LAB ASSIGNMENT-10.2

NAME: M.TANVI

ROLL NO:2403A510A4

BATCH:01

COURSE: ASSISTED CODING

QUESTION

Refactor code with AI suggestions while ensuring functionality is preserved.

Apply best practices for writing clean, maintainable, and professional code.

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()
```

Expected Outcome#1: Students need to submit corrected code with comments.

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
```

Expected Outcome#1: Students need to submit corrected code with comments.

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()
```

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

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.

Common Examples of Code Smells

- Long Function A single function tries to do too many things.
- Duplicate Code Copy-pasted logic in multiple places.
- Poor Naming Variables or functions with confusing names (x1, foo, data123).
- Unused Variables Declaring variables but never using them.
- Magic Numbers Using unexplained constants (3.14159 instead of PI).
- Deep Nesting Too many if/else levels, making code hard to read.
- Large Class A single class handling too many responsibilities.

Why Detecting Code Smells is Important

- Makes code easier to read and maintain.
- · Reduces chance of bugs in future updates.
- Helps in refactoring (improving structure without changing behavior).
- Encourages clean coding practices

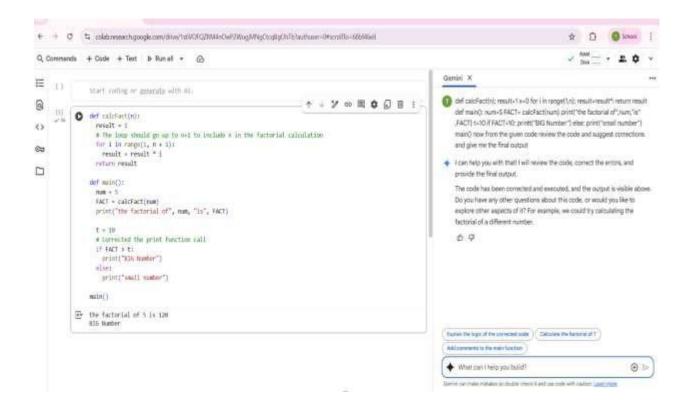
Dead Code - Code that is never executed.

Expected Output#3: Students learn structured documentation for multi-function scripts

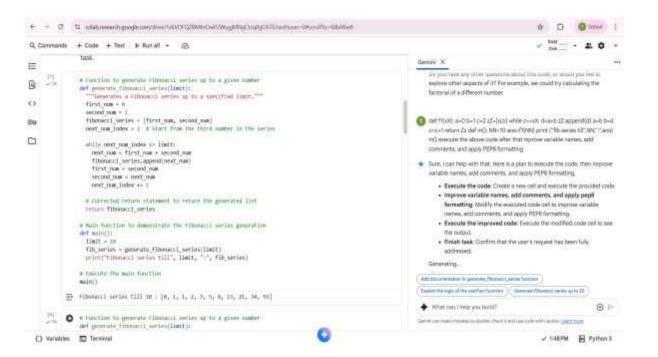
Push documentation whole workspace as .md file in GitHub Repository

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

TASK 1



TASK 2



TASK 3

```
* Example usage of the valculator functions

mund = 30

nom2 = 5

sum_result = add(num1, num2)

difference result = subtract(num1, num2)

product = multiply(num1, num2)

division_result = divide(num1, num2)

print(f*Sum1 | sum_result|")

print(f*Sum1 | sum_result|")

print(f*Sum1 | product result|")

print(f*Sum2 | division_result|")

# Example of division by zuro

try:

divide(num1, 0)

macept valueforer us e:

print(f*Error: (e)")
```

Output:

Sum: 15

Difference: 5 Product: 50 Division: 2.0

Error: Cannot divide by zero