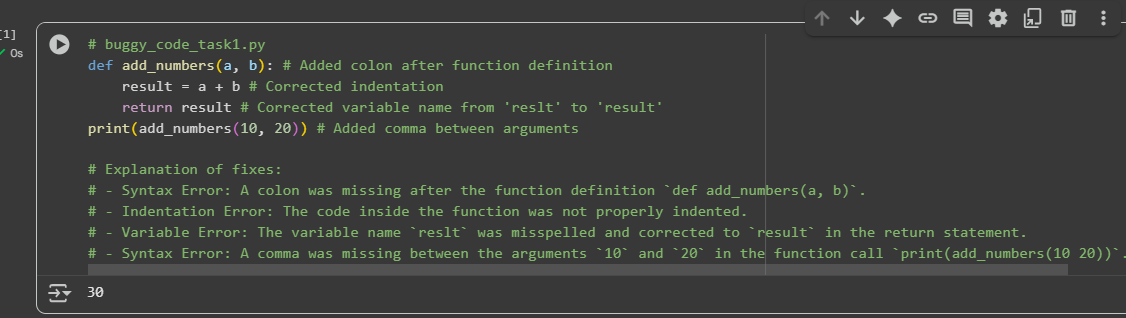
# AI ASSISTED CODING

## ASSIGNMENT-10

### TASK-1

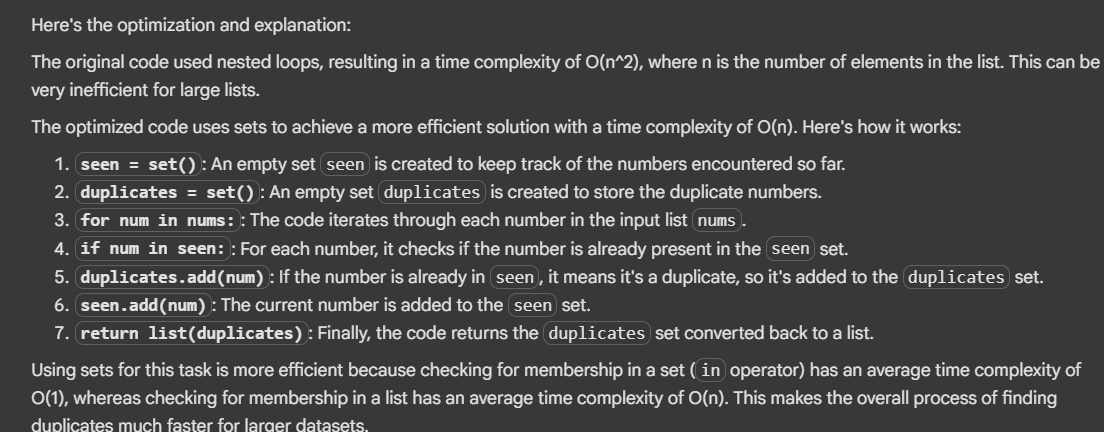
Task 1: Syntax and Error Detection  
Task: Identify and fix syntax, indentation, and variable errors in the  
given script.  
# buggy\_code\_task1.py  
def add\_numbers(a, b)  
result = a + b  
return reslt  
print(add\_numbers(10 20))  
Expected Output:  
• Corrected code with proper syntax (: after function, fixed variable  
name, corrected function call).  
• AI should explain what was fixed.



### Task-2

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

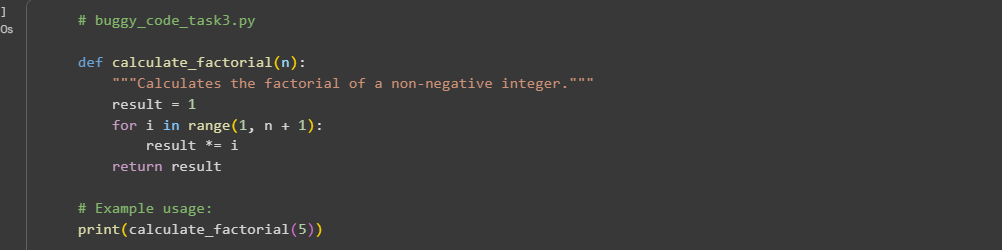


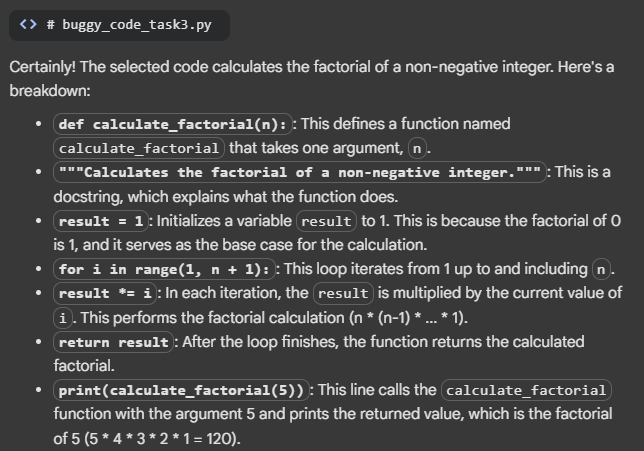


### TASK-3

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

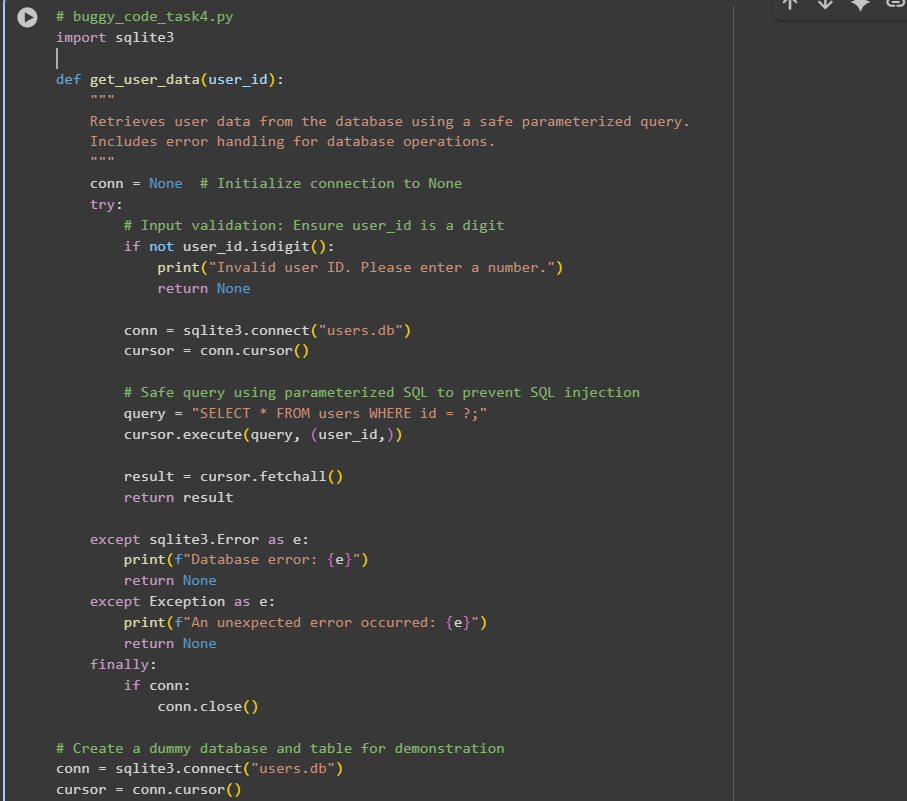
def c(n):  
x=1  
for i in range(1,n+1):  
x=x\*i  
return x  
print(c(5))  
Expected Output:  
Function renamed to calculate\_factorial.  
Proper indentation, variable naming, docstrings, and formatting.  
AI should provide a more readable version

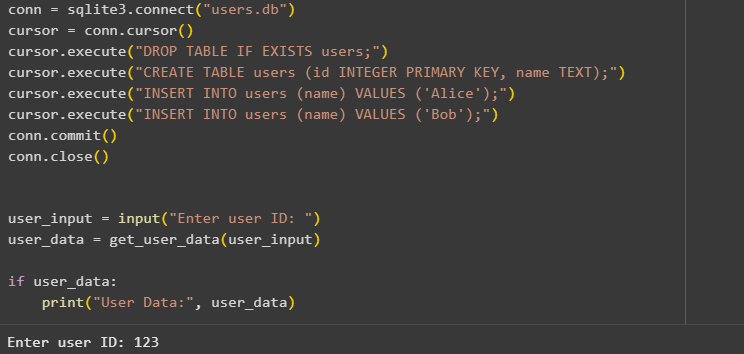


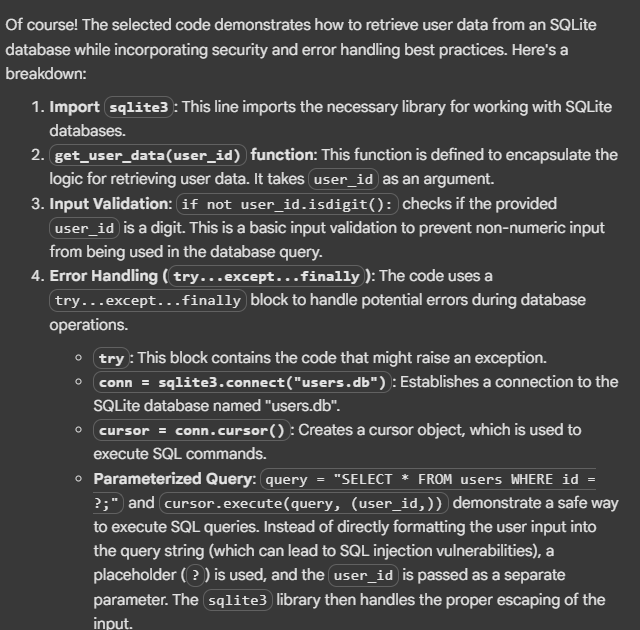


### Task-4

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))  
Expected Output:  
Safe query using parameterized SQL (? placeholders).  
Try-except block for database errors.  
Input validation before query execution









### Task-5

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"))  
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

