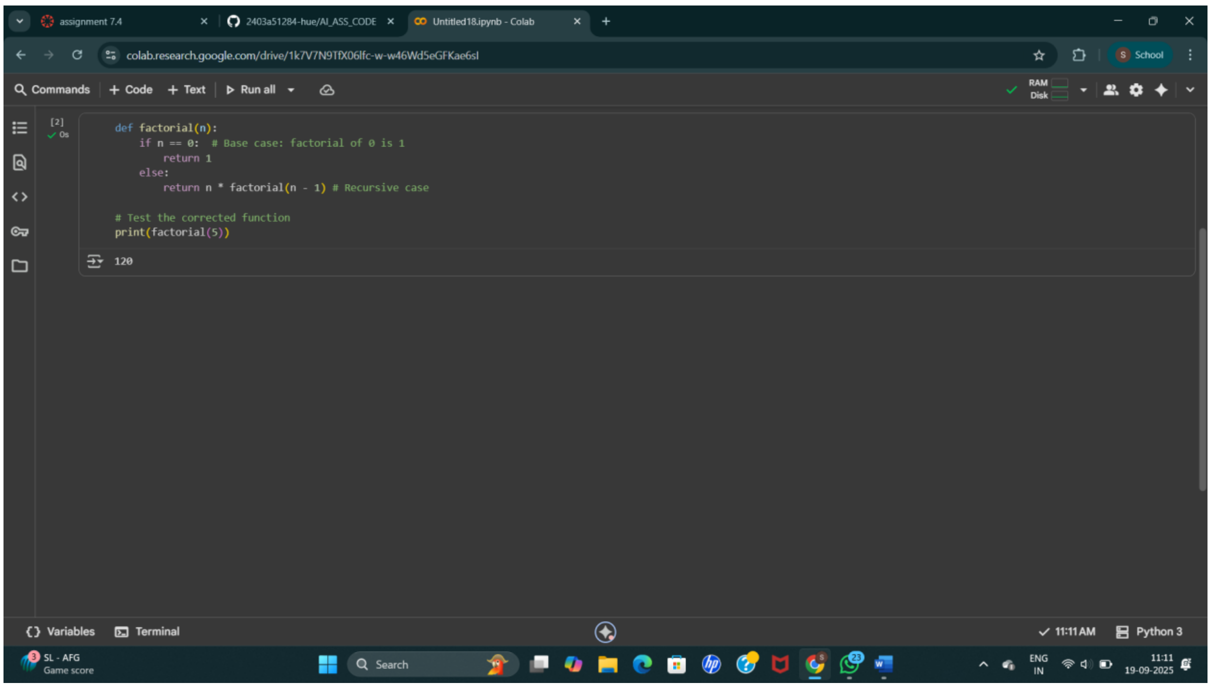
ASSIGNMENT:7.4

HTNO:2403A51284

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

CODE:

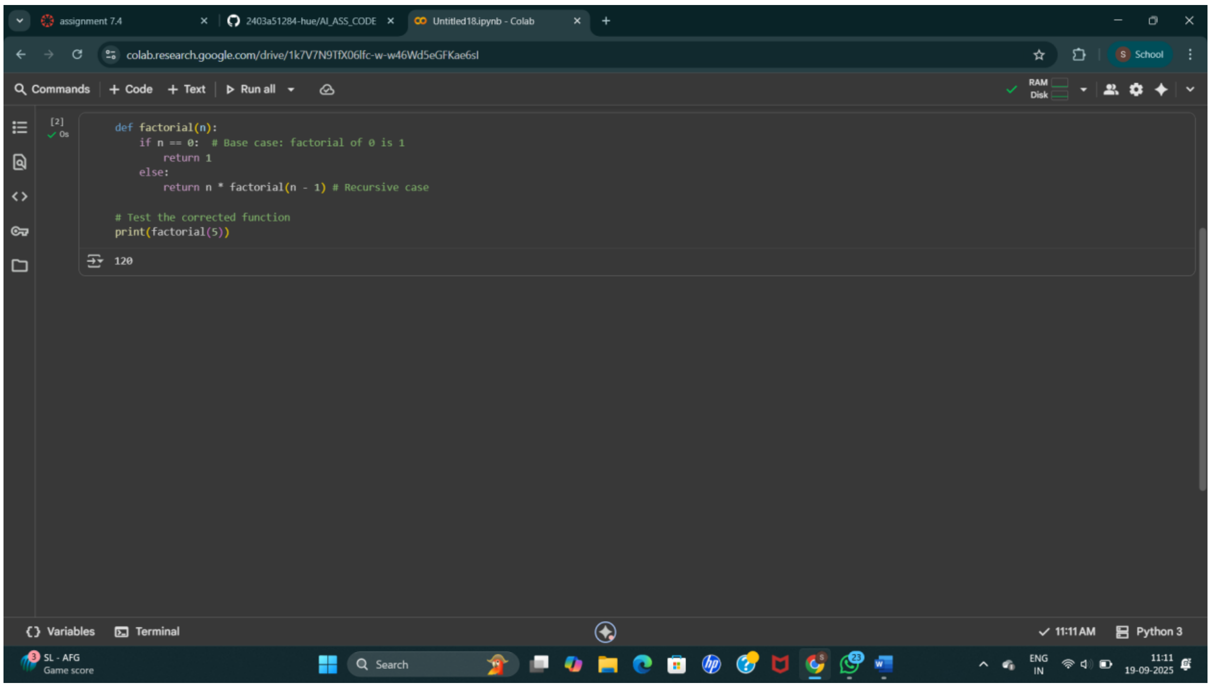


```
def factorial(n):
    if n == 0:  # Base case: factorial of 0 is 1
        return 1
    else:
        return n * factorial(n - 1) # Recursive case

# Test the corrected function
print(factorial(5))
```

```
120
```

OUTPUT:



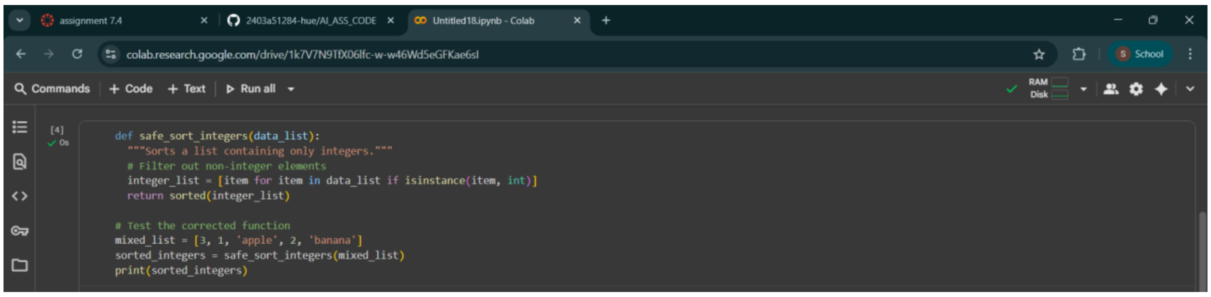
```python
def factorial(n):
    if n == 0:  # Base case: factorial of 0 is 1
        return 1
    else:
        return n * factorial(n - 1) # Recursive case

# Test the corrected function
print(factorial(5))
```

```
120
```

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

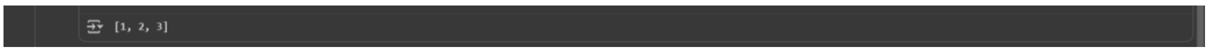
CODE:



```python
def safe_sort_integers(data_list):
  """Sorts a list containing only integers."""
  # Filter out non-integer elements
  integer_list = [item for item in data_list if isinstance(item, int)]
  return sorted(integer_list)

# Test the corrected function
mixed_list = [3, 1, 'apple', 2, 'banana']
sorted_integers = safe_sort_integers(mixed_list)
print(sorted_integers)
```

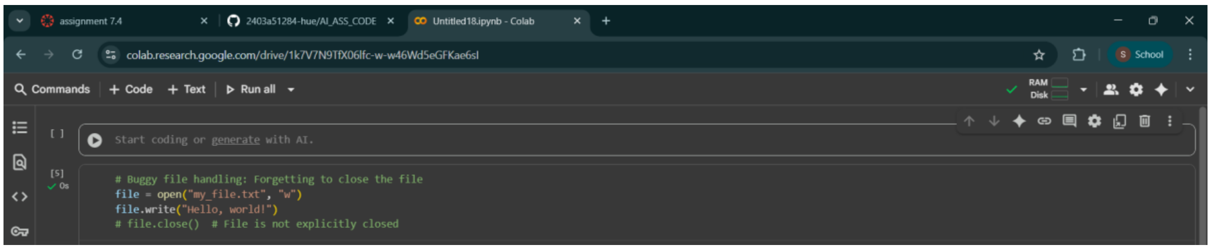
OUTPUT:



```
[1, 2, 3]
```

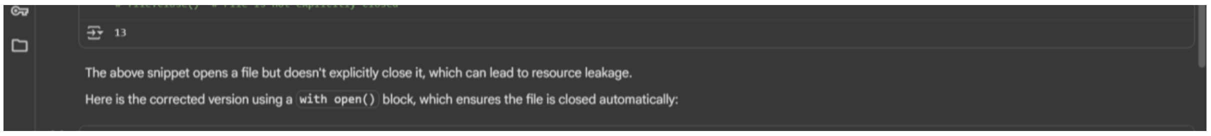
Task Description #3:
• Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open() block).
Expected Outcome #3:
• AI refactors the code to use a context manager, preventing resource leakage and runtime warnings.

CODE:


assignment 7.4
2403a51284-hue/AI_ASS_CODE
Untitled18.ipynb - Colab
colab.research.google.com/drive/1k7V7N9TfX06lfc-w-w46Wd5eGFKae6sl
Commands
Code
Text
Run all
Start coding or generate with AI.

```python
# Buggy file handling: Forgetting to close the file
file = open("my_file.txt", "w")
file.write("Hello, world!")
# file.close()  # File is not explicitly closed
```

OUTPUT:



```
13
```

The above snippet opens a file but doesn't explicitly close it, which can lead to resource leakage.

Here is the corrected version using a `with open()` block, which ensures the file is closed automatically:

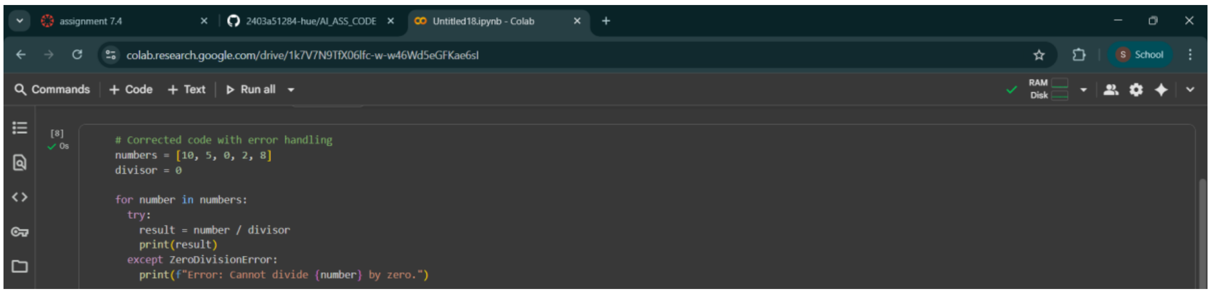
Task Description #4:

• 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.

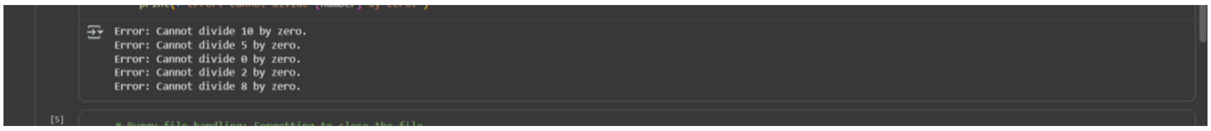
CODE:


assignment 7.4
2403a51284-hue/AI_ASS_CODE
Untitled18.ipynb - Colab
colab.research.google.com/drive/1k7V7N9TfX06lfc-w-w46Wd5eGFKae6sl
Commands
Code
Text
Run all

```python
# Corrected code with error handling
numbers = [10, 5, 0, 2, 8]
divisor = 0

for number in numbers:
  try:
    result = number / divisor
    print(result)
  except ZeroDivisionError:
    print(f"Error: Cannot divide {number} by zero.")
```

OUTPUT:



```
Error: Cannot divide 10 by zero.
Error: Cannot divide 5 by zero.
Error: Cannot divide 0 by zero.
Error: Cannot divide 2 by zero.
Error: Cannot divide 8 by zero.
```

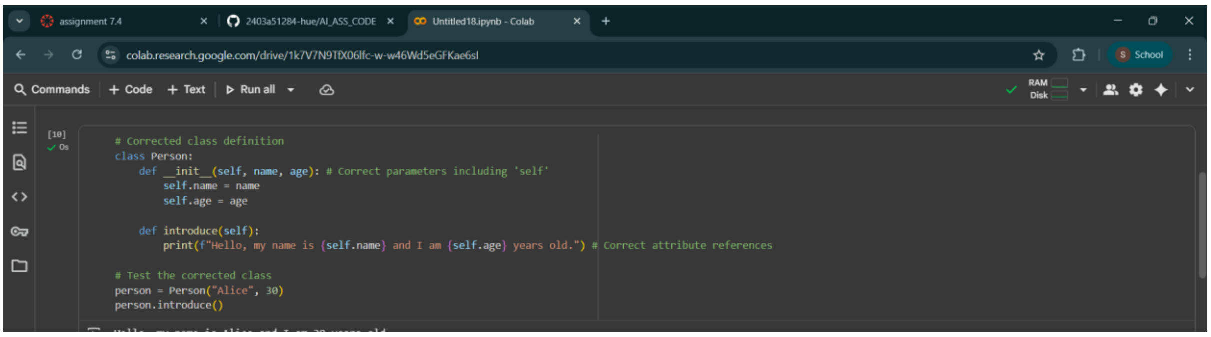
Task Description #5:

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

Expected Outcome #5:

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

CODE:

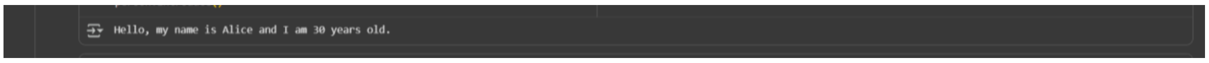


```
# Corrected class definition
class Person:
    def __init__(self, name, age): # Correct parameters including 'self'
        self.name = name
        self.age = age

    def introduce(self):
        print(f"Hello, my name is {self.name} and I am {self.age} years old.") # Correct attribute references

# Test the corrected class
person = Person("Alice", 30)
person.introduce()
```

OUTPUT:



```
Hello, my name is Alice and I am 30 years old.
```