

# 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:

The screenshot shows a Google Colab notebook titled "Untitled29.ipynb". In cell [13], there is a syntax error in the definition of the 'add' function. The code is as follows:

```
[13] ⚡ 0s
def add(a, b)
    return a + b

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

A tooltip "Next steps: Explain error" is visible below the code. To the right of the notebook area, there is a "Release notes" sidebar. It includes a link to a blog post about new features and a section for "2026-01-20" with a bulleted list of changes. At the bottom of the sidebar, there is a "Python package upgrades" section with a list of packages and their versions.

2) corrected code:

The screenshot shows the Google Colab interface. On the left, a code cell titled 'Untitled29.ipynb' contains Python code for defining a function 'add' and printing its result. The output shows the function definition and the printed result 'The sum is: 30'. On the right, a 'Release notes' sidebar is open, dated 2026-01-20, which includes a link to a blog post about new features and tips. The sidebar also lists several recent updates, such as the launch of Data Explorer and the availability of Gemini 3 in Colab. At the bottom right, it shows the time as 10:28 AM and the Python version as Python 3.

Output:

The screenshot shows the Google Colab interface with the output of the previous code cell. The output cell displays the text 'The sum is: 30'.

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:

The screenshot shows a Google Colab notebook titled "Untitled29.ipynb". In the code cell, there is an infinite loop where `i` is printed and then decremented by 1. A tooltip from the browser indicates the error: "# Wrong update causing infinite loop". The sidebar shows a list of recent notebooks. On the right, a "Release notes" panel is open, dated 2026-01-20, listing various updates and fixes.

```
i = 1
while i <= 5:
    print(i)
    i -= 1 # Wrong update causing infinite loop
```

Release notes

Please follow our [blog](#) to see more information 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

Corrected code:

The screenshot shows a Google Colab interface. The top bar has tabs for "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. The URL is https://colab.research.google.com/drive/1jTrnY3hzh6\_a-77g7l49jyJco3xXKb3I#scrollTo=GNhQ04ph-wWz. The main area shows a notebook titled "Untitled29.ipynb" with a status bar indicating it's saving. The menu bar includes File, Edit, View, Insert, Runtime, Tools, and Help. Below the menu is a toolbar with search, code, text, and run all buttons. 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 pane shows the results of the execution:

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

Output:

The screenshot shows the same Google Colab interface as the previous one, but the code cell has been executed. The output pane now displays the results of the print statements and the final message:

```
1
2
3
4
5
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". In the code editor, there is a cell containing the following Python code:

```
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".

## Corrected Code:

The screenshot shows the same Google Colab notebook after the code has been corrected. The code now includes a try-except block to handle the `ZeroDivisionError`:

```
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}")

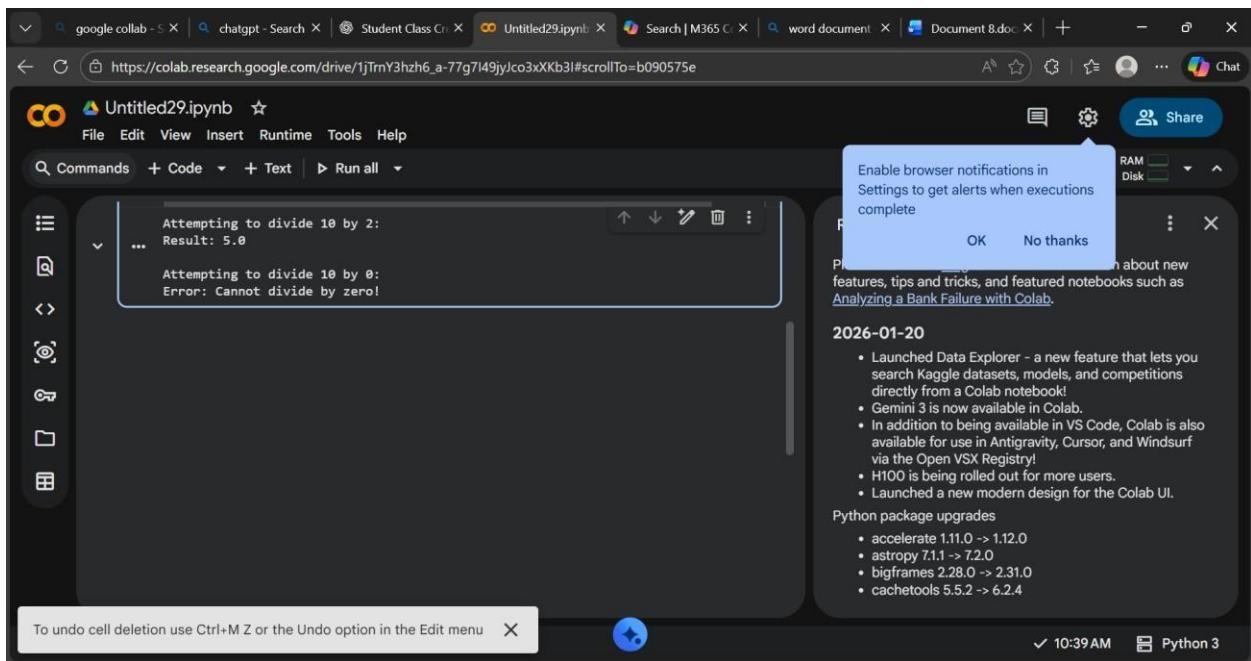
print("\nAttempting to divide 10 by 0:")
result2 = divide(10, 0)
if result2 is not None:
    print(f"Result: {result2}")
```

The output shows that the first division works as expected, but the second division triggers the error handling code:

```
Result: 5.0
```

A tooltip message is displayed in the top right corner: "Enable browser notifications in Settings to get alerts when executions complete".

## 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

Untitled29.ipynb

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

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

RAM Disk

Enable browser notifications in Settings to get alerts when executions complete

OK No thanks

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:

Untitled29.ipynb

```
[21]  Os
  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()
```

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Enable browser notifications in Settings to get alerts when executions complete

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
```

2026-01-20

- Launched Data search Kaggle directly from Colab
- Gemini 3 is now available for users via the Open Assistant
- H100 is being launched
- Launched a new Python package update

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

Untitled29.ipynb

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

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

Next steps: Explain error

Enable browser notifications in Settings to get alerts when executions complete

OK No thanks

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 Python 3 10:47 AM

Corrected Code:

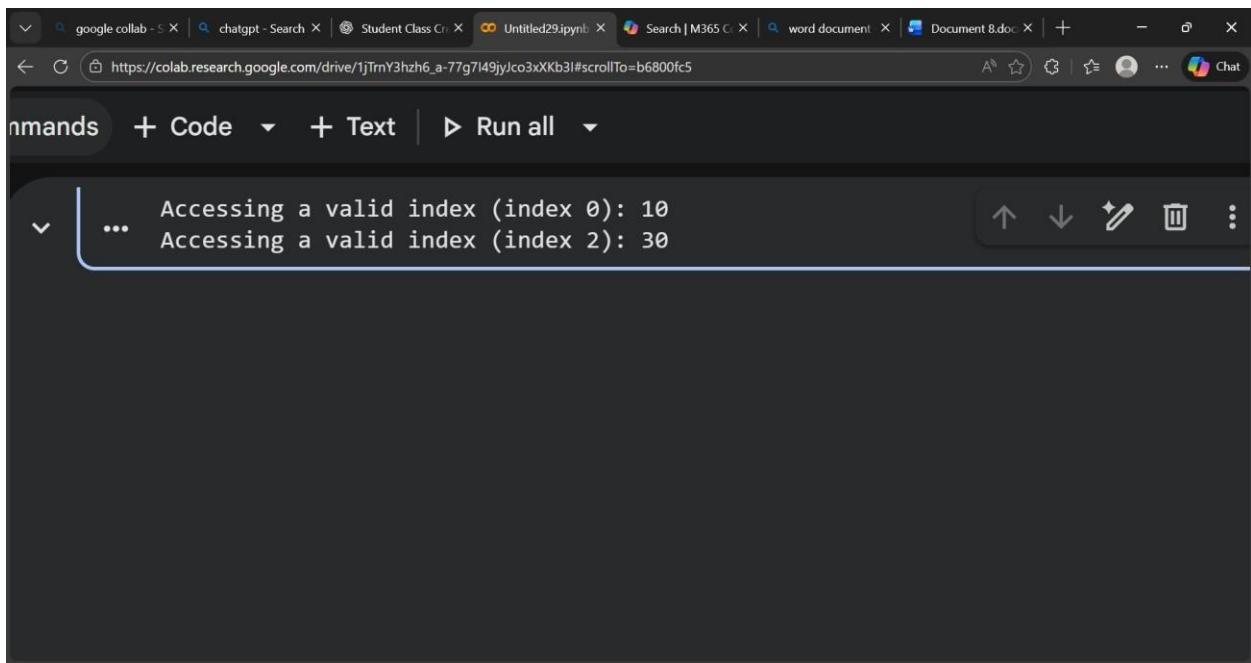
Untitled29.ipynb

```
[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 Google Colab interface with multiple tabs at the top: "google collab - S", "chatgpt - Search", "Student Class Cr", "Untitled29.ipynb", "Search | M365 C", "word document", "Document 8.doc", and a new tab icon. The URL in the address bar is [https://colab.research.google.com/drive/1jTrnY3hzh6\\_a-77g7I49yJco3xXKb3I#scrollTo=b6800fc5](https://colab.research.google.com/drive/1jTrnY3hzh6_a-77g7I49yJco3xXKb3I#scrollTo=b6800fc5). Below the tabs is a toolbar with "Commands", "+ Code", "+ Text", and "Run all". A dropdown menu is open, showing two items: "Accessing a valid index (index 0): 10" and "Accessing a valid index (index 2): 30". To the right of the dropdown are icons for 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.