

# Assignment -7.3

Name: Shiva Shankar

2303A51294

Btach:05

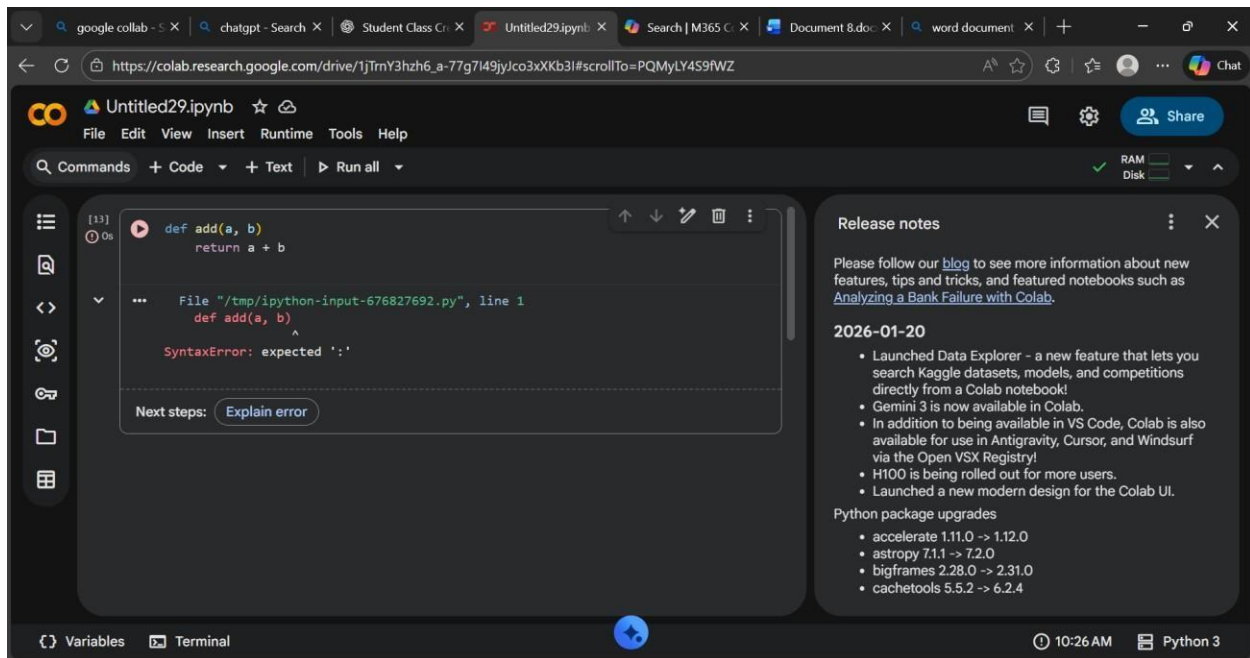
## Ask 1: Fixing Syntax Errors

Prompt: The following Python function has a syntax error. Identify the issue and correct it. Also explain what the syntax error is.

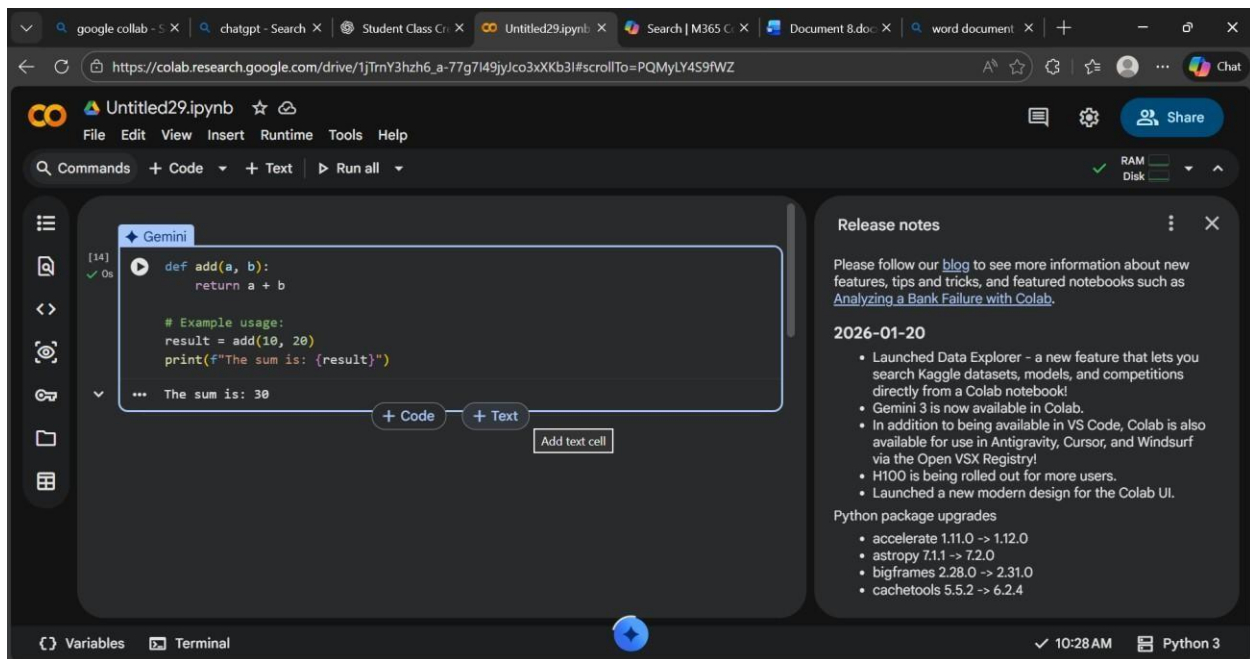
```
def add(a, b)
```

return a + b Input:

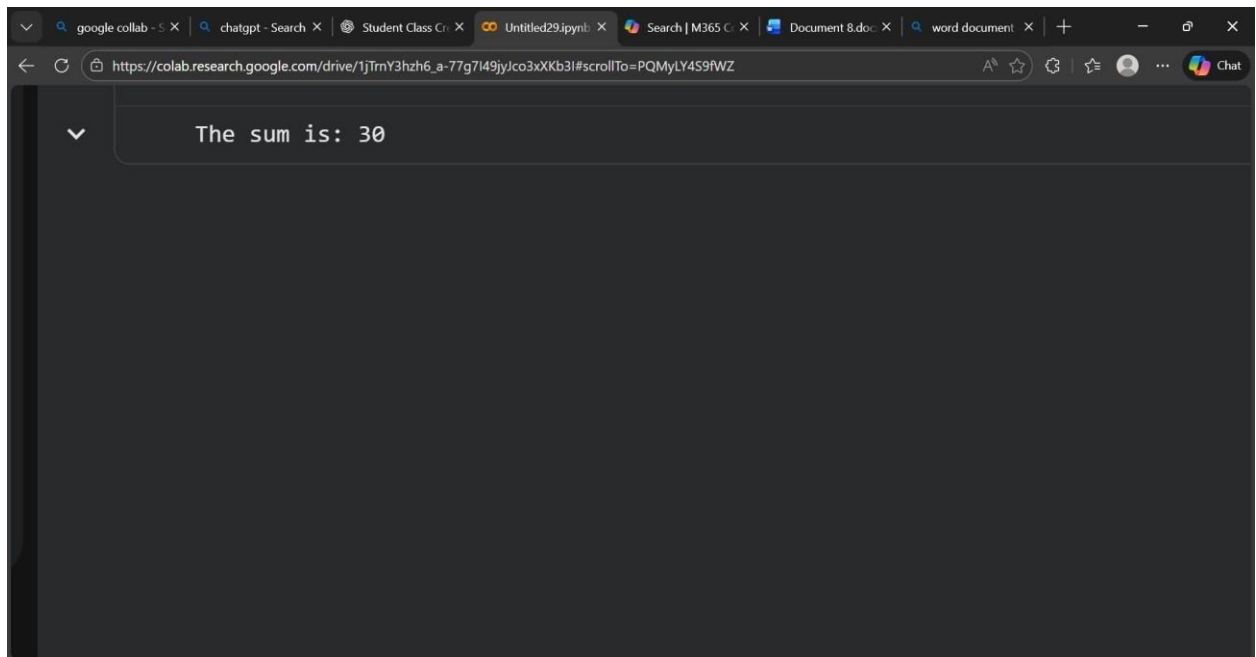
Bug Code:



2) corrected code:



Output:



Explanation:

- In Python, a colon `:` is required after defining a function header.
- Without the colon, Python cannot recognize the start of the function block, causing a **SyntaxError**.
- AI correctly identified the missing colon and fixed the function definition.

## Task 2: Debugging Logic Errors in Loops

Prompt: The following Python loop runs infinitely. Identify the logic error, correct the loop, and explain the issue.

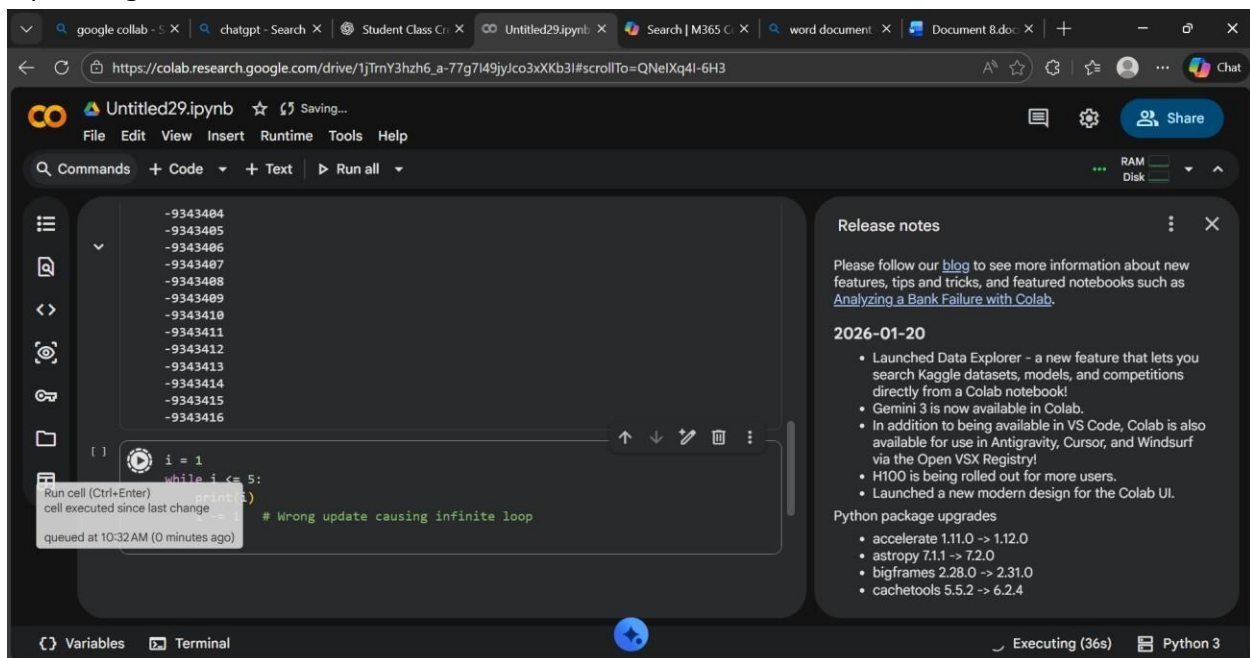
```
i = 1 while i
```

```
<= 5:
```

```
    print(i)
```

```
    i -= 1
```

Input: Bug code:



Corrected code:

The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'google collab', 'chatgpt - Search', 'Student Class Cr...', 'Untitled29.ipynb', 'Search | M365 C...', 'word document', and 'Document 8.doc'. The notebook title is 'Untitled29.ipynb' with a 'Saving...' status. 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'. The code cell [16] contains the following Python code:

```
i = 1
while i <= 5:
    print(i)
    i += 1 # Corrected: increment i instead of decrementing

print("Loop finished.")
```

The output of the code cell shows the numbers 1, 2, and 3, followed by an ellipsis (...).

Output:

The screenshot shows the same Google Colab notebook interface, but the code cell is now empty. The output of the previous code cell is displayed, showing the numbers 1, 2, 3, 4, and 5, followed by the text 'Loop finished.'.

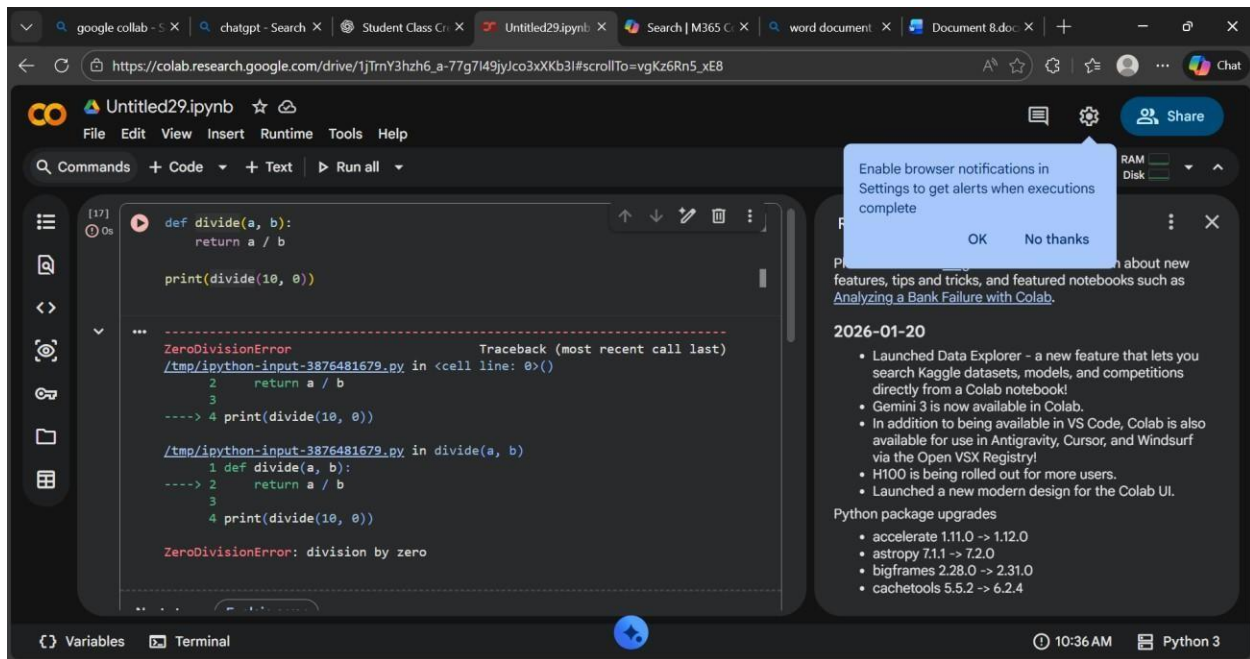
Explanation: The variable `i` was decreasing (`i -= 1`) while the condition required it to increase, causing an infinite loop.

Changing it to `i += 1` allows the loop to reach the stopping condition and terminate correctly.

### Task 3: Handling Runtime Errors (Division by Zero)

Prompt: This Python code causes a runtime error. Identify the problem, fix it using `tryexcept`, and explain the issue. `def divide(a, b): return a / b print(divide(10, 0))`

## Input:Bug Code

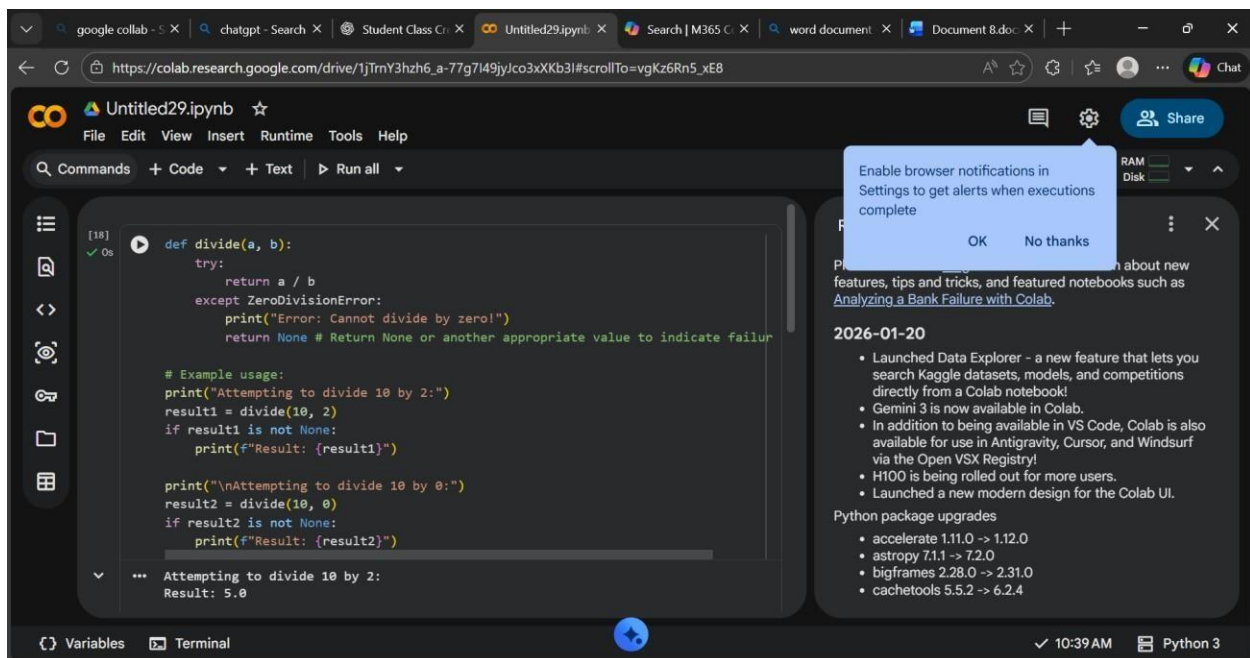


The screenshot shows a Google Colab notebook titled 'Untitled29.ipynb'. The code in the cell is as follows:

```
[17] def divide(a, b):  
      return a / b  
  
      print(divide(10, 0))  
  
      ...  
      -----  
      ZeroDivisionError                                Traceback (most recent call last)  
      /tmp/ipython-input-3876481679.py in <cell line: 0>()  
          2     return a / b  
          3  
      ----> 4 print(divide(10, 0))  
  
      /tmp/ipython-input-3876481679.py in divide(a, b)  
          1 def divide(a, b):  
      ----> 2     return a / b  
          3  
          4 print(divide(10, 0))  
  
      ZeroDivisionError: division by zero
```

The error message 'ZeroDivisionError: division by zero' is displayed at the bottom of the cell. A notification bubble in the top right corner says 'Enable browser notifications in Settings to get alerts when executions complete' with 'OK' and 'No thanks' buttons. The right sidebar shows a list of updates for January 20, 2026, including 'Launched Data Explorer', 'Gemini 3 is now available in Colab', and 'Python package upgrades'.

## Corrected Code:

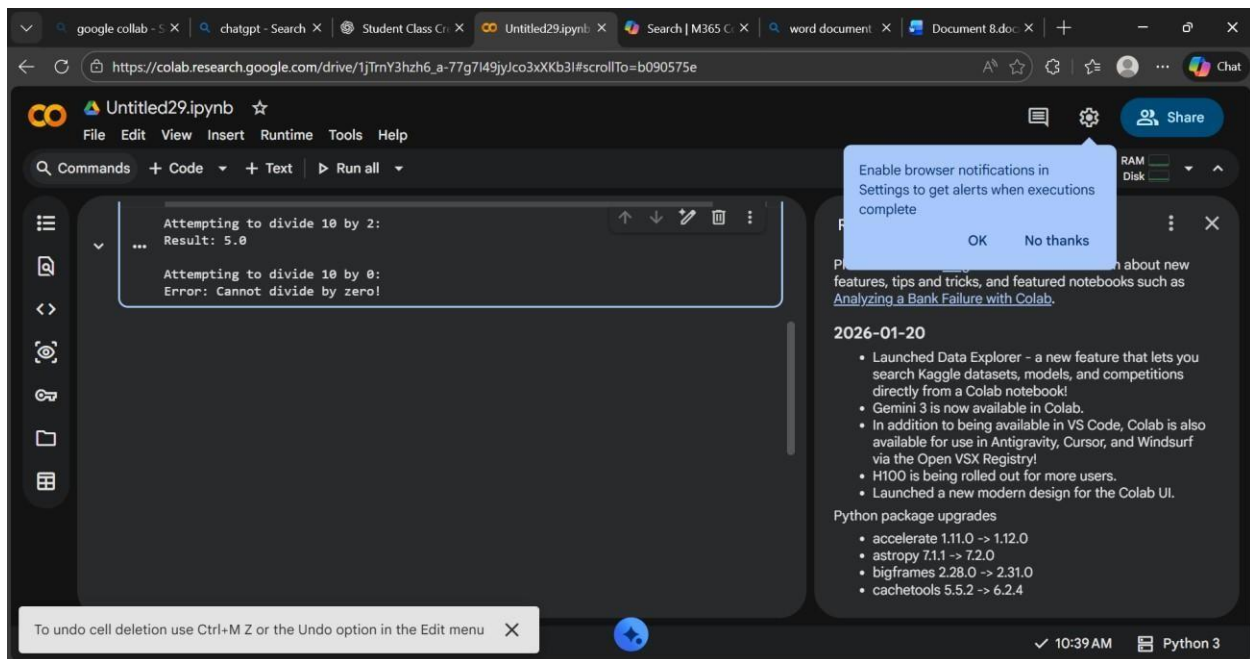


The screenshot shows the same Google Colab notebook with the corrected code. The code in the cell is as follows:

```
[18] def divide(a, b):  
      try:  
          return a / b  
      except ZeroDivisionError:  
          print("Error: Cannot divide by zero!")  
          return None # Return None or another appropriate value to indicate failure  
  
      # Example usage:  
      print("Attempting to divide 10 by 2:")  
      result1 = divide(10, 2)  
      if result1 is not None:  
          print(f"Result: {result1}")  
  
      print("\nAttempting to divide 10 by 0:")  
      result2 = divide(10, 0)  
      if result2 is not None:  
          print(f"Result: {result2}")  
  
      ...  
      Attempting to divide 10 by 2:  
      Result: 5.0
```

The code now handles the ZeroDivisionError by printing an error message and returning None. The output shows the result of dividing 10 by 2 as 5.0. The notification bubble and the right sidebar are the same as in the previous screenshot.

## Output:



Explanation: the program crashes because division by zero is not allowed in Python, causing a `ZeroDivisionError`.

Using `try-except` prevents the crash and safely handles the error.

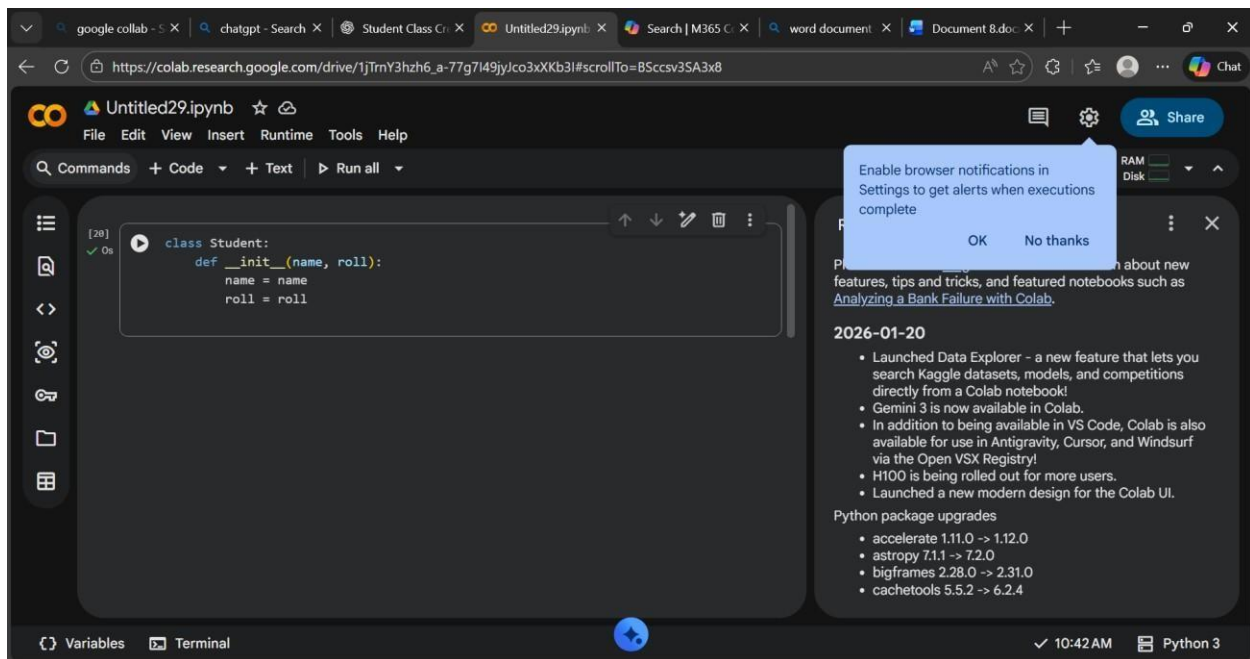
## Task 4: Debugging Class Definition Errors

Prompt: The following Python class has an error in the constructor. Identify the issue, correct the class definition, and explain why the fix is needed.

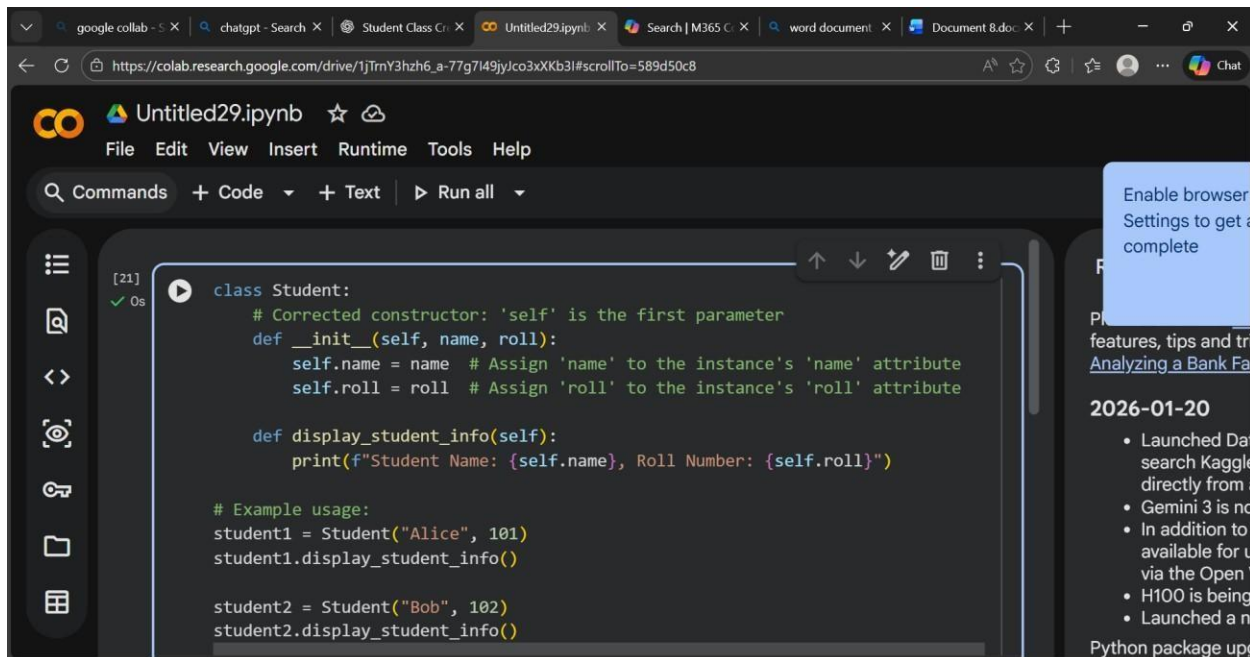
```
class Student: def init(name, roll): name = name roll = roll
```

Input: Bug Code

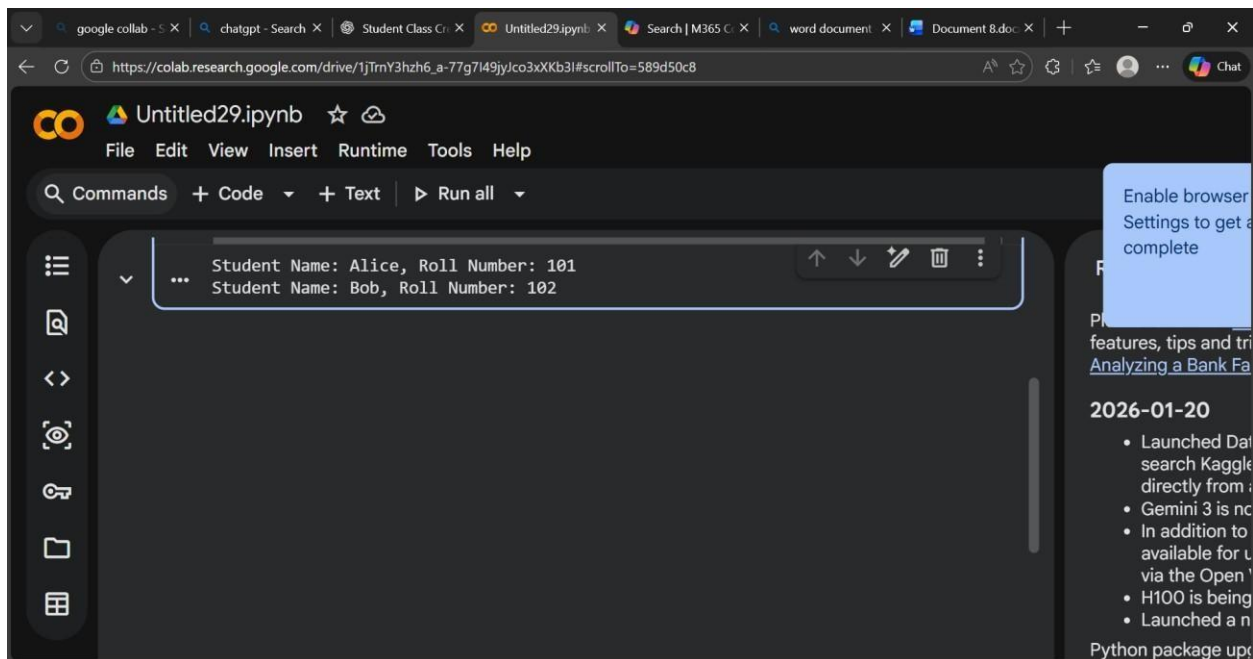




Corrected code:



Output:



Explanation: The constructor was missing the `self` parameter, which is required to refer to the object instance.

Using `self.name` and `self.roll` stores values inside the object properly. Task 5:

## Resolving Index Errors in Lists

Prompt: This Python code causes an `IndexError`. Identify the issue, correct the code using safe access methods, and explain the problem.

```
numbers = [10, 20, 30]
```

```
(numbers[5])
```

Input: Bug code



Enable browser notifications in Settings to get alerts when executions complete

OK No thanks

```
numbers = [10, 20, 30]
print(numbers[5]) # Invalid index
```

Traceback (most recent call last)

```
/tmp/ipython-input-1643708134.py in <cell line: 0>()
1 numbers = [10, 20, 30]
2
----> 3 print(numbers[5]) # Invalid index
```

IndexError: list index out of range

Next steps: [Explain error](#)

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4

Corrected Code:

```
numbers = [10, 20, 30]

# Attempt to access an element safely using try-except
try:
    print(f"Attempting to access index 5: {numbers[5]}")
except IndexError:
    print("Error: Index out of bounds! The list does not have an element at this index.")

# Example of valid access:
print(f"\nAccessing a valid index (index 0): {numbers[0]}")
print(f"Accessing a valid index (index 2): {numbers[2]}")
```

Output:

The screenshot shows a Google Colab notebook interface. At the top, there's a browser address bar with the URL [https://colab.research.google.com/drive/1jTmY3hzh6\\_a-77g7l49jylco3xXKb3l#scrollTo=b6800fc5](https://colab.research.google.com/drive/1jTmY3hzh6_a-77g7l49jylco3xXKb3l#scrollTo=b6800fc5). Below the address bar, there's a toolbar with buttons for '+ Code', '+ Text', and 'Run all'. The main area of the notebook displays two lines of output in a light blue box. The first line is 'Accessing a valid index (index 0): 10' and the second line is 'Accessing a valid index (index 2): 30'. To the right of the output box, there are icons for up, down, edit, delete, and a menu.

Explanation: The program tried to access an index that does not exist in the list, causing an `IndexError`.

Using `len()` to check bounds prevents the program from crashing.