HINO:2403A51286  
 Assignment:10.3

**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

**Prompt:**

You are given the following buggy Python script:

# buggy\_code\_task1.py

def add\_numbers(a, b)

result = a + b

return reslt

print(add\_numbers(10 20))

Tasks:

1. Identify and fix all syntax, indentation, and variable errors.
2. Provide the corrected code in a single cell (so it can run directly).
3. Explain what was fixed (e.g., missing colon, indentation issues, typos, missing comma).
4. Show the expected output when the corrected code is executed.

**Code:**



**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))  
Expected Output:  
• More efficient duplicate detection (e.g., using sets).  
• AI should explain the optimization.

**Prompt:**

You are given the following inefficient Python script:

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

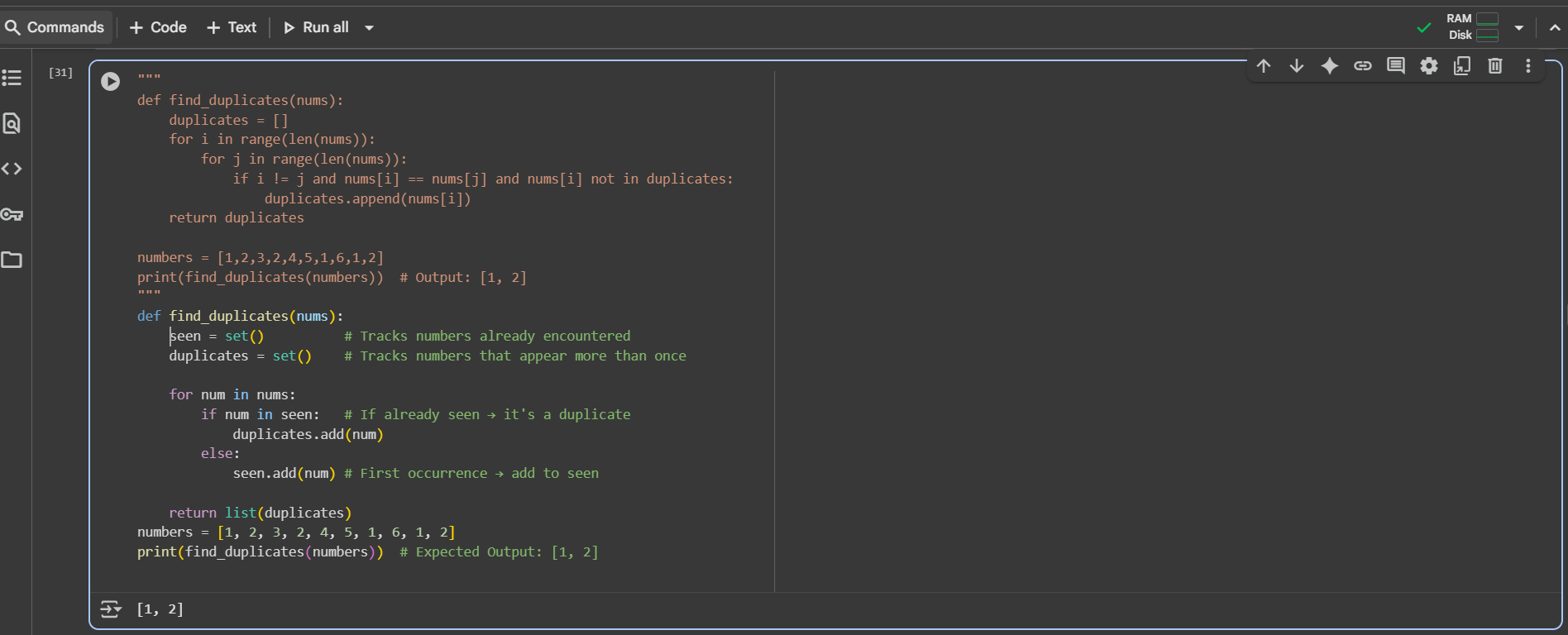
numbers = [1,2,3,2,4,5,1,6,1,2]

print(find\_duplicates(numbers))

**Tasks:**

1. Identify logical and performance issues in the code.
2. Rewrite the function with a more efficient approach (e.g., using sets).
3. Provide the corrected code in a single cell.
4. Explain how the optimization improves performance while keeping results correct.
5. Show the expected output.

**Code:**

****

**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))

Expected Output:

Function renamed to calculate\_factorial.

Proper indentation, variable naming, docstrings, and formatting.

AI should provide a more readable version.

**Prompt:**

You are given the following messy Python script:

def c(n):

x=1

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

x=x\*i

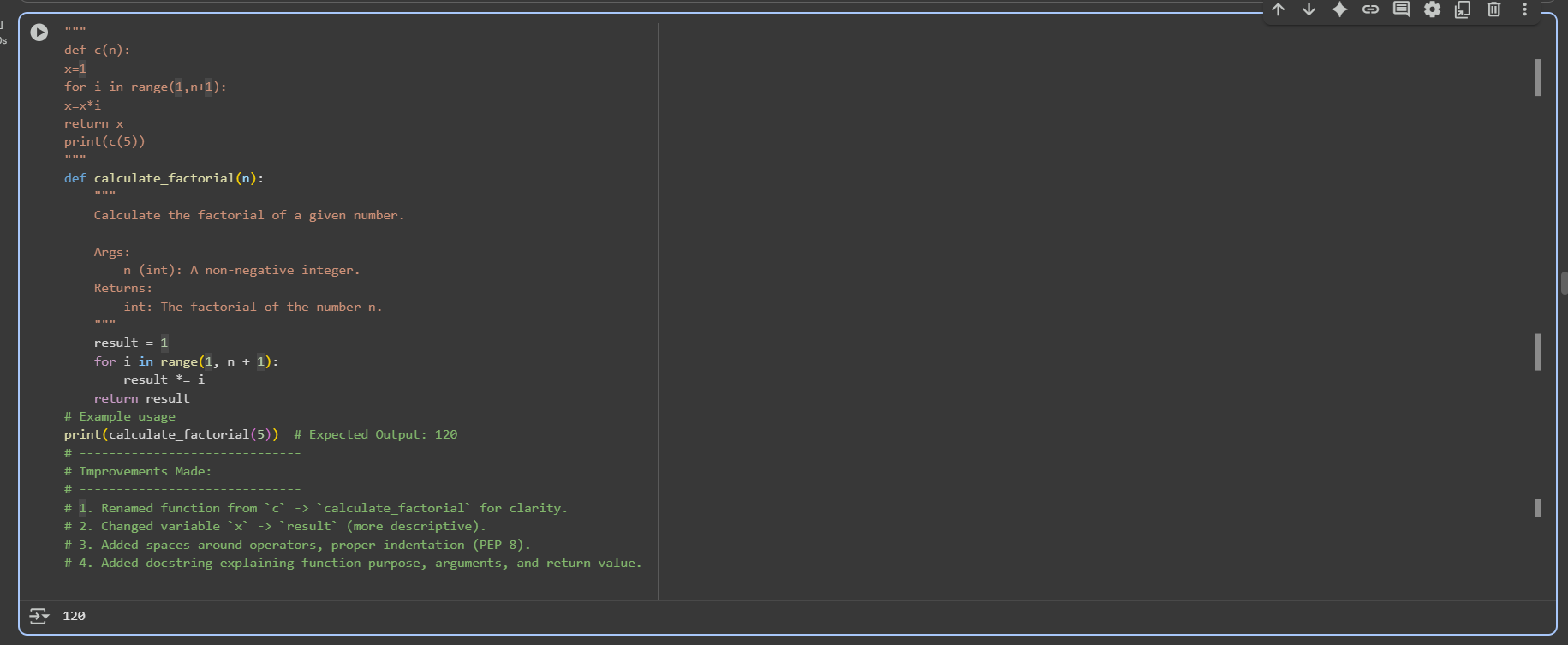
return x

print(c(5))

**Tasks:**

1. Refactor the code to follow **PEP 8 standards** (clean indentation, spacing, and formatting).
2. Rename the function to calculate\_factorial and use meaningful variable names.
3. Add a **docstring** that explains the function’s purpose, inputs, and return value.
4. Provide the cleaned code in a single cell.
5. Show the expected output of the function.

**Code:**



**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.

**Prompt:**

You are given the following insecure Python script:

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

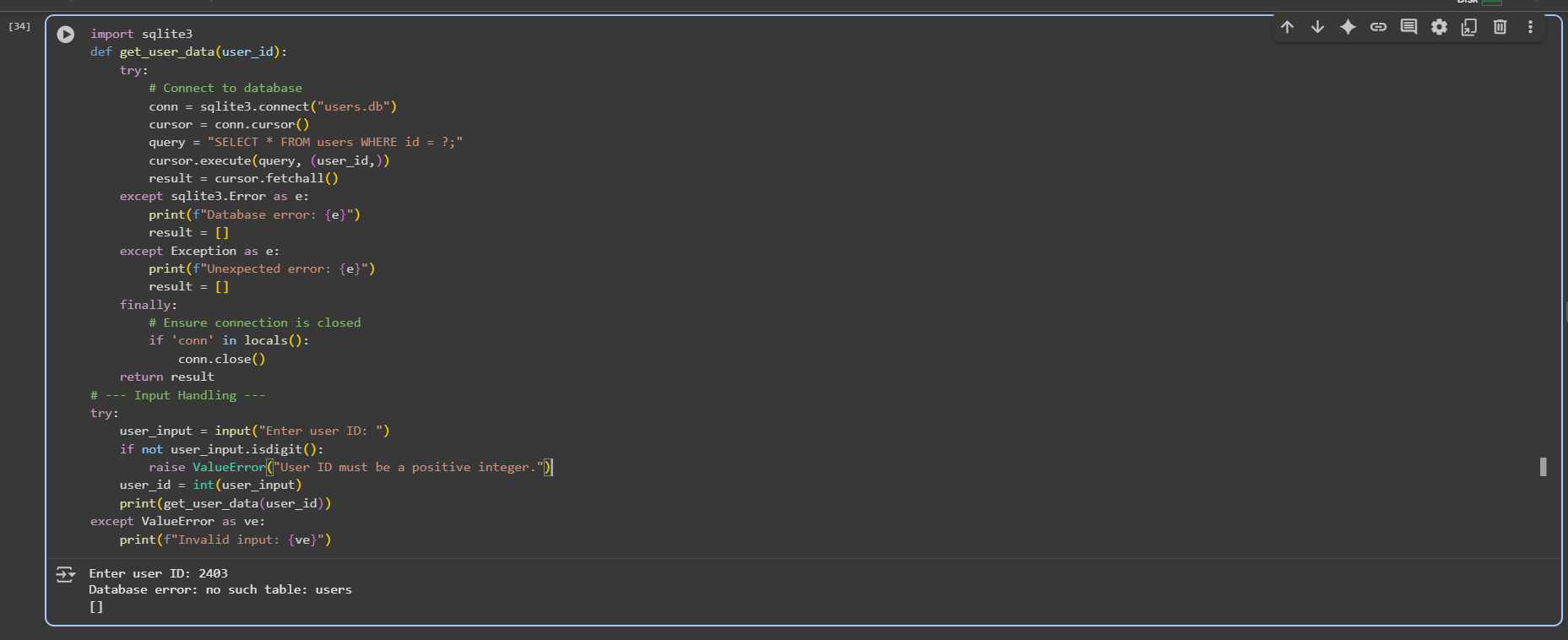
user\_input = input("Enter user ID: ")

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

**Tasks:**

1. Identify security risks (e.g., SQL injection).
2. Fix the code by using **parameterized SQL queries** with ? placeholders.
3. Add **exception handling** for database errors and unexpected errors.
4. Validate user input to ensure only integers are accepted as IDs.
5. Provide the corrected, secure code in a single cell.
6. Show example outputs for valid and invalid input.

**Code:**

****