

# Assignment -7.3

Name:K.Hithesh

2303A51291

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:

The screenshot shows the Google Colab interface. On the left, there's a sidebar with various icons. In the center, a code cell contains the following Python code:

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

Below the code, an error message is displayed:

```
File "/tmp/ipython-input-676827692.py", line 1
    def add(a, b)
               ^
SyntaxError: expected ':'
```

At the bottom of the code cell, there's a button labeled "Explain error". To the right of the code cell, there's a "Release notes" panel. The "2026-01-20" section lists several updates, including the launch of Data Explorer and the availability of Gemini 3 in Colab. The "Python package upgrades" section lists three packages: accelerate, astropy, and bioframes, all updated from version 1.x to 2.x.

2) corrected code:

The screenshot shows a Colab notebook interface. On the left, there's a sidebar with various icons for file operations. The main workspace contains a code cell titled "Gemini". The code defines a function `add(a, b)` that returns the sum of `a` and `b`. It includes an example usage where `add(10, 20)` prints "The sum is: 30". Below the code cell are buttons for "+ Code", "+ Text", and "Add text cell". The status bar at the bottom shows "Variables" and "Terminal" icons, along with the date "✓ 10:28 AM" and "Python 3". On the right, a "Release notes" sidebar is open, dated "2026-01-20", which includes a link to a blog post about new features like Data Explorer and Gemini 3.

Output:

The screenshot shows the output cell from the previous screenshot. It displays the text "The sum is: 30" in a monospaced font, indicating the execution result of the code cell.

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. On the left, a sidebar displays a list of recent notebooks with IDs: -9343404, -9343405, -9343406, -9343407, -9343408, -9343409, -9343410, -9343411, -9343412, -9343413, -9343414, -9343415, and -9343416. The main workspace contains a single code cell with the following content:

```
i = 1
while i <= 5:
    print(i)
    i -= 1
```

A tooltip above the cell indicates: "Run cell (Ctrl+Enter) cell executed since last change # Wrong update causing infinite loop queued at 10:32 AM (0 minutes ago)". To the right of the workspace is a "Release notes" panel for January 2026, which includes a list of changes such as launching Data Explorer and Gemini 3, and upgrading various Python packages like accelerate, astropy, bigframes, and cachetools.

A screenshot of a Jupyter Notebook interface. The top bar has buttons for Commands, Code, Text, Run all, and a search bar. On the left is a sidebar with icons for file operations, a refresh, a question mark, and a key. A code cell is open with the following content:

```
[16] ✓ 0s
i = 1
while i <= 5:
    print(i)
    i += 1 # Corrected: increment i instead of decrementing

print("Loop finished.")

... 1
2
3
```

Output:

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

A screenshot of a Jupyter Notebook interface. The sidebar icons are visible on the left. A code cell is open with the following content:

```
print( Loop finished. )

... 1
2
3
4
5
Loop finished.
```

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". In cell [17], there is a function definition:

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

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

When run, it results in a `ZeroDivisionError`:

```
ZeroDivisionError: division by zero
```

A tooltip message is displayed in the top right corner:

Enable browser notifications in Settings to get alerts when executions complete

OK No thanks

Below the code cell, a sidebar displays a news item and package upgrades:

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

Variables Terminal 10:36 AM Python 3

## Corrected Code:

The screenshot shows the same Google Colab notebook. The corrected function definition is:

```
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}")
```

The output shows the function handling the division by zero case:

```
Attempting to divide 10 by 2:
Result: 5.0

Attempting to divide 10 by 0:
Error: Cannot divide by zero!
```

A tooltip message is displayed in the top right corner:

Enable browser notifications in Settings to get alerts when executions complete

OK No thanks

Below the code cell, a sidebar displays a news item and package upgrades:

2026-01-20

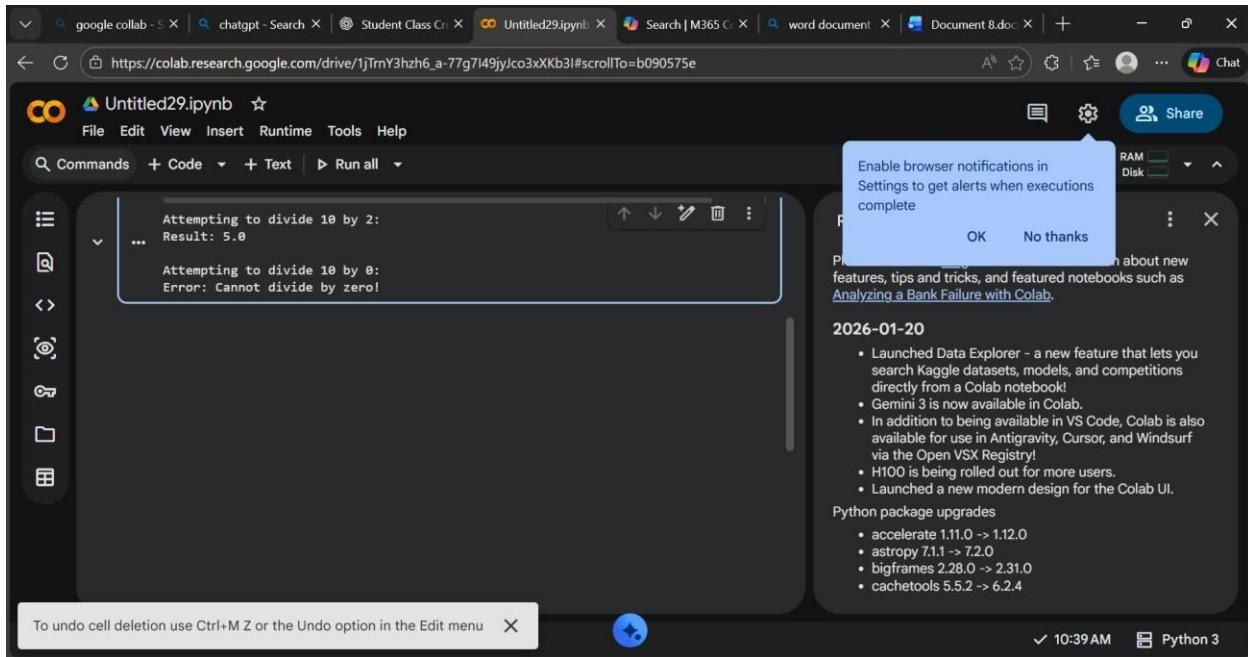
- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open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

Variables Terminal 10:39 AM Python 3

## 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

A screenshot of a web browser window displaying a Google Colab notebook titled "Untitled29.ipynb". The notebook contains the following Python code:

```
[28] class Student:
    def __init__(name, roll):
        name = name
        roll = roll
```

The code cell has a green checkmark icon and the text "0s" next to it, indicating it has run successfully. A small blue circular icon with a play button is positioned to the left of the code. On the right side of the screen, there is a floating blue notification box with the following text:

Enable browser notifications in Settings to get alerts when executions complete

OK No thanks

Please enable browser notifications to receive updates about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

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

Variables Terminal ✓ 10:42 AM Python 3

Corrected code:

A screenshot of a web browser window displaying a Google Colab notebook titled "Untitled29.ipynb". The notebook contains the following Python code:

```
[21] class Student:
    # Corrected constructor: 'self' is the first parameter
    def __init__(self, name, roll):
        self.name = name # Assign 'name' to the instance's 'name' attribute
        self.roll = roll # Assign 'roll' to the instance's 'roll' attribute

    def display_student_info(self):
        print(f"Student Name: {self.name}, Roll Number: {self.roll}")

    # Example usage:
    student1 = Student("Alice", 101)
    student1.display_student_info()

    student2 = Student("Bob", 102)
    student2.display_student_info()
```

The code cell has a green checkmark icon and the text "0s" next to it, indicating it has run successfully. A small blue circular icon with a play button is positioned to the left of the code. On the right side of the screen, there is a floating blue notification box with the following text:

Enable browser notifications in Settings to get alerts when executions complete

OK No thanks

Please enable browser notifications to receive updates about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

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

Output:

```
Student Name: Alice, Roll Number: 101
Student Name: Bob, Roll Number: 102
```

Enable browser  
Settings to get a  
complete  
Pl  
features, tips and tri  
Analyzing a Bank Fa  
2026-01-20

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

The screenshot shows a Google Colab interface with multiple tabs at the top. The active tab is 'Untitled29.ipynb'. In cell 22, there is an error message:

```
[22] 0s
numbers = [10, 20, 30]
print(numbers[5]) # Invalid index

...
IndexError: list index out of range
```

A tooltip is displayed over the error message, providing information about browser notifications and execution alerts.

On the right side of the screen, there is a sidebar with a 'Data Explorer' section showing RAM and Disk usage, and a 'Recent' section listing various notebooks and documents. A notification bar at the bottom right indicates '2026-01-20' and lists several updates, including the launch of Data Explorer and Python package upgrades.

Corrected Code:

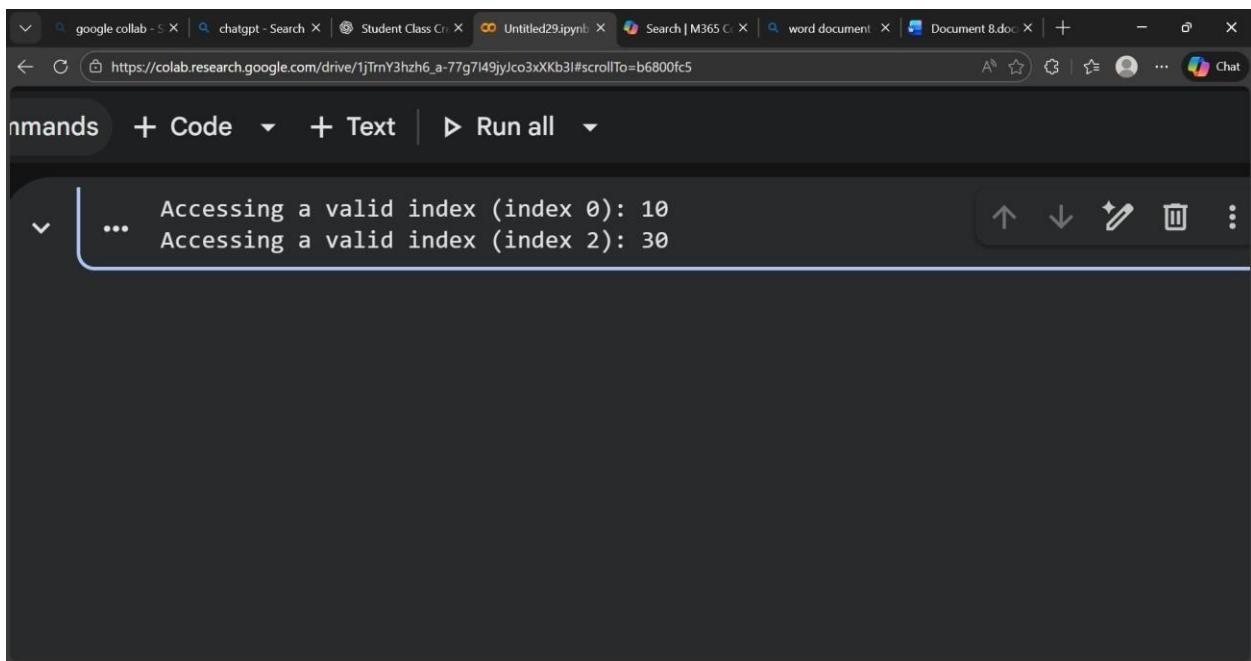
The screenshot shows a Google Colab interface with multiple tabs at the top. The active tab is 'Untitled29.ipynb'. In cell 23, the code has been corrected to handle index errors safely:

```
[23] 0s
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 Jupyter Notebook interface in Google Colab. At the top, there are several tabs: "google collab - S X", "chatgpt - Search X", "Student Class Cr X", "Untitled29.ipynb X", "Search | M365 C X", "word document X", "Document 8.doc X", and a "+" button. Below the tabs, the URL is https://colab.research.google.com/drive/1jTrnY3hzh6\_a-77g7I49yJco3xXKb3I#scrollTo=b6800fc5. The toolbar includes "Commands", "+ Code", "+ Text", and "Run all". The main code cell contains the following Python code:

```
Accessing a valid index (index 0): 10
...
Accessing a valid index (index 2): 30
```

Below the code cell are standard Jupyter Notebook controls: up, down, edit, delete, and more.

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.