AI-ASSITED

ASSIGNMENT-10.4

ROLLNO:2403A52084

BATCH:04

TASK:01

PROMPT:

Identify and fix syntax, indentation, and variable errors in the following Python script:

def add\_numbers(a, b)

result = a + b

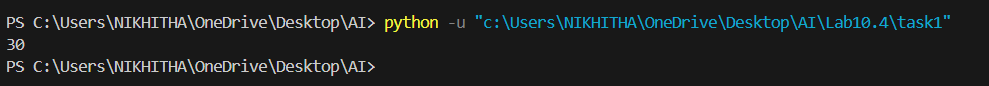
return reslt

print(add\_numbers(10 20)).

CORRECTED CODE:

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

OBSEVATOIN: MISSING SEMICLONE,INDENTATOIN ERROR,MISSING COMA,BETWEEN ARGUMENTS.

TASK02:

PROMPT:

Optimize inefficient logic while keeping the result correct.

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]

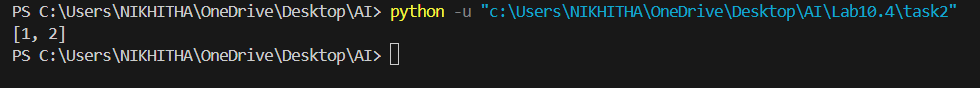
print(find\_duplicates(numbers))

CODE:

A screen shot of a computer code

AI-generated content may be incorrect.

OUTPUT:



TASK03:

PROMPT:

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

def c(n):

x=1

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

x=x\*i

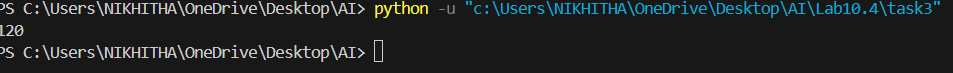
return x

print(c(5))

CODE:

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

OBSEVATOIN:

INDENTTATION ERROR,FUNCTOIN IS ALSO WRONG

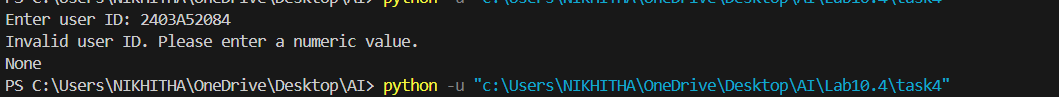
TASK04:

PROMPT**: 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.**

CODE:A screen shot of a computer program

AI-generated content may be incorrect.

OUTPUT:



OBSEVATOIN:

This code securely fetches user data from a SQLite database using a user ID.

It uses a parameterized query (?) to prevent SQL injection attacks.

Exception handling ensures that database errors are caught and reported, and the connection is always closed.

The user is prompted for an ID, and matching records are printed.

TASK05:

PROMPT:

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

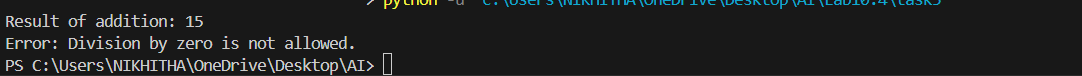
def calc(x,y,z):  
if z=="add":  
return x+y  
elif z=="sub": return x-y  
elif z=="mul":  
return x\*y  
elif z=="div":  
return x/y  
else: print("wrong")  
print(calc(10,5,"add"))  
print(calc(10,0,"div"))  
Expected Output:  
AI-generated review report should mention:  
o Missing docstrings  
o Inconsistent formatting (indentation, inline return)  
o Missing error handling for division by zero  
o Non-descriptive function/variable names  
o Suggestions for readability and PEP 8 compliance.

CODE:

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:



OBSERVATION:ERRORS: INDENTATION ERROR AND FORMATTING,ERROR HANDLING,FUNCTION RETURN CONSISTENCY,CODE STRUCTURE.

SUMMARY: The function calc(x, y, z) performs basic arithmetic operations based on the value of z. The code is functional but suffers from poor formatting, lack of error handling, and inconsistent structure.

RECOMMENDATION:

• Fix indentation and formatting to comply with PEP 8.

• Add exception handling for division by zero.

• Ensure the function always returns a value.

• Replace inline prints with proper error messages or exceptions.

• Consider using a dictionary to map operations for scalability.