Name:K.Rajavardhan reddy

2403A51108

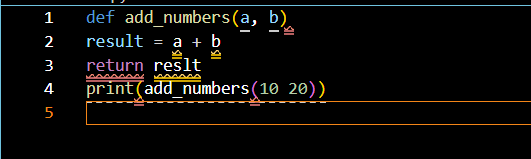
Lab-10.3

Task 1: Syntax and Error Detection  
Task: 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))

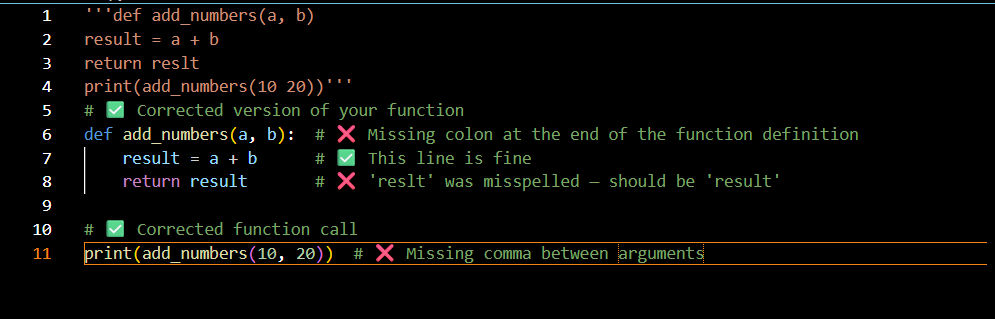
Prompt:

Give the above code and ask ai to give fixed code with comments on where its gone wrong.

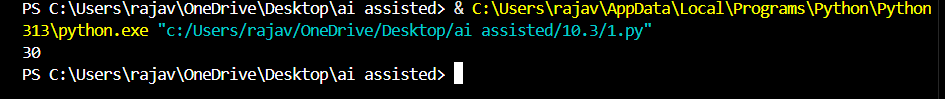
Errorcode:



Fixedcode:



Output:



Explanation:

The original function contains three key issues:

• Syntax Error:

def add\_numbers(a, b) is missing a colon (:) at the end of the function definition.

• Typo in Variable Name:

return reslt should be return result — the variable name is misspelled.

• Incorrect Function Call:

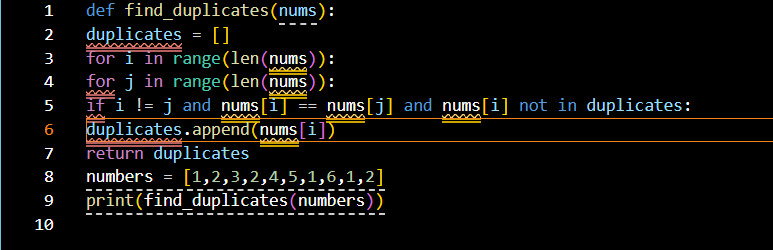
print(add\_numbers(10 20)) is missing a comma between arguments — should be print(add\_numbers(10, 20))

Task 2: Logical and Performance Issue Review  
Task: 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)).

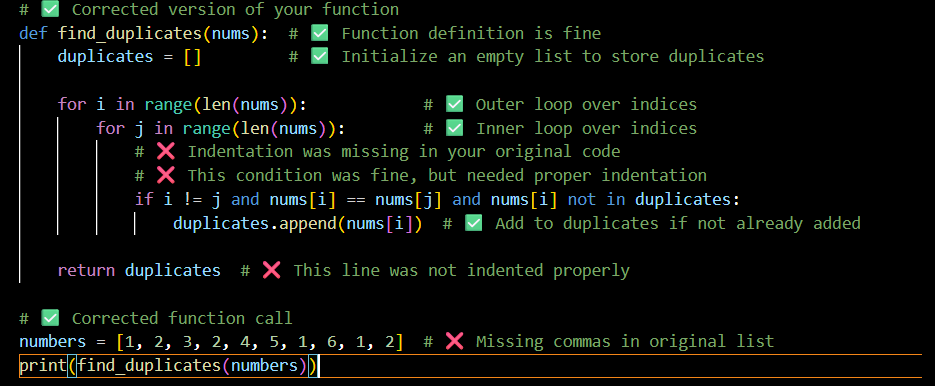
Prompt:

Give the above code and ask ai to give fixed code with comments on where its gone wrong

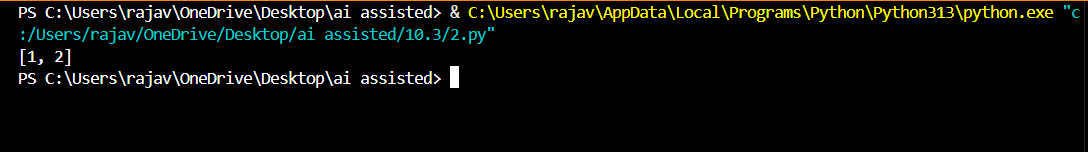
Errorcode:



Fixedcode:



Output:



Explanation:

The original function contains three key issues:

* Syntax Error:  
  def add\_numbers(a, b) is missing a colon (:) at the end of the function definition.
* Typo in Variable Name:  
  return reslt should be return result — the variable name is misspelled.
* Incorrect Function Call:  
  print(add\_numbers(10 20)) is missing a comma between arguments — should be print(add\_numbers(10, 20)).

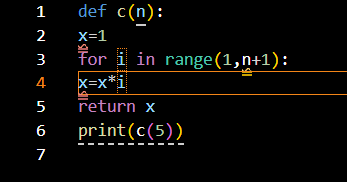
Task 3: Code Refactoring for Readability  
Task: Refactor messy code into clean, PEP 8–compliant, well-  
structured code.  
# buggy\_code\_task3.py

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

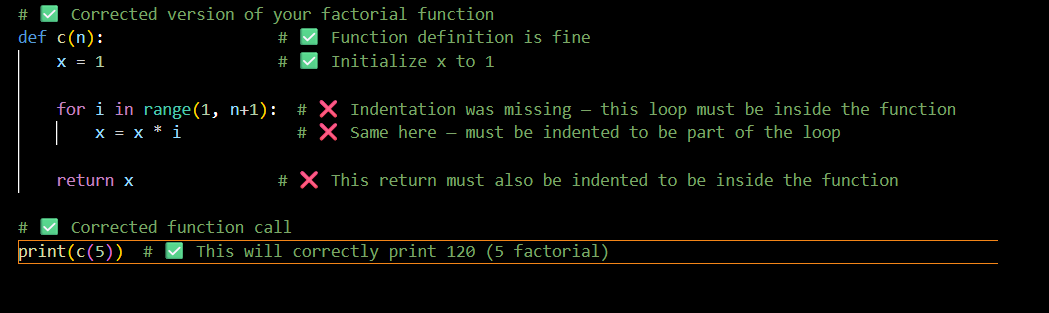
Prompt:

Give the above code and ask ai to give fixed code with comments on where its gone wrong

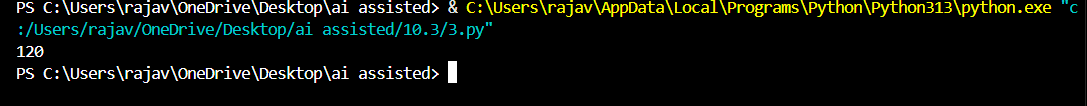
Errorcode:



Fixedcode:



Output:



Explanation:

**The original function contains three key issues:**

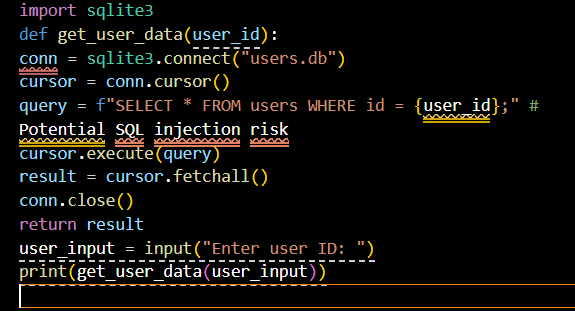
* **Indentation Error:  
  The for loop was not properly indented — it must be inside the function block.**
* **Misplaced Calculation:  
  The line x = x \* i was outside the loop due to missing indentation, so the multiplication wouldn't repeat correctly.**
* **Misaligned Return Statement:  
  return x was not indented properly — it must be inside the function to return the result.**

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

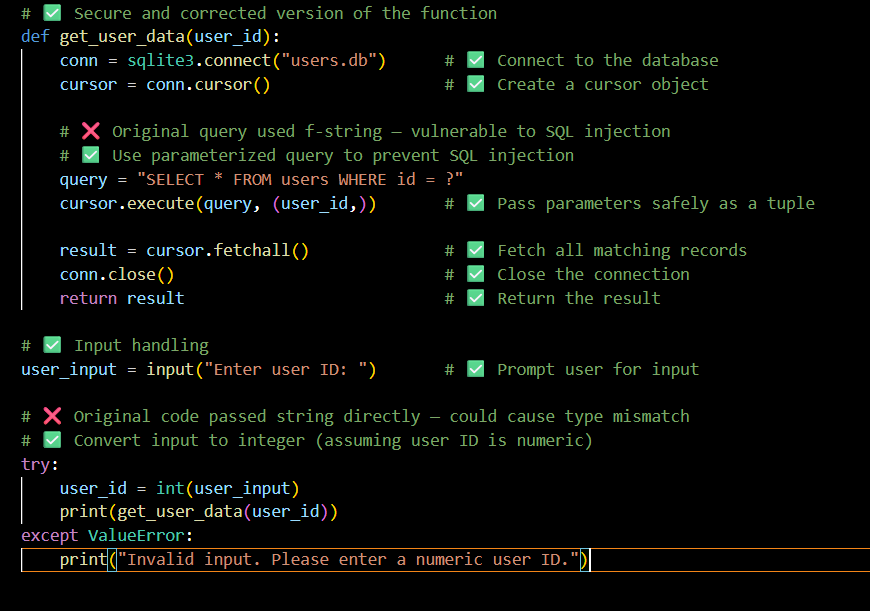
Prompt:

Give the above code and ask ai to give fixed code with comments on where its gone wrong

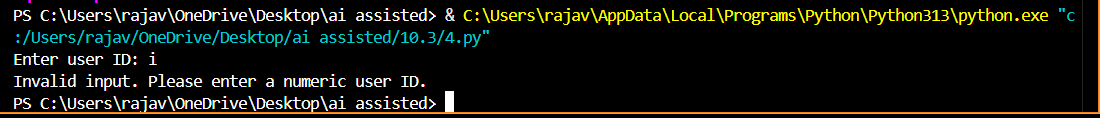
Errorcode:



Fixedcode:



Output:



Explanation:

**🔍 Breakdown of Issues:**

* **SQL Injection Risk**: Using f"{user\_id}" directly in the query allows malicious input. We fixed this using ? placeholders and parameterized queries.
* **Type Safety**: input() returns a string. If user\_id is expected to be an integer, we should convert it and handle errors gracefully.
* **Code Structure**: Everything is now properly indented and secure.

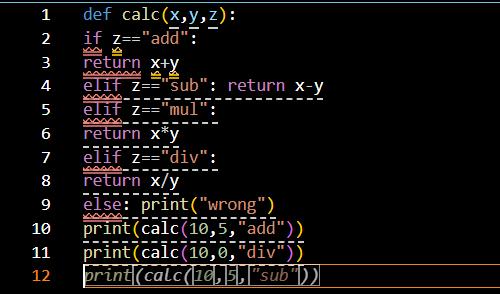
Task 5: Automated Code Review Report Generation  
Task: Generate a review report for this messy code.  
# buggy\_code\_task5.py

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"))

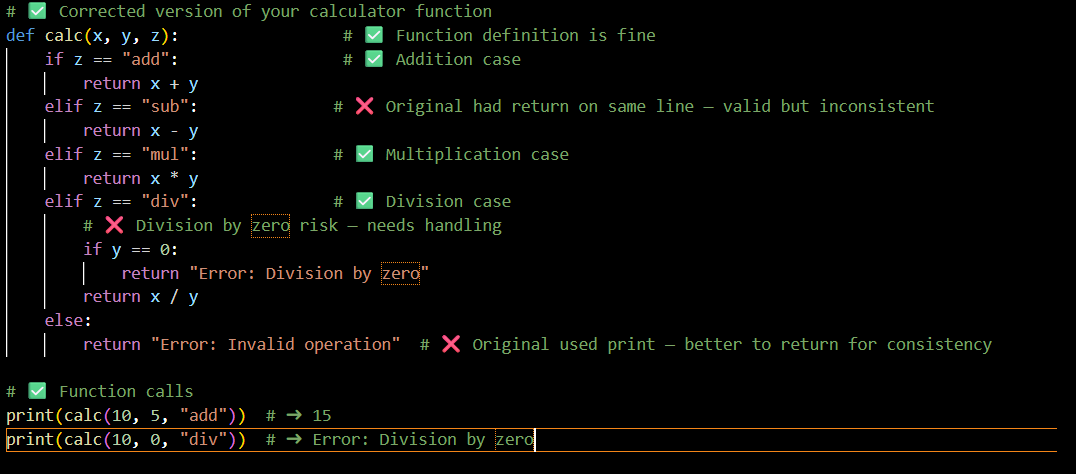
prompt:

Give the above code and ask ai to give fixed code with comments on where its gone wrong

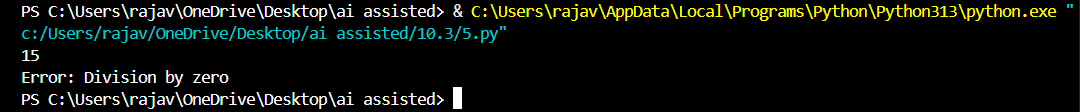
Errorcode:



Fixedcode:



Output:



Explantion:

**🔍 Breakdown of Issues:**

* **Inconsistent return placement**: Mixing inline and block-style elif returns can reduce readability.
* **Division by Zero**: You didn’t handle y == 0, which would crash the program. Now it returns a clear error.
* **Error Handling**: Instead of printing "wrong", it's better to return an error string so the caller can decide what to do with it.