

**Program :**B.tech(CSE)

**Specialization :**AIML

**Course Title :**AI Assisted Coding

**Course Code :**24CS002PC215

**Semester :**3rd semester

**Academic Session :**2025-2026

**Name of Student :**Challa Sravya

**Enrollment No. :**2403A52033

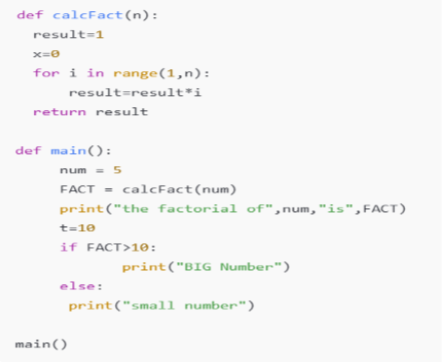
**Batch No. :**02

**Date :**07/10/2025

LAB ASSIGNMENT-10.2

TASK DESCRIPTION-1: Assisted Code Review (Basic Errors)  
• Write python program as shown below.  
• Use an AI assistant to review and suggest corrections.

Given Incorrected Code:-

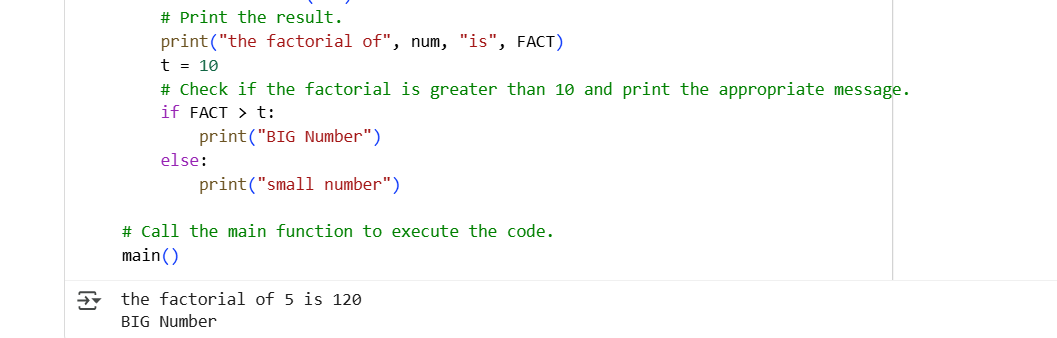
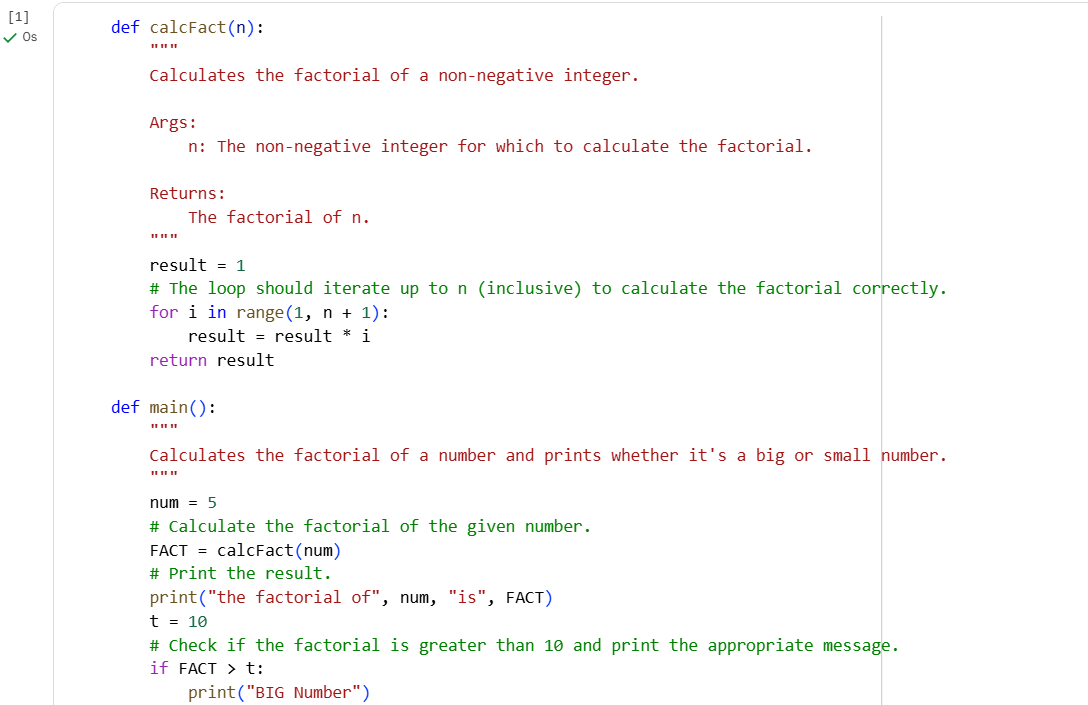


Expected Outcome#1:

Students need to submit corrected code with comments.

Prompt:-

def calcFact(n): result=1 x=0 for i in range(1,n): result=result\*i return result def main(): num = 5 FACT = calcFact(num) print("the factorial of",num,"is",FACT) t=10 if FACT>10: print("BIG Number") else: print("small number") main() correct the code and give me the corrected code include comments as well  
Corrected Code:-



TASK DESCRIPTION-2: Automatic Inline Comments  
• Write the Python code for Fibonacci as shown below and execute.  
• Ask AI to improve variable names, add comments, and apply PEP8 formatting  
(cleaned up).  
• Students evaluate which suggestions improve readability most. One.

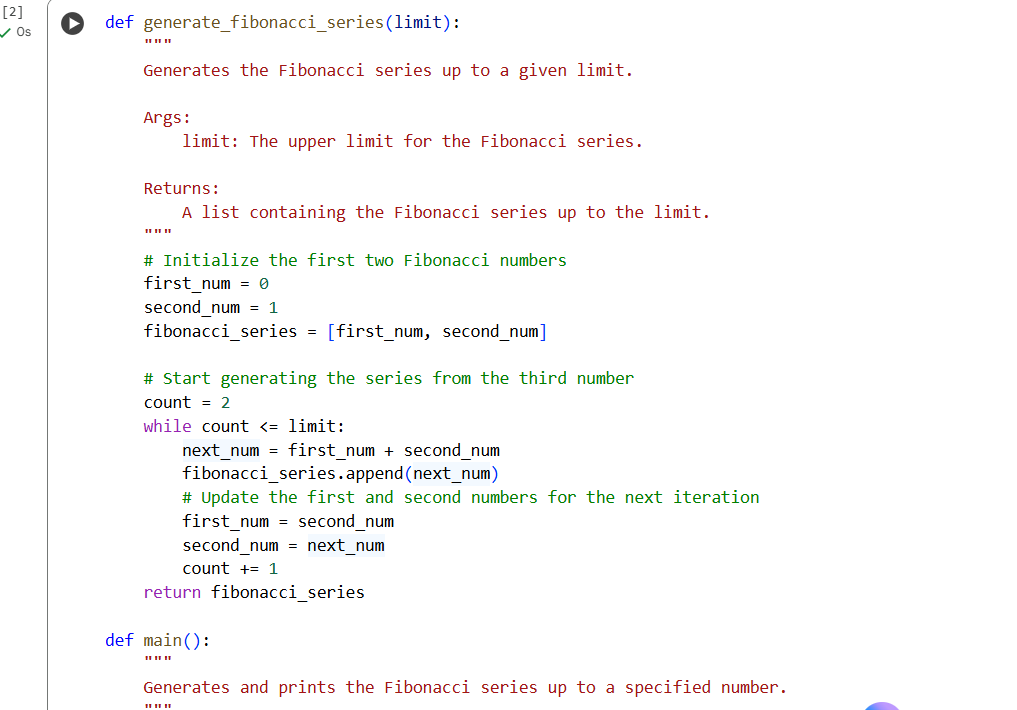
Given Code:-

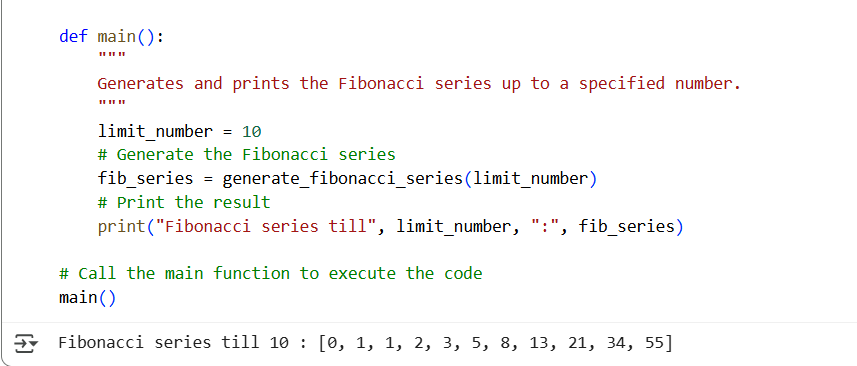


Expected Output#2: Clean format python code with much readability.

Prompt:- def f1(xX): a=0 b=1 c=2 Zz=[a,b] while c<=XX: d=a+b Zz.append(d) a=b b=d c=c+1 return Zz def m(): NN=10 ans=f1(NN) print("fib series till",NN,":",ans) m() give me a corrected code with improving variable name , including comments, and apply PEP8 Formatting

Code:-





TASK DESCRIPTION-3: • Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply,  
divide).  
• Incorporate manual docstring in code with NumPy Style  
• Use AI assistance to generate a module-level docstring + individual function  
docstrings.  
• Compare the AI-generated docstring with your manually written one.

Manual docstring with Numpy Style:-

def add(a, b):

return a + b

def subtract(a, b):

return a – b

def multiply(a, b):

return a \* b

def divide(a, b):

if b == 0:

raise ZeroDivisionError("Cannot divide by zero.")

return a / b

Ai generated module-level docstring + individual function:-

"""

This module provides basic calculator functions.

It includes functions for addition, subtraction, multiplication, and division.

Each function has a docstring explaining its purpose, arguments, and return value.

"""

def add(x, y):

    """

    Adds two numbers.

    Args:

        x: The first number.

        y: The second number.

    Returns:

        The sum of x and y.

    """

    return x + y

def subtract(x, y):

    """

    Subtracts the second number from the first.

    Args:

        x: The first number.

        y: The second number.

    Returns:

        The difference between x and y.

    """

    return x - y

def multiply(x, y):

    """

    Multiplies two numbers.

    Args:

        x: The first number.

        y: The second number.

    Returns:

        The product of x and y.

    """

    return x \* y

def divide(x, y):

    """

    Divides the first number by the second.

    Args:

        x: The first number (dividend).

        y: The second number (divisor).

    Returns:

        The result of the division.

    Raises:

        ZeroDivisionError: If the divisor (y) is zero.

    """

    if y == 0:

        raise ZeroDivisionError("Division by zero is not allowed.")

    return x / y

# Example usage (optional, you can remove this if you only need the functions)

# if \_\_name\_\_ == "\_\_main\_\_":

#     num1 = 10

#     num2 = 5

#     print(f"{num1} + {num2} = {add(num1, num2)}")

#     print(f"{num1} - {num2} = {subtract(num1, num2)}")

#     print(f"{num1} \* {num2} = {multiply(num1, num2)}")

#     try:

#         print(f"{num1} / {num2} = {divide(num1, num2)}")

#         print(f"{num1} / 0 = {divide(num1, 0)}") # Example of division by zero

#     except ZeroDivisionError as e:

#         print(f"Error: {e}")

Compare:-

Prompt:- compare this code and """ This module provides basic calculator functions. It includes functions for addition, subtraction, multiplication, and division. Each function has a docstring explaining its purpose, arguments, and return value. """ def add(x, y):     """     Adds two numbers.     Args:         x: The first number.         y: The second number.     Returns:         The sum of x and y.     """     return x + y def subtract(x, y):     """     Subtracts the second number from the first.     Args:         x: The first number.         y: The second number.     Returns:         The difference between x and y.     """     return x - y def multiply(x, y):     """     Multiplies two numbers.     Args:         x: The first number.         y: The second number.     Returns:         The product of x and y.     """     return x \* y def divide(x, y):     """     Divides the first number by the second.     Args:         x: The first number (dividend).         y: The second number (divisor).     Returns:         The result of the division.     Raises:         ZeroDivisionError: If the divisor (y) is zero.     """     if y == 0:         raise ZeroDivisionError("Division by zero is not allowed.")     return x / y # Example usage (optional, you can remove this if you only need the functions) # if \_\_name\_\_ == "\_\_main\_\_": #     num1 = 10 #     num2 = 5 #     print(f"{num1} + {num2} = {add(num1, num2)}") #     print(f"{num1} - {num2} = {subtract(num1, num2)}") #     print(f"{num1} \* {num2} = {multiply(num1, num2)}") #     try: #         print(f"{num1} / {num2} = {divide(num1, num2)}") #         print(f"{num1} / 0 = {divide(num1, 0)}") # Example of division by zero #     except ZeroDivisionError as e: #         print(f"Error: {e}") And this code def add(a, b):     return a + b    def subtract(a, b):     return a - b def multiply(a, b):     return a \* b def divide(a, b):     if b == 0:         raise ZeroDivisionError("Cannot divide by zero.")     return a / b

Output:-

Here's a breakdown of the differences and similarities:

**Code Snippet 1 (from the cell with id 4574aa78):**

* **Docstrings:** This code includes a module-level docstring explaining the overall purpose of the module and individual docstrings for each function. These docstrings explain what the function does, its arguments (Args:), what it returns (Returns:), and in the case of divide, what errors it might raise (Raises:). This adheres to good documentation practices (like the Google Style Guide for docstrings) and makes the code much easier to understand and use.
* **Variable Names:** The variable names within the functions (x, y) are simple and clear.
* **Error Handling:** The divide function explicitly checks for division by zero and raises a ZeroDivisionError with a descriptive message.
* **Example Usage:** Includes an optional if \_\_name\_\_ == "\_\_main\_\_": block demonstrating how to use the functions and handle the ZeroDivisionError.

**Code Snippet 2:**

* **Docstrings:** This code lacks any docstrings at the module or function level. This makes it harder for someone reading the code to quickly understand what each function does without reading the code itself.
* **Variable Names:** Uses simple variable names (a, b).
* **Error Handling:** The divide function also checks for division by zero and raises a ZeroDivisionError, but the error message is slightly different ("Cannot divide by zero." vs "Division by zero is not allowed.").
* **Example Usage:** Does not include any example usage.

**Similarities:**

* Both snippets implement the same basic calculator functions: add, subtract, multiply, and divide.
* Both divide functions handle the case of division by zero by raising a ZeroDivisionError.
* Both snippets use simple and functional approaches for the calculations.

**In summary:**

Code Snippet 1 is significantly better documented and more robust due to the inclusion of docstrings and an example usage block that demonstrates error handling. Code Snippet 2 is functional but lacks the documentation that is crucial for code readability and maintainability, especially in larger projects or when collaborating with others.

THANK YOU