AI ASSISTED CODING

LAB-10.4

NAME:P.LEKHANAND

ENROLL.NO:2403A52044

BATCH:0

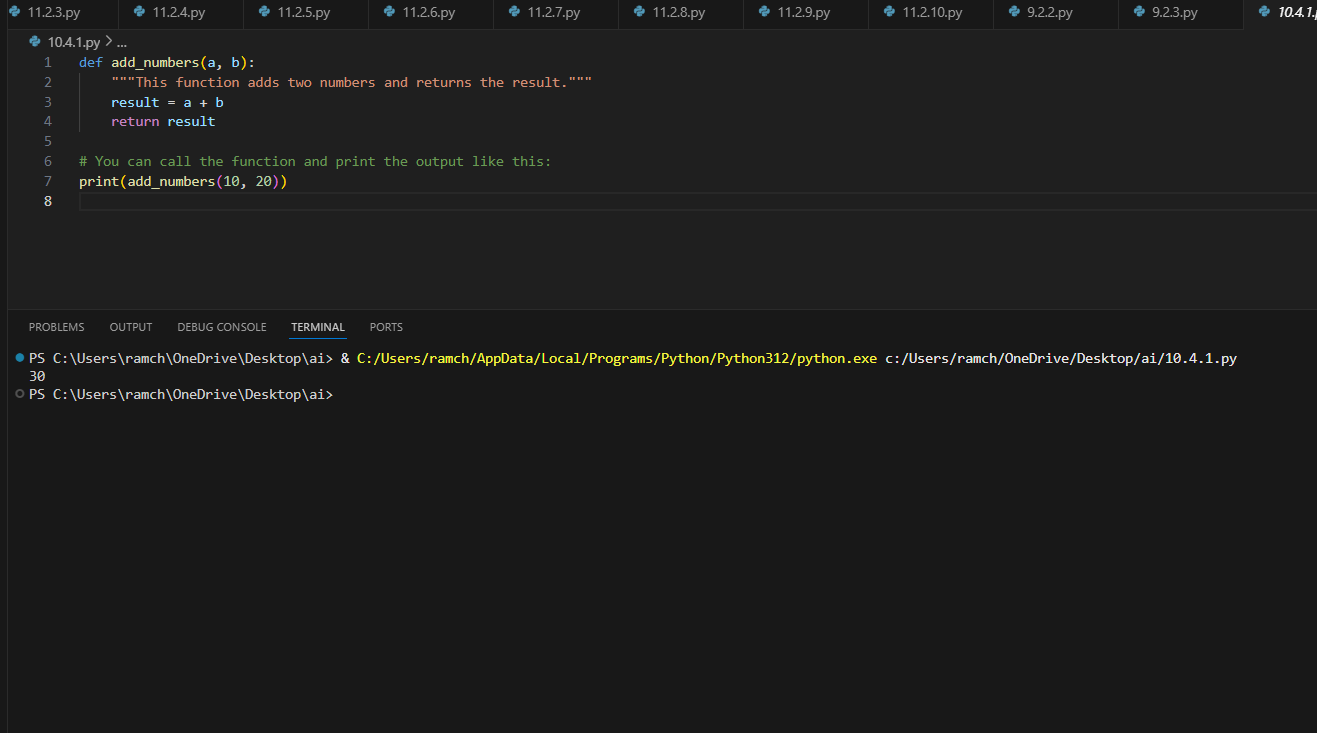
**TASK-01:**

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))

**PROMPT:**

Identify the errors in the given code and give the corrected code.

**CODE & OUTPUT:**

****

**OBSERVATION:**

The ai corrected all the errors in the given code and make the code much more efficient to understand and it also gave the identation to the code.

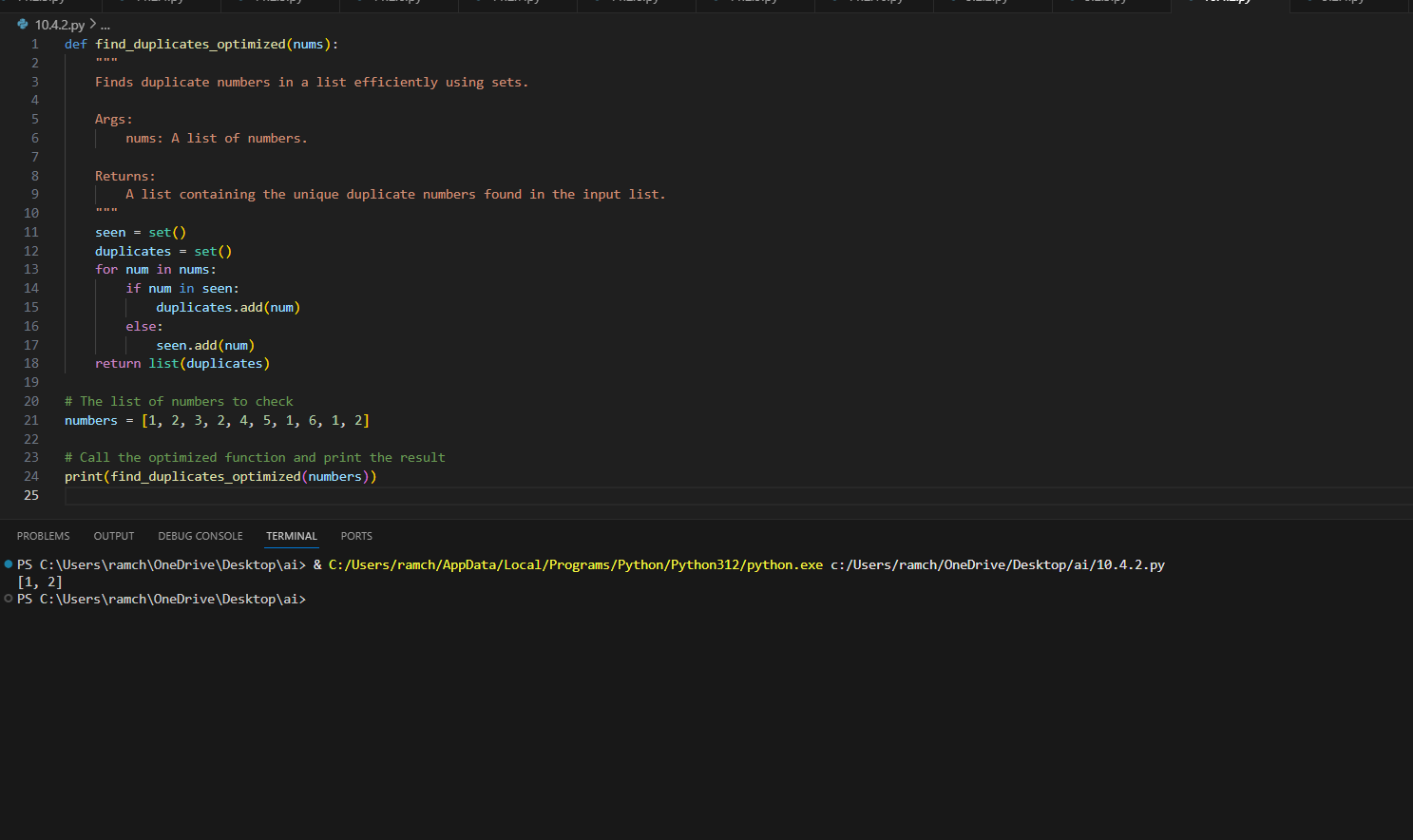
**TASK-02:**

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))

**PROMPT:**

Correct the given code with the proper indentation and give the corrected code.

**CODE & OUTPUT:**



**OBSERVATION:**

The ai edited the code with proper indentation and also corrected all the errors and the code is finding the duplicate numbers efficiently.

**TASK-03:**

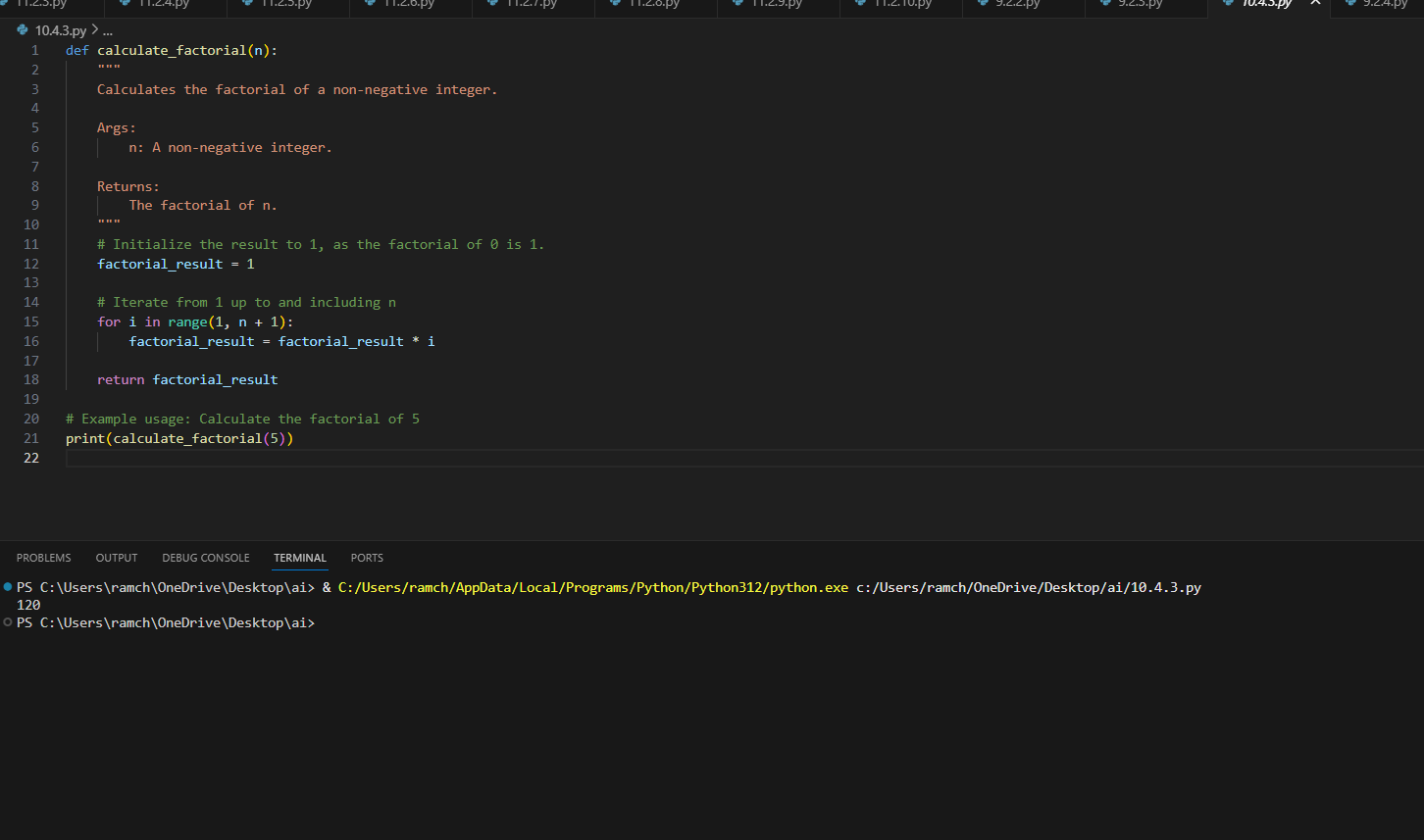
Refactor messy code into clean, PEP 8–compliant, well-  
structured code.  
# buggy\_code\_task3.py

def c(n):  
x=1  
for i in range(1,n+1):  
x=x\*i  
return x  
print(c(5))

**PROMPT:**

rename the function name in the given code and correct code with correcting all the errors and also give the proper indentation.

**CODE & OUTPUT:**

****

**OBSERVATION:**

The code is corrected by the ai according to the prompt given by me in which I have mentioned that update the function name and correct the errors and give the proper indentation.

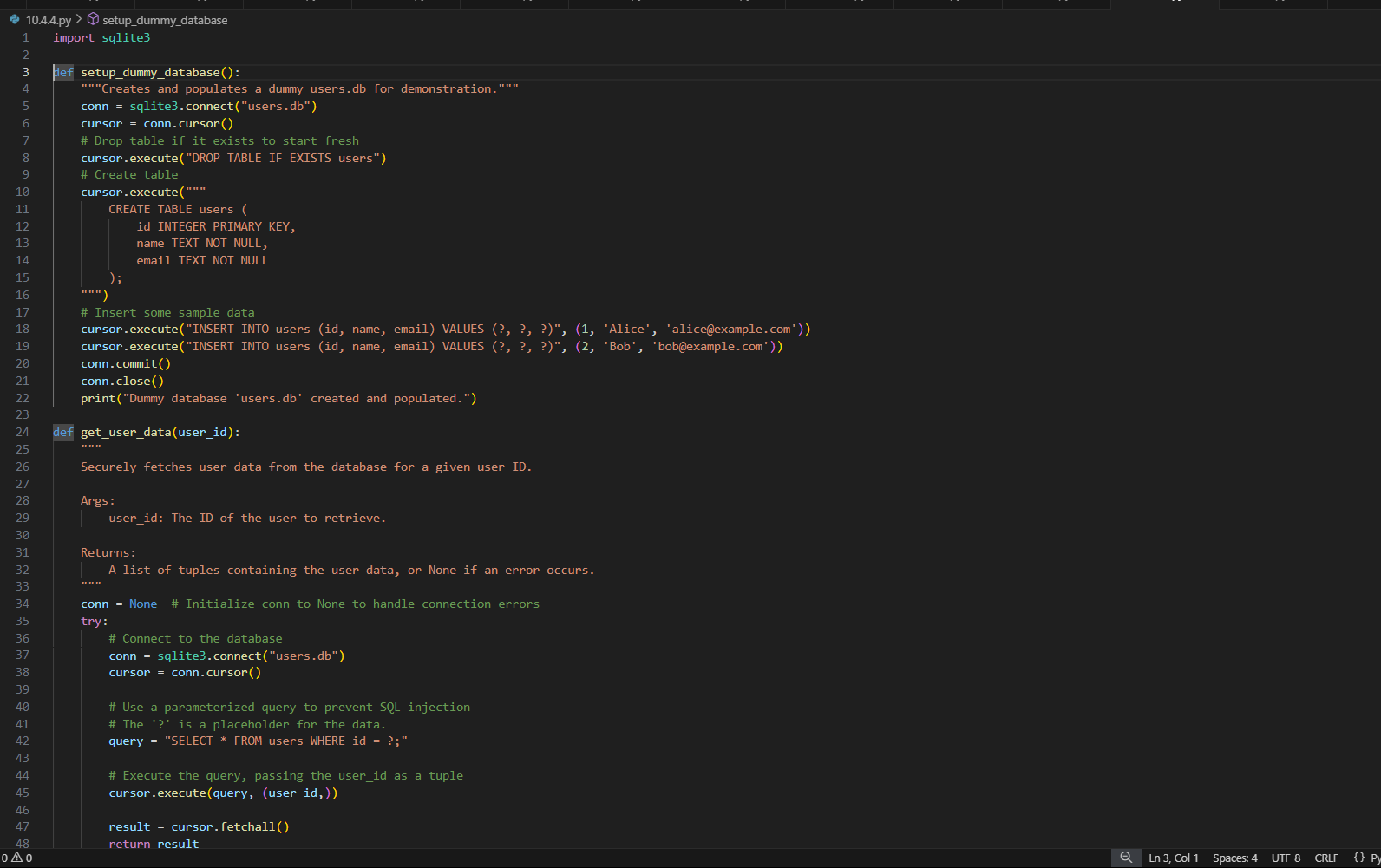
**TASK-04:**

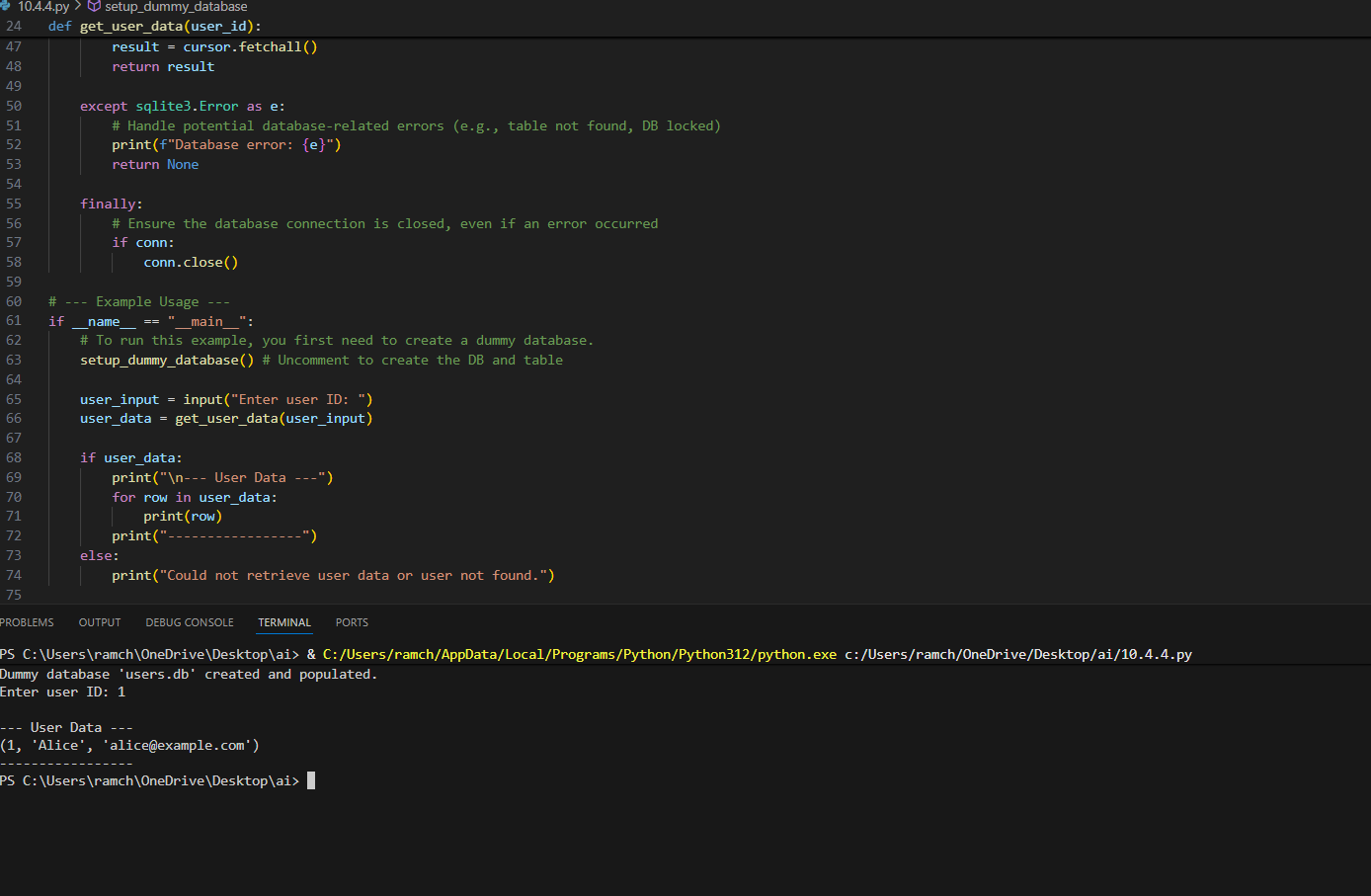
Add security practices and exception handling to the code.  
# buggy\_code\_task4.py  
import sqlite3  
def get\_user\_data(user\_id):  
conn = sqlite3.connect("users.db")  
cursor = conn.cursor()  
query = f"SELECT \* FROM users WHERE id = {user\_id};" #  
Potential SQL injection risk  
cursor.execute(query)  
result = cursor.fetchall()  
conn.close()  
return result  
user\_input = input("Enter user ID: ")  
print(get\_user\_data(user\_input)

**PROMPT:**

Correct the code by applying proper indentations and identify all the errors and correct them give the corrected code.

**CODE & OUTPUT:**

****

****

**OBSERVATION:**

The code corrects the code by removing all the errors and applies the exception handling to the code to make it more efficient and accurate.

**TASK-05:**

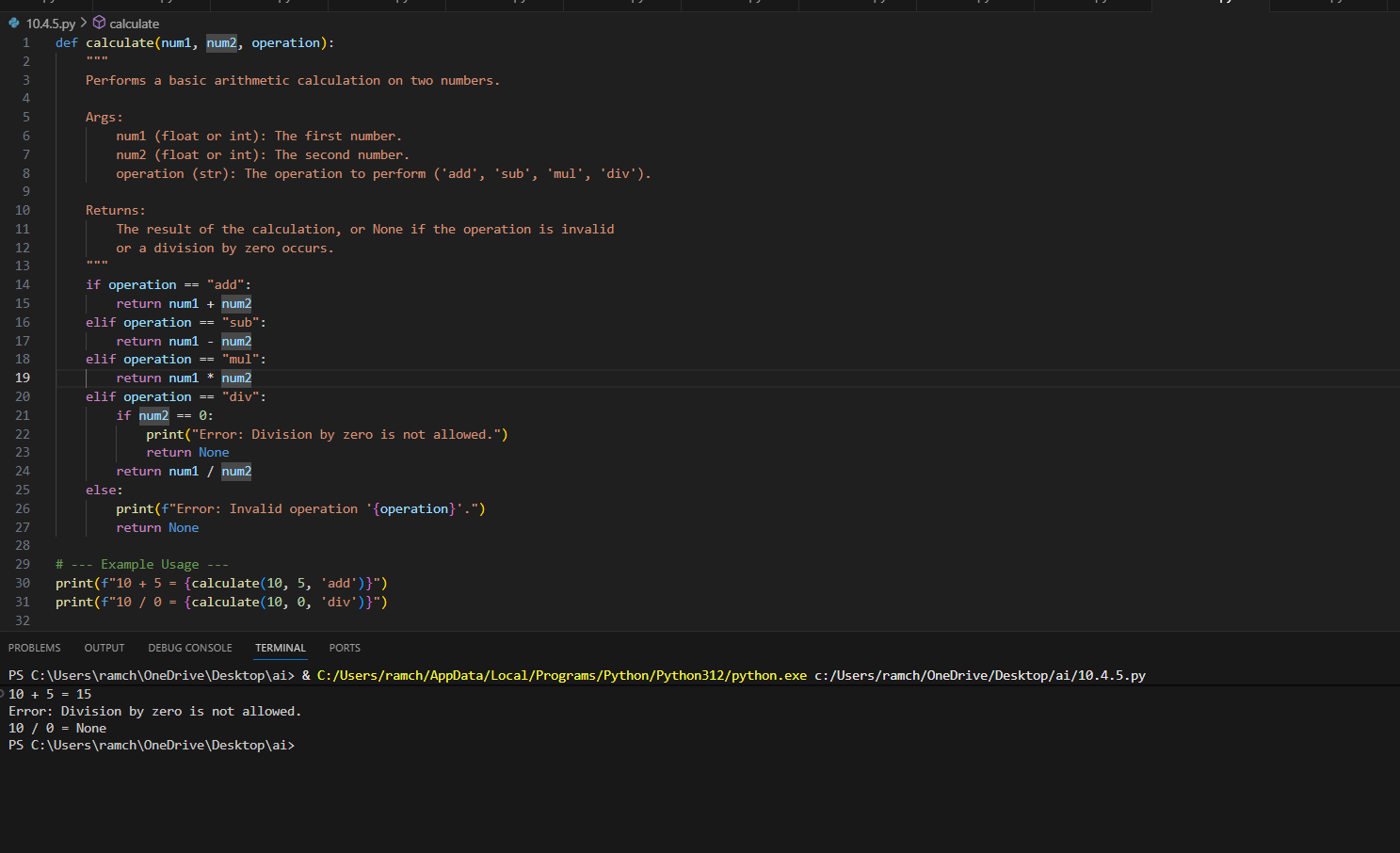
Generate a review report for this messy code.  
# buggy\_code\_task5.py

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"))

**PROMPT:**

Correct the code and give the report explaining the code.

**CODE & OUTPUT:**

****

**EXPLANATION:**

AI-Generated Code Review Report

The provided Python code snippet for a simple calculator has several areas that can be improved to meet standard coding practices. Below is a detailed analysis based on your criteria.

1. Missing Docstrings

The function calc lacks a docstring. Docstrings are crucial for explaining what a function does, its parameters, and what it returns. This makes the code easier for others (and your future self) to understand and use.

2. Inconsistent Formatting and PEP 8 Compliance

The code does not follow standard Python formatting guidelines (PEP 8), which harms readability.

Indentation: The bodies of the if, elif, and else statements are not correctly indented. Python relies on indentation to define code blocks.

Inline return: The line elif z=="sub": return x-y has the return statement on the same line as the condition. While syntactically allowed, it's inconsistent with the other branches and generally discouraged for readability.

3. Missing Error Handling

A critical bug exists in the code. The test case print(calc(10, 0, "div")) will cause the program to crash with a ZeroDivisionError because it attempts to divide by zero. The function should handle this case gracefully.

4. Non-Descriptive Function and Variable Names

The names used (calc, x, y, z) are very generic. Using more descriptive names like calculate, num1, num2, and operation would make the function's purpose immediately clear without needing to read the implementation

Summary of Improvements:

Clarity: Function and variable names are now descriptive.

Documentation: A comprehensive docstring has been added.

Robustness: It now includes a check to prevent ZeroDivisionError and handles invalid operations gracefully.

Readability: The code is formatted consistently according to PEP 8 guidelines.

**OBSERVATION:**

AI generated the accurate report of the code and it also corrected the code according to the prompt in an efficient way.