AI ASSISTED CODING
LAB-7.4

NAME: N.PRANAYKUMAR
ENROLL NO: 2403A52087
BATCH:04

TASK :01
Introduce a buggy Python function that calculates the factorial of a number using recursion. Use Copilot or Cursor AI to detect and fix the logical or syntax errors.

PROMPT:

lab7.4.1.py > ...

```python
def buggy_recursive_factorial(n):
    """
    This function attempts to calculate the factorial of a number using recursion,
    but it has a bug that causes infinite recursion.

    For any integer n > 0, it will cause a RecursionError.

    Args:
        n: A non-negative integer.

    Returns:
        The calculated factorial of n.
    """
    if not isinstance(n, int) or n < 0:
        raise ValueError("Input must be a non-negative integer")

    # The base case is correct.
    if n == 0:
        return 1

    # The bug is in the recursive step below.
    # It calls itself with the same number 'n' instead of 'n - 1',
    # so it never reaches the base case.
    return n * buggy_recursive_factorial(n)

# Example of the bug in action:
# try:
#     # This will raise a RecursionError
#     print(f"buggy_recursive_factorial(3) = {buggy_recursive_factorial(3)}")
# except RecursionError as e:
#     print(f"Calling buggy_recursive_factorial(3) caused an error: {e}")
#
# Expected output for factorial(3): 6
# Actual output: RecursionError: maximum recursion depth exceeded in comparison
buggy_recursive_factorial(3)

```

Detect the bugs in the code and correct it .

CODE:

```python
def recursive_factorial(n):
    """
    This function calculates the factorial of a number using recursion.

    Args:
        n: A non-negative integer.

    Returns:
        The calculated factorial of n.
    """
    if not isinstance(n, int) or n < 0:
        raise ValueError("Input must be a non-negative integer")

    # Base case: factorial of 0 is 1.
    if n == 0:
        return 1

    # Recursive step: n * factorial(n-1)
    return n * recursive_factorial(n - 1)

# Example of the corrected function in action:
try:
    result = recursive_factorial(int(input("Enter a non-negative integer: ")))
    print(f"recursive_factorial(5) = {result}")
except ValueError as e:
    print(f"Calling recursive_factorial(3) caused an error: {e}")
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/python.exe c:/Users/ramch/OneDrive/Desktop/ai/
lab7.4.1.py
Enter a non-negative integer: 5
recursive_factorial(5) = 120
PS C:\Users\ramch\OneDrive\Desktop\ai>
```

OBSERVATION:

The given prompt code has bugs which are resolved by the AI and rewrite the code.

TASK :02

Provide a list sorting function that fails due to a type error (e.g., sorting list with mixed integers and strings). Prompt AI to detect the issue and fix the code for consistent sorting.

PROMPT:

lab7.4.2.py > buggy_type_error_sort

```python
def buggy_type_error_sort(items):
    """
    Attempts to sort a list using a bubble sort algorithm, but has a bug.

    This function will raise a TypeError if the list contains non-numeric
    types (like strings), as it incorrectly tries to perform an arithmetic
    operation on the elements before comparing them.

    Args:
        items: A list of items to sort.

    Returns:
        A new list with the items sorted.
    """
    # Create a copy to avoid modifying the original list in place
    sorted_items = list(items)
    n = len(sorted_items)

    for i in range(n):
        swapped = False
        for j in range(0, n - i - 1):
            # THE BUG: This comparison assumes elements are numeric.
            # Adding an integer (0) to a string will raise a TypeError.
            if sorted_items[j] + 0 > sorted_items[j + 1] + 0:
                sorted_items[j], sorted_items[j + 1] = sorted_items[j + 1], sorted_items[j]
                swapped = True

        if not swapped:
            break

    return sorted_items

# --- Example of the bug in action ---

# This will work because the elements are numbers
numbers = [10, 2, 88, 1, -5]
print(f"Sorting numbers: {buggy_type_error_sort(numbers)}")
# Expected and Actual Output: Sorting numbers: [-5, 1, 2, 10, 88]

# This will fail with a TypeError
strings = ["cherry", "apple", "banana"]
try:
    buggy_type_error_sort(strings)
except TypeError as e:
    print(f"\nSorting strings failed as expected: {e}")
# Expected Output: TypeError: can only concatenate str (not "int") to str
```

Detect the issue in the code and correct it.

CODE:

lab7.4.2.py > bubble_sort

```python
def bubble_sort(items):
    """
    Sorts a list using a bubble sort algorithm.

    Args:
        items: A list of comparable items to sort (e.g., all numbers or
               all strings).

    Returns:
        A new list with the items sorted.
    """
    # Create a copy to avoid modifying the original list in place
    sorted_items = list(items)
    n = len(sorted_items)

    for i in range(n):
        swapped = False
        for j in range(0, n - i - 1):
            # The comparison now works for any comparable types.
            if sorted_items[j] > sorted_items[j + 1]:
                sorted_items[j], sorted_items[j + 1] = sorted_items[j + 1], sorted_items[j]
                swapped = True

        if not swapped:
            break

    return sorted_items

# --- Example of the corrected function ---

# This will work because the elements are numbers
numbers = [10, 2, 88, 1, -5]
print(f"Sorting numbers: {bubble_sort(numbers)}")
# Expected Output: Sorting numbers: [-5, 1, 2, 10, 88]

# This now works correctly with strings as well
strings = ["cherry", "apple", "banana"]
print(f"Sorting strings: {bubble_sort(strings)}")
# Expected Output: Sorting strings: ['apple', 'banana', 'cherry']
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/python.exe c:/Users/ramch/OneDrive/Desktop/ai/lab7.4.2.py
Sorting numbers: [-5, 1, 2, 10, 88]
Sorting strings: ['apple', 'banana', 'cherry']
PS C:\Users\ramch\OneDrive\Desktop\ai>
```

OBSERVATION:

The code has a bug of not sorting the list of names lexographically it only sorts the list of numbers in ascending order whereas in the code given by gemini has both the ability.

TASK :03

Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open() block).

PROMPT:

```
def log_message(filename, message):

    file_handle = open(filename, 'a')
    file_handle.write(f"{message}\n")

    print(f"Wrote to {filename}, but forgot to close it.")
log_file = "app.log"
log_message(log_file, "User logged in.")
log_message(log_file, "Data processing started.")
log_message(log_file, "Error encountered during processing.")
```

Detect the bug and correct the code.

CODE:

```
lab7.4.3  Welcome
def log_message(filename, message):
    """
    Appends a message to a log file, ensuring the file is properly closed.

    Args:
        filename (str): The path to the log file.
        message (str): The message to write to the file.
    """
    # Using a 'with' statement is the standard, safe way to handle files.
    # It automatically closes the file handle when the block is exited,
    # even if an error occurs, preventing resource leaks.
    with open(filename, 'a') as file_handle:
        file_handle.write(f"{message}\n")
    print(f"Successfully logged message to {filename}.")

log_file = "app.log"
log_message(log_file, "User logged in.")
log_message(log_file, "Data processing started.")
log_message(log_file, "Error encountered during processing.")
```

OUTPUT:

```
Sorting strings: ['apple', 'banana', 'cherry']
PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/python.exe c:/Users/ramch/OneDrive/Desktop/ai/
lab7.4.3.py
Successfully logged message to app.log.
Successfully logged message to app.log.
Successfully logged message to app.log.
PS C:\Users\ramch\OneDrive\Desktop\ai>
```

OBSERVATION:

The bug in the code is it opens a file to store some data btu forgots to close it. The bug is resolved by the AI

TASK :4

Provide a piece of code with a ZeroDivisionError inside a loop. Ask AI to add error handling using try-except and continue execution safely

PROMPT:

Generate a python code to perform the ZeroDivisionError and include try-except methods to improve the efficiency of the code.

CODE:

```python
def safe_division_loop(numerator, divisors):
    """
    Demonstrates handling a ZeroDivisionError within a loop.

    This function iterates through a list of divisors and attempts to divide
    the numerator by each one. It uses a try-except block to catch the
    ZeroDivisionError and continue the loop without crashing.

    Args:
        numerator (int or float): The number to be divided.
        divisors (list): A list of numbers to divide the numerator by.
    """
    print(f"Starting division loop with numerator: {numerator}\n")

    for divisor in divisors:
        try:
            # Attempt the division operation which might cause an error
            result = numerator / divisor
            print(f"Successfully calculated: {numerator} / {divisor} = {result}")
        except ZeroDivisionError:
            # This block executes ONLY if a ZeroDivisionError occurs
            print(f"Error: Cannot divide by zero. Skipping divisor '{divisor}'.")
            # The loop will automatically continue to the next item

    print("\nLoop finished. The program continued execution safely.")

# --- Example Usage ---

# A list of numbers to use as divisors, including 0 to trigger the error.
numbers_to_divide_by = [10, 5, 2, 0, 8, 4]
main_number = 100

safe_division_loop(main_number, numbers_to_divide_by)
```

OUTPUT:

```
PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/py
lab7.4.4.py
Starting division loop with numerator: 100

Successfully calculated: 100 / 10 = 10.0
Successfully calculated: 100 / 5 = 20.0
Successfully calculated: 100 / 2 = 50.0
Error: Cannot divide by zero. Skipping divisor '0'.
Successfully calculated: 100 / 8 = 12.5
Successfully calculated: 100 / 4 = 25.0

Loop finished. The program continued execution safely.
PS C:\Users\ramch\OneDrive\Desktop\ai>
```

OBSERVATION:

The code generated by AI has the ability to perform the operations like ZeroDivisionError and giving an efficient output.

TASK :05

Include a buggy class definition with incorrect __init__ parameters or attribute references. Ask AI to analyze and correct the constructor and attribute usage.

PROMPT:

lab7.4.5.py > BuggyUser > get_greeting

```python
class BuggyUser:
    def __init__(self, username, email):

        print(f"Initializing user with username: {username}")
        username = username
        self.email = email
        self.is_active = True

    def get_greeting(self):
        return f"Welcome back, {self.username}!"
print("Creating a BuggyUser instance...")
user = BuggyUser("alex_123", "alex@example.com")

print("\nAttempting to call a method on the instance...")
try:
    # This call will fail.
    greeting = user.get_greeting()
    print(greeting)
except AttributeError as e:
    print(f"Caught an expected error: {e}")
    print("This happened because 'self.username' was not set in __init__.")


```

Detect the bug in the code and rewrite the correct code with the correct class.

CODE:

```python
class User:
    """A simple class to represent a user."""
    def __init__(self, username, email):
        """
        Initializes a User object.

        Args:
            username (str): The user's username.
            email (str): The user's email address.
        """
        # THE FIX: Assign the 'username' parameter to an instance attribute.
        self.username = username
        self.email = email
        self.is_active = True

    def get_greeting(self):
        """Returns a welcome message for the user."""
        return f"Welcome back, {self.username}!"

print("Creating a User instance...")
user = User("alex_123", "alex@example.com")

print("\nCalling the get_greeting method...")
greeting = user.get_greeting()
print(greeting)
```

OUTPUT:

```
PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/python.exe c:/Users/ramch/OneDrive/Desktop/ai/
lab7.4.5.py
Creating a User instance...

Calling the get_greeting method...
Welcome back, alex_123!
PS C:\Users\ramch\OneDrive\Desktop\ai>
```

OBSERVATION:

The AI has efficiently corrected the bugs in the code provided by the user and gave the debugged code.