

AI ASSISTED CODING

ASSIGNMENT - 7.3

ROLL NO. 2303A510J0

Task 1: Fixing Syntax Errors

Scenario

You are reviewing a Python program where a basic function definition contains a syntax error.

```
python

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

Requirements

- Provide a Python function `add(a, b)` with a **missing colon**
- Use an AI tool to detect the syntax error
- Allow AI to correct the function definition
- Observe how AI explains the syntax issue

Expected Output

- Corrected function with proper syntax
- Syntax error resolved successfully
- AI-generated explanation of the fix

Wrong code:

```
_000 > pz > 2044b.py > add
1  def add(a, b)
2  |    return a + b
```

Output:

```
PS C:\Users\rohit\OneDrive\Desktop\Code\DSA\codeforces> & C:/Users/rohit/AppData/Local/Programs/Python/Python314/python.exe c:/Users/rohit/OneD  
rive/Desktop/Code/DSA/codeforces/r_800/p2/2044b.py  
File "c:/Users/rohit/OneDrive/Desktop/Code/DSA/codeforces/r_800/p2/2044b.py", line 1  
    def add(a, b)  
          ^  
SyntaxError: expected ':'  
PS C:\Users\rohit\OneDrive\Desktop\Code\DSA\codeforces> & C:/Users/rohit/AppData/Local/Programs/Python/Python314/python.exe c:/Users/rohit/OneD  
rive/Desktop/Code/DSA/codeforces/r_800/p2/2044b.py
```

Explanation:

The screenshot shows a code editor interface with a tooltip explaining a syntax error. The tooltip content is as follows:

CHAT + ⌂ ⚙️ ⋮ | ☰ ×

← SYNTAX ERROR TROUBLESHOOTING IN CODE

@workspace /explain Expected ":"

Workspace

> Used 1 reference

The code in `2044b.py` has a syntax error on line 1. In Python, function definitions must end with a colon (`:`) after the parameter list to indicate the start of the function body. Without it, the parser expects a colon and raises an error.

Here's the corrected version:

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

This follows Python's syntax rules for defining functions, ensuring the code runs without errors. If you have more context or other issues, provide details for further help.

G ↲ ⌘ ⌘ Grok Code Fast 1 • 1x

2044b.py

Describe what to build next

Agent ⌄ Auto ⌄ 🎨 ↵ ➤

Spaces: 4 UTF-8 CRLF { } Python 3.14.2 Go Live

Prompt:

Fix the code

Code:

```
2044b.py ⑥ X

800 > p2 > 2044b.py > ...
1 def add(a, b):
2     return a + b
3
4 # Example usage
5 result = add(5, 3)
6 print(result)
```

Output:

```
PS C:\Users\rohit\OneDrive\Desktop\Code\DSA\codeforces> & C:/Users/rohit/AppData/Local/Programs/Python/Python314/python.exe c:/Users/rohit/OneDrive/Desktop/Code/DSA/codeforces/r_800/p2/2044b.py
8
PS C:\Users\rohit\OneDrive\Desktop\Code\DSA\codeforces>
```

Justification:

This task demonstrates how AI tools can quickly identify and resolve basic syntax errors in Python programs. Syntax errors, such as missing colons in function definitions, prevent code execution entirely. Using AI for detection helps reduce manual debugging time and provides clear explanations of language rules. This improves understanding of Python syntax and reinforces correct coding practices.

Task 2: Debugging Logic Errors in Loops

Scenario

You are debugging a loop that runs infinitely due to a logical mistake.

```
python

def count_down(n):
    while n >= 0:
        print(n)
        n += 1 # Should be n -= 1
```

Requirements

- Provide a loop with an **increment or decrement error**
- Use AI to identify the cause of infinite iteration
- Let AI fix the loop logic
- Analyze the corrected loop behavior

Expected Output

- Infinite loop issue resolved
 - Correct increment/decrement logic applied
 - AI explanation of the logic error
-

Wrong code:

```
def count_down(n):
    while n >= 0:
        print(n)
        n += 1

print(count_down(5))
```

Output: (infinite loop had to keyboard interrupt it)

```
200650
200651
200652
200653
200654
Traceback (most recent call last):
  File "c:\Users\rohit\OneDrive\Desktop\Code\DSA\codeforces\r_800\p2\2044b.py", line 6, in <module>
    print(count_down(5))
    ~~~~~^~^
  File "c:\Users\rohit\OneDrive\Desktop\Code\DSA\codeforces\r_800\p2\2044b.py", line 3, in count_down
    print(n)
    ~~~~~^~^
KeyboardInterrupt
```

Explanation:

Gemini

Please explain this code:

```
<> def count_down(n):
```

◆ This Python code defines a function called `count_down` that takes one argument `n`. The function contains a `while` loop that intends to print the value of `n` as long as `n` is greater than or equal to 0. However, inside the loop, `n` is incremented (`n += 1`), which means if `n` starts non-negative, the condition `n >= 0` will always remain true, leading to an infinite loop. The last line then calls this function with `n=5` and tries to print its return value, but the function will never return due to the infinite loop.

Like Dislike

How can I install Python libraries? Load data from Google Drive

Show an example of training a simple ML model

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](#)

Prompt:

Fix the logical bug in the function in the code

Code:

```
def count_down(n):
    while n >= 0:
        print(n)
        n -= 1

count_down(5)
```

Output:

```
... 5
4
3
2
1
0
```

Justification:

This task focuses on identifying logical errors that cause unintended infinite loops. Such errors are harder to detect than syntax errors because the program runs but behaves incorrectly. AI-assisted debugging helps analyze loop conditions and variable updates, ensuring proper termination. This task highlights the importance of correct logic flow and demonstrates how AI can explain and correct algorithmic mistakes.

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

A Python function crashes during execution due to a division by zero error.

```
# Debug the following code
def divide(a, b):
    return a / b

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

Requirements

- Provide a function that performs division without validation
- Use AI to identify the runtime error
- Let AI add try-except blocks for safe execution
- Review AI's error-handling approach

Expected Output

- Function executes safely without crashing
- Division by zero handled using try-except
- Clear AI-generated explanation of runtime error handling

Wrong code:

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

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

Output:

```
*** ZeroDivisionError                                     Traceback (most recent call last)
/tmp/ipython-input-2311834636.py in <cell line: 0>()
      2     return a / b
      3
----> 4 print(divide(10, 0))

/tmp/ipython-input-2311834636.py in divide(a, b)
      1 def divide(a, b):
----> 2     return a / b
      3
      4 print(divide(10, 0))

ZeroDivisionError: division by zero
```

Next steps: [Explain error](#)

Explanation:

Gemini

Please explain this error:

! `ZeroDivisionError: division by zero`

◆ The error `ZeroDivisionError: division by zero` occurs because you are trying to divide a number by zero in the `divide` function, specifically when calling `divide(10, 0)`. In mathematics, division by zero is undefined, and Python correctly raises this error to prevent an invalid operation.

👍 🙅

[How can I install Python libraries?](#) [Load data from Google Drive](#)

[Show an example of training a simple ML model](#)

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](#)

⌚ 4:19 PM Python 3

The screenshot shows the Gemini AI interface. At the top, there's a dark header with the word "Gemini". Below it, a user asks "Please explain this error:" followed by a red exclamation mark icon and the text "ZeroDivisionError: division by zero". A blue diamond-shaped bullet point explains that this error occurs when dividing by zero in the "divide" function, specifically with "divide(10, 0)". It notes that division by zero is undefined in mathematics and that Python correctly raises this error. Below the explanation are two small icons: a thumbs-up and a thumbs-down. Further down, there are three buttons: "How can I install Python libraries?", "Load data from Google Drive", and "Show an example of training a simple ML model". A large input field asks "What can I help you build?" with a plus sign icon to its left. To the right of the input field is the text "Gemini 2.5 Flash" with a dropdown arrow and a right-pointing arrow. At the bottom, a note says "Gemini can make mistakes so double-check it and use code with caution." with a link to "Learn more". The bottom bar shows the time "4:19 PM" and the Python 3 logo.

Prompt:

Fix the logical bug in the code

Fixed code:

```
def divide(a, b):  
    return a / b  
  
try:  
    print(divide(10, 0))  
except ZeroDivisionError:  
    print("Error: Cannot divide by zero")
```

Output:

```
... Error: Cannot divide by zero
```

Justification:

Runtime errors occur during program execution and can cause unexpected crashes. This task illustrates how AI detects runtime exceptions such as division by zero and suggests structured error handling using try-except blocks. Implementing exception handling improves program robustness and reliability. AI explanations help understand defensive programming techniques and safe execution strategies.

Task 4: Debugging Class Definition Errors

Scenario

You are given a faulty Python class where the constructor is incorrectly defined.

python

```
class Rectangle:  
    def __init__(length, width):  
        self.length = length  
        self.width = width
```

Requirements

- Provide a class definition with **missing self-parameter**
- Use AI to identify the issue in the `__init__()` method
- Allow AI to correct the class definition
- Understand why `self` is required

Expected Output

- Corrected `__init__()` method
 - Proper use of `self` in class definition
 - AI explanation of object-oriented error
-

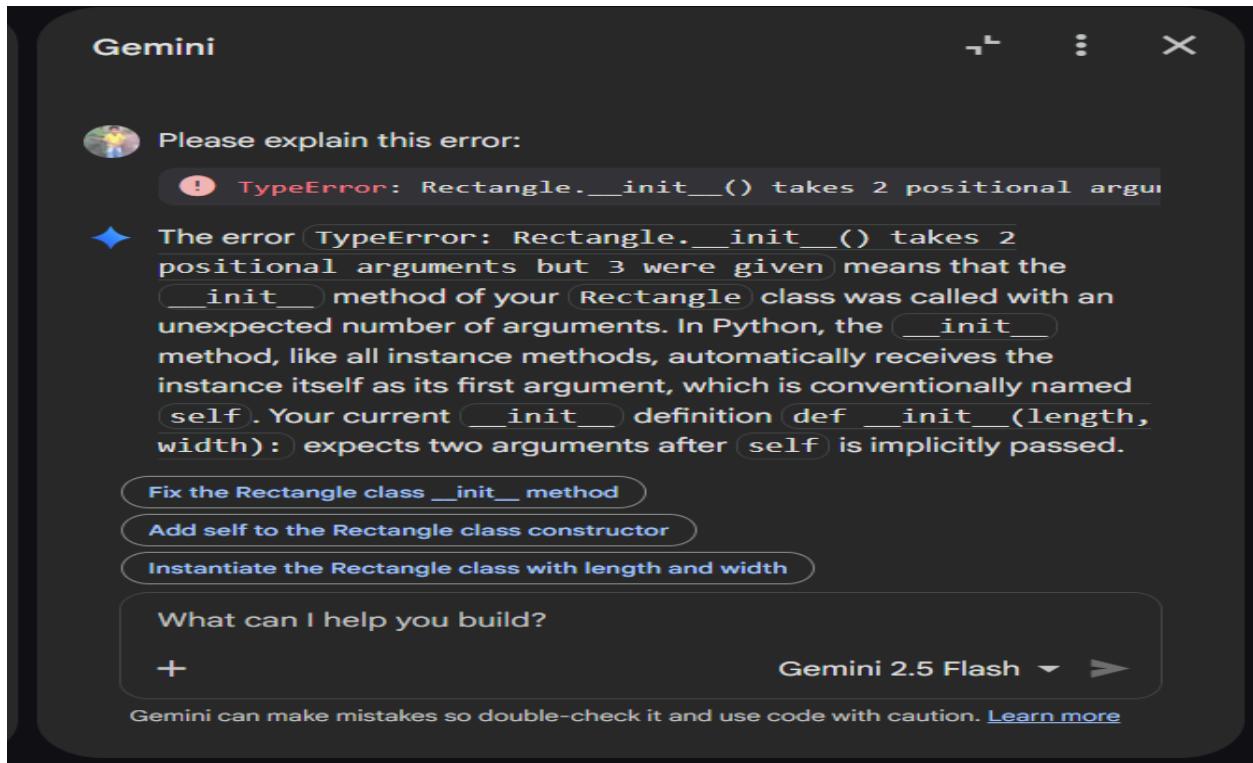
Wrong code:

```
class Rectangle:  
    def __init__(length, width):  
        self.length = length  
        self.width = width  
  
r1 = Rectangle(5, 6)
```

Output:

```
-----  
TypeError                                 Traceback (most recent call last)  
/tmp/ipython-input-159090773.py in <cell line: 0>()  
      4     self.width = width  
      5  
----> 6 r1 = Rectangle(5, 6)  
  
TypeError: Rectangle.__init__() takes 2 positional arguments but 3 were given
```

Explanation:



The screenshot shows the Gemini AI interface with the following details:

- Gemini** is the AI's name.
- The user asks: "Please explain this error:"
- The AI responds: "The error `TypeError: Rectangle.__init__() takes 2 positional arguments but 3 were given` means that the `__init__` method of your `Rectangle` class was called with an unexpected number of arguments. In Python, the `__init__` method, like all instance methods, automatically receives the instance itself as its first argument, which is conventionally named `self`. Your current `__init__` definition `def __init__(length, width):` expects two arguments after `self` is implicitly passed."
- Three buttons are shown below the explanation:
 - Fix the Rectangle class `__init__` method
 - Add `self` to the Rectangle class constructor
 - Instantiate the Rectangle class with length and width
- A message at the bottom says: "What can I help you build?"
- At the bottom right, it says "Gemini 2.5 Flash" with a dropdown arrow.
- At the very bottom, a footer note says: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)"

Prompt:

Fix the class in the code

Fixed code:

```
class Rectangle:  
    def __init__(self, length, width):  
        self.length = length  
        self.width = width  
  
r1 = Rectangle(5, 6)  
print(r1.length)  
print(r1.width)
```

Output:

```
[... 5  
   6]
```

Justification:

This task addresses common object-oriented programming mistakes, specifically incorrect constructor definitions. The absence of the `self` parameter leads to runtime errors and improper object initialization. AI-assisted debugging identifies this issue and explains the role of `self` in accessing instance variables. This task strengthens understanding of Python's class structure and object behavior.

Task 5: Resolving Index Errors in Lists

Scenario

A program crashes when accessing an invalid index in a list.

```
python
```

```
numbers = [1, 2, 3]
print(numbers[5])
```

Requirements

- Provide code that accesses an **out-of-range list index**
- Use AI to identify the Index Error
- Let AI suggest safe access methods
- Apply bounds checking or exception handling

Expected Output

- Index error resolved
- Safe list access logic implemented

Wrong code:

```
numbers = [1, 2, 3]
print(numbers[5])
```

Output:

```
...
IndexError                                 Traceback (most recent call last)
/tmp/ipython-input-1537628302.py in <cell line: 0>()
      1 numbers = [1, 2, 3]
----> 2 print(numbers[5])

IndexError: list index out of range
```

Next steps: [Explain error](#)

Explanation:

The screenshot shows the Gemini AI interface. At the top, it says "Gemini". Below that, a user icon and the text "Please explain this error:". A tooltip-like box contains the error message "IndexError: list index out of range". To the right of the error message is a blue diamond-shaped bullet point followed by a detailed explanation: "The error `IndexError: list index out of range` means you are trying to access an item in the list using an index that is beyond the list's boundaries. In Python, lists are zero-indexed, meaning the first element is at index 0. Your `numbers` list `[1, 2, 3]` has elements at indices 0, 1, and 2. When you try to access `numbers[5]`, there is no element at that position, hence the error." Below the explanation are two small icons: a thumbs up and a thumbs down. At the bottom of the main window, there are three buttons: "Explain the IndexError in the last cell", "Fix the code for list index out of range", and "Show how to safely access list elements". Below these buttons is a text input field with the placeholder "What can I help you build?". To the right of the input field is the text "Gemini 2.5 Flash" with a dropdown arrow and a right-pointing arrow. At the very bottom of the interface, there is a status bar with the text "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)". On the far left of the status bar is a small warning icon. On the far right are icons for time ("4:31PM") and Python 3.

Prompt:

Fix the Error in the code

Fixed code:

```
numbers = [1, 2, 3]

try:
    print(numbers[5])
except IndexError:
    print("Error: Index is out of range for the list.")
```

Output:

```
... Error: Index is out of range for the list.
```

Justification:

Index errors occur when attempting to access elements outside the valid range of a list. This task demonstrates how AI identifies such errors and recommends safe access techniques like bounds checking or exception handling. Proper list handling prevents program crashes and ensures data safety. The AI explanations promote writing more reliable and error-resistant code.