

Lab Assignment-7.4

Hall.No: 2303A51873

Name: R.Vineesha

Batch: 14

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 "Untitled109.ipynb". The code cell contains the following Python code:

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

    num = int(input("Enter a number: "))
    print("Factorial is:", factorial(num))
```

The error message displayed is:

```
... File "/tmp/ipython-input-3913272354.py", line 1
      def factorial(n)
          ^
SyntaxError: expected ':'
```

Next steps: [Explain error](#)

Explanation:

AI identified that the syntax error was caused by a missing colon in the function definition. After fixing the syntax and ensuring a proper base condition, the recursive function executed correctly without errors.

The screenshot shows the same Jupyter Notebook interface, but with a Gemini AI chat window open on the right. The chat window contains the following text:

Please explain this error:

SyntaxError: expected ':'

The error `SyntaxError: expected ':'` means that a colon is missing at the end of the `def factorial(n)` line. Python requires a colon after the function parameters in a function definition.

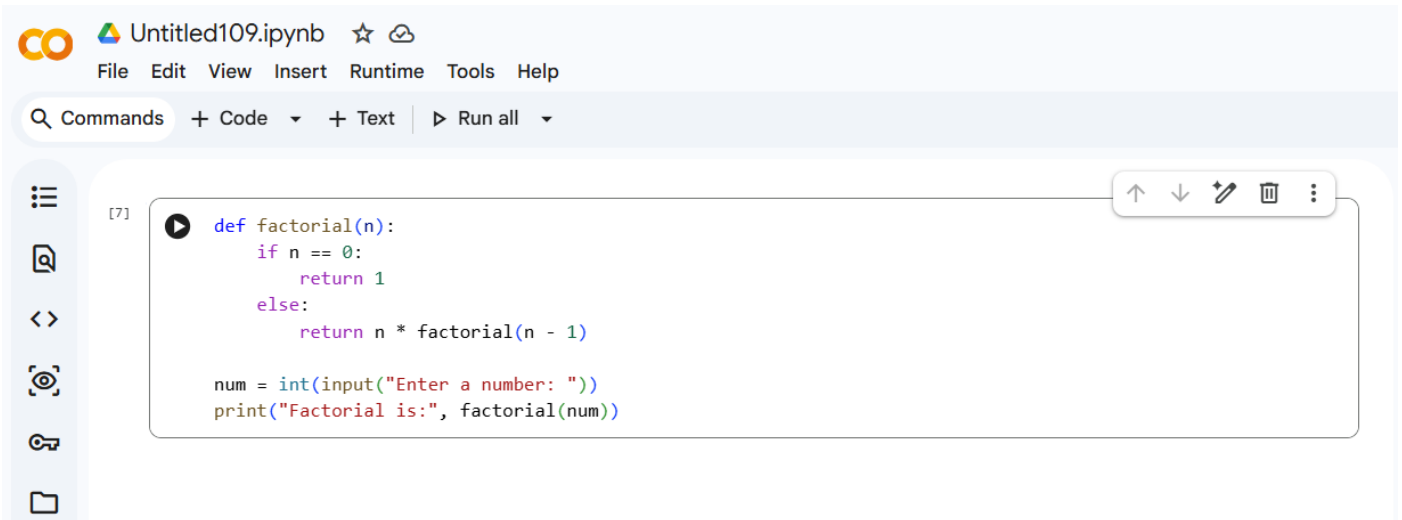
Next steps: [Explain error](#)

The Gemini AI chat window shows a detailed explanation of the syntax error and a corrected code snippet. The text reads:

◆ The error `SyntaxError: expected ':'` means that a colon is missing at the end of the `def factorial(n)` line. Python requires a colon after the function parameters in a function definition.

It looks like the cell executed, but produced no output. This often happens when the main execution logic is not at the top level. Currently, the `input()` and `print()` lines are indented inside the `factorial` function, which means they are never called. Also, the `factorial` function itself needs a base case (e.g., `if n == 0: return 1`) to prevent infinite recursion. I'll fix these issues by adjusting the indentation and adding the base case to the `factorial` function.

Corrected Code (AI-Suggested Solution)



The screenshot shows a Jupyter Notebook titled 'Untitled109.ipynb'. The interface includes a top bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus. Below the menu bar is a search bar and tabs for 'Commands', '+ Code', '+ Text', and 'Run all'. On the left side, there is a sidebar with icons for file explorer, search, and other notebook functions. The main area displays a code cell with the following Python code:

```
[7] def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n - 1)  
  
num = int(input("Enter a number: "))  
print("Factorial is:", factorial(num))
```

Explanation:

The error was resolved by correcting the function syntax and maintaining proper recursive structure. The program now takes user input and produces the correct factorial output without crashing.

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 titled "Untitled109.ipynb". The code cell contains the following Python code:

```
[8] values = [10, "5", 3, "20", 7]
     values.sort()
     print(values)
```

The code has been executed, resulting in a `TypeError`. The traceback shows the error occurred in the `values.sort()` call. The error message is: `TypeError: '<' not supported between instances of 'str' and 'int'`.

Next steps: [Explain error](#)

Explanation:

AI identified that the runtime error occurs because the list contains mixed data types. Python requires all elements to be of the same data type for sorting operations.

The screenshot shows the same Jupyter Notebook interface, but with a Gemini AI chat window open on the right. The code cell in the notebook now includes a correction to the original code:

```
[8] values = [10, "5", 3, "20", 7]
     values = [10, int("5"), 3, int("20"), 7]
     values.sort()
     print(values)
```

The Gemini chat window displays the following response:

Please explain this error:

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

The error is a `TypeError` because you're trying to sort a list containing both strings and integers. Python doesn't know how to compare a string directly with an integer. To fix this, I'll convert all the string numbers in the `values` list to integers, so the `sort()` method can work as expected.

Accept & Run Accept Cancel

What can I help you build?

Gemini 2.5 Flash

R Please explain this error:

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

◆ The error is a `TypeError` because you're trying to sort a list containing both strings and integers. Python doesn't know how to compare a string directly with an integer. To fix this, I'll convert all the string numbers in the `values` list to integers, so the `sort()` method can work as expected.



Corrected Code (AI-Suggested Solution)

The screenshot shows a Jupyter Notebook titled 'Untitled109.ipynb'. The code cell contains the following Python code:

```
values = [10, int("5"), 3, int("20"), 7]
values.sort()
print(values)
```

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

```
[3, 5, 7, 10, 20]
```

Explanation:

The error was fixed by converting string elements into integers. This ensured consistent data types in the list, allowing the sorting function to work correctly without runtime errors.

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:

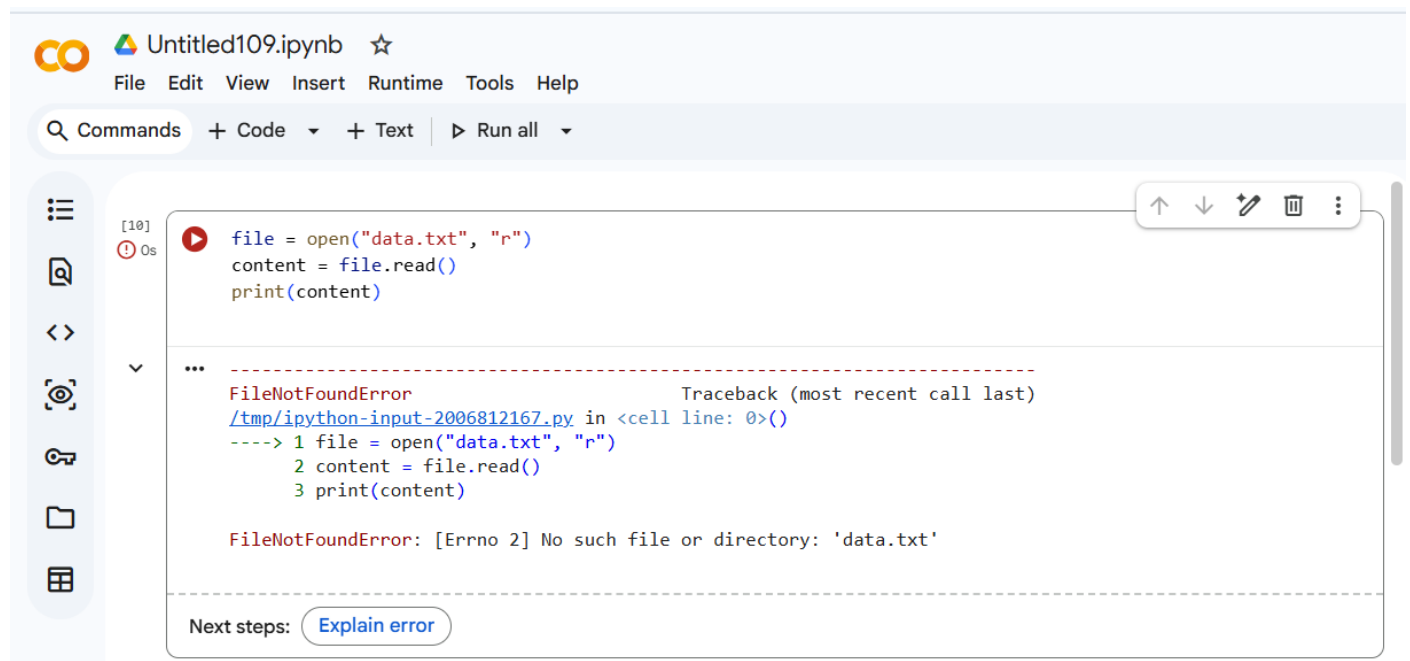
- Identify the potential problem in the code
- Refactor it using best practices (such as a context manager)
- Ensure safe and reliable file handling

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 code cell contains the following Python code:

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

The code cell is followed by a traceback showing a `FileNotFoundError` error. The traceback indicates that the file `data.txt` does not exist in the current directory.

```
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 code reads data from a file but does not close the file after use. This may cause resource management issues and is not considered a best practice in Python.

Explanation:

The corrected code uses a context manager to safely handle file operations. This ensures automatic file closure and prevents resource leaks, making the program more reliable and efficient.

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:

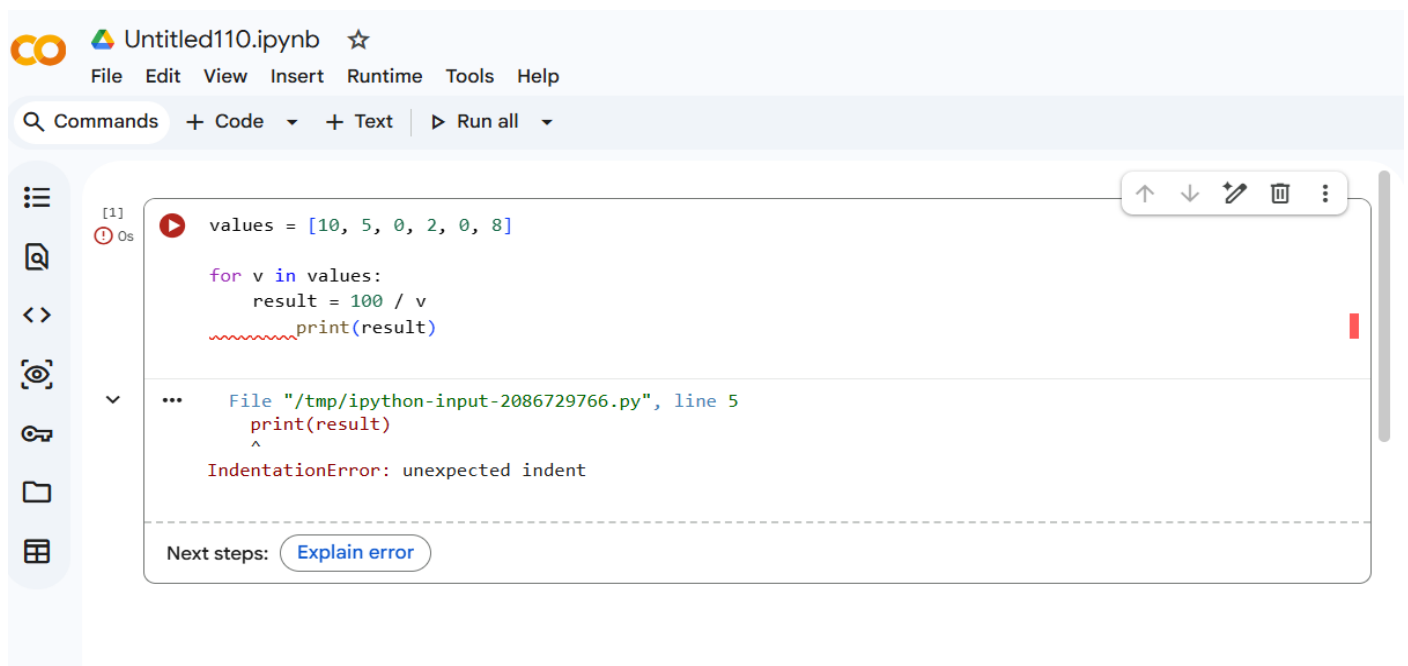
- Detect the exact location of the error
- Add appropriate exception handling using `try-except`
- Ensure the loop continues executing safely

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




Explanation:

The error occurs in the line `result = 100 / v` when the loop reaches a value of 0 in the list. At that moment, the program tries to divide 100 by 0, which is not allowed in Python. This causes a `ZeroDivisionError`, and as a result, the program stops executing immediately. Because of this crash, the remaining values in the list are not processed.

Gemini



R Please explain this error:

 File `"/tmp/ipython-input-2086729766.py"`, line 5 `print(result`

◆ The error you encountered was an `IndentationError: unexpected indent`. This means that the line `print(result)` was indented too much compared to the code block it belonged to. Python uses indentation to define code structure, so incorrect indentation leads to this type of error. Your recent edit to the cell has successfully fixed this by correctly indenting the code within the `if` and `else` blocks, and also added a check to prevent division by zero, which is a great improvement!

▶ Accept & Run ✓ Accept ✕ Cancel

Corrected Code (AI-Suggested Solution)



Untitled110.ipynb ☆

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all



[2]
✓ 0s

◆ Gemini

```
values = [10, 5, 0, 2, 0, 8]

for v in values:
    if v != 0:
        result = 100 / v
        print(result)
    else:
        print(f"Skipping division by zero for value: {v}")
```

...
10.0
20.0
Skipping division by zero for value: 0
50.0
Skipping division by zero for value: 0
12.5

Explanation:

After fixing the error, the code uses a try-except block to handle division by zero safely. When the program divides 100 by a valid number, the result is printed normally. If the value is 0, the ZeroDivisionError is caught by the except block, and a meaningful error message is displayed instead of crashing the program. This allows the loop to continue running and process all remaining values in the list without stopping.

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
- Missing or incorrect attribute references (e.g., missing `self`)

Use GitHub Copilot or Cursor AI to:

- Analyze the class definition
- Identify constructor and attribute issues
- Correct the class so objects initialize and behave correctly

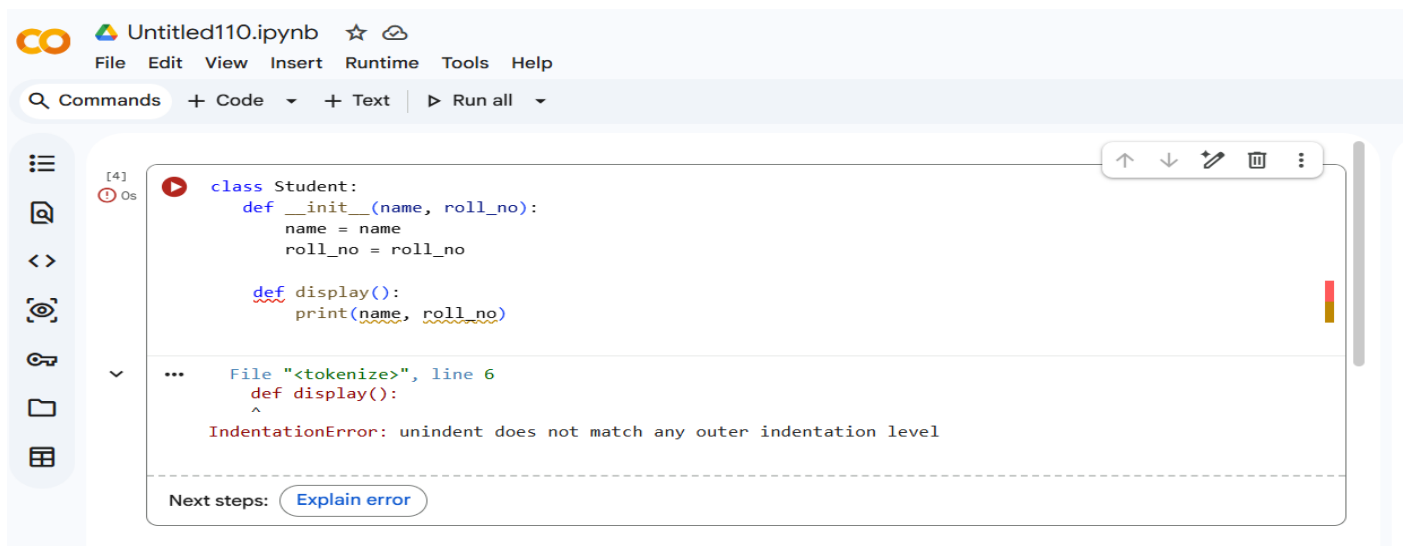
Explain the corrections suggested by the AI.

Expected Outcome

- A corrected class definition
- Proper use of `self` and constructor parameters
- AI-assisted explanation of the original errors and fixes
- Sample object creation and method usage

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

Code With Error



The screenshot shows a Jupyter Notebook window titled "Untitled110.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for commands, code, text, and running. The code cell contains the following Python code:

```
[4]: class Student:
      def __init__(name, roll_no):
          name = name
          roll_no = roll_no

      def display():
          print(name, roll_no)
```

The code is highlighted with syntax coloring. Below the code, an error message is displayed:

```
... File "<tokenize>", line 6
      def display():
      ^
IndentationError: unindent does not match any outer indentation level
```

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

Explanation:

Gemini

Please explain this error:

File "<tokenize>", line 6 def display(): ^ IndentationError:

✦ The main issue is an `IndentationError`. The `display` method is not correctly indented, so Python doesn't recognize it as part of the `Student` class. It needs to be indented to the same level as the `__init__` method. Beyond that, both your `__init__` and `display` methods are missing the `self` parameter, which is crucial for defining instance methods and accessing object-specific attributes in Python classes. I've corrected the indentation and added `self` where necessary, ensuring the `name` and `roll_no` are properly initialized and accessed as instance variables.



```
class Student:
    def __init__(name, roll_no):
        name = name
        roll_no = roll_no
    def __init__(self, name, roll_no):
        self.name = name
        self.roll_no = roll_no
    def display():
        print(name, roll_no)
    def display(self):
        print(self.name, self.roll_no)
```

File "<tokenize>", line 6
def display():
^
IndentationError: unindent does not match any outer indentation level

Next steps: [Explain error](#)

Gemini

Please explain this error:

File "<tokenize>", line 6 def display(): ^ IndentationError:

✦ The main issue is an `IndentationError`. The `display` method is not correctly indented, so Python doesn't recognize it as part of the `Student` class. It needs to be indented to the same level as the `__init__` method. Beyond that, both your `__init__` and `display` methods are missing the `self` parameter, which is crucial for defining instance methods and accessing object-specific attributes in Python classes. I've corrected the indentation and added `self` where necessary, ensuring the `name` and `roll_no` are properly initialized and accessed as instance variables.

[Accept & Run](#) [Accept](#) [Cancel](#)

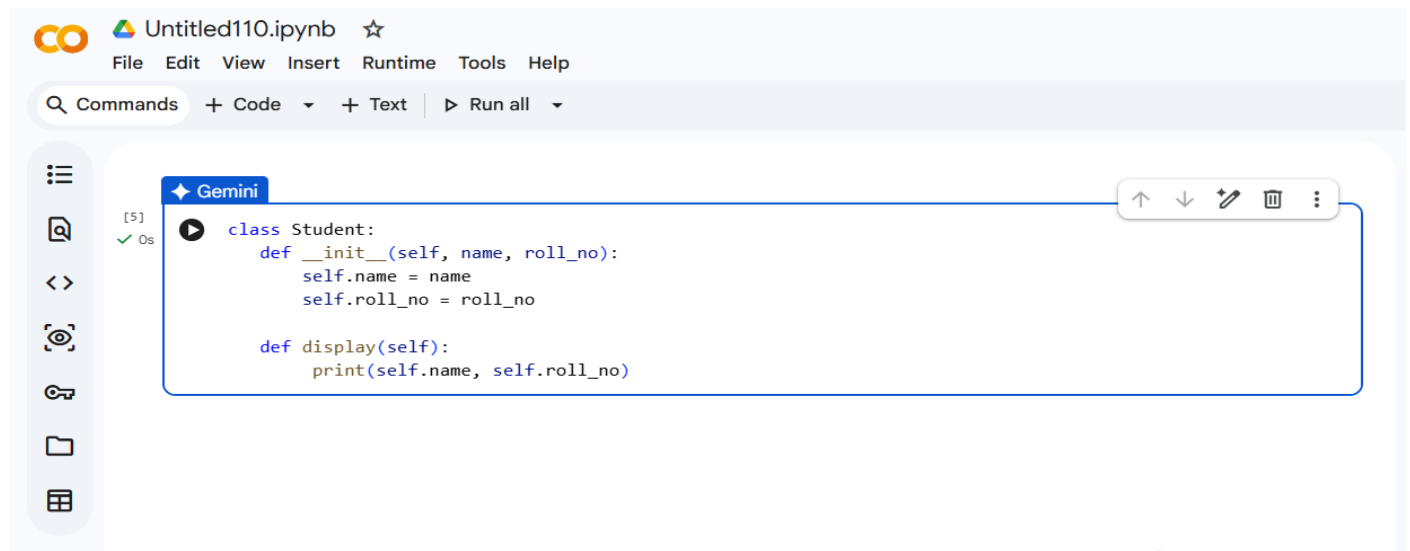
What can I help you build?

[+](#) Gemini 2.5 Flash [▶](#)

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

The original class code contains several mistakes that cause errors during object creation and method execution. In the constructor, the `self` parameter is missing, so Python cannot link the variables to the object being created. The statements `name = name` and `roll_no = roll_no` do not store values inside the object, which means the attributes are never properly initialized. Additionally, the `display` method does not include `self`, so it cannot access object data. Because of these issues, the program throws errors when trying to create an object or access its attributes.

corrected Code (AI-Suggested Solution)



The screenshot shows a Jupyter Notebook interface with the title "Untitled110.ipynb". The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu bar is a search bar labeled "Commands" and buttons for "+ Code", "+ Text", and "Run all". On the left side, there is a sidebar with icons for file operations. The main area displays a code cell with the following Python code:

```
class Student:
    def __init__(self, name, roll_no):
        self.name = name
        self.roll_no = roll_no

    def display(self):
        print(self.name, self.roll_no)
```

The code cell is titled "Gemini" and has a status bar showing "[5]" and "0s".

Explanation:

The corrected code properly uses the `self` keyword to refer to the current object. In the `__init__` constructor, `self.name` and `self.roll_no` are used to store the values passed while creating the object, ensuring that each object has its own data. The `display` method also includes `self`, which allows it to access and print the object's attributes. Because of these corrections, the object initializes correctly, attributes are stored properly, and methods work without errors.

