**PROGRAM : B.TECH/CSE**

**SPECIALIZATION : AIML**

**COURSE TITLE : AI ASSISTED CODING**

**COURSE CODE : 24CS101PC214**

**SEMESTER : 3RD**

**NAME OF THE STUDENT : KARNAKANTI GODADEVI**

**ENROLLMENT NO : 2403A52003**

**BATCH NO : 01**

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

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

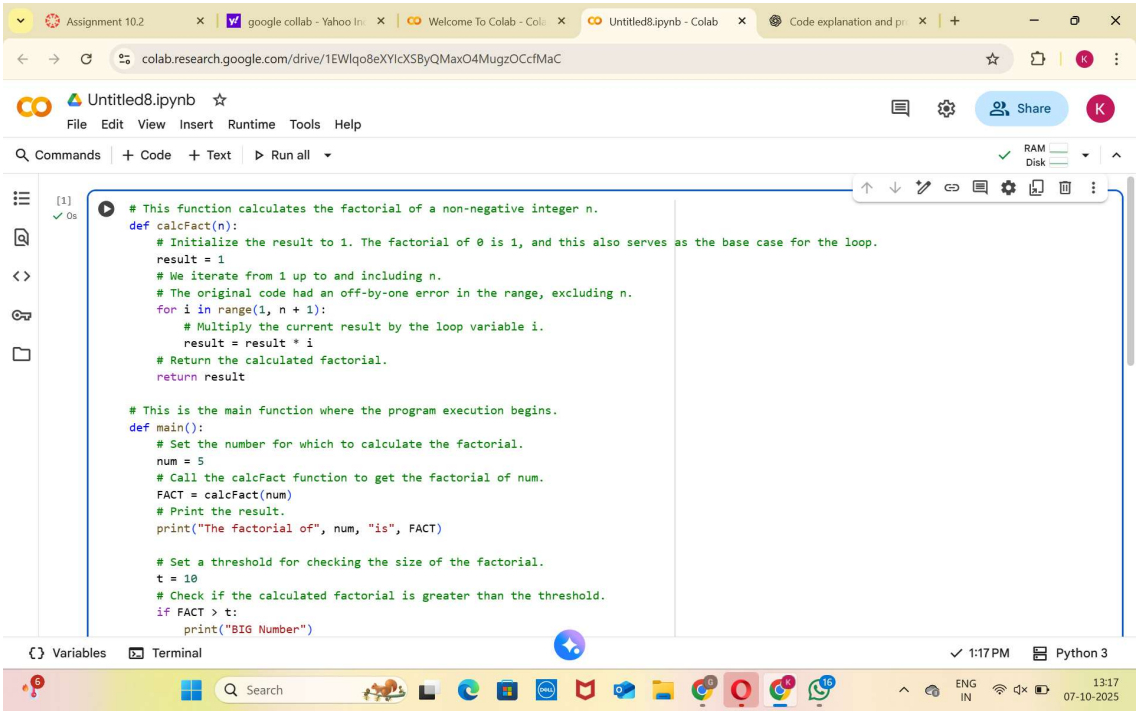
```
def calcFact(n):
  result=1
  x=0
  for i in range(1,n):
      result=result*i
  return result

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()
```

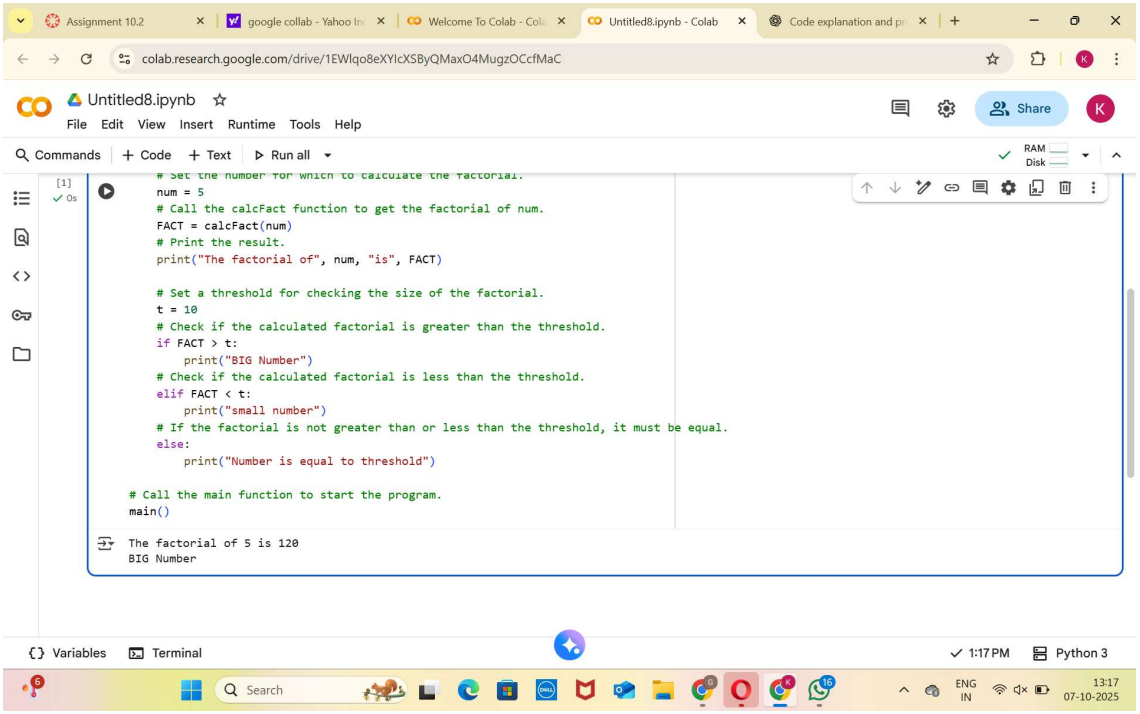
**Expected Outcome#1:** Students need to submit corrected code with comments.

**Prompt:**

Review this Python code for factorial calculation, identify syntax or logic errors, and suggest corrected and well-commented version using best coding practices.

**Code&Output:**




**Observation:**

- The code defines a function calcFact to compute the factorial of a non-negative integer n.
- It uses a loop to multiply numbers from 1 up to n to get the factorial.
- The main function sets a number (5) and calculates its factorial using calcFact.

- It then compares the calculated factorial to a threshold (10) and prints "BIG Number", "small number", or "Number is equal to threshold".
- The output shows the factorial of 5 is 120 and correctly identifies it as a "BIG Number" because 120 is greater than 10.

**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.

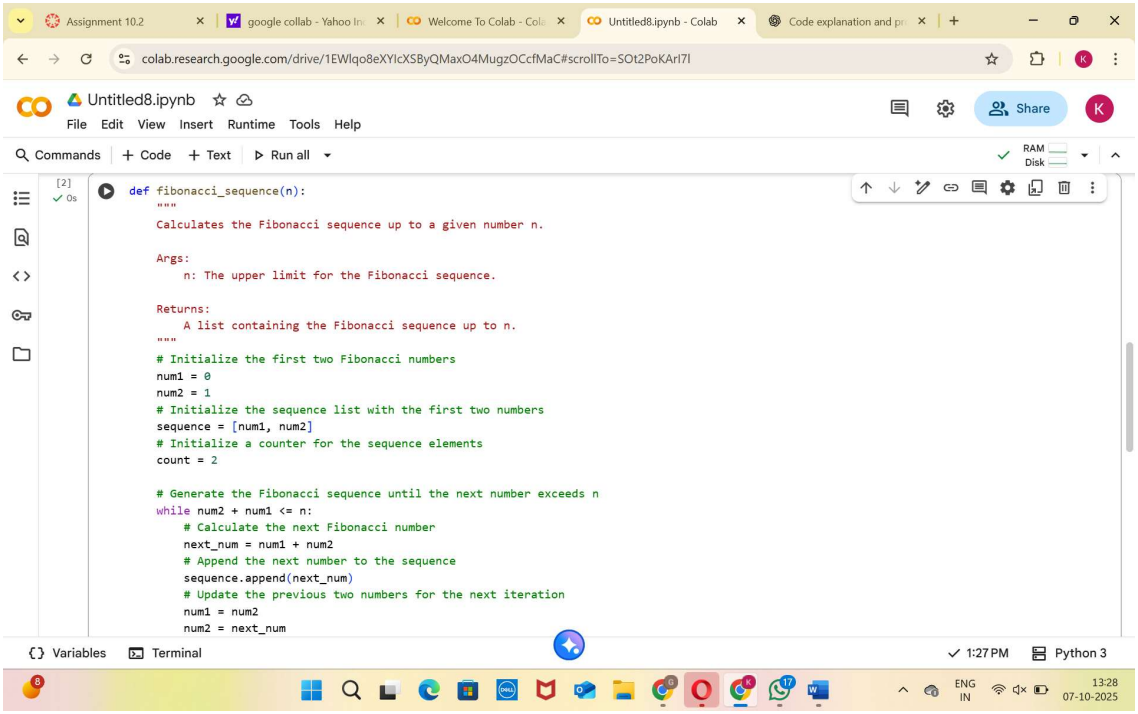
```
def f1(xX):
    a=0
    b=1
    c=2
    Zz=[a,b]
    while c<=xX:
        d=a+b
        Zz.append(d)
        a=b
        b=d
        c=c+1
    return Zz

def m():
      NN=10
      ans=f1(NN)
      print("fib series till",NN,":",ans)

m()
```

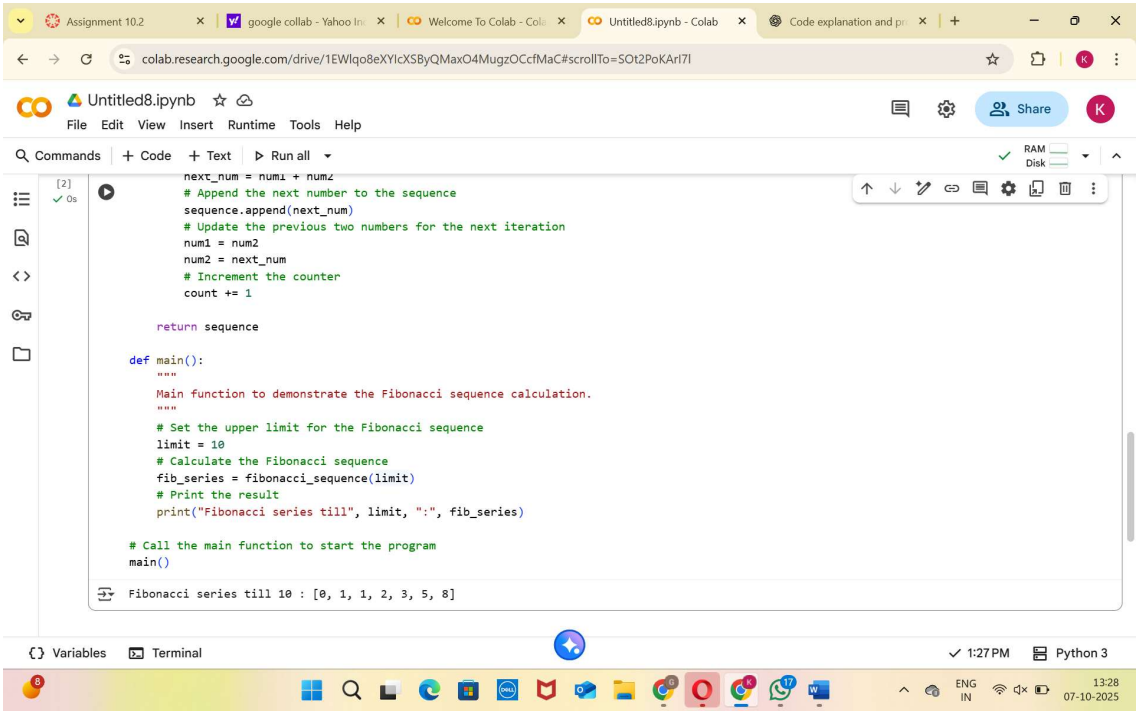
**Expected Output#2:** Clean format python code with much readability.

**Prompt:**

Review this Fibonacci Python code for readability and logic. Apply PEP8 formatting, rename unclear variables, add inline comments, and include docstrings for clarity.

**Code&Output:**




**Observation:**

- The code defines a function fibonacci_sequence that calculates the Fibonacci sequence up to a given limit n.
- It initializes the first two Fibonacci numbers (0 and 1) and a list to store the sequence.

- A while loop iteratively calculates and appends the next Fibonacci number to the sequence until the next number exceeds the limit.
- The main function demonstrates how to call fibonacci_sequence and prints the resulting series.
- The output shows the Fibonacci numbers generated up to the specified limit (10 in this case), which are [0, 1, 1, 2, 3, 5, 8].

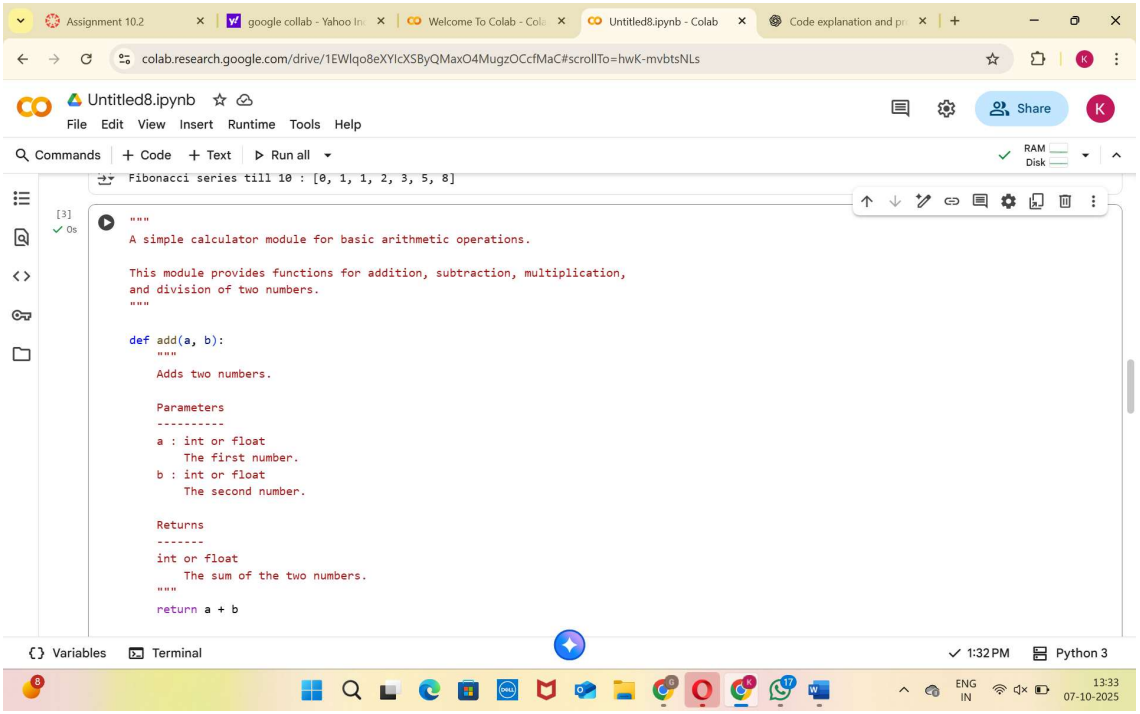
**Task Description#3**

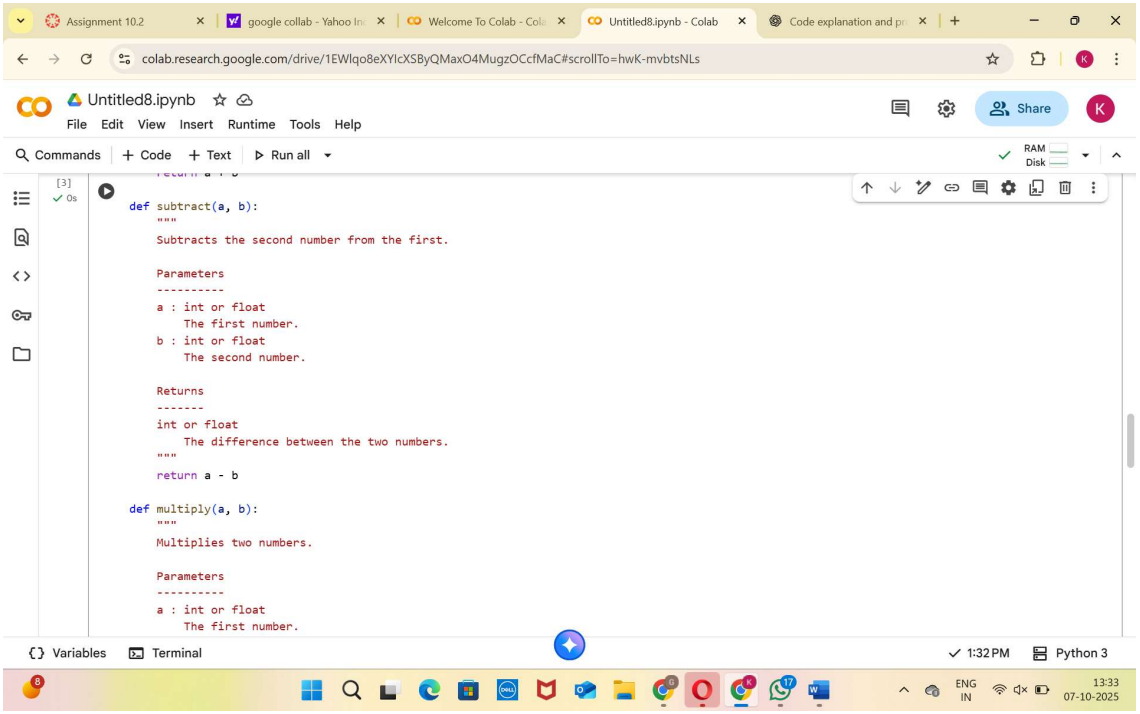
• Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
• Incorporate manual docstring in code with NumPy Style
• Use AI assistance to generate a module-level docstring + individual function docstrings.
• Compare the AI-generated docstring with your manually written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

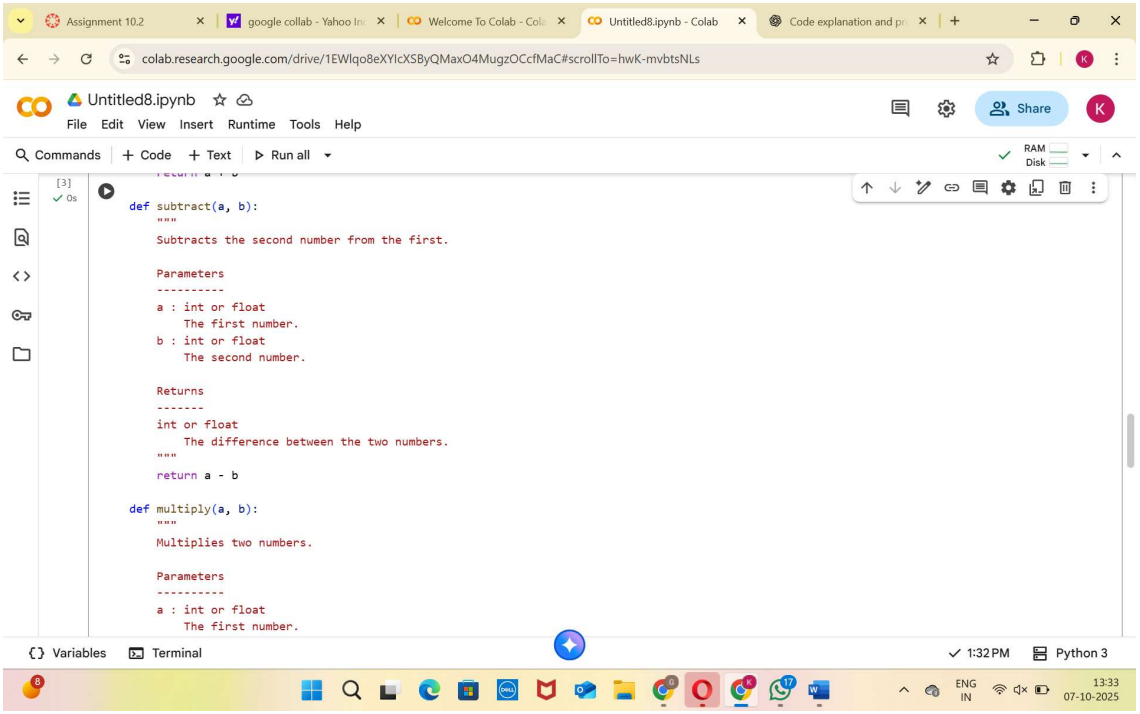
**Prompt:**

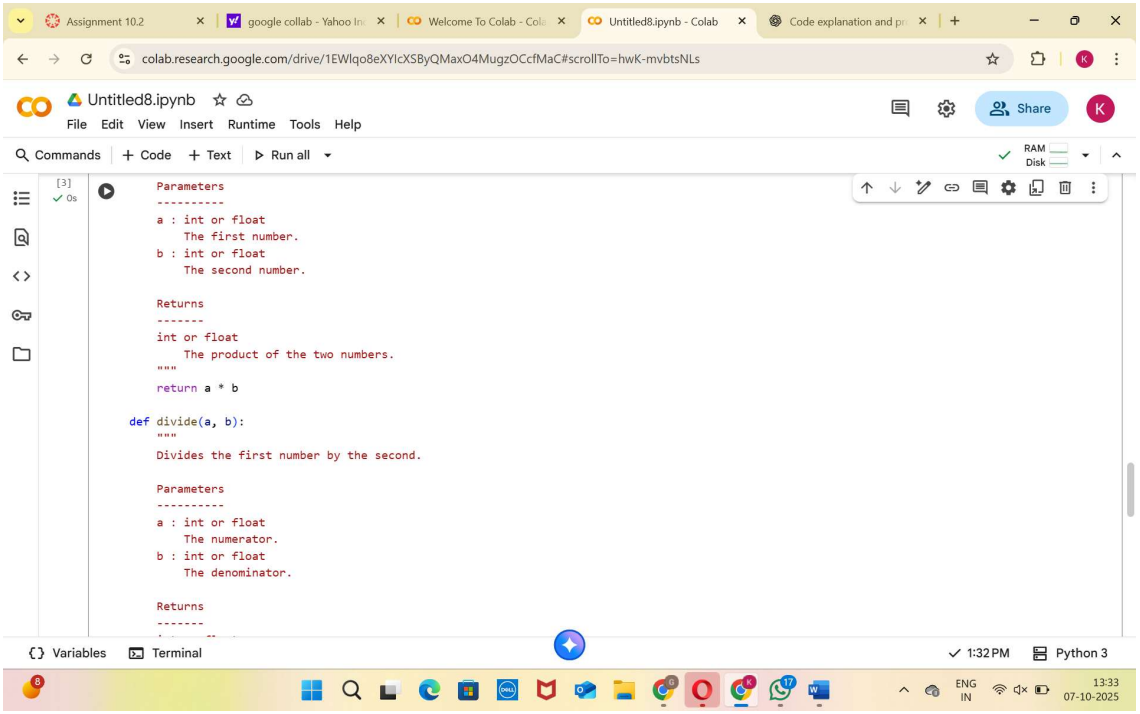
Generate a module-level and function-level docstring for my Python calculator script using NumPy style. Ensure clear descriptions, parameters, return values, and exception handling. Also, check for code smells and suggest refactoring improvements.

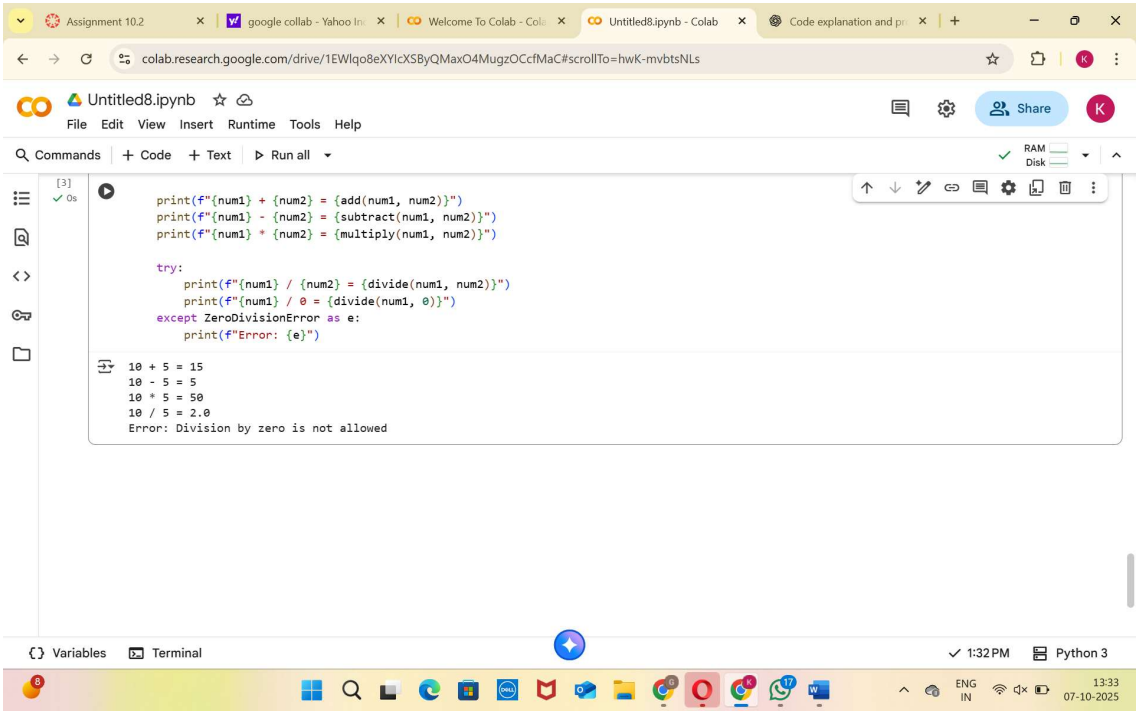
**Code&Output:**


Assignment 10.2
google collab - Yahoo In
Welcome To Colab - Cola
Untitled8.ipynb - Colab
Code explanation and pr
colab.research.google.com/drive/1EWlqo8eXYIcXSByQMaxO4MugzOCcfMaC#scrollTo=hwK-mvbtsNLs
Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
Fibonacci series till 10 : [0, 1, 1, 2, 3, 5, 8]
[3]
0s
"""
A simple calculator module for basic arithmetic operations.
This module provides functions for addition, subtraction, multiplication,
and division of two numbers.
"""
def add(a, b):
"""
Adds two numbers.
Parameters
a : int or float
The first number.
b : int or float
The second number.
Returns
int or float
The sum of the two numbers.
"""
return a + b
Variables
Terminal
1:32 PM
Python 3
ENG IN
13:33 07-10-2025


Assignment 10.2
google collab - Yahoo In
Welcome To Colab - Cola
Untitled8.ipynb - Colab
Code explanation and pr
colab.research.google.com/drive/1EWlqo8eXYIcXSByQMaxO4MugzOCcfMaC#scrollTo=hwK-mvbtsNLs
Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[3]
0s
def subtract(a, b):
"""
Subtracts the second number from the first.
Parameters
a : int or float
The first number.
b : int or float
The second number.
Returns
int or float
The difference between the two numbers.
"""
return a - b
def multiply(a, b):
"""
Multiplies two numbers.
Parameters
a : int or float
The first number.
Variables
Terminal
1:32 PM
Python 3
ENG IN
13:33 07-10-2025


Assignment 10.2
google collab - Yahoo In
Welcome To Colab - Cola
Untitled8.ipynb - Colab
Code explanation and pr
colab.research.google.com/drive/1EWlqo8eXYIcXSByQMaxO4MugzOCcfMaC#scrollTo=hwK-mvbtsNLs
Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[3]
0s
def subtract(a, b):
"""
Subtracts the second number from the first.
Parameters
a : int or float
The first number.
b : int or float
The second number.
Returns
int or float
The difference between the two numbers.
"""
return a - b
def multiply(a, b):
"""
Multiplies two numbers.
Parameters
a : int or float
The first number.
Variables
Terminal
1:32 PM
Python 3
ENG
IN
13:33
07-10-2025


Assignment 10.2
google collab - Yahoo In
Welcome To Colab - Cola
Untitled8.ipynb - Colab
Code explanation and pr
colab.research.google.com/drive/1EWlqo8eXYIcXSByQMaxO4MugzOCcfMaC#scrollTo=hwK-mvbtsNLs
Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[3]
0s
Parameters
a : int or float
The first number.
b : int or float
The second number.
Returns
int or float
The product of the two numbers.
"""
return a * b
def divide(a, b):
"""
Divides the first number by the second.
Parameters
a : int or float
The numerator.
b : int or float
The denominator.
Returns
Variables
Terminal
1:32 PM
Python 3
ENG
IN
13:33
07-10-2025




**Observation:**

- The code defines functions for basic arithmetic operations: addition, subtraction, multiplication, and division.
- It includes clear NumPy-style docstrings explaining each function's purpose, parameters, return values, and potential exceptions.
- The divide function explicitly checks for division by zero and raises a ZeroDivisionError.
- The example usage demonstrates how to use the functions and includes a try...except block to handle the ZeroDivisionError gracefully.
- The if __name__ == "__main__": block ensures the example usage only runs when the script is executed directly.