

Assignment 7.3 Ai Assisted Coding

Htno:2303a51288

Btno:05

Task 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 interface. In the code editor, there is a cell with the following content:

```
[13] In [1] def add(a, b)
      return a + b

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

Below the code, there is a "Next steps:" button with the option "Explain error". To the right of the code editor, there is a "Release notes" sidebar with the following text:

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 1.1.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4

At the bottom of the sidebar, it says "10:26 AM" and "Python 3".

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, detailing various updates like the launch of Data Explorer and Gemini 3 availability. The bottom status bar indicates the time is 10:28 AM and the Python version is Python 3.

```
[14]: def add(a, b):
    return a + b

# Example usage:
result = add(10, 20)
print(f"The sum is: {result}")

... The sum is: 30
```

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

Variables Terminal ✓ 10:28 AM Python 3

Output:

The screenshot shows the Google Colab interface with the output cell expanded. It displays the printed result "The sum is: 30" from the previous code execution.

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 editor, there is a cell containing the following Python code:

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

A tooltip above the code highlights the line `i -= 1` with the note: "# Wrong update causing infinite loop". The cell status bar indicates it was run at 10:32 AM (0 minutes ago). To the right of the code editor, a "Release notes" sidebar is open, showing the date 2026-01-20 and a list of updates. The sidebar also mentions "Python package upgrades" with a list of packages and their versions.

Corrected code:

The screenshot shows a Google Colab notebook titled "Untitled29.ipynb". The code cell contains the following Python script:

```
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 notebook. The output pane displays the results of the previous execution:

```
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

A screenshot of a Google Colab notebook titled "Untitled29.ipynb". The code defines a simple division function:

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

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

The output shows 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:

A screenshot of a Google Colab notebook titled "Untitled29.ipynb". The code now includes exception handling to catch `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}")
```

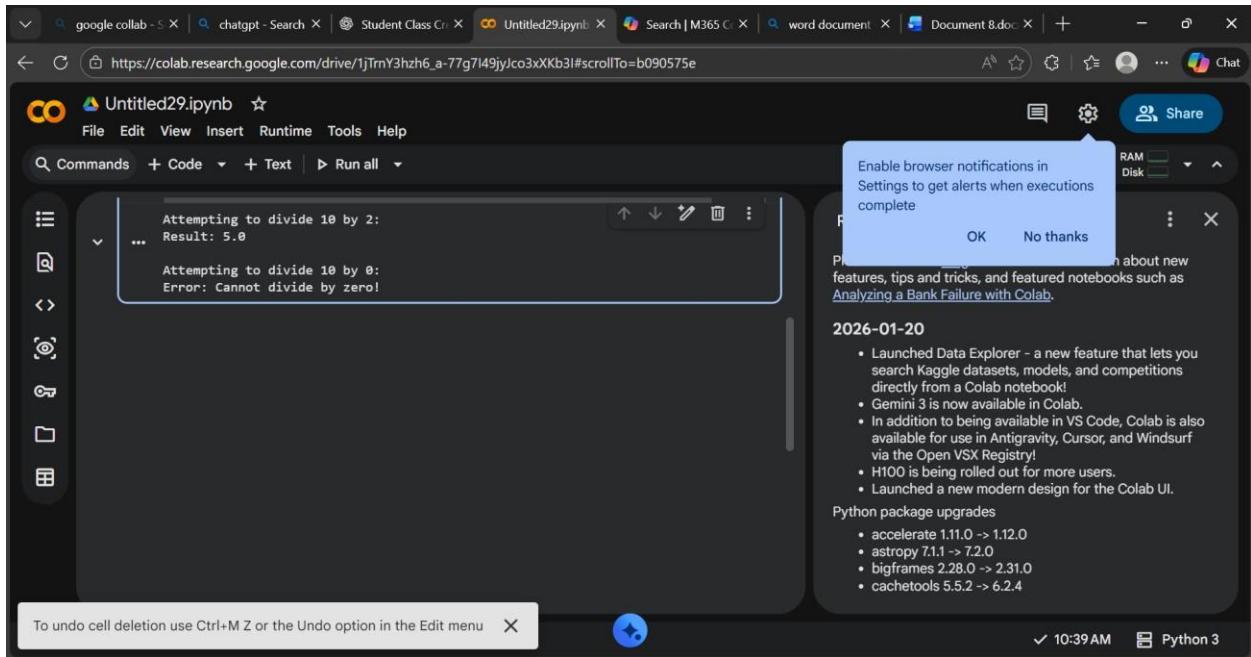
The output shows the results of running the corrected code:

```
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".

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 Google Colab notebook titled "Untitled29.ipynb". The code cell contains:

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

A tooltip is displayed in the top right corner, showing a message about browser notifications and a link to "Analyzing a Bank Failure with Colab".

Corrected code:

A screenshot of a Google Colab notebook titled "Untitled29.ipynb". The code cell now contains the corrected 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 output of the code is shown below the code cell.

Output:

The screenshot shows a Google Colab notebook titled "Untitled29.ipynb". The code cell contains the following Python code:

```
class Student:
    def __init__(self, name, roll):
        self.name = name
        self.roll = roll
    def __str__(self):
        return f"Student Name: {self.name}, Roll Number: {self.roll}"
s1 = Student("Alice", 101)
s2 = Student("Bob", 102)
print(s1)
print(s2)
```

The output of the code cell is:

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

A sidebar on the right provides a summary of recent activity and a news feed.

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. In cell [22], there is an error message:

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

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

A tooltip provides a link to "Explain error". On the right, a sidebar displays a notification about browser notifications and a news item from 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 listed include:

- 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

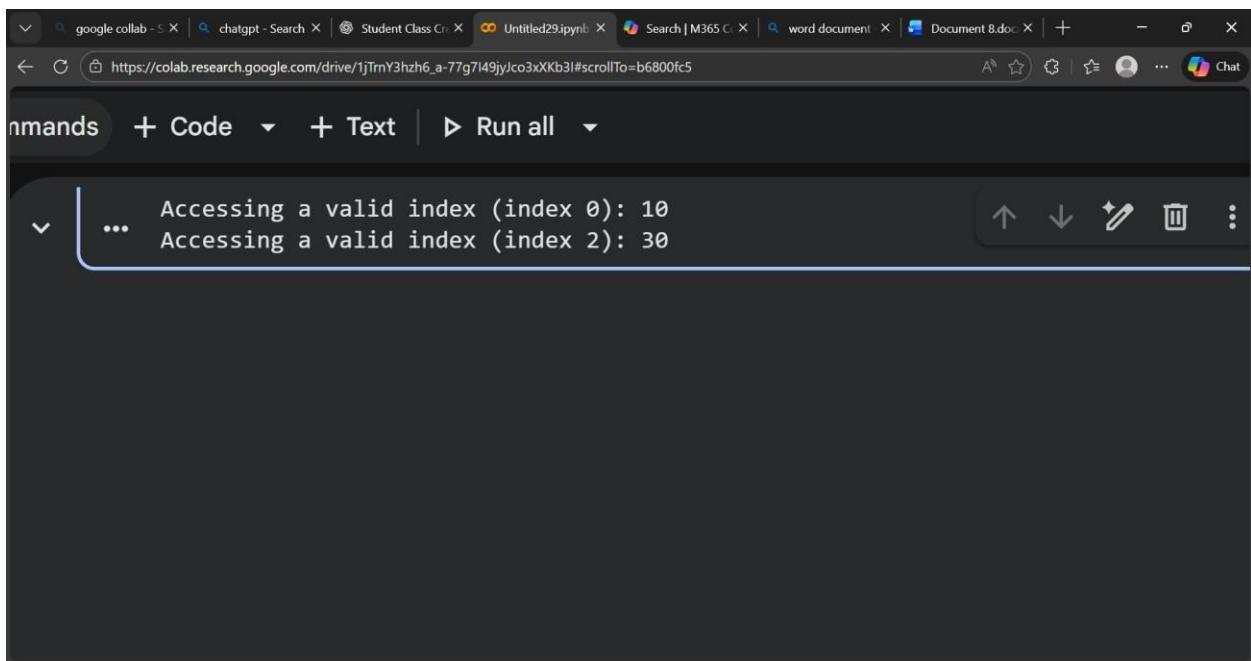
Corrected Code:

The screenshot shows a Google Colab interface. In cell [23], the code has been corrected to handle index errors safely:

```
# 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-77g7l49yJco3xXKb3I#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.