ASSIGNMENT-7.4

NAME : M.Nithisha

ENROLL NO : 2503A51458

BATCH : 16

Task Description #1:
• 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.
Expected Outcome #1:
• Copilot or Cursor AI correctly identifies missing base condition or incorrect recursive call and suggests a functional factorial implementation.

add.py

```python
def factorial(n):
    # Buggy implementation: missing base case and incorrect recursive call
    return n * factorial(n - 1)

def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)
print(factorial(5))  # Expected output: 120
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter Code

```
[Running] python -u "c:\Users\ajayk\OneDrive\Attachments\Desktop\AI&CC\add.py"
120

[Done] exited with code=0 in 0.219 seconds
```

Observation :

- The **first factorial function is redundant** and never used.
- The **correct implementation** is the second one, which works fine.
- Best practice: remove the buggy first definition to avoid confusion.

Task Description #2:
• 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.
Expected Outcome #2:
• AI detects the type inconsistency and either filters or converts list elements, ensuring successful sorting without a crash.

The program defines a function sort_list(lst) that:

- Converts all elements of the input list into **integers**.
- Sorts them in ascending order using Python's sorted() function.

```python
sort.py > ...
1   # def sort_list(lst):
2   #     return sorted(lst)
3   # mixed_list = [3, "2", 5, "1"]
4   # print(sort_list(mixed_list))
5
6   def sort_list(lst):
7       # Convert all elements to integers before sorting
8       return sorted([int(x) for x in lst])
9
10  # Example usage
11  mixed_list = [3, "2", 5, "1"]
12  print(sort_list(mixed_list))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter Code

```
[Running] python -u "c:\Users\ajayk\OneDrive\Attachments\Desktop\AI&CC\sort.py"
[1, 2, 3, 5]

[Done] exited with code=0 in 0.317 seconds
```

Observation :

1. **Commented-out code at the top** shows the **initial attempt**:
   - Directly using sorted(lst) would fail because Python cannot compare integers and strings (TypeError).
2. The **final implementation** (lines 6–8) correctly resolves this by:
   - Converting all elements to integers before sorting.
   - This makes the function robust for mixed-type numeric inputs.
3. **Output is correct** → elements are sorted numerically as expected.

Task Description #3:

• 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).

Expected Outcome #3:

• AI refactors the code to use a context manager, preventing resource leakage and runtime warnings.

```python
file.py
file.py > ...
f = open("example.txt", "w")
f.write("Hello, world!")
# Forgot to close the file

with open("example.txt", "w") as f:
    f.write("Hello, world!")
# File is automatically closed after the block
```

Observation :

- But the file is **not closed explicitly**, which can lead to:
  - Data not being saved properly (buffer not flushed).
  - File handle leaks.
- Uses the open() method and **with statement**.
- Automatically **closes the file** once the block is exited, even if an error occurs.
- This is the **best practice** for file handling in Python.

Task Description #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.

Expected Outcome #4:

• Copilot adds a try-except block around the risky operation, preventing crashes and printing a meaningful error message.

error.py

```python
#This will raise a ZeroDivisionError when n is 0
numbers = [5, 2, 0, 4]
for n in numbers:
    result = 10 / n
    print(f"10 / {n} = {result}")

# Fixed code with exception handling
numbers = [5, 2, 0, 4]
for n in numbers:
    try:
        result = 10 / n
        print(f"10 / {n} = {result}")
    except ZeroDivisionError:
        print(f"Cannot divide by zero for n = {n}")
        continue
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter Code

```
[Running] python -u "c:\Users\ajayk\OneDrive\Attachments\Desktop\AI&CC\error.py"
Traceback (most recent call last):
  File "c:\Users\ajayk\OneDrive\Attachments\Desktop\AI&CC\error.py", line 3, in <module>
10 / 5 = 2.0
10 / 2 = 5.0
    result = 10 / n
             ~~~^~~
ZeroDivisionError: division by zero

[Done] exited with code=1 in 0.305 seconds

[Running] python -u "c:\Users\ajayk\OneDrive\Attachments\Desktop\AI&CC\error.py"
10 / 5 = 2.0
10 / 2 = 5.0
Cannot divide by zero for n = 0
10 / 4 = 2.5

[Done] exited with code=0 in 0.299 seconds
```

Observation :

- **Without try-except :** The loop starts and successfully computes 100/10 and 100/5. When it reaches n = 0, the program raises a **ZeroDivisionError** and **stops execution immediately**. The last element 20 is never processed because the crash interrupts the loop.

- **With try-except :** The loop starts normally. At n = 0, instead of crashing, the program enters the except block. A **meaningful error message** is displayed (Division by zero is not allowed...). The continue statement ensures the loop safely moves on to the next value. Final output contains results for 10, 5, and 20 plus one error message for 0.

Task Description #5:

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

```
para.py > ...
class Person:
    def __init__(name, age):  # Missing 'self'
        name = name
        age = age

    def greet(self):
        print(f"Hello, my name is {name} and I am {age} years old.")  # Missing 'self.'

# Usage
p = Person("Alice", 30)
p.greet()
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, **/*.ts, !**/node_modules/**)

para.py 2

"name" is not defined Pylance(reportUndefinedVariable) [Ln 7, Col 36]

"age" is not defined Pylance(reportUndefinedVariable) [Ln 7, Col 52]

Expected Outcome #5:

• Copilot identifies mismatched parameters or missing self references and rewrites the class with accurate initialization and usage

```
para.py > ...
#The corrected code is as follows:
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
        (method) def greet(self: Self@Person) -> None
    def greet(self):
        print(f"Hello, my name is {self.name} and I am {self.age} years old.")

# Usage
p = Person("Alice", 30)
p.greet()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter Code

```
[Running] python -u "c:\Users\ajayk\OneDrive\Attachments\Desktop\AI&CC\para.py"
Hello, my name is Alice and I am 30 years old.
```

Observation :

The original class definition of Student contained errors such as a missing self parameter in the __init__ method, incorrect attribute assignment without self, and undefined attribute references in the display() method. These issues caused initialization and runtime errors when creating objects. After correction, the class was rewritten with self included in the constructor, and attributes (name, age) were properly assigned as self.name and self.age. The display() method now works correctly, printing the student's details.