AI ASSITED CODING

LAB ASSIGNMENT -10.2

Program :BTech

Specialisation :AIML

Student Name :M.varshith reddy
Enrollment No :2403A51289
Batch No :B01
Date :07-10-2025

TASK1

Task Description#1 AI-Assisted Code Review (Basic Errors)

Write python program as shown below.

Use an AI assistant to review and suggest corrections.

Code that is given in Assignment

GIVEN CODE:

```
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()
```

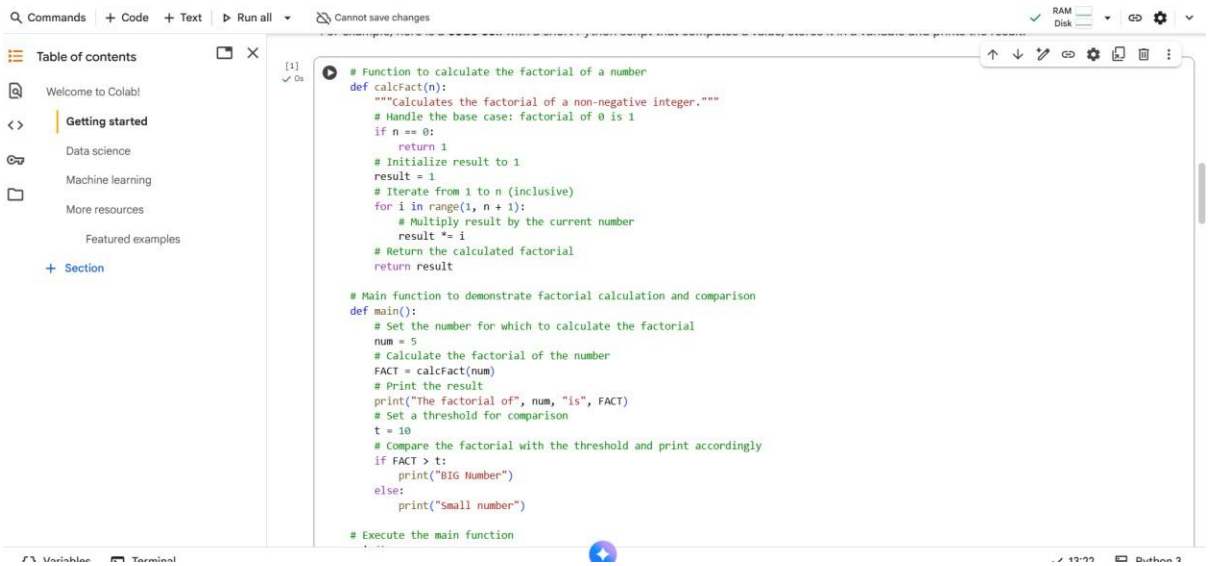
main.py

```python
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()
```

PROMPT:

Modify the above code and suggest for better implementation if possible and add comments to it.

Generated by AI



OUPUT:

```
The factorial of 5 is 120
BIG Number
```

OBSERVATION:

As per my observation in the given code ,function call is not properly written and variables x and t are unwanted variables in the code.

It has a logical error in it i.e in for loop
It should be for i in range(1,n+1)

TASK2

Task Description:

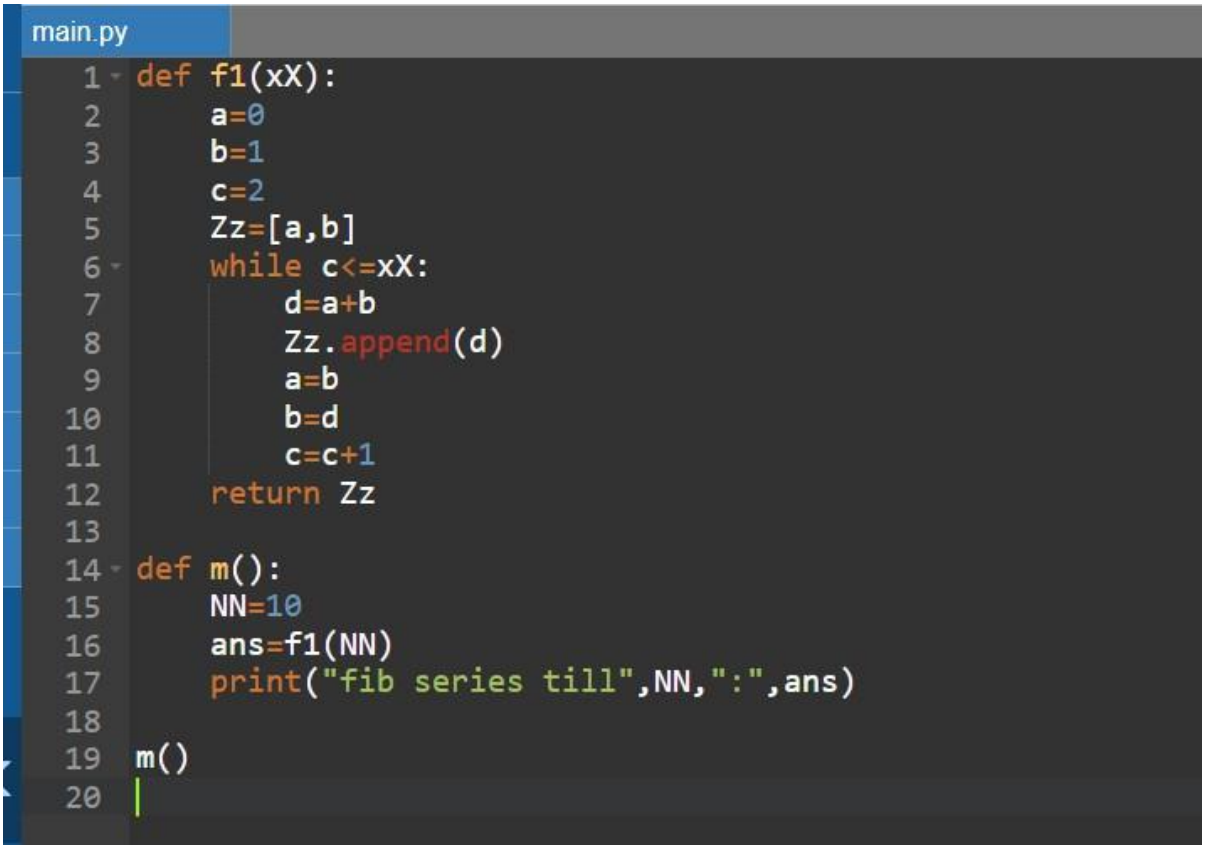
Write the Python code for Fibonacci as shown below and execute.

• Ask AI to improve variable names, add comments, and apply PEP8 formatting
(cleaned up).

• Students evaluate which suggestions improve readability most. one.

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

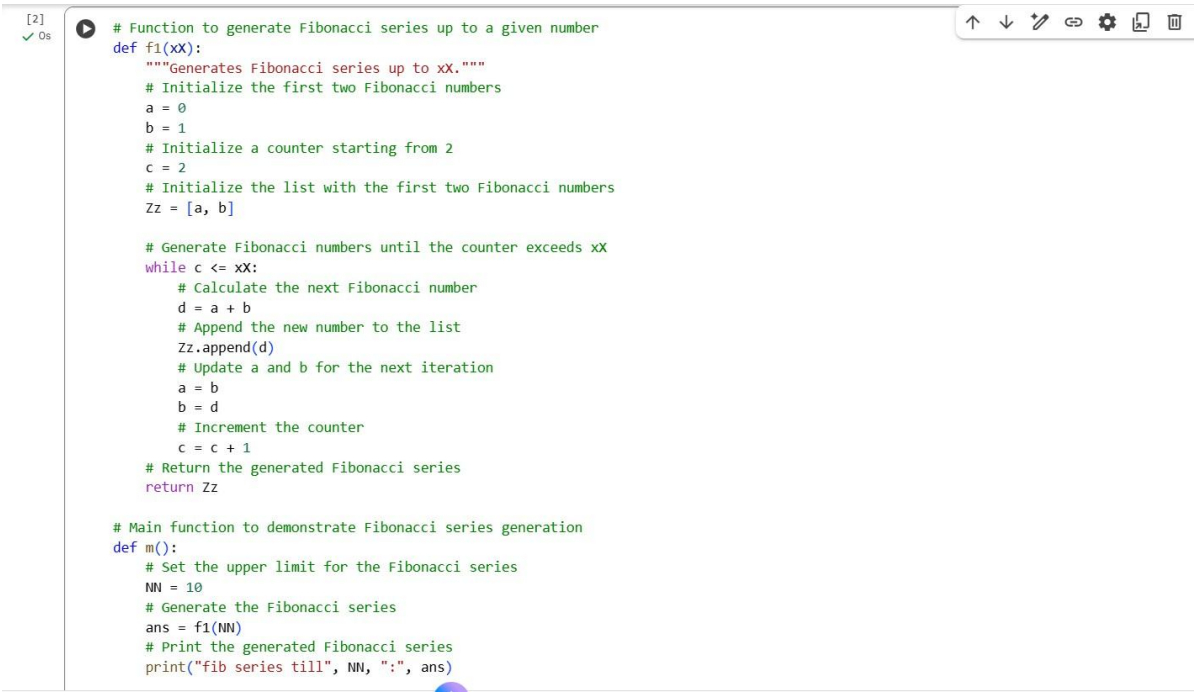
Given Code:



PROMPT:

From the Given Code ,modify it if it has any logical or syntactical errors it.Put comments for better understanding

Generated by AI



```
[2]
✓ 0s
# Function to generate Fibonacci series up to a given number
def f1(xX):
    """Generates Fibonacci series up to xX."""
    # Initialize the first two Fibonacci numbers
    a = 0
    b = 1
    # Initialize a counter starting from 2
    c = 2
    # Initialize the list with the first two Fibonacci numbers
    Zz = [a, b]

    # Generate Fibonacci numbers until the counter exceeds xX
    while c <= xX:
        # Calculate the next Fibonacci number
        d = a + b
        # Append the new number to the list
        Zz.append(d)
        # Update a and b for the next iteration
        a = b
        b = d
        # Increment the counter
        c = c + 1
    # Return the generated Fibonacci series
    return Zz

# Main function to demonstrate Fibonacci series generation
def m():
    # Set the upper limit for the Fibonacci series
    NN = 10
    # Generate the Fibonacci series
    ans = f1(NN)
    # Print the generated Fibonacci series
    print("fib series till", NN, ":", ans)
```

OUPUT:

```
fib series till 10 : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

OBSERVATION:

In given Code The variable names (xX, c, Zz, NN, ans) are not very

descriptive, which can make the code harder to understand at a glance.

Where as in **AI generated Code it** better names for variables and used a better function name

TASK 3

TASK Description:

• 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.

Write a Python Program in which it has multiple methods like add,substract,multiply,divide

```
class Calculator:
    """A simple calculator class with basic arithmetic operations."""

    def add(self, x, y):
        """Adds two numbers."""
        return x + y

    def subtract(self, x, y):
        """Subtracts the second number from the first."""
        return x - y

    def multiply(self, x, y):
        """Multiplies two numbers."""
```

```
        return x * y

    def divide(self, x, y):
        """Divides the first number by the second.
        Returns an error message if the divisor is zero.
        """
        if y == 0:
            return "Error: Division by zero"
        return x / y

# Example usage:
calculator = Calculator()
```

```
num1 = 10
num2 = 5

print(f"{num1} + {num2} = {calculator.add(num1, num2)}")
print(f"{num1} - {num2} = {calculator.subtract(num1, num2)}")
print(f"{num1} * {num2} = {calculator.multiply(num1, num2)}")
print(f"{num1} / {num2} = {calculator.divide(num1, num2)}")

num3 = 10
num4 = 0
```

```
print(f"{num3} / {num4} =
{calculator.divide(num3, num4)}")
```

Incorporate manual docstring in code with NumPy Style

```
class Calculator:
    """
    A simple calculator class with
basic arithmetic operations.
    """


    def add(self, x, y):
        """
        Adds two numbers.
```

Parameters

---

x : int or float

    The first number.

y : int or float

    The second number.

Returns

---

int or float

    The sum of x and y.

"""

return x + y

```
    def subtract(self, x, y):
        """
        Subtracts the second number from the first.

        Parameters
        -------
        x : int or float
            The first number.
        y : int or float
            The second number.

        Returns
        -----
```

```
    int or float
        The difference between x and y.
    """
    return x - y

def multiply(self, x, y):
    """
    Multiplies two numbers.

    Parameters
    -------
    x : int or float
        The first number.
    y : int or float
```

```
        The second number.

        Returns
        -------
        int or float
            The product of x and y.
        """
        return x * y

    def divide(self, x, y):
        """
        Divides the first number by the second.

        if y == 0:
```

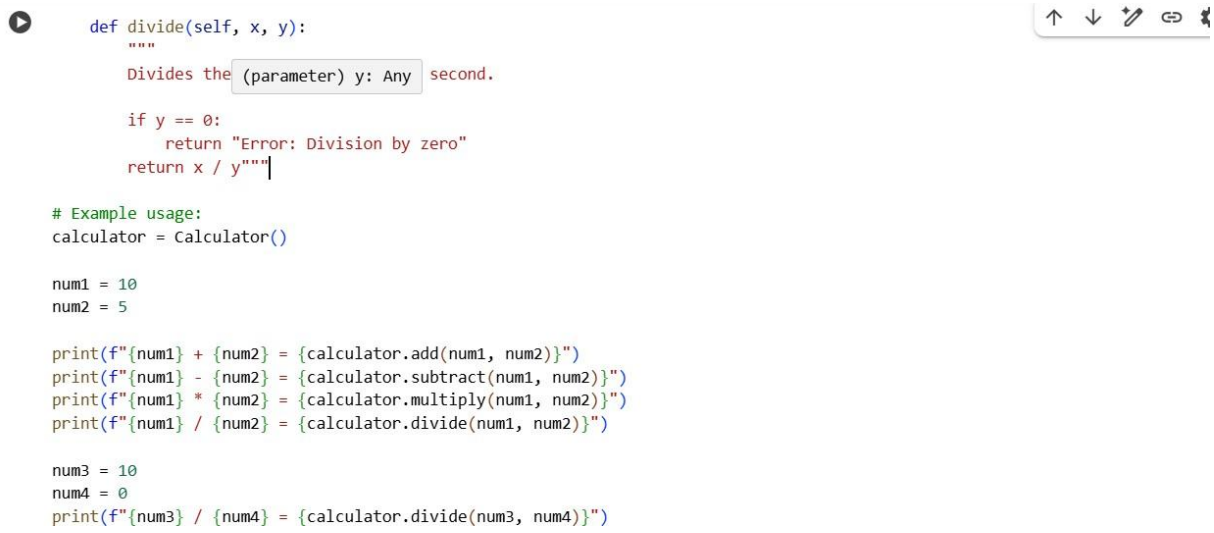
```
        return "Error: Division by zero"
    return x / y"""

# Example usage:
calculator = Calculator()

num1 = 10
num2 = 5

print(f"{num1} + {num2} = {calculator.add(num1, num2)}")
print(f"{num1} - {num2} = {calculator.subtract(num1, num2)}")
print(f"{num1} * {num2} = {calculator.multiply(num1, num2)}")
```

```
print(f"{num1} / {num2} =
{calculator.divide(num1, num2)}")


num3 = 10
num4 = 0
print(f"{num3} / {num4} =
{calculator.divide(num3, num4)}")
```

Use AI assistance to generate a module-level docstring + individual function docstrings.

```python
class Calculator:
    """
    A simple calculator class with basic arithmetic operations.
    """

    def add(self, x, y):
        """
        Adds two numbers.

        Parameters
        ----------
        x : int or float
            The first number.
        y : int or float
            The second number.

        Returns
        -------
        int or float
            The sum of x and y.
        """
        return x + y

    def subtract(self, x, y):
        """
        Subtracts the second number from the first.

        Parameters
        ----------
        x : int or float
            The first number.
        y : int or float
            The second number.
```

```python
    def subtract(self, x, y):
        """
        Subtracts the second number from the first.

        Parameters
        ----------
        x : int or float
            The first number.
        y : int or float
            The second number.

        Returns
        -------
        int or float
            The difference between x and y.
        """
        return x - y

    def multiply(self, x, y):
        """
        Multiplies two numbers.

        Parameters
        ----------
        x : int or float
            The first number.
        y : int or float
            The second number.

        Returns
        -------
        int or float
            The product of x and y.
        """
```



```
    def divide(self, x, y):
        """
        Divides the (parameter) y: Any second.

        if y == 0:
            return "Error: Division by zero"
        return x / y"""

# Example usage:
calculator = Calculator()

num1 = 10
num2 = 5

print(f"{num1} + {num2} = {calculator.add(num1, num2)}")
print(f"{num1} - {num2} = {calculator.subtract(num1, num2)}")
print(f"{num1} * {num2} = {calculator.multiply(num1, num2)}")
print(f"{num1} / {num2} = {calculator.divide(num1, num2)}")

num3 = 10
num4 = 0
print(f"{num3} / {num4} = {calculator.divide(num3, num4)}")
```

OBSERVATION:

As of my observation the manual code has docstrings in simple english in it.It just have comments and docstrings

Where as in AI generated Code it used numpy and docstrings for better understanding and more efficient for use case,it has gone with better

approach