
16
17     # Task 1 Execution
18     print("-" * 50)
19     print("TASK 1: STUDENT CLASS")
20     print("-" * 50)
21     student1 = Student("Harsha Vardhan", 101, "Computer Science")
22     student1.display_details()
23
24
25
```

- Terminal:** Shows the command PS C:\Users\HARSHAVARDHAN\Desktop\AI-CODING\ & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/Desktop/AI-CODING/ASS-6.3.py followed by the output:

```
=====
TASK 1: STUDENT CLASS
=====
Name: Harsha Vardhan
Roll Number: 101
Branch: Computer Science
PS C:\Users\HARSHAVARDHAN\Desktop\AI-CODING>
```

- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS.

Justification

This prompt was designed to:

- Clearly specify the **class name and required attributes**
- Request a **constructor (`__init__`)**, ensuring object-oriented principles are followed
- Ask for a **method implementation** and **sample execution**
- Allow the AI to generate a **complete, testable class** rather than partial code

The prompt ensures that the AI output aligns with the lab requirement of demonstrating **basic OOP concepts** in Python.

Task Description #2: Loops (Multiples of a Number)

Scenario

You are writing a utility function to display multiples of a given number.

Task

- Prompt the AI tool to generate a function that prints the first 10 multiples of a given number

using a loop.

- Analyze the generated loop logic.

- Ask the AI to generate the same functionality using another controlled looping structure (e.g.,

while instead of for).

Expected Output #2

- Correct loop-based Python implementation.
- Output showing the first 10 multiples of a number.
- Comparison and analysis of different looping approaches

AI Prompt Used

“Generate a Python function that prints the first 10 multiples of a given number using a for loop.”

“Generate the same functionality using a while loop instead of a for loop.”

```
File Edit Selection View Go Run Terminal Help ← → Q AI-CODING
EXPLORER Welcome ASS-6.3.py
ASS-6.3.py
1 def print_multiples_for(num, count=10):
2     """Print first n multiples using for loop."""
3     print(f"\nFirst {count} multiples of {num} (using for loop):")
4     for i in range(1, count + 1):
5         print(f"{num} | {i} = {num * i}", end=" ")
6     print()
7
8 def print_multiples_while(num, count=10):
9     """Print first n multiples using while loop."""
10    print(f"\nFirst {count} multiples of {num} (using while loop):")
11    i = 1
12    while i <= count:
13        print(f"{num} | {i} = {num * i}", end=" ")
14        i += 1
15    print()
16
17 # Task 2 Execution
18 print("\n" + "=" * 50)
19 print("TASK 2: LOOPS - MULTIPLES OF A NUMBER")
20 print("=" * 50)
21 print_multiples_for(5, 10)
22 print_multiples_while(7, 10)
23
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & c:/Users/HARSHAVARDHAN/appData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ASS-6.3.py
=====
First 10 multiples of 5 (using for loop):
5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50

First 10 multiples of 7 (using while loop):
7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 7 x 4 = 28 7 x 5 = 35 7 x 6 = 42 7 x 7 = 49 7 x 8 = 56 7 x 9 = 63 7 x 10 = 70
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING>
```

Justification

This prompt:

- Explicitly defines the **task scope** (first 10 multiples)
- Forces use of a **for loop**, allowing analysis of fixed-iteration looping
- Helps evaluate AI's understanding of **range-based iteration**
- Produces predictable and verifiable output for correctness checking
- Encourages the AI to use an **alternative looping construct**
- Enables **comparison between for and while loops**
- Helps analyze how AI handles **manual loop control and termination conditions**
- Supports the lab objective of evaluating different looping approaches

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

Task

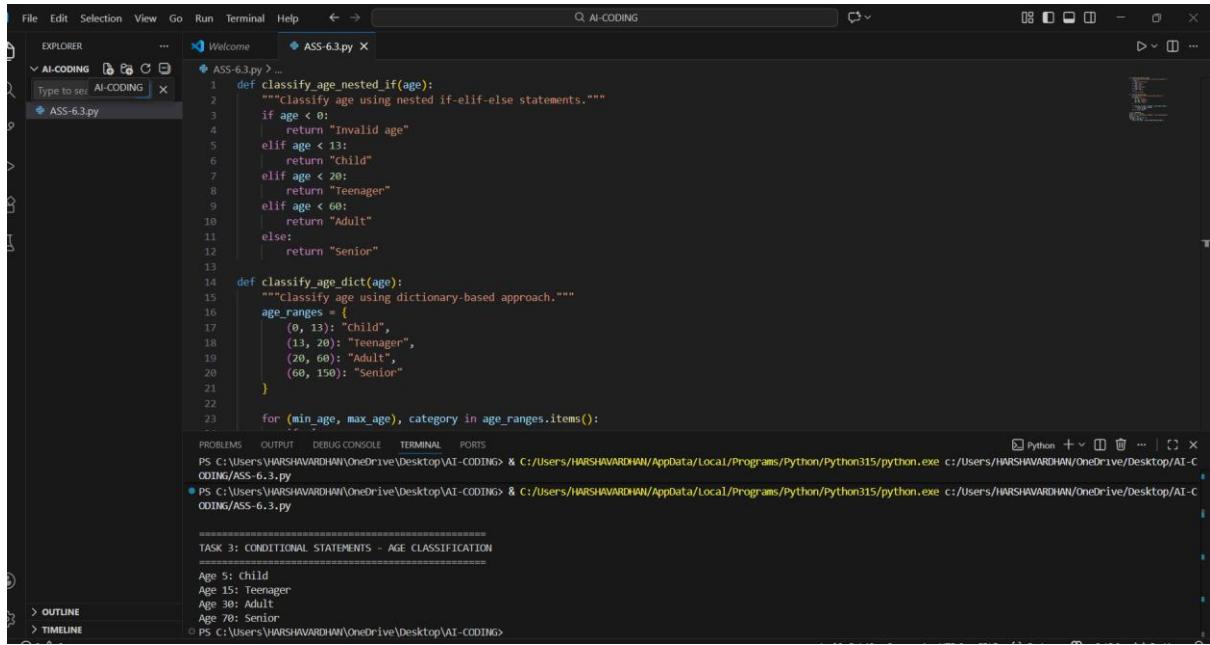
- Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups
(e.g., child, teenager, adult, senior).
- Analyze the generated conditions and logic.
- Ask the AI to generate the same classification using alternative conditional structures
(e.g., simplified conditions or dictionary-based logic).

Expected Output #3

- A Python function that classifies age into appropriate groups.
- Clear and correct conditional logic.
- Explanation of how the conditions work.

AI Prompt Used

“Generate a Python function that uses nested if-elif-else statements to classify age into child, teenager, adult, and senior categories.” “Generate an alternative implementation for age classification using simplified conditions or a different logical structure.”



```
File Edit Selection View Go Run Terminal Help <- > AI-CODING ASS-6.3.py

EXPLORER AI-CODING ASS-6.3.py
Type to see AI-CODING ...
Welcome ASS-6.3.py

1 def classify_age_nested_if(age):
2     """Classify age using nested if-elif-else statements."""
3     if age < 0:
4         return "Invalid age"
5     elif age < 13:
6         return "Child"
7     elif age < 20:
8         return "Teenager"
9     elif age < 60:
10        return "Adult"
11    else:
12        return "Senior"
13
14 def classify_age_dict(age):
15     """Classify age using dictionary-based approach."""
16     age_ranges = {
17         (0, 13): "Child",
18         (13, 20): "Teenager",
19         (20, 60): "Adult",
20         (60, 150): "Senior"
21     }
22
23     for (min_age, max_age), category in age_ranges.items():
24
25         if min_age <= age < max_age:
26             return category
27
28     return "Invalid age"

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ASS-6.3.py
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ASS-6.3.py

TASK 3: CONDITIONAL STATEMENTS - AGE CLASSIFICATION
-----
Age 5: Child
Age 15: Teenager
Age 30: Adult
Age 70: Senior
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING>
```

Justification

This follow-up prompt:

- Encourages the AI to use an **alternative looping construct**
- Enables **comparison between for and while loops**
- Helps analyze how AI handles **manual loop control and termination conditions**
- Supports the lab objective of evaluating different looping approaches

This prompt was chosen to:

- Require **conditional branching logic**
- Ensure use of **nested if-elif-else statements**
- Test AI's ability to create **non-overlapping and logically ordered conditions**
- Provide a real-world classification problem suitable for control-flow analysis

- Encourages AI to explore **different logical designs**
- Helps assess **code readability and maintainability**
- Allows comparison between **traditional conditionals and optimized logic**
- Aligns with the lab goal of critical evaluation of AI-generated solutions

Task Description #4: For and While Loops (Sum of First n Numbers)

Scenario

You need to calculate the sum of the first n natural numbers.

Task

- Use AI assistance to generate a `sum_to_n()` function using a for loop.
- Analyze the generated code.
- Ask the AI to suggest an alternative implementation using a while loop or a mathematical formula.

Expected Output #4

- Python function to compute the sum of first n numbers.
- Correct output for sample inputs.
- Explanation and comparison of different approaches.

AI Prompt Used

“Generate a Python function `sum_to_n()` that calculates the sum of the first n natural numbers using a for loop. “Suggest an alternative implementation of the same function using a while loop or a mathematical formula.”

```

File Edit Selection View Go Run Terminal Help ← → Q AI-CODING
EXPLORER Type to search 🔎 ×
AI-CODING
ASS-6.3.py
1 def sum_to_n_for(n):
2     """Calculate sum of first n natural numbers using for loop."""
3     total = 0
4     for i in range(1, n + 1):
5         total += i
6     return total
7
8 def sum_to_n_while(n):
9     """Calculate sum of first n natural numbers using while loop."""
10    total = 0
11    i = 1
12    while i <= n:
13        total += i
14        i += 1
15    return total
16
17 def sum_to_n_formula(n):
18     """Calculate sum using mathematical formula: n(n+1)/2."""
19     return n * (n + 1) // 2
20
21 # Task 4 Execution
22 print("\n" + "=" * 50)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & c:/users/HARSHAVARDHAN/appData/local/programs/python/python315/python.exe c:/users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ASS-6.3.py
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & c:/users/HARSHAVARDHAN/appData/local/programs/python/python315/python.exe c:/users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ASS-6.3.py

=====
TASK 4: FOR AND WHILE LOOPS - SUM OF FIRST N NUMBERS
=====
Sum of first 10 numbers (for loop): 55
Sum of first 10 numbers (while loop): 55
Sum of first 10 numbers (formula): 55
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING>

```

> OUTLINE > TIMELINE

Justification

This prompt:

- Clearly defines **input, output, and looping structure**
- Forces AI to implement **iterative accumulation**
- Makes it easy to test correctness with sample values
- Demonstrates how AI handles **basic algorithmic problems**

This prompt:

- Encourages AI to suggest **more efficient or varied solutions**
- Helps analyze **time complexity differences**
- Supports comparison between **iterative and formula-based approaches**
- Reinforces understanding of optimization concepts

Task Description #5: Classes (Bank Account Class)

Scenario

You are designing a basic banking application.

Task

- Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(), and check_balance().
- Analyze the AI-generated class structure and logic.
- Add meaningful comments and explain the working of the code.

Expected Output #5

- Complete Python Bank Account class.
- Demonstration of deposit and withdrawal operations with updated balance.
- Well-commented code with a clear explanation.

Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.

AI Prompt Used

“Generate a Python BankAccount class with methods deposit(), withdraw(), and check_balance(). Include validation logic and demonstrate usage with sample transactions.”

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

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print.
- Explorer:** Shows a tree view with "AI-CODING" expanded, containing "ASS-6.3.py".
- Editor:** Displays the Python code for a `BankAccount` class with methods `deposit` and `withdraw`.
- Terminal:** Shows the command line output of running the script, displaying account balance changes and validation messages.
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS, along with Python-related icons.

Justification

This prompt was structured to:

- Require a **real-world class design**
- Test AI's ability to implement **state management (balance)**
- Include **input validation** for deposits and withdrawals
- Demonstrate **method interaction and object behavior**
- Produce code suitable for explanation and commenting

It aligns strongly with evaluating **AI-generated OOP logic and correctness**.