AI ASSISTED CODING

ASSIGNMENT: 10.3

NAME: KOUDAGANI SAHITHYA

BATCH: 02

HT.NO: 2403A52063

TASK 1:

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

- Write python program as shown below.
- Use an AI assistant to review and suggest corrections.

```
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:

Rewrite the above code by clearing the errors

Code:

```python
def calcFact(n):
    result = 1
    # Loop from 1 to n (inclusive) to calculate factorial
    for i in range(1, n + 1):
        result = result * i
    return result

def main():
    # Get input from the user
    num = int(input("Enter a non-negative integer: "))
    # Check if the number is non-negative
    if num < 0:
        print("Factorial is not defined for negative numbers.")
    else:
        # Calculate the factorial
        FACT = calcFact(num)
        # Print the result
        print("The factorial of", num, "is", FACT)

        # Check if the factorial is greater than 10
        t = 10
        if FACT > t:
            print("BIG Number")
        else:
            print("Small number")

# Call the main function to start the program
main()
```

Output:

```
Enter a non-negative integer: 5
The factorial of 5 is 120
BIG Number
```

Code explanation:

This Python code calculates the factorial of a non-negative integer entered by the user.

Calc Fact(n) function: Takes a number n and calculates its factorial by multiplying all integers from 1 to n.

Main () function:

Gets a non-negative integer input from the user.

If the input is negative, it prints an error.

If non-negative, it calls calcFact to get the factorial.

Prints the calculated factorial.

Compares the factorial to 10 and prints "BIG Number" if it's greater than 10, otherwise prints "Small number".

The program starts by calling main().

In short, it calculates the factorial of a user-provided number and tells you if the result is greater than 10.

TASK 2:

Task Description#2 Automatic Inline Comments

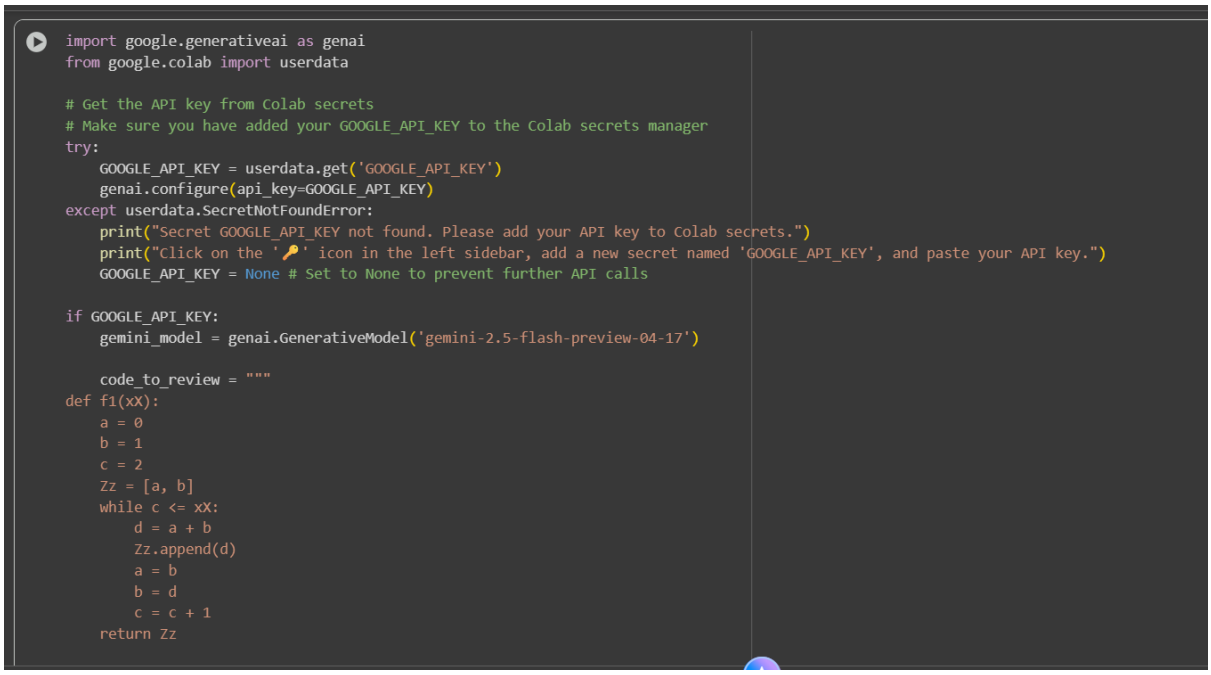
- Write the Python code for Fibonacci as shown below and execute.
- Ask AI to improve variable names, add comments, and apply PEP8 formatting (cleaned up).
- Students evaluate which suggestions improve readability most. one.

```
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()
```

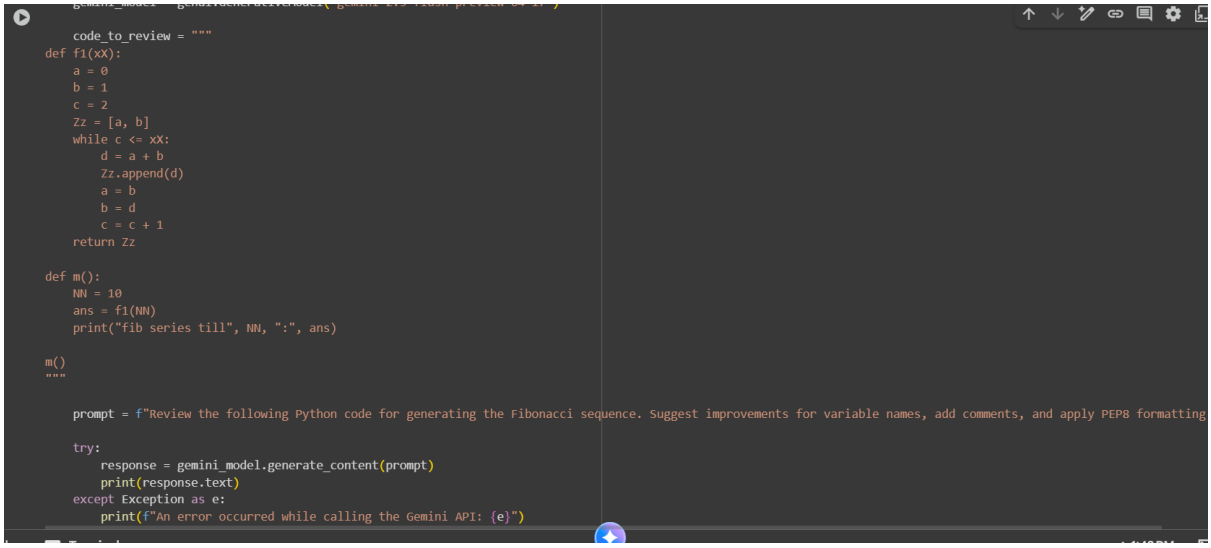
Prompt:

Write a python code for Fibonacci series

CODE:

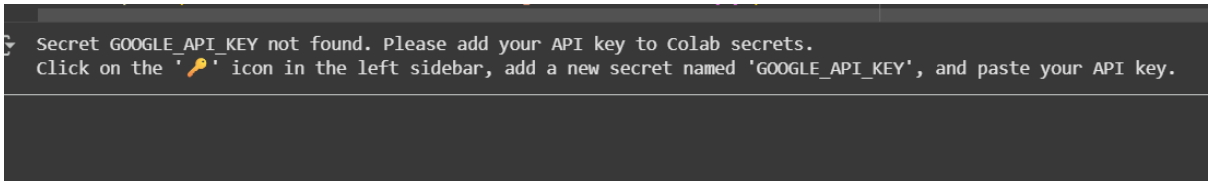


```python
import google.generativeai as genai
from google.colab import userdata

# Get the API key from Colab secrets
# Make sure you have added your GOOGLE_API_KEY to the Colab secrets manager
try:
    GOOGLE_API_KEY = userdata.get('GOOGLE_API_KEY')
    genai.configure(api_key=GOOGLE_API_KEY)
except userdata.SecretNotFoundError:
    print("Secret GOOGLE_API_KEY not found. Please add your API key to Colab secrets.")
    print("Click on the '🔑' icon in the left sidebar, add a new secret named 'GOOGLE_API_KEY', and paste your API key.")
    GOOGLE_API_KEY = None # Set to None to prevent further API calls

if GOOGLE_API_KEY:
    gemini_model = genai.GenerativeModel('gemini-2.5-flash-preview-04-17')

    code_to_review = """
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
```



```python
    code_to_review = """
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()
"""

    prompt = f"Review the following Python code for generating the Fibonacci sequence. Suggest improvements for variable names, add comments, and apply PEP8 formatting

    try:
        response = gemini_model.generate_content(prompt)
        print(response.text)
    except Exception as e:
        print(f"An error occurred while calling the Gemini API: {e}")
```

OUTPUT:



```
Secret GOOGLE_API_KEY not found. Please add your API key to Colab secrets.
Click on the '🔑' icon in the left sidebar, add a new secret named 'GOOGLE_API_KEY', and paste your API key.
```

Code explanation:

This Python code uses the Google Generative AI library to get code review suggestions. It retrieves your API key from Colab secrets, configures the

Gemini model, and then sends the provided Fibonacci code as a prompt to the model. The model analyzes the code and generates suggestions for improving variable names, adding comments, and applying PEP8 formatting, which are then printed as output.

TASK 3:

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.

Prompt:

Write a python script with 3-4 functions

CODE:

```
# Example usage of the calculator functions
num1 = 10
num2 = 5

sum_result = add(num1, num2)
difference_result = subtract(num1, num2)
product_result = multiply(num1, num2)
division_result = divide(num1, num2)

print(f"Sum: {sum_result}")
print(f"Difference: {difference_result}")
print(f"Product: {product_result}")
print(f"Division: {division_result}")

# Example of division by zero
try:
    divide(num1, 0)
except ValueError as e:
    print(f"Error: {e}")
```

Output:

```
Sum: 15
Difference: 5
Product: 50
Division: 2.0
Error: Cannot divide by zero
```

Code explanation:

The first cell defines basic calculator functions: add, subtract, multiply, and divide. Each function has a docstring explaining its purpose, parameters, and

return value. The second cell demonstrates how to use these functions with example numbers, printing the results of each operation. It also includes error handling for division by zero using a try-except block.