

AI-assisted coding

Assignment -7.4

Name: V. Vyshnavi

Batch-14

HT.No.: 2303A51968

Task 1: Debugging a Recursive Calculation Module

Scenario

You are maintaining a utility module in a software project that performs mathematical computations. One function is meant to calculate the factorial of a number, but users are reporting crashes or incorrect outputs.

Task Description

You are given a Python function intended to calculate the factorial of a number using recursion, but it contains logical or syntactical errors (such as a missing base condition or incorrect recursive call).

Use GitHub Copilot or Cursor AI to:

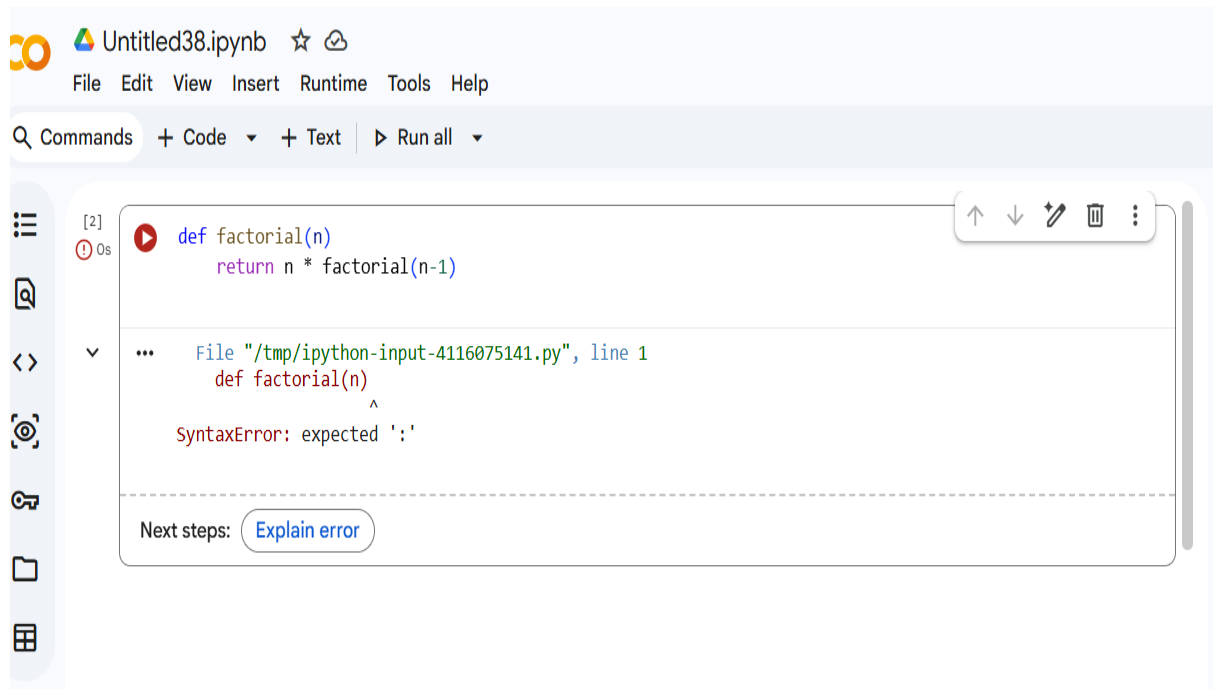
- Analyze the faulty code
- Identify the exact cause of the error
- Suggest and apply corrections to make the function work

Correctly Document how the AI detected the issue and what changes were made.

Expected Outcome

- A corrected recursive factorial function
- AI-generated explanation identifying:
 - o The missing or incorrect base case
 - o The corrected recursive logic
- Sample input/output demonstrating correct execution

Code with error:



The screenshot shows a Jupyter Notebook titled "Untitled38.ipynb". The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with "Commands", "+ Code", "+ Text", and "Run all". On the left is a sidebar with icons for file operations. The main area shows a code cell with the following Python code:

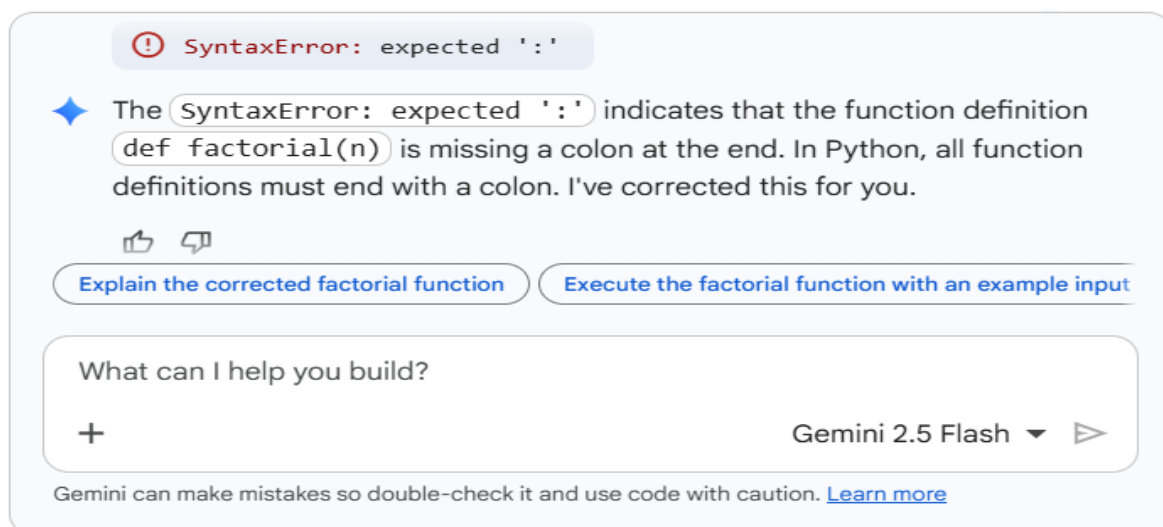
```
[2] def factorial(n):  
    return n * factorial(n-1)
```

The code cell has a red error icon and a tooltip showing the error message:

```
... File "/tmp/ipython-input-4116075141.py", line 1  
    def factorial(n)  
        ^  
SyntaxError: expected ':'
```

Below the error message, there is a "Next steps:" section with a button labeled "Explain error".

Explanation:



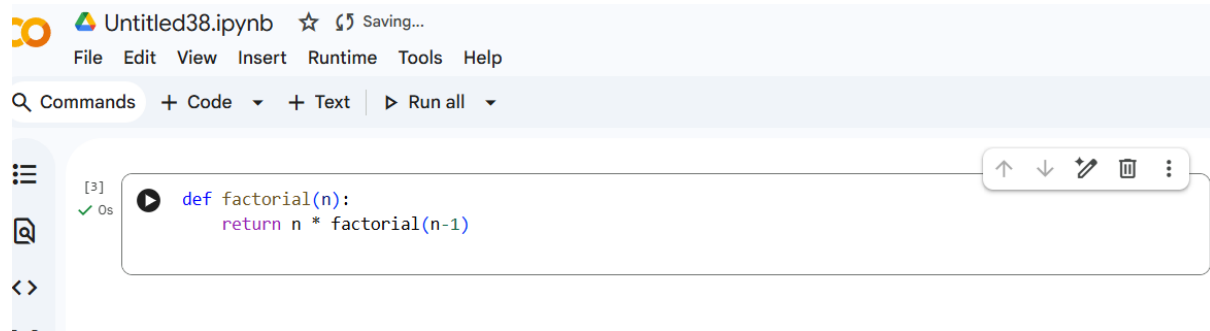
The explanation section shows a red error icon and the message "SyntaxError: expected ':'". Below this, a blue star icon is followed by text explaining the error:

✦ The `SyntaxError: expected ':'` indicates that the function definition `def factorial(n)` is missing a colon at the end. In Python, all function definitions must end with a colon. I've corrected this for you.

Below the text are two buttons: "Explain the corrected factorial function" and "Execute the factorial function with an example input".

At the bottom, there is a chat interface with the prompt "What can I help you build?" and a plus sign icon. The chat is powered by "Gemini 2.5 Flash". A disclaimer at the bottom states: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

Actual code:



Task 2: Fixing Data Type Errors in a Sorting Utility

Scenario

You are developing a data processing script that sorts user input values.

The program crashes when users enter mixed data types.

Task Description

You are provided with a list-sorting function that fails due to a

`TypeError` caused by mixed data types (e.g., integers and strings).

Use GitHub Copilot or Cursor AI to:

- Detect the root cause of the runtime error
- Modify the code to ensure consistent sorting (by filtering or type conversion)
- Prevent the program from crashing Explain the debugging steps followed by the AI.

Expected Outcome

- A corrected sorting function
- AI-generated solution handling type inconsistencies
- Successful sorting without runtime errors
- Explanation of how the fix improves robustness

Code with error:

The screenshot shows a Jupyter Notebook window titled "Untitled38.ipynb". The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a "Commands" bar with options: "Commands", "+ Code", "+ Text", and "Run all". On the left is a sidebar with icons for file operations. The main area displays a code cell with the following code:

```
[4] data = [10, "20", 5, "apple", 15]
    print(sorted(data))
```

The cell has a status bar showing "[4]" and "0s". Below the code, a traceback is visible:

```
...
TypeError                                Traceback (most recent call last)
/tmp/ipython-input-3788855612.py in <cell line: 0>()
      1 data = [10, "20", 5, "apple", 15]
----> 2 print(sorted(data))

TypeError: '<' not supported between instances of 'str' and 'int'
```

At the bottom of the cell, there is a "Next steps:" section with a button labeled "Explain error".

Explanation:

The screenshot shows a Gemini AI chat interface. The chat history includes a user message asking for an explanation of an error and a Gemini response.

User: Please explain this error:

Gemini: `TypeError: '<' not supported between instances of 'str' and 'int'`

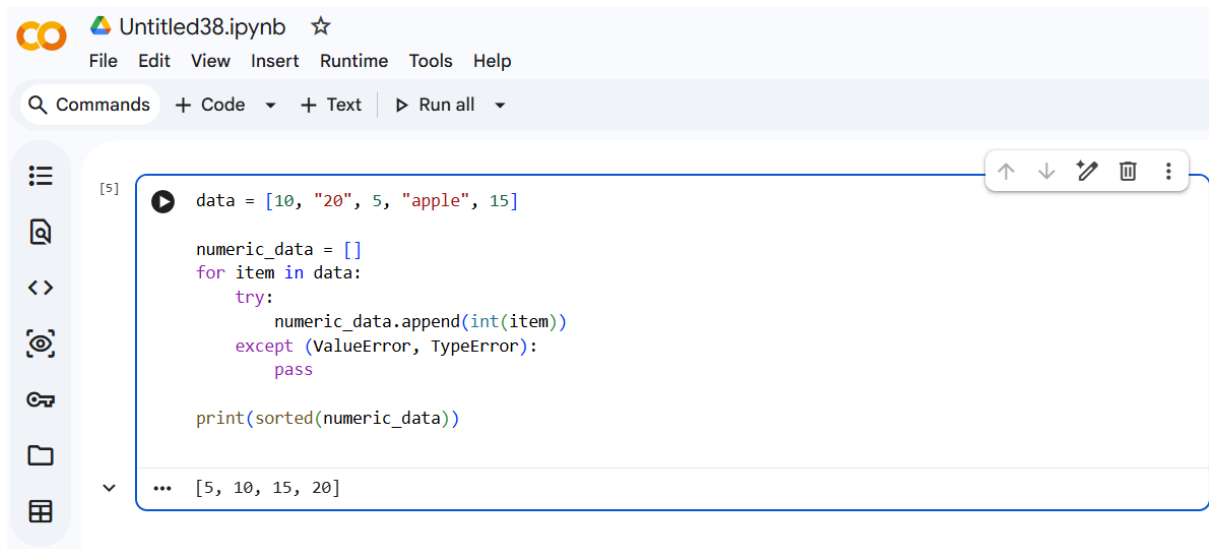
User: The error `TypeError: '<' not supported between instances of`

Gemini buttons: "Explain the 'TypeError' in the last cell", "Show the contents of 'numeric_data' before sorting"

User input field: "What can I help you build?"

Gemini footer: Gemini 2.5 Flash

Actual code:



The screenshot shows a Jupyter Notebook window titled 'Untitled38.ipynb'. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu is a toolbar with 'Commands', '+ Code', '+ Text', and 'Run all'. On the left is a sidebar with icons for file operations. The main area contains a code cell with the following Python code:

```
[5] data = [10, "20", 5, "apple", 15]

numeric_data = []
for item in data:
    try:
        numeric_data.append(int(item))
    except (ValueError, TypeError):
        pass

print(sorted(numeric_data))
```

Below the code cell, the output is displayed as a list: `[5, 10, 15, 20]`.

Task 3: Improving File Handling Reliability

Scenario

A backend script reads data from files regularly. Over time, the system shows performance issues due to improper resource management.

Task Description

You are given a Python file-handling snippet that opens a file but does not explicitly close it.

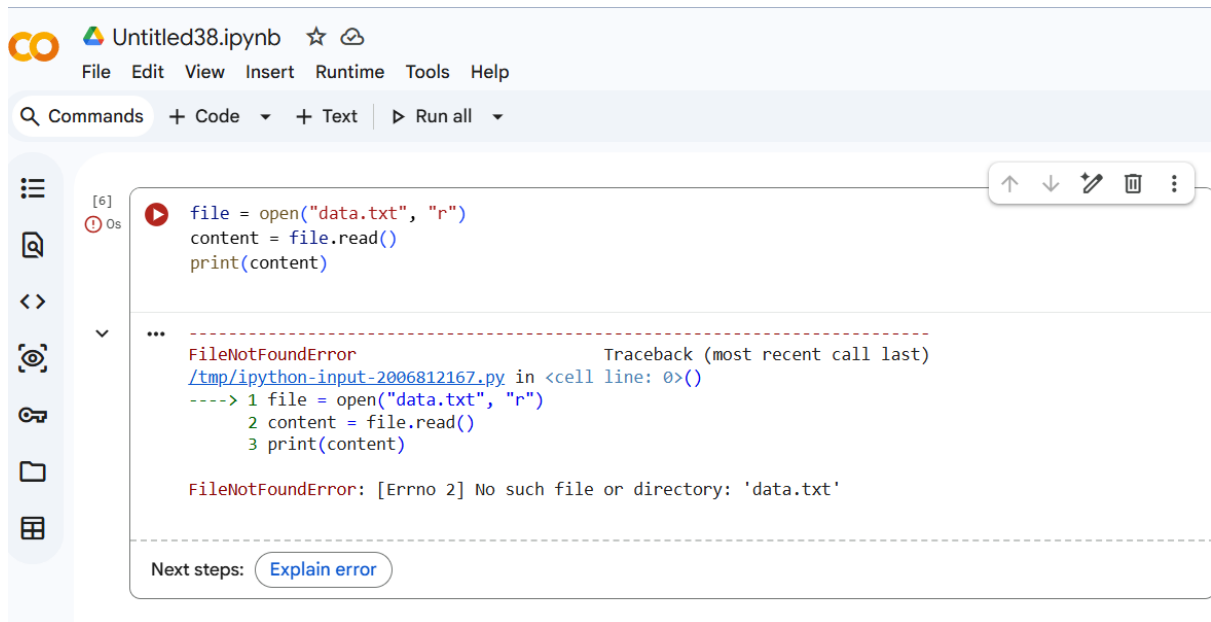
Use GitHub Copilot or Cursor AI to:

- Identify the potential problem in the code
- Refactor it using best practices (such as a context manager)
- Ensure safe and reliable file handling Briefly describe why the revised approach is better.

Expected Outcome

- Refactored code using the `with open()` statement
- AI explanation highlighting prevention of resource leaks
- Clean execution without warnings or errors

Code with error:



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar (Commands, + Code, + Text, Run all). The notebook is titled 'Untitled38.ipynb'. A code cell contains the following Python code:

```
file = open("data.txt", "r")
content = file.read()
print(content)
```

The code cell has a status bar showing '[6] 0s' and a red error icon. Below the code cell, a traceback is displayed:

```
FileNotFoundError                                Traceback (most recent call last)
/tmp/ipython-input-2006812167.py in <cell line: 0>()
----> 1 file = open("data.txt", "r")
      2 content = file.read()
      3 print(content)

FileNotFoundError: [Errno 2] No such file or directory: 'data.txt'
```

Below the traceback, there is a 'Next steps:' section with a button labeled 'Explain error'.

Explanation:



The screenshot shows a Gemini chat interface. The user's prompt is: "Please explain this error:". The Gemini response is: "FileNotFoundError: [Errno 2] No such file or directory: 'data.txt'". Below the response, there are three buttons: "Run the code cell again", "Explain why the FileNotFoundError might still be present", and "Show more". At the bottom of the chat interface, there is a text input field with the placeholder text "What can I help you build?", a plus sign icon, and a dropdown menu showing "Gemini 2.5 Flash". Below the input field, there is a link: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

Actual code:



Task 4: Handling Runtime Errors Gracefully in Loops Scenario

You are working on a data analysis script that processes a list of values.

Some values cause runtime errors, but the program should continue processing remaining data.

Task Description

You are provided with a code snippet containing a `ZeroDivisionError` inside a loop.

Use GitHub Copilot or Cursor AI to:

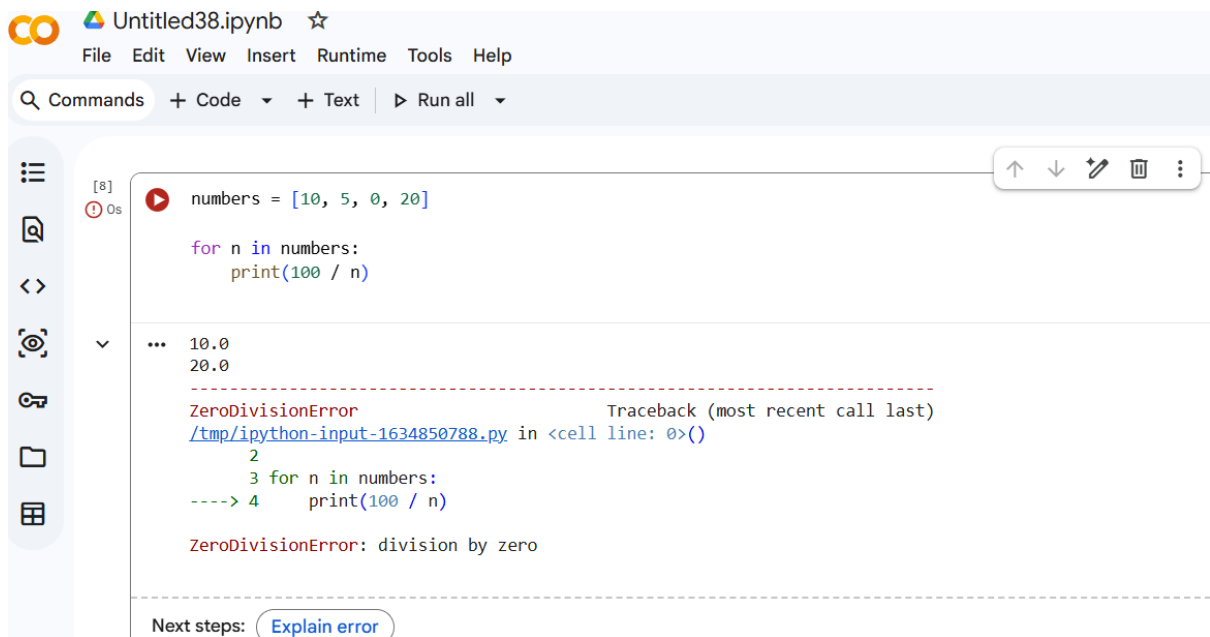
- Detect the exact location of the error
- Add appropriate exception handling using `try-except`
- Ensure the loop continues executing safely

Document how AI improved the fault tolerance of the program.

Expected Outcome

- Updated code with proper exception handling
- Meaningful error messages instead of program crashes
- Successful execution for all valid inputs

Code with error:



The screenshot shows a Jupyter Notebook titled "Untitled38.ipynb". The code cell contains the following Python code:

```
numbers = [10, 5, 0, 20]

for n in numbers:
    print(100 / n)
```

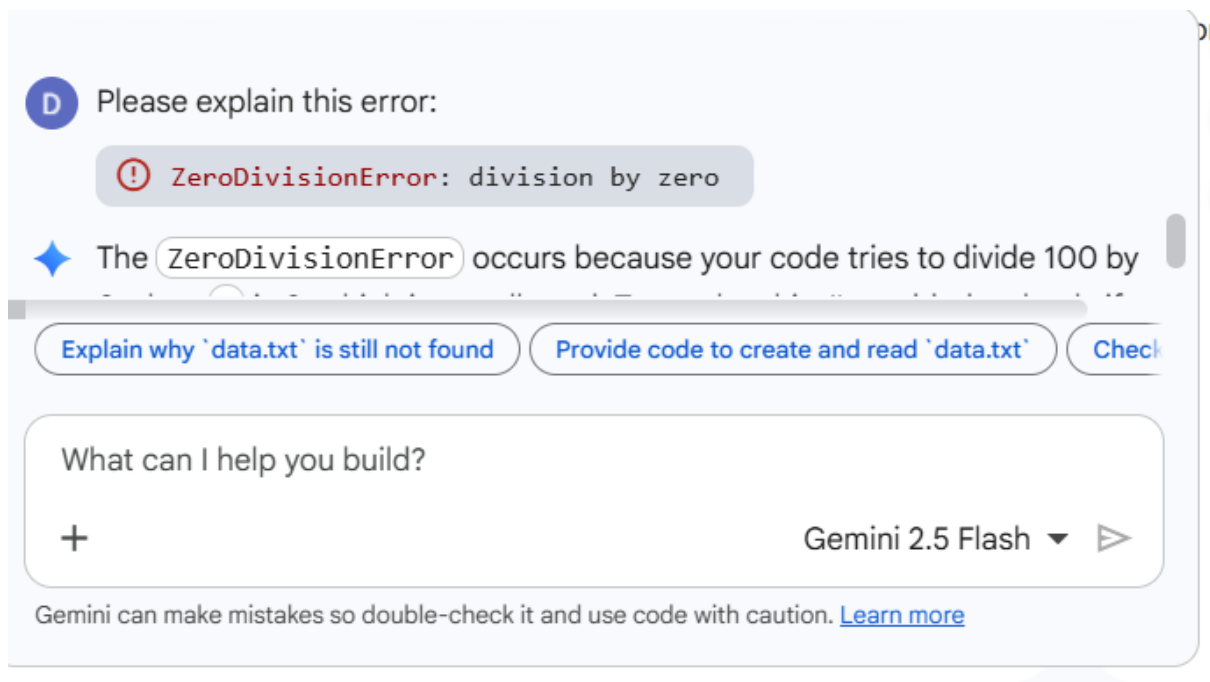
The output shows the first two divisions: 10.0 and 20.0. Then, a `ZeroDivisionError` is raised because of the division by zero. The traceback shows the error occurred in the cell at line 0, column 4.

```
ZeroDivisionError                                Traceback (most recent call last)
/tmp/ipython-input-1634850788.py in <cell line: 0>()
      2
      3 for n in numbers:
----> 4     print(100 / n)

ZeroDivisionError: division by zero
```

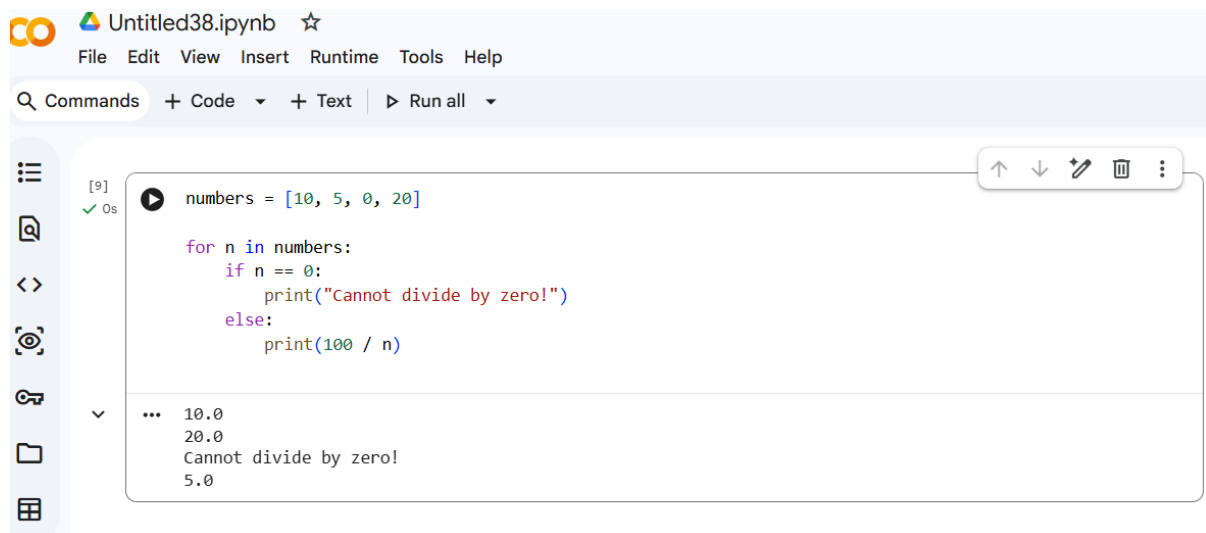
Below the error, there is a "Next steps" section with a button labeled "Explain error".

Explanation:



The screenshot shows a chat interface with a Gemini 2.5 Flash model. The user asks: "Please explain this error:". The model responds with the error message: `ZeroDivisionError: division by zero`. The user then asks: "The `ZeroDivisionError` occurs because your code tries to divide 100 by". Below the chat, there are three buttons: "Explain why 'data.txt' is still not found", "Provide code to create and read 'data.txt'", and "Check". At the bottom, there is a text input field with the placeholder "What can I help you build?", a plus sign icon, and the model name "Gemini 2.5 Flash" with a play button icon. A disclaimer at the bottom states: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

Actual code:



```
numbers = [10, 5, 0, 20]

for n in numbers:
    if n == 0:
        print("Cannot divide by zero!")
    else:
        print(100 / n)
```

```
... 10.0
     20.0
     Cannot divide by zero!
     5.0
```

Task 5: Debugging Class Initialization Errors Scenario

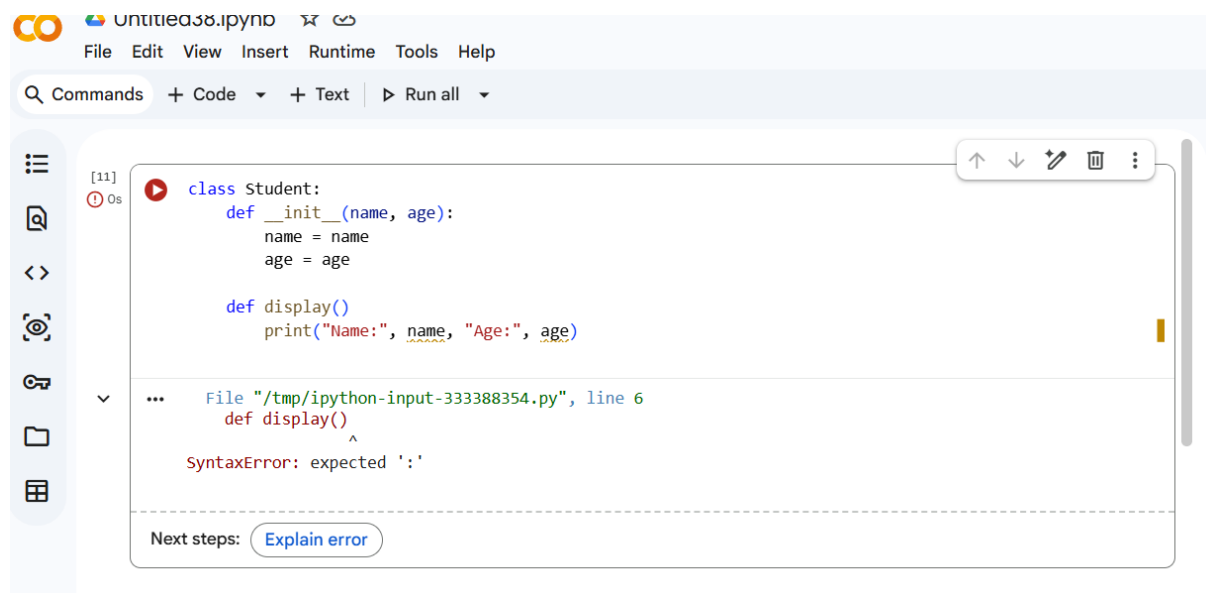
A class written by a junior developer is throwing unexpected errors when objects are created or attributes are accessed.

Task Description

You are given a Python class with:

- Incorrect `__init__` parameters

Code with error:



```
class Student:
    def __init__(name, age):
        name = name
        age = age


    def display()
        print("Name:", name, "Age:", age)
```

```
... File "/tmp/ipython-input-333388354.py", line 6
      def display()
        ^
SyntaxError: expected ':'
```

Next steps: [Explain error](#)

Explanation:

D Please explain this error:

 **ZeroDivisionError:** division by zero

◆ The `ZeroDivisionError` occurs because your code tries to divide 100 by 0 when `n` is 0, which is not allowed. To resolve this, I've added a check: if

[Explain why 'data.txt' is still not found](#) [Provide code to create and read 'data.txt'](#) [Check](#)

What can I help you build?

+ Gemini 2.5 Flash ▶

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Actual code:

```
Untitled38.ipynb ☆
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all
[12]
✓ Os
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print("Name:", self.name, "Age:", self.age)
```