**SR UNIVERSITY**

**AI ASSISTED CODING**

# NAME:MOHAMMAD NASRIN BATCH:19

**Using AI to Improve Code Quality and Readability**

**Lab Objectives**

* Use AI for automated code review and quality enhancement.
* Identify and fix syntax, logical, performance, and security issues in Python code.
* Improve readability and maintainability through structured refactoring and comments.
* Apply prompt engineering for targeted improvements.
* Evaluate AI-generated suggestions against PEP 8 standards and software engineering best practices

**TASK 1: Syntax and Error Detection**

Identify and fix syntax, indentation, and variable errors in the given script.

# buggy\_code\_task1.py

def add\_numbers(a, b)

result = a + b

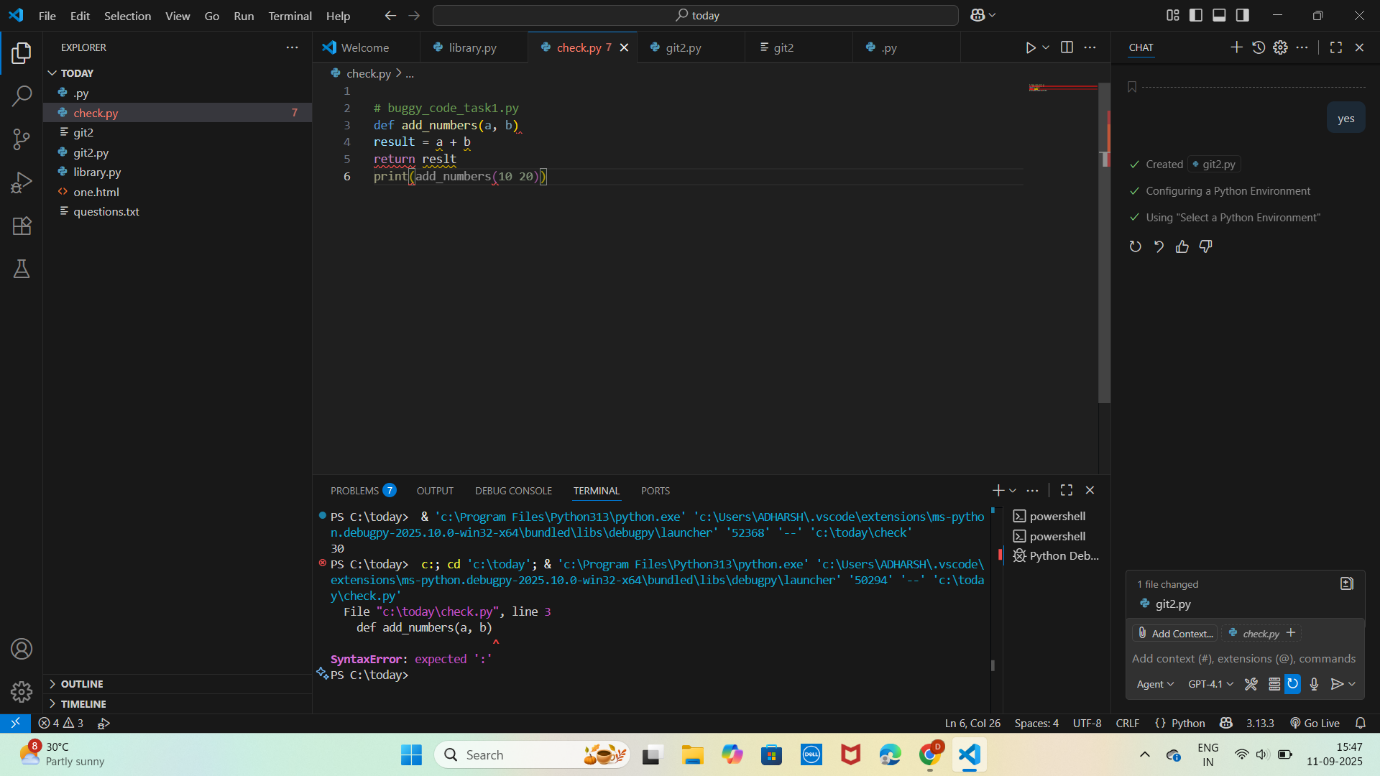
return reslt

print(add\_numbers(10 20))

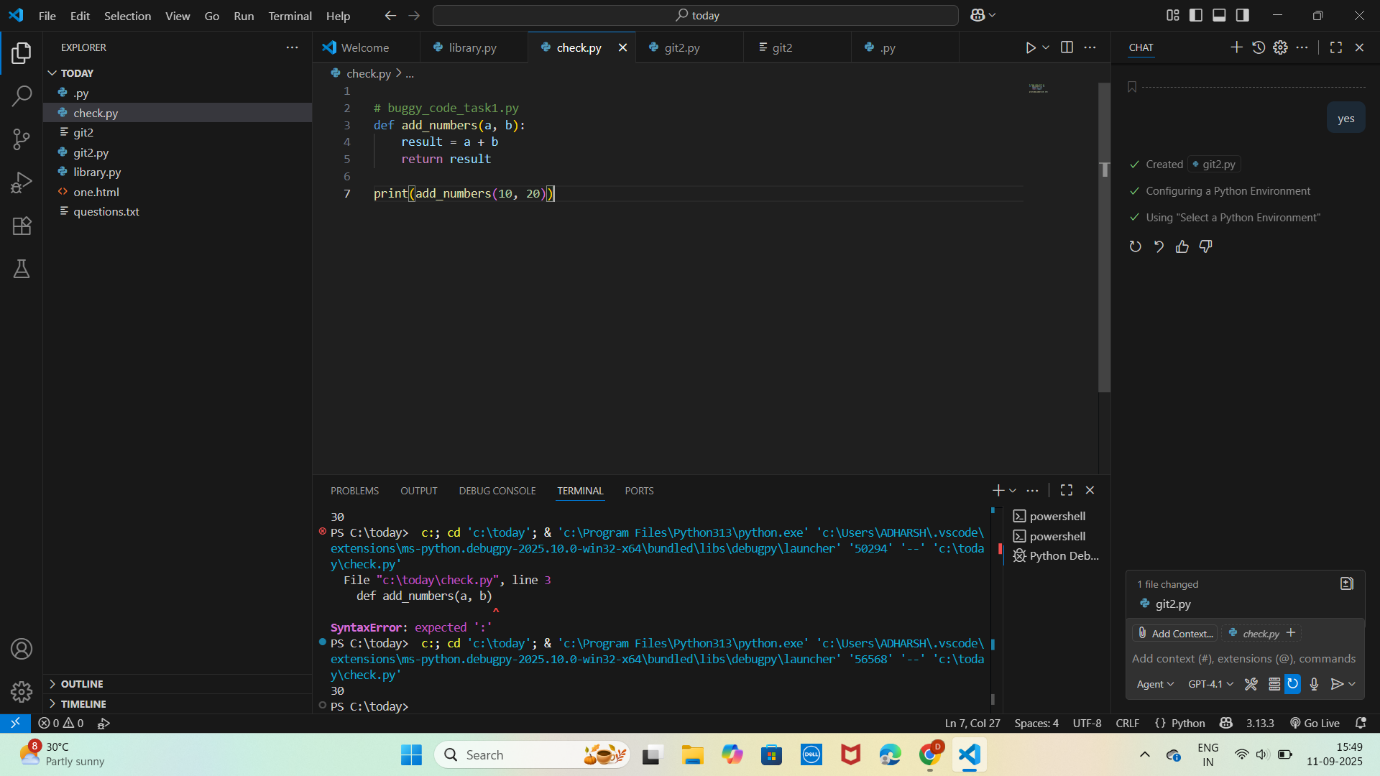
**Expected Output**:

* Corrected code with proper syntax (: after function, fixed variable name, corrected function call).
* AI should explain what was fixed.

**Before Detection:**

****

**After Detection:**

****

**Task 2: Logical and Performance Issue Review**

Optimize inefficient logic while keeping the result correct.

# buggy\_code\_task2.py

def find\_duplicates(nums):

duplicates = []

for i in range(len(nums)):

for j in range(len(nums)):

if i != j and nums[i] == nums[j] and nums[i] not in duplicates:

duplicates.append(nums[i])

return duplicates

numbers = [1,2,3,2,4,5,1,6,1,2]

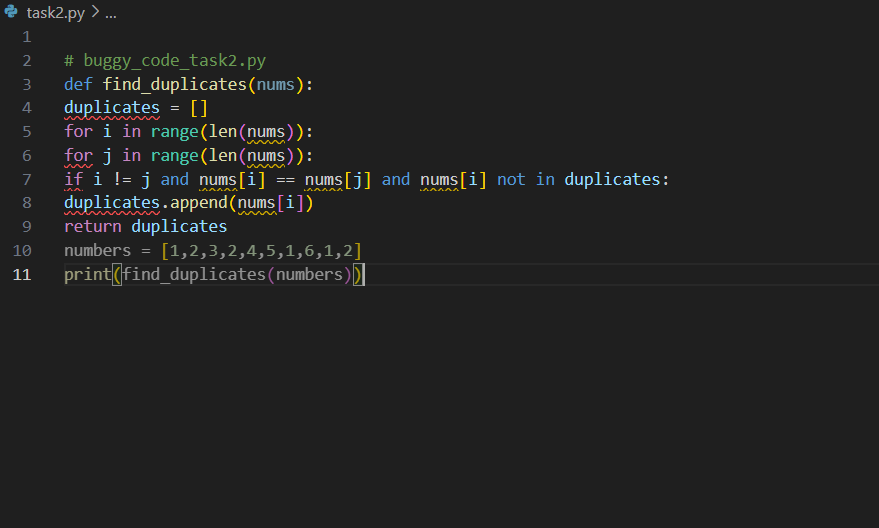
print(find\_duplicates(numbers))

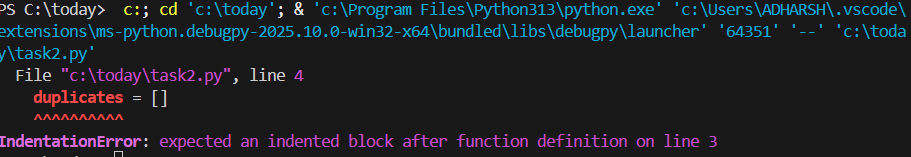
**Expected Output:**

More efficient duplicate detection (e.g., using sets).

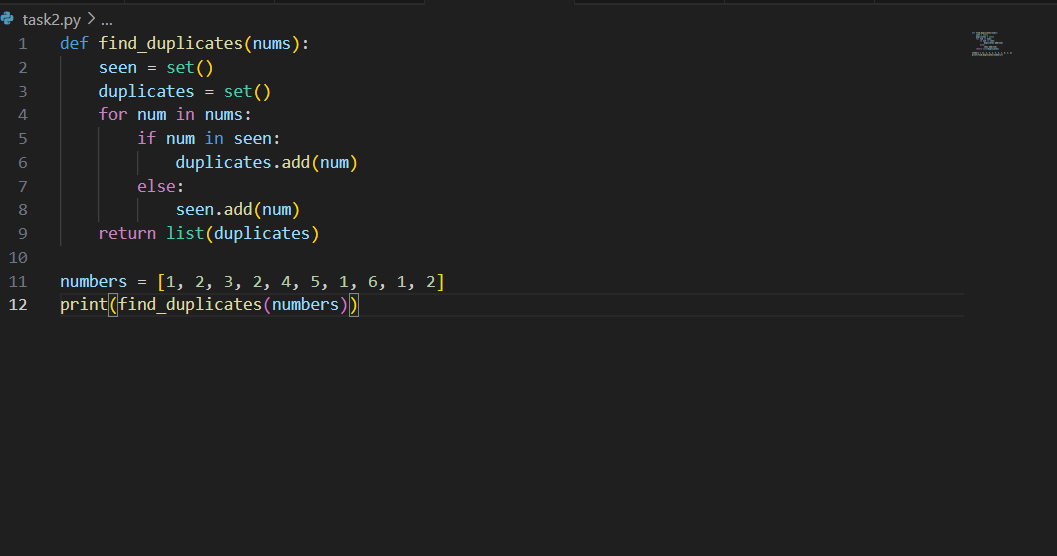
AI should explain the optimization.

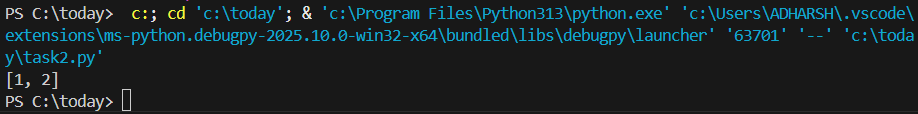
**Before detection:**





**After Detection:**

****

****

**Task 3: Code Refactoring for Readability**

Refactor messy code into clean, PEP 8–compliant, well-structured code.

# buggy\_code\_task3.py

def c(n):

x=1

for i in range(1,n+1):

x=x\*i

return x

print(c(5))

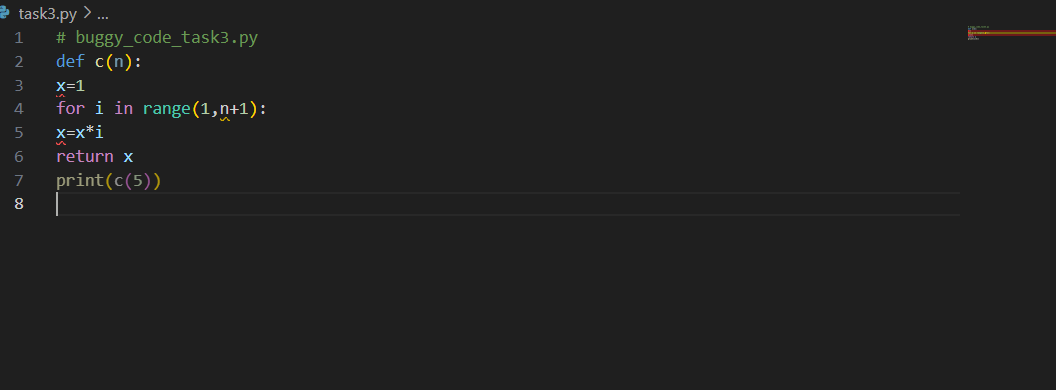
**Expected Output:**

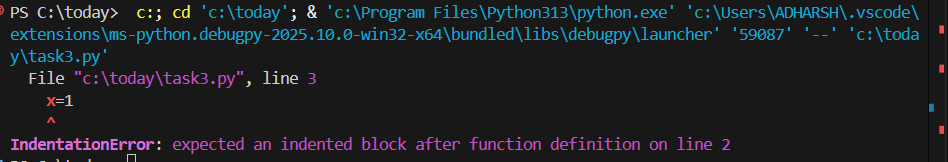
Function renamed to calculate\_factorial.

Proper indentation, variable naming, docstrings, and formatting.

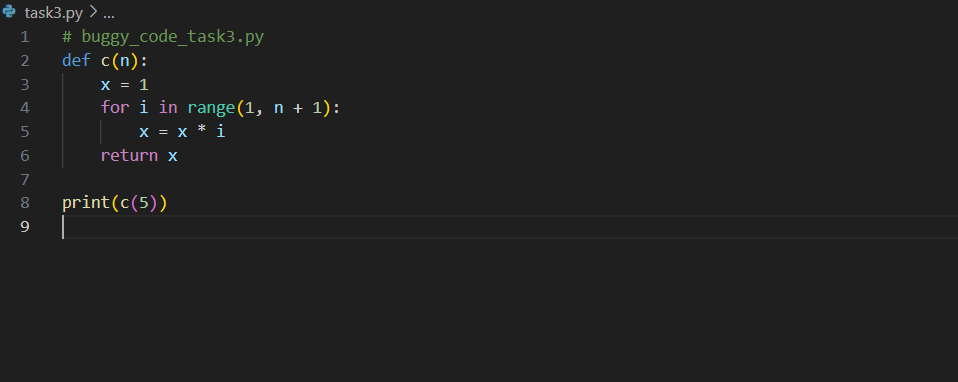
AI should provide a more readable version.

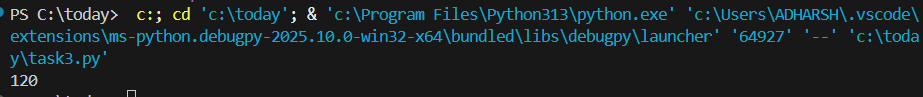
**Before Detection:**

****

****

**After Detection:**

****

****

**Task 4: Security and Error Handling Enhancement**

Add security practices and exception handling to the code.

# buggy\_code\_task4.py

import sqlite3

def get\_user\_data(user\_id):

conn = sqlite3.connect("users.db")

cursor = conn.cursor()

query = f"SELECT \* FROM users WHERE id = {user\_id};" # Potential SQL injection risk

cursor.execute(query)

result = cursor.fetchall()

conn.close()

return result

user\_input = input("Enter user ID: ")

print(get\_user\_data(user\_input))

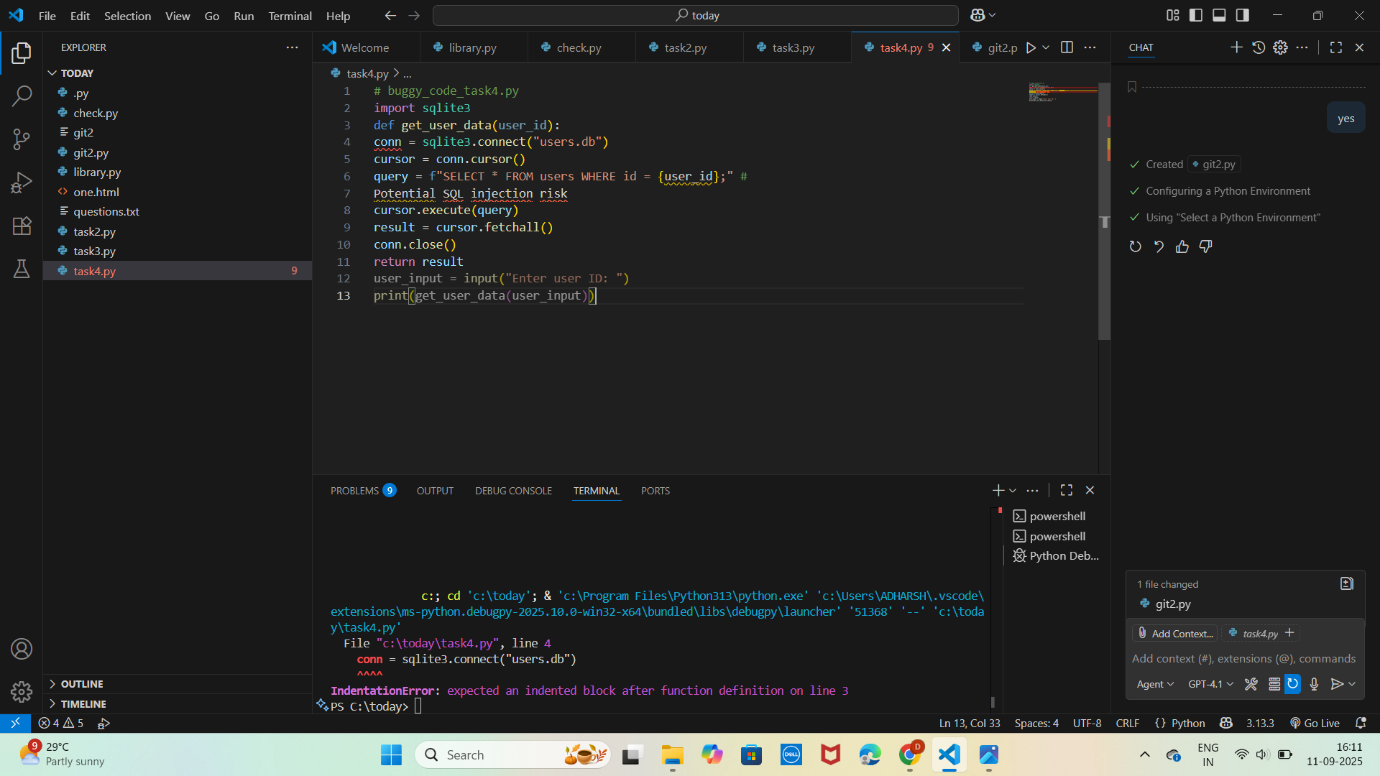
**Expected Output:**

Safe query using parameterized SQL (? placeholders).

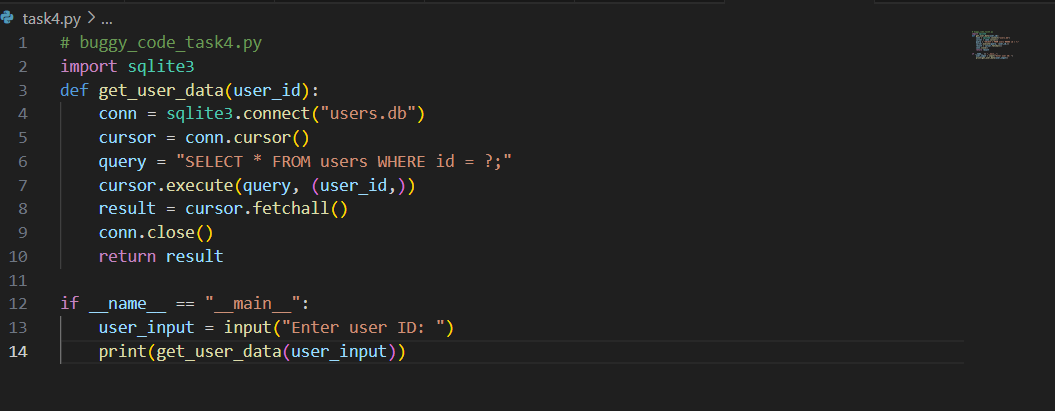
Try-except block for database errors.

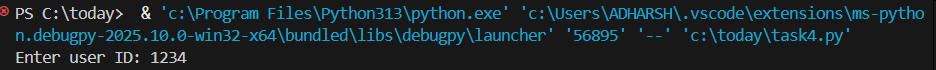
Input validation before query execution.

**Before Detection:**

****

**After Detection:**

****

****

**Task 5: Automated Code Review Report Generation**

Generate a review report for this messy code.

# buggy\_code\_task5.py

def calc(x,y,z):

if z=="add":

return x+y

elif z=="sub": return x-y

elif z=="mul":

return x\*y

elif z=="div":

return x/y

else: print("wrong")

print(calc(10,5,"add"))

print(calc(10,0,"div"))

**Expected Output:**

AI-generated review report should mention:

Missing docstrings

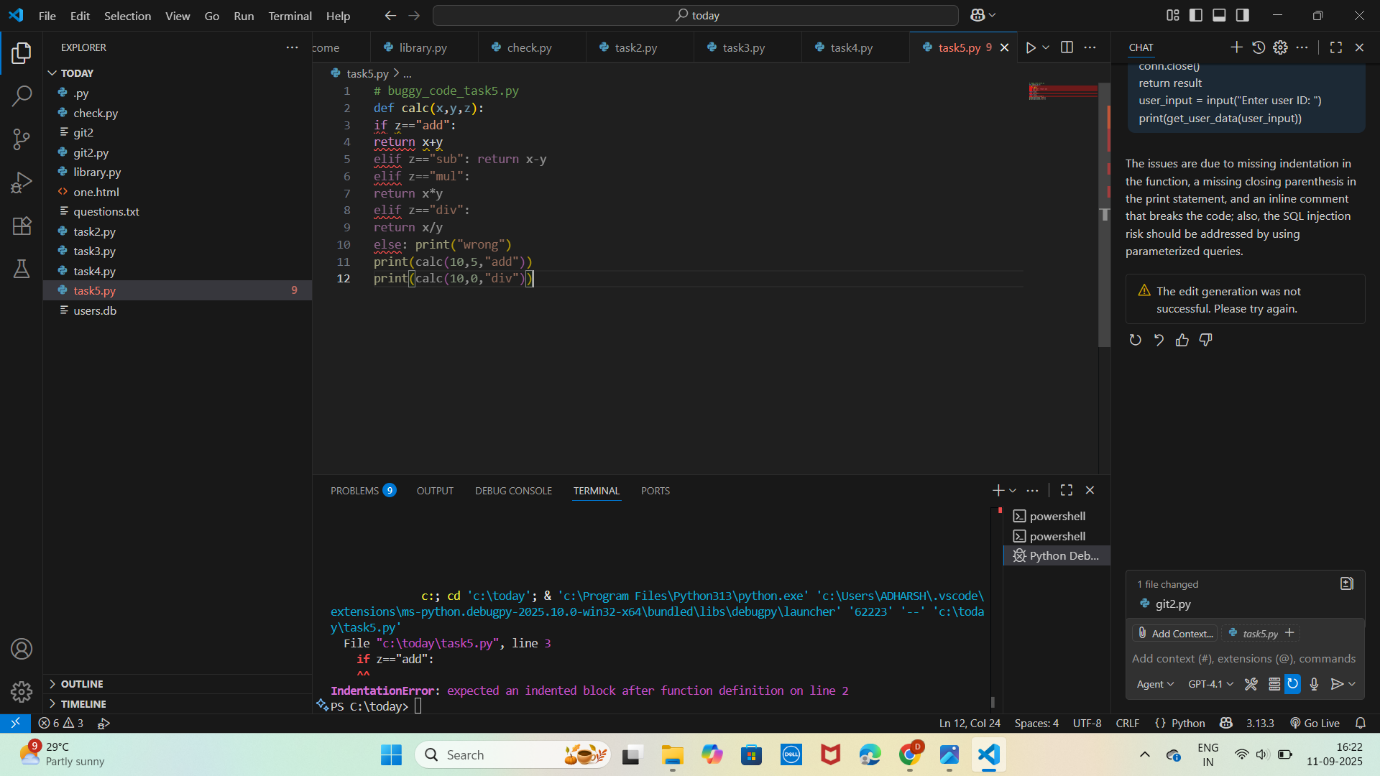
Inconsistent formatting (indentation, inline return)

Missing error handling for division by zero

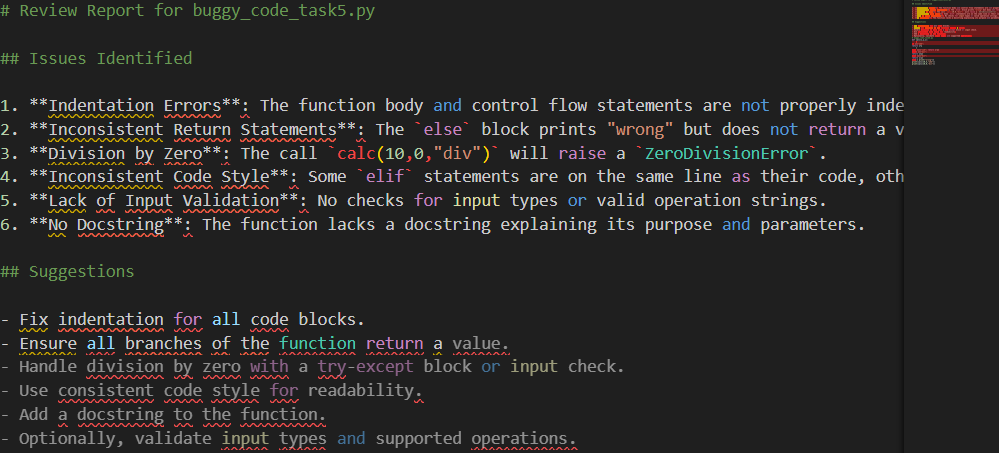
Non-descriptive function/variable names

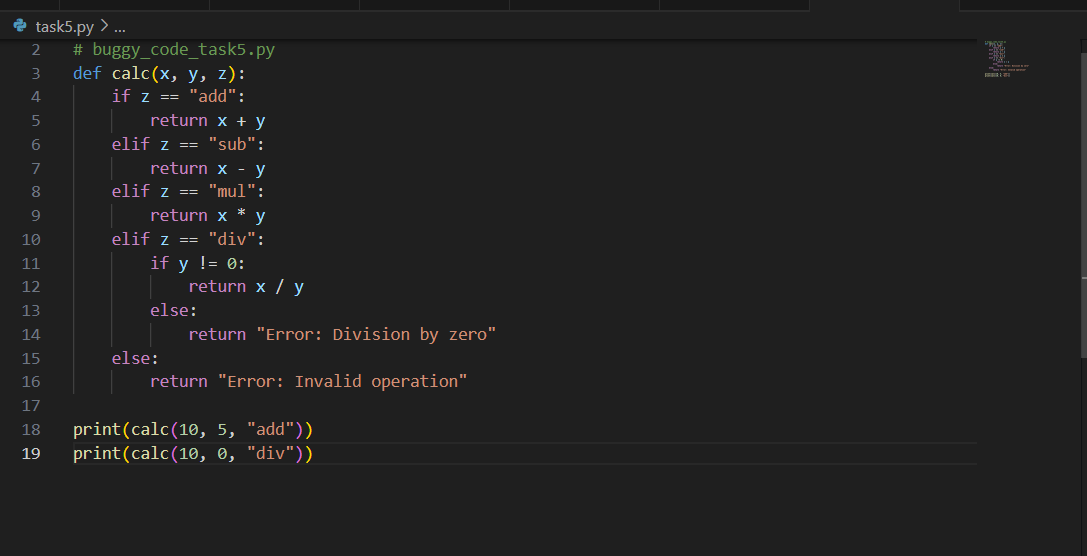
Suggestions for readability and PEP 8 compliance

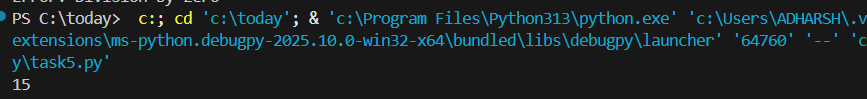
**Before Detection:**

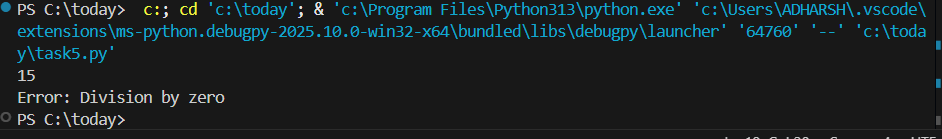


After Detection:









**D.ANKITHA**

**2503A51L09**