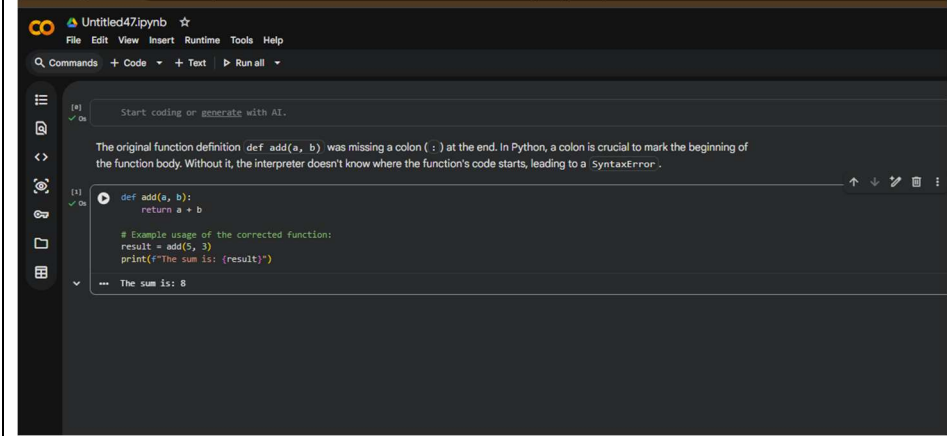


K.Akshay
2303A51817
B-26

| | | | | | | | | | | | | | | | | | | | | |
|---|-------------------|---|---------------------------------|--------------------|---------------|------------------------------|-------------------|----------------|----------------|--------------|----------------------------------|-----------------------|--------------------|-----------------|-------------------|---------------|---------------|-------------------------|-----------------------|----------------|
| SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE | | DEPARTMENT OF COMPUTER SCIENCE ENGINEERING | | | | | | | | | | | | | | | | | | |
| Program Name: B. Tech | | Assignment Type: Lab | Academic Year: 2025-2026 | | | | | | | | | | | | | | | | | |
| Course Coordinator Name | | Dr. Rishabh Mittal | | | | | | | | | | | | | | | | | | |
| Instructor(s) Name | | <table border="1"> <tr><td>Mr. S Naresh Kumar</td></tr> <tr><td>Ms. B. Swathi</td></tr> <tr><td>Dr. Sasanko Shekhar Gantayat</td></tr> <tr><td>Mr. Md Sallauddin</td></tr> <tr><td>Dr. Mathivanan</td></tr> <tr><td>Mr. Y Srikanth</td></tr> <tr><td>Ms. N Shilpa</td></tr> <tr><td>Dr. Rishabh Mittal (Coordinator)</td></tr> <tr><td>Dr. R. Prashant Kumar</td></tr> <tr><td>Mr. Ankushavali MD</td></tr> <tr><td>Mr. B Viswanath</td></tr> <tr><td>Ms. Sujitha Reddy</td></tr> <tr><td>Ms. A. Anitha</td></tr> <tr><td>Ms. M.Madhuri</td></tr> <tr><td>Ms. Katherashala Swetha</td></tr> <tr><td>Ms. Velpula sumalatha</td></tr> <tr><td>Mr. Bingi Raju</td></tr> </table> | | Mr. S Naresh Kumar | Ms. B. Swathi | Dr. Sasanko Shekhar Gantayat | Mr. Md Sallauddin | Dr. Mathivanan | Mr. Y Srikanth | Ms. N Shilpa | Dr. Rishabh Mittal (Coordinator) | Dr. R. Prashant Kumar | Mr. Ankushavali MD | Mr. B Viswanath | Ms. Sujitha Reddy | Ms. A. Anitha | Ms. M.Madhuri | Ms. Katherashala Swetha | Ms. Velpula sumalatha | Mr. Bingi Raju |
| Mr. S Naresh Kumar | | | | | | | | | | | | | | | | | | | | |
| Ms. B. Swathi | | | | | | | | | | | | | | | | | | | | |
| Dr. Sasanko Shekhar Gantayat | | | | | | | | | | | | | | | | | | | | |
| Mr. Md Sallauddin | | | | | | | | | | | | | | | | | | | | |
| Dr. Mathivanan | | | | | | | | | | | | | | | | | | | | |
| Mr. Y Srikanth | | | | | | | | | | | | | | | | | | | | |
| Ms. N Shilpa | | | | | | | | | | | | | | | | | | | | |
| Dr. Rishabh Mittal (Coordinator) | | | | | | | | | | | | | | | | | | | | |
| Dr. R. Prashant Kumar | | | | | | | | | | | | | | | | | | | | |
| Mr. Ankushavali MD | | | | | | | | | | | | | | | | | | | | |
| Mr. B Viswanath | | | | | | | | | | | | | | | | | | | | |
| Ms. Sujitha Reddy | | | | | | | | | | | | | | | | | | | | |
| Ms. A. Anitha | | | | | | | | | | | | | | | | | | | | |
| Ms. M.Madhuri | | | | | | | | | | | | | | | | | | | | |
| Ms. Katherashala Swetha | | | | | | | | | | | | | | | | | | | | |
| Ms. Velpula sumalatha | | | | | | | | | | | | | | | | | | | | |
| Mr. Bingi Raju | | | | | | | | | | | | | | | | | | | | |
| Course Code | 23CS002PC304 | Course Title | AI Assisted Coding | | | | | | | | | | | | | | | | | |
| Year/Sem | III/II | Regulation | R23 | | | | | | | | | | | | | | | | | |
| Date and Day of Assignment | Week4 – Wednesday | Time(s) | 23CSBTB01 To 23CSBTB52 | | | | | | | | | | | | | | | | | |
| Duration | 2 Hours | Applicable to Batches | All batches | | | | | | | | | | | | | | | | | |
| AssignmentNumber: 7.3(Present assignment number)/24(Total number of assignments) | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | |
| Q.No. | Question | Expected Time to complete | | | | | | | | | | | | | | | | | | |
| 1 | | Week4 - | | | | | | | | | | | | | | | | | | |

| | | |
|--|--|------------------|
| | <div data-bbox="358 191 1299 268"><p>Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs</p></div> <div data-bbox="358 268 1299 430"><p>Lab Objectives</p><ul style="list-style-type: none">• To identify and correct syntax, logic, and runtime errors in Python programs using AI tools• To understand common programming bugs and AI-assisted debugging suggestions• To evaluate how AI explains, detects, and fixes different types of coding errors• To build confidence in using AI for structured debugging practices</div> <div data-bbox="358 430 1299 625"><p>Lab Outcomes (LOs)</p><p>After completing this lab, students will be able to:</p><ul style="list-style-type: none">• Use AI tools to detect and correct syntax, logic, and runtime errors• Interpret AI-suggested bug fixes and explanations• Apply systematic debugging strategies using AI-generated insights• Refactor buggy code using reliable programming patterns</div> <div data-bbox="358 625 1299 682"><p>Task 1: Fixing Syntax Errors</p></div> <div data-bbox="358 682 1299 745"><p>Scenario</p><p>You are reviewing a Python program where a basic function definition contains a syntax error.</p></div> <div data-bbox="386 745 695 919"><pre>python def add(a, b) return a + b</pre></div> <div data-bbox="358 951 1299 1092"><p>Requirements</p><ul style="list-style-type: none">• Provide a Python function add(a, b) with a missing colon• Use an AI tool to detect the syntax error• Allow AI to correct the function definition• Observe how AI explains the syntax issue</div> <div data-bbox="358 1113 1299 1228"><p>Expected Output</p><ul style="list-style-type: none">• Corrected function with proper syntax• Syntax error resolved successfully• AI-generated explanation of the fix</div> <div data-bbox="358 1228 1299 1285"><p>PROMPT: Here is a Python function with a syntax error. Please find the error, fix it, and explain what was wrong. def add(a, b) return a + b</p></div> <div data-bbox="358 1285 1299 1341"><p>EXPLANATION: The error was caused by a missing colon (:) at the end of the function definition.</p></div> <div data-bbox="358 1341 1299 1398"><p>In Python, a colon is required to start the function body. Adding it fixes the syntax and allows the code to run properly.</p></div> <div data-bbox="358 1398 1299 1831"></div> | <p>Wednesday</p> |
|--|--|------------------|

Task 2: Debugging Logic Errors in Loops

Scenario

You are debugging a loop that runs infinitely due to a logical mistake.

```
python

def count_down(n):
    while n >= 0:
        print(n)
        n += 1 # Should be n -= 1
```

Requirements

- Provide a loop with an **increment or decrement error**
- Use AI to identify the cause of infinite iteration
- Let AI fix the loop logic
- Analyze the corrected loop behavior

Expected Output

- Infinite loop issue resolved
- Correct increment/decrement logic applied
- AI explanation of the logic error

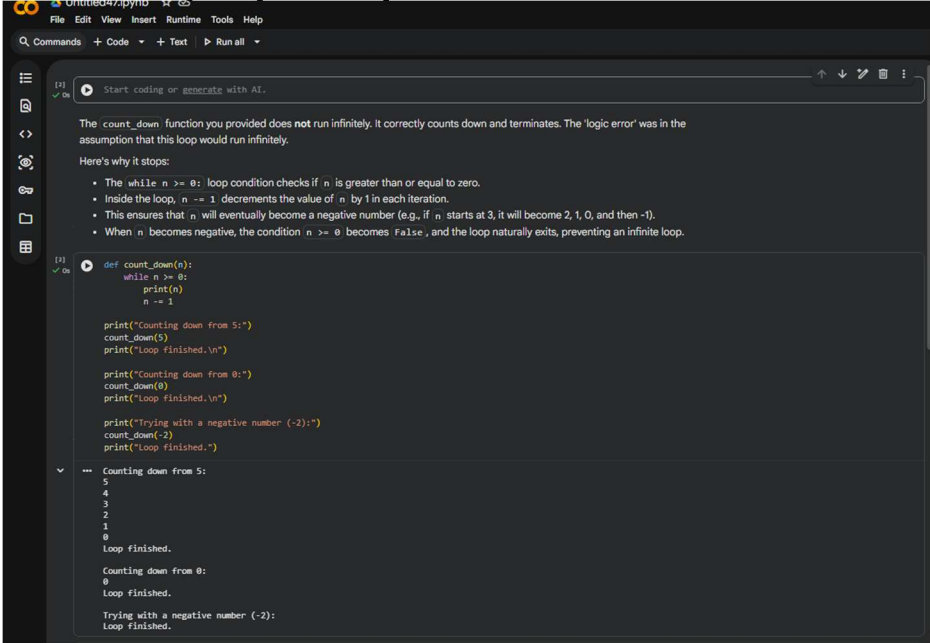
PROMPT: Here is a loop that runs infinitely. Please identify why it does not stop, fix the logic error, and explain the correction.

```
def count_down(n):
    while n >= 0:
        print(n)
        n -= 1
```

EXPLANATION: The loop was infinite because the counter variable was never updated correctly inside the loop.

By adding the proper increment/decrement, the condition eventually becomes false.

This allows the loop to stop after the expected number of iterations.



Task 3: Handling Runtime Errors (Division by Zero)

Scenario

A Python function crashes during execution due to a division by zero error.

```
# Debug the following code

def divide(a, b):
    return a / b

print(divide(10, 0))
```

Requirements

- Provide a function that performs division without validation
- Use AI to identify the runtime error
- Let AI add try-except blocks for safe execution
- Review AI's error-handling approach

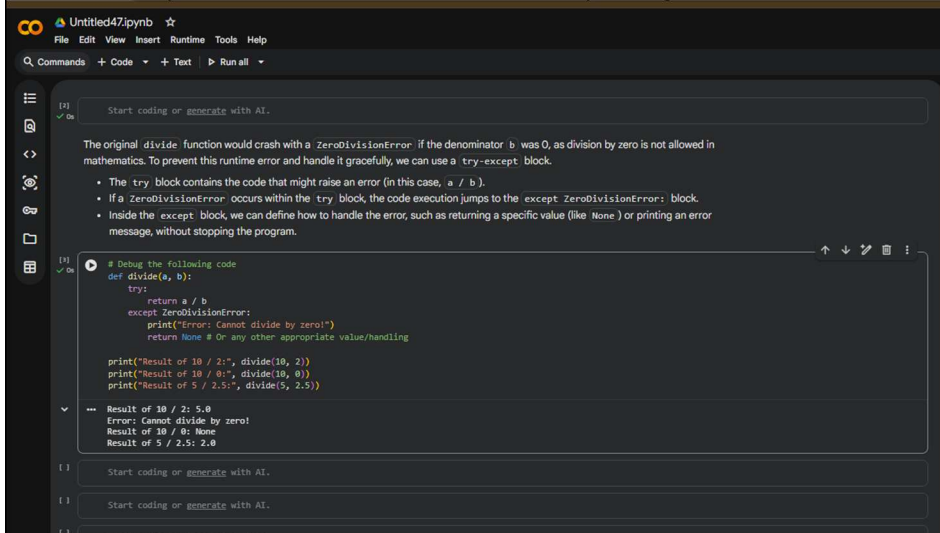
Expected Output

- Function executes safely without crashing
- Division by zero handled using try-except
- Clear AI-generated explanation of runtime error handling

PROMPT: Here is a Python function that performs division but crashes at runtime. Please identify the error, fix it using try-except, and explain how the error is handled.”

EXPLANATION: The program crashes when division by zero occurs, which raises a runtime error.

Using a try-except block prevents the program from stopping suddenly. Instead, it safely catches the error and shows a user-friendly message.



Task 4: Debugging Class Definition Errors

Scenario

You are given a faulty Python class where the constructor is incorrectly defined.

```
python

class Rectangle:
    def __init__(length, width):
        self.length = length
        self.width = width
```

Requirements

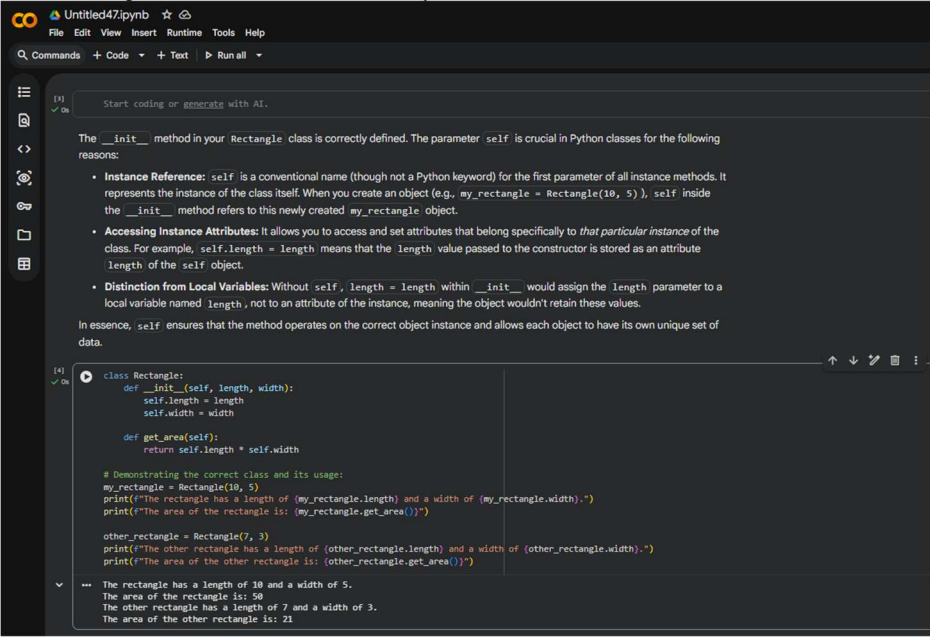
- Provide a class definition with **missing self-parameter**
- Use AI to identify the issue in the `__init__()` method
- Allow AI to correct the class definition
- Understand why self is required

Expected Output

- Corrected `__init__()` method
- Proper use of self in class definition
- AI explanation of object-oriented error

PROMPT: Here is a Python class with an incorrect constructor. Please find the error in the `__init__` method, fix it, and explain why self is required.

EXPLANATION: The constructor was missing self as the first parameter. self is required to store values inside the object. After adding it, the class works correctly.



```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def get_area(self):
        return self.length * self.width

# Demonstrating the correct class and its usage:
my_rectangle = Rectangle(10, 5)
print(f"The rectangle has a length of {my_rectangle.length} and a width of {my_rectangle.width}." )
print(f"The area of the rectangle is: {my_rectangle.get_area()}")

other_rectangle = Rectangle(7, 3)
print(f"The other rectangle has a length of {other_rectangle.length} and a width of {other_rectangle.width}." )
print(f"The area of the other rectangle is: {other_rectangle.get_area()}")
```

The rectangle has a length of 10 and a width of 5.
The area of the rectangle is: 50
The other rectangle has a length of 7 and a width of 3.
The area of the other rectangle is: 21

Task 5: Resolving Index Errors in Lists

Scenario

A program crashes when accessing an invalid index in a list.

```
python

numbers = [1, 2, 3]
print(numbers[5])
```

Requirements

- Provide code that accesses an **out-of-range list index**
- Use AI to identify the Index Error
- Let AI suggest safe access methods
- Apply bounds checking or exception handling

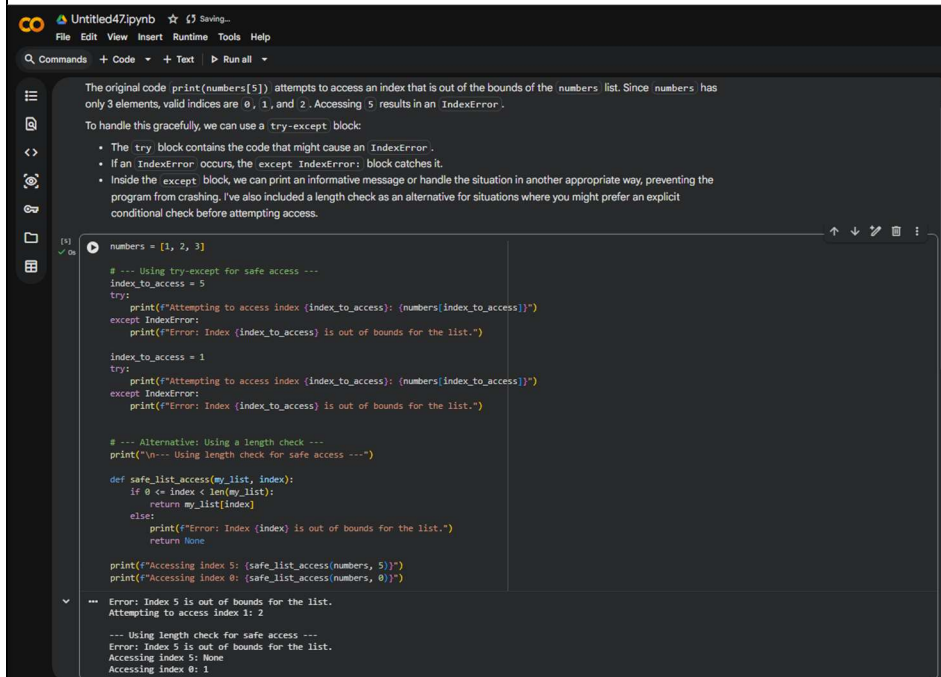
Expected Output

- Index error resolved
- Safe list access logic implemented
- AI suggestion using length checks or exception handling.

PROMPT: Here is a Python program that crashes due to an invalid list index. Please identify the `IndexError`, fix it using safe list access (length check or try-except), and explain the solution.

```
numbers = [1, 2, 3]
print(numbers[5])
```

EXPLANATION: The error occurs because the program tries to access an index that does not exist in the list.
By checking the list length or using try-except, we prevent the crash.
This makes the program run safely even when the index is invalid.



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar indicates the file is 'Untitled47.ipynb' and is 'Saving...'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu is a search bar and a 'Run all' button. The left sidebar contains icons for file explorer, search, and other notebook functions. The main area displays a code cell with the following content:

```
[5]: numbers = [1, 2, 3]

# --- Using try-except for safe access ---
index_to_access = 5
try:
    print(f"Attempting to access index (index_to_access): {numbers[index_to_access]}")
except IndexError:
    print(f"Error: Index (index_to_access) is out of bounds for the list.")

index_to_access = 1
try:
    print(f"Attempting to access index (index_to_access): {numbers[index_to_access]}")
except IndexError:
    print(f"Error: Index (index_to_access) is out of bounds for the list.")

# --- Alternative: Using a length check ---
print("\n--- Using length check for safe access ---")

def safe_list_access(my_list, index):
    if 0 <= index < len(my_list):
        return my_list[index]
    else:
        print(f"Error: Index (index) is out of bounds for the list.")
        return None

print(f"Accessing index 5: {safe_list_access(numbers, 5)}")
print(f"Accessing index 0: {safe_list_access(numbers, 0)}")
```

The output of the code cell is shown below the code:

```
*** Error: Index 5 is out of bounds for the list.
Attempting to access index 1: 2

--- Using length check for safe access ---
Error: Index 5 is out of bounds for the list.
Accessing index 5: None
Accessing index 0: 1
```

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots