

AI Assisted Coding Lab ass-6.1

Name:M.Mohan venkatesh

Batch:14

2303A510f0

Task Description #1 (AI-Based Code Completion for Loops)

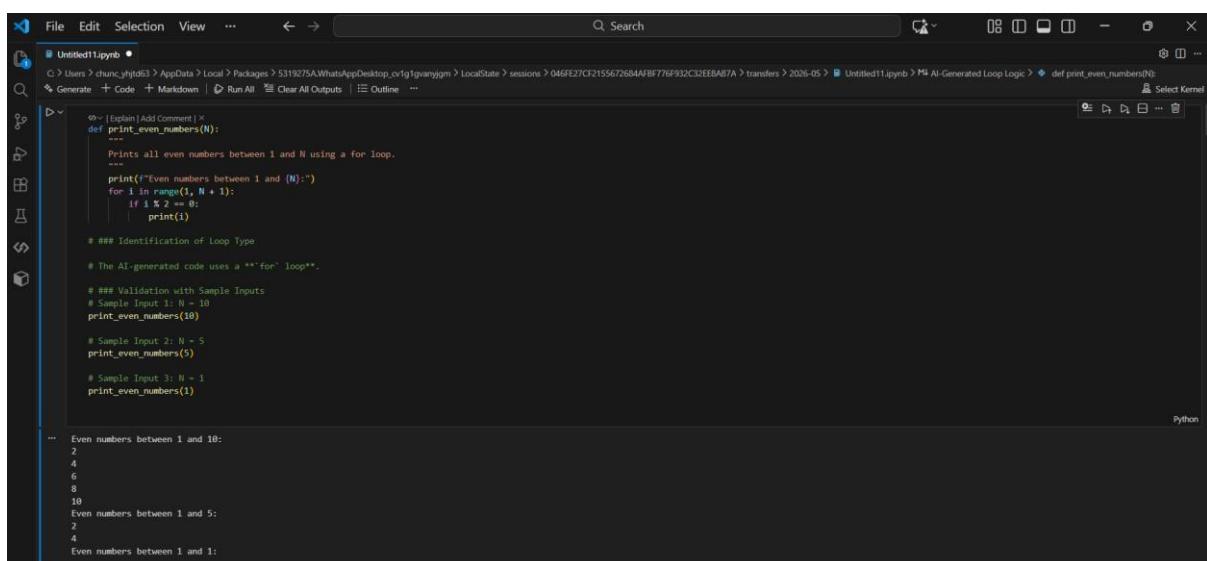
Task: Use an AI code completion tool to generate a loop-based program.

Prompt:

“Generate Python code to print all even numbers between 1 and N using a loop.”

Expected Output:

- AI-generated loop logic.
- Identification of loop type used (for or while).
- Validation with sample inputs.



The screenshot shows a Jupyter Notebook interface with a single cell containing Python code. The code defines a function `print_even_numbers(N)` that prints even numbers from 1 to N using a for loop. The code includes comments explaining the purpose of the loop and its validation with sample inputs (N=10, N=5, N=1). The output of the code execution shows the even numbers for each case: 2, 4, 6, 8, 10 for N=10; 2, 4 for N=5; and 2 for N=1.

```
def print_even_numbers(N):
    """Prints all even numbers between 1 and N using a for loop.
    """
    print("Even numbers between 1 and " + str(N) + ":")
    for i in range(1, N + 1):
        if i % 2 == 0:
            print(i)

    # *** Identification of Loop Type
    # The AI-generated code uses a "for" loop.

    # *** Validation with Sample Inputs
    # Sample Input 1: N = 10
    print_even_numbers(10)

    # Sample Input 2: N = 5
    print_even_numbers(5)

    # Sample Input 3: N = 1
    print_even_numbers(1)

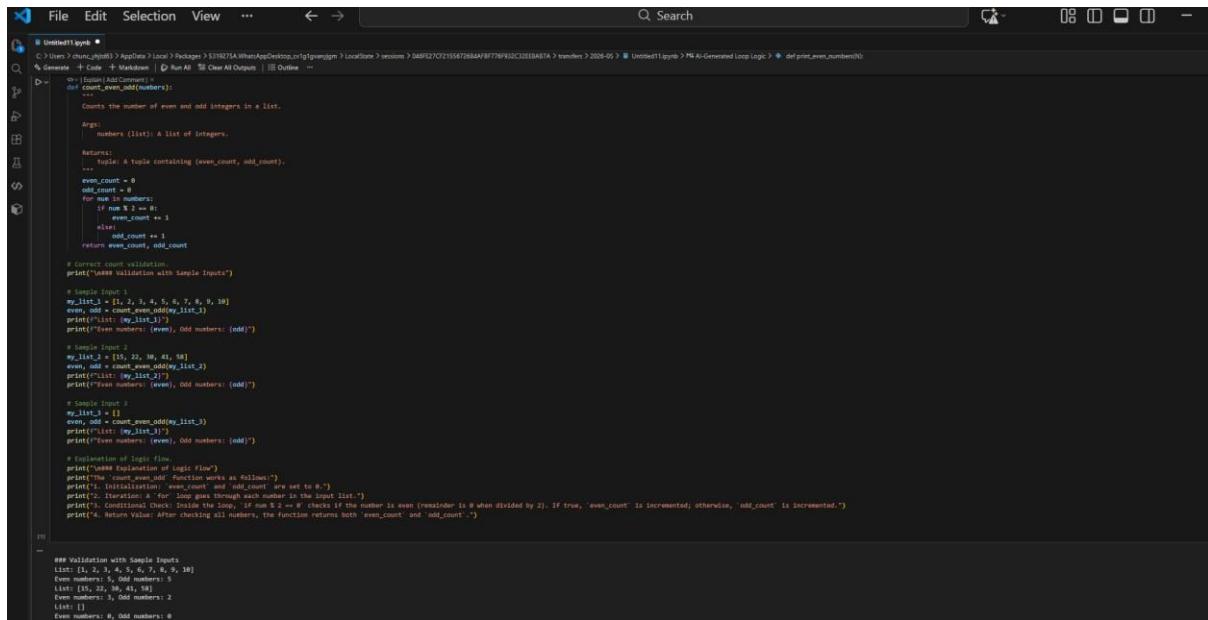
... Even numbers between 1 and 10:
2
4
6
8
10
Even numbers between 1 and 5:
2
4
Even numbers between 1 and 1:
2
```

Task Description #2 (AI-Based Code Completion for Loop with Conditionals)

Task: Use an AI code completion tool to combine loops and conditionals.

Prompt:

“Generate Python code to count how many numbers in a list are even and odd.”



The screenshot shows a code editor window with an AI-generated Python script. The script defines a function `count_even_odd_numbers` that takes a list of integers and returns a tuple of even and odd counts. It includes validation for sample inputs and an explanation of logic flow.

```
% Generate + Code + Markdown
def count_even_odd_numbers():
    """
    Counts the number of even and odd integers in a list.

    Args:
        numbers (list): A list of integers.

    Returns:
        tuple: A tuple containing (even_count, odd_count).

    """
    even_count = 0
    odd_count = 0
    for num in numbers:
        if num % 2 == 0:
            even_count += 1
        else:
            odd_count += 1
    return even_count, odd_count

# Correct count validation.
print("## Validation with Sample Inputs")

# Sample Input 1
my_list_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even, odd = count_even_odd(my_list_1)
print("List: ", my_list_1)
print("Even numbers: {}, Odd numbers: {}".format(even, odd))

# Sample Input 2
my_list_2 = [15, 22, 30, 41, 58]
even, odd = count_even_odd(my_list_2)
print("List: ", my_list_2)
print("Even numbers: {}, Odd numbers: {}".format(even, odd))

# Sample Input 3
my_list_3 = []
even, odd = count_even_odd(my_list_3)
print("List: ", my_list_3)
print("Even numbers: {}, Odd numbers: {}".format(even, odd))

# Explanation of logic flow.
print("## Explanation of logic flow")
print("1. The function works as follows:")
print("1.1. Initialization: 'even_count' and 'odd_count' are set to 0.")
print("1.2. Iteration: It iterates through each number in the list 'numbers'.")
print("1.3. Condition: Checks the value of 'num % 2 == 0'. Checks if the number is even (remainder is 0 when divided by 2). If true, 'even_count' is incremented; otherwise, 'odd_count' is incremented.")
print("1.4. Return Value: After checking all numbers, the function returns both 'even_count' and 'odd_count'.")

## Validation with Sample Inputs
List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Even numbers: 5, Odd numbers: 5
List: [15, 22, 30, 41, 58]
Even numbers: 3, Odd numbers: 2
List: []
Even numbers: 0, Odd numbers: 0
```

Expected Output:

- AI-generated code using loop and if condition.
- Correct count validation.
- Explanation of logic flow.

Task Description #3 (AI-Based Code Completion for Class)

Attributes Validation)

Task: Use an AI tool to complete a Python class that validates user input.

Prompt:

“Generate a Python class User that validates age and email using conditional statements.”

Expected Output:

- AI-generated class with validation logic.
 - Verification of condition handling.
 - Test cases for valid and invalid inputs.

Task Description #4 (AI-Based Code Completion for Classes) Task:

Use an AI code completion tool to generate a Python class for managing student details.

Prompt:

“Generate a Python class Student with attributes (name, roll number, marks) and methods to calculate total and average marks.”

```

class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def calculate_total_marks(self):
        # Calculate the total marks.
        # The student object will have the attribute 'marks' which is a list of marks across different subjects.
        if not self.marks:
            raise ValueError("Marks must be a list of numbers")
        else:
            return sum(self.marks)

    def calculate_average_marks(self):
        # Calculate the average marks of the student.
        # Marks must be a list of numbers.
        # If no marks, average is 0.0.
        # Otherwise, calculate the average by dividing total marks by the count of marks.
        if not self.marks:
            return 0.0
        else:
            return self.calculate_total_marks() / len(self.marks)

    def __str__(self):
        # Returns a string representation of the student object.
        return f"Student Name: {self.name}, Roll No: {self.roll_number}, Total Marks: {self.calculate_total_marks()}"

```

```

student = Student("John", "1001", [90, 80, 70])
print(f"Total Marks for student John: {student.calculate_total_marks()}")
print(f"Average Marks for student John: {student.calculate_average_marks()}")

```

Expected Output:

- AI-generated class code.
- Verification of correctness and completeness of class structure.
- Minor manual improvements (if needed) with justification.

Task Description 5 (AI-Assisted Code Completion Review) Task:

Use an AI tool to generate a complete Python program using classes, loops, and conditionals together.

Prompt:

“Generate a Python program for a simple bank account system using class, loops, and conditional statements.”

- Identification of strengths and limitations of AI suggestions.
- Reflection on how AI assisted coding productivity.