AI ASSISTED CODING

LAB-7.4

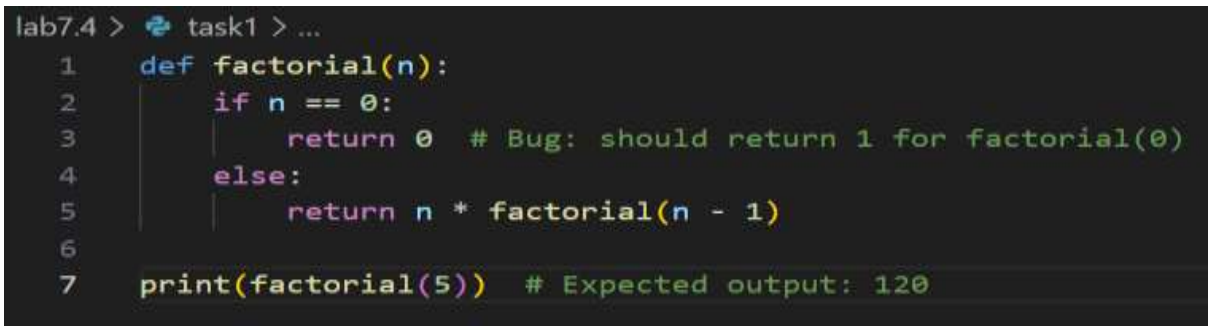
BATCH-04

ROLLNO:2403A52084

TASK-1:

PROMPT: Introduce a buggy Python function that calculates the factorial of a number using recursion.

CODE:

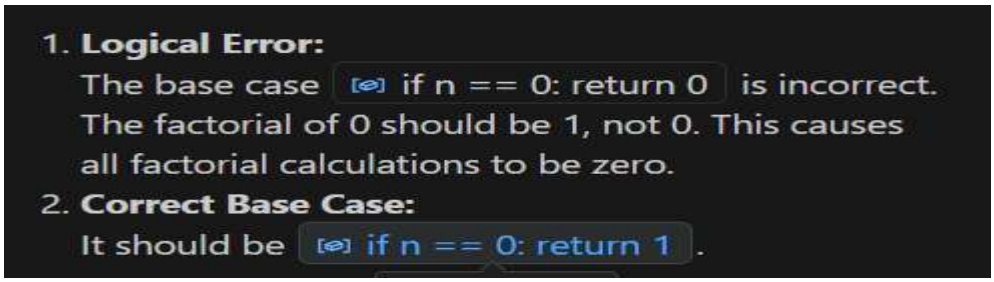


```
lab7.4 > task1 > ...
1   def factorial(n):
2       if n == 0:
3           return 0  # Bug: should return 1 for factorial(0)
4       else:
5           return n * factorial(n - 1)
6
7   print(factorial(5))  # Expected output: 120
```

OUTPUT:

```
PS C:\Users\Lasya\Desktop\AI ASSITED CODING> & C:\Users\Lasya\AppData\Local\Programs\Python\Python313
\python.exe "c:/Users/Lasya/Desktop/AI ASSITED CODING/lab7.4/task1"
0
```

ERRORS:



1. **Logical Error:**
   The base case `if n == 0: return 0` is incorrect. The factorial of 0 should be 1, not 0. This causes all factorial calculations to be zero.
2. **Correct Base Case:**
   It should be `if n == 0: return 1`.

TASK-2:

PROMPT: 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.

CODE:

```
def buggy_sort(lst):
    return sorted(lst)  # This will fail if lst contains both integers and strings

# Example usage:
mixed_list = [3, 'apple', 1, 'banana']
print(buggy_sort(mixed_list))  # TypeError: '<' not supported between instances of 'st
```

OUTPUT:

```
PS C:\Users\Lasya\Desktop\AI ASSITED CODING> & C:\Users\Lasya\AppData\Local\Programs\Python\Python313\pyt
on.exe "c:/Users/Lasya/Desktop/AI ASSITED CODING/lab7.4/TASK-2"
Traceback (most recent call last):
  File "c:\Users\Lasya\Desktop\AI ASSITED CODING\lab7.4\TASK-2", line 6, in <module>
    print(buggy_sort(mixed_list))  # TypeError: '<' not supported between instances of 'str' and 'int'
          ~~~~~~~~~~^^^^^^^^^^^^
  File "c:\Users\Lasya\Desktop\AI ASSITED CODING\lab7.4\TASK-2", line 2, in buggy_sort
    return sorted(lst)  # This will fail if lst contains both integers and strings
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\Lasya\Desktop\AI ASSITED CODING>
```

ERRORS:

The error occurs because the `sorted()` function cannot compare integers and strings directly. Python raises a `TypeError` when trying to sort a list with mixed types, as it doesn't know how to order numbers and text together.

TASK-3:

PROMPT: Write a Python snippet for file handling that opens a file but forgets to close it.

CODE:

```
lab7_4 > task3 > ...
1   # Buggy file handling: forgets to close the file
2   f = open('example.txt', 'w')
3   f.write('Hello, world!')
4
5   # Read and print the file contents to show output
6   f.close()  # Manually close so we can read
7   with open('example.txt', 'r') as f2:
8       print(f2.read())
9   # Copilot/AI: Improve this using best practice (with open() block)
```

OUTPUT:

```
e "c:/Users/User/OneDrive/Desktop/ai assisted coding/lab7_4/task3"
Hello, world!
```

EXPLANATION: Using the with open() statement is the standard best practice for file handling in Python. It automatically manages closing the file for you, even if an error occurs inside the block. This prevents potential resource leaks and makes the code safer and more readable..

TASK-4:

PROMPT: Write a python code that Provide a piece of code with a ZeroDivisionError inside a loop.

CODE:

```
task4
# This code will crash due to a ZeroDivisionError.
dividends = [10, 20, 30, 40, 50]
divisors = [2, 5, 0, 8, 4]

print("Starting division process...")

for i in range(len(dividends)):
    dividend = dividends[i]
    divisor = divisors[i]

    print(f"Attempting to divide {dividend} by {divisor}.")
    result = dividend / divisor
    print(f"Result: {result}")
    print("-" * 20)

print("Division process finished.")

# Copilot/AI: Please add error handling to the loop above.
# Use a try-except block to catch the ZeroDivisionError.
# When the error is caught, print a user-friendly message like "Error: Cannot divide by zero."
# and ensure the loop continues to the next iteration safely.
```

OUTPUT:

```
g/lab7_4/task4"
Starting division process...
Attempting to divide 10 by 2.
Result: 5.0
--------------------
Attempting to divide 20 by 5.
Result: 4.0
--------------------
Attempting to divide 30 by 0.
Traceback (most recent call last):
  File "c:\Users\User\OneDrive\Desktop\ai assisted coding\lab7_4\task4", line 12, in <module>
    result = dividend / divisor
             ~~~~~~~~~^~~~~~~~~
ZeroDivisionError: division by zero
```

EXPLANATION:

This script performs element-wise division on numbers from two lists. It will crash with a ZeroDivisionError because it attempts to divide by zero. The comments suggest adding try-except error handling to catch the error, allowing the program to continue running safely

TASK-5:

PROMPT: Write a python code that Include a buggy class definition with incorrect _init_ parameters or attribute references

CODE:

```python
# task5
# This class definition contains several bugs related to its constructor
# and attribute handling, which will cause it to crash.

class Vehicle:
    """Represents a vehicle with a make, model, and year."""

    def __init__(self, make, model_name, year):
        # The 'make' parameter is correctly received.
        self.make = make

        # Bug 1: The parameter is 'model_name', but the code tries to use 'model'.
        # This will raise a NameError because 'model' is not defined.
        self.model = model

        # Bug 2: The 'year' parameter is received but never assigned to an
        # instance attribute (e.g., self.year).

    def display_info(self):
        """Returns a string with the vehicle's details."""
        # Bug 3: This method tries to access 'self.year', but it was never
        # created in the __init__ constructor. This will raise an AttributeError.
        return f"Vehicle Details: {self.year} {self.make} {self.model}"

# Attempting to create and use an instance of the Vehicle class will fail.
print("Creating a new vehicle...")
my_car = Vehicle("Tesla", "Model S", 2023)
print(my_car.display_info())


# Copilot/AI: Please analyze the 'Vehicle' class and fix the following issues:
# 1. Correct the NameError in the __init__ method by using the proper parameter name.
# 2. Ensure the 'year' parameter is correctly assigned to an instance attribute
#    within the constructor.
# 3. Fix the AttributeError in the display_info() method so that it can correctly
#    access the vehicle's year.
```

OUTPUT:

```
g/lab7_4/task5
Creating a new vehicle...
Traceback (most recent call last):
Traceback (most recent call last):
  File "c:\Users\User\OneDrive\Desktop\ai assisted coding\lab7_4\task5", line 26, in <module>
    my_car = Vehicle("Tesla", "Model S", 2023)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\User\OneDrive\Desktop\ai assisted coding\lab7_4\task5", line 13, in __init__
    self.model = model
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\User\OneDrive\Desktop\ai assisted coding\lab7_4\task5", line 13, in __init__
    self.model = model
  File "c:\Users\User\OneDrive\Desktop\ai assisted coding\lab7_4\task5", line 13, in __init__
    self.model = model
                 ^^^^^
NameError: name 'model' is not defined
```

EXPLANATION: Include a buggy class definition with incorrect _init_ parameters or attribute references. This Vehicle class is buggy. Its constructor (_init_) fails to correctly assign the model and year attributes. This causes the program to crash with an error when you try to create a Vehicle object and display its information.