

# ASSIGNMENT-7.3

**Name:** M. Preethi Meghana

**Ht.No:** 2303A51584

**Batch:** 05

## Task-01:Fixing Syntax Errors

### Identification of error:



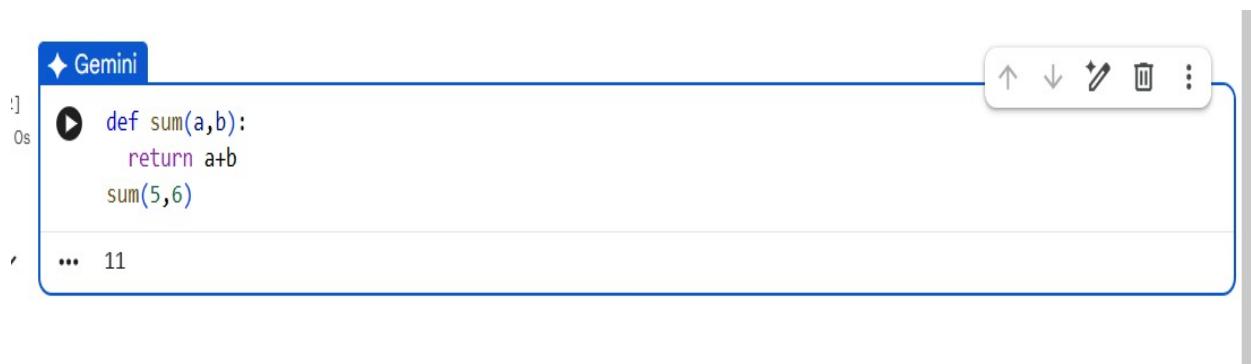
The screenshot shows a Jupyter Notebook cell with the following content:

```
[1] In [1]: def sum(a,b)
        return a+b
    sum(5,6)

Out[1]: ... File "/tmp/ipython-input-2461189405.py", line 1
        def sum(a,b)
                  ^
SyntaxError: expected ':'
```

A red play button icon is next to the cell number. A context menu with icons for up, down, copy, paste, and more is visible above the cell. At the bottom of the cell, there is a "Next steps:" button with the text "Explain error".

## Fixing Error:



The screenshot shows a code editor window titled "Gemini". The code is as follows:

```
def sum(a,b):
    return a+b
sum(5,6)
... 11
```

The code contains a syntax error: it is missing a colon (:) after the parameter list "a,b" in the function definition. The editor highlights this error with a red squiggle under the colon.

## Explanation of Error:



The Gemini interface displays the following message:

M Please explain this error:  
! SyntaxError: expected ':'  
◆ The error 'SyntaxError: expected ':'' indicates that a colon is missing after the parameter list in the `def sum(a,b)` line. In Python, function definitions require a colon after the parentheses enclosing the parameters. I've updated the code to include the missing colon.  
👎 👍

## Explanation of code:

### EXPLANATION:

- A Python function named `add(a, b)` is created to perform addition.
- The function definition initially has a syntax error because the colon (:) is missing.
- When the program is executed, Python raises a syntax error.

- The AI tool analyzes the code and identifies the missing colon as the issue.
- The function definition is corrected by adding the required colon.
- The corrected function follows proper Python syntax rules.
- The function executes successfully and returns the correct result.

## Task-02: Debugging Logic Errors in Loops

### Identification of Error:

The screenshot shows the Gemini AI interface. In the code editor, there is a Python script named 'Gemini' containing a function 'count\_down'. The code has a syntax error where the assignment 'n+=1' is highlighted in red, indicating it's incorrect. The function is called with 'count\_down(10)'. Below the code, the output shows a series of numbers from 23613123 to 23613145, which is an infinite loop.

A tooltip window is open, providing the following information:

- M** identify why i am getting infinite loop
- def count\_down(n):**
- ◆** The code (count\_down(n)) in cell (idszBEVuBTL5) is causing an infinite loop because inside the while loop, the value of n is increasing with n += 1.
- ▷ Accept & Run    ✓ Accept    ✘ Cancel**

At the bottom of the tooltip, there is a message: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

### Fixing Error:

```
s  def count_down(n):
    while(n>=0):
        print(n)
        n-=1
    count_down(10)

...
10
9
8
7
6
5
4
3
2
1
0
```

## Explanation of Code:

- A loop is written with a logical error in its increment/decrement statement.
- Due to this mistake, the loop condition never becomes false, causing infinite execution.
- When the program is run, the loop continues indefinitely without termination.
- The AI tool analyzes the loop condition and identifies the incorrect increment/decrement logic.
- The loop logic is corrected so that the loop control variable moves toward the terminating condition.
- After correction, the loop executes a finite number of times.
- The infinite loop issue is resolved, and the program behaves as expected.

## Task-03: Handling Runtime Errors (Division by Zero)

### Identification of Error:

```
▶ def divide(a,b):
    return a/b
print(divide(10,0))

...
ZeroDivisionError                                     Traceback (most recent call last)
/tmp/ipython-input-3986490526.py in <cell line: 0>()
      1 def divide(a,b):
      2     return a/b
----> 3 print(divide(10,0))

/tmp/ipython-input-3986490526.py in divide(a, b)
      1 def divide(a,b):
----> 2     return a/b
      3 print(divide(10,0))

ZeroDivisionError: division by zero
```

Next steps: [Explain error](#)

## Explanation of Error:

Release notes      Gemini X

M Please explain this error:

! ZeroDivisionError: division by zero

◆ The error `ZeroDivisionError: division by zero` occurs because you are trying to divide the number 10 by 0, which is not mathematically allowed. To prevent this, you can add a check for the divisor before performing the division. I've updated the code to handle this scenario.

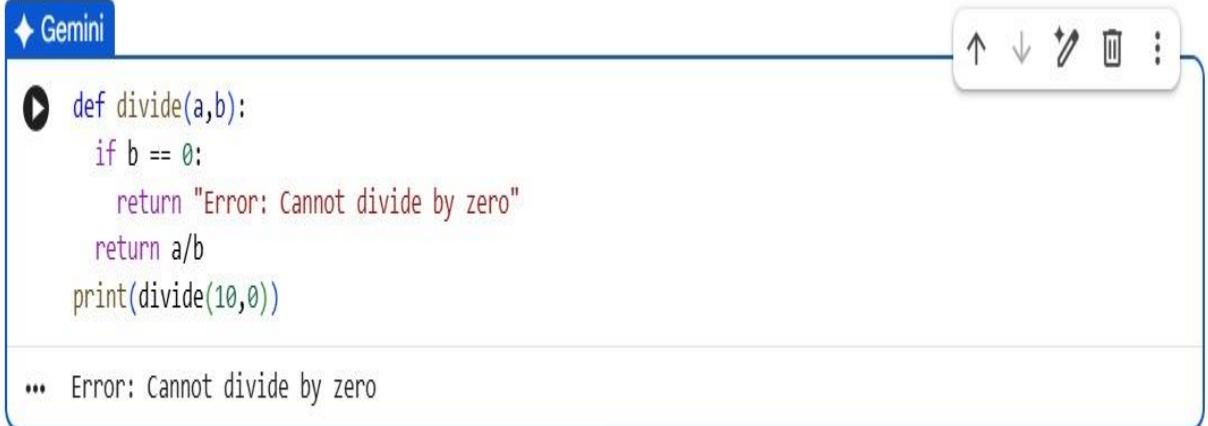
↶ ↷

▷ Accept & Run    ✓ Accept    ✘ Cancel

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](#)

## Fixing Error:



The screenshot shows a code editor window titled "Gemini". The code in the editor is as follows:

```
def divide(a,b):
    if b == 0:
        return "Error: Cannot divide by zero"
    return a/b
print(divide(10,0))
```

Below the code, the output is displayed as:

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

## EXPLANATION:

- A Python function is defined to perform division of two numbers without input validation.
- When the divisor value is zero, the function crashes during execution.
- Python raises a runtime error called `ZeroDivisionError`.
- The AI tool analyzes the error and identifies division by zero as the cause.
- A try-except block is added to handle the division operation safely.
- The try block performs the division, and the except block catches the runtime error.
- The function now executes without crashing and handles division by zero gracefully.

# Task-04: Debugging Class Definition Errors

## Identification of Error:

The screenshot shows a code editor window with the following code:

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

Below the code, a traceback is displayed:

```
...
TypeError Traceback (most recent call last)
/tmp/ipython-input-1857360067.py in <cell line: 0>()
      5     def area(self):
      6         return self.length * self.width
----> 7 print(Rectangle(5,6).area())

TypeError: Rectangle.__init__() takes 2 positional arguments but 3 were given
```

At the bottom left, there is a "Next steps" button labeled "Explain error".

## Explanation of Error:

The screenshot shows a code editor window with the same Python script as before. The line `- def __init__(length, width):` is highlighted in red, indicating it's the source of the error.

On the right side, a Gemini AI interface provides an explanation of the error:

Release notes Gemini X

ⓘ **TypeError: Rectangle.\_\_init\_\_() takes 2 positional arguments but 3 were given**

◆ The error `TypeError: Rectangle.__init__() takes 2 positional arguments but 3 were given` means that the `__init__` method of your `Rectangle` class is missing the `self` parameter. All instance methods in Python, including `__init__`, automatically receive the instance itself as their first argument. You called `Rectangle(5, 6)`, which implicitly passes the instance object, plus `5` and `6`, making it three arguments in total for a method defined to accept only two (`length` and `width`). I've updated the `__init__` method definition to include `self` as its first parameter.

Accept & Run Accept Cancel

Next steps: Explain error

## Fixing Error:



The screenshot shows a code editor window titled "Gemini". The code is as follows:

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

At the bottom left, there is a status bar with "... 30". On the right side of the editor, there is a toolbar with icons for up, down, edit, delete, and more.

## EXPLANATION:

- A Python class is defined with a constructor method `__init__()` that is missing the `self` parameter.
- When an object of the class is created, Python raises an error due to the incorrect constructor definition.
- The AI tool analyzes the class and identifies the missing `self` parameter as the issue.
- The constructor is corrected by adding `self` as the first parameter in the `__init__()` method.
- The `self` parameter allows instance variables to be associated with the current object.
- After correction, the class initializes objects properly.
- The corrected class follows object-oriented programming principles and works as expected.

# Task-05:Resolving Index Errors in Lists

## Identification of Error:

The screenshot shows a code editor window with the following content:

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

...
IndexError Traceback (most recent call last)
/tmp/ipython-input-3809996345.py in <cell line: 0>()
      1 numbers=[1,2,3]
----> 2 print(numbers[5])

IndexError: list index out of range
```

Below the code, there is a button labeled "Next steps: Explain error".

## Explanation of Error:

The screenshot shows a code editor window with the following content:

```
◀ Gemini
▶ numbers=[1,2,3]
- print(numbers[5])
+try:
+ print(numbers[5])
+except IndexError:
+ print("Error: Index is out of range. Please use a valid index.")

...
IndexError Traceback (most recent call last)
/tmp/ipython-input-3809996345.py in <cell line: 0>()
      1 numbers=[1,2,3]
----> 2 print(numbers[5])

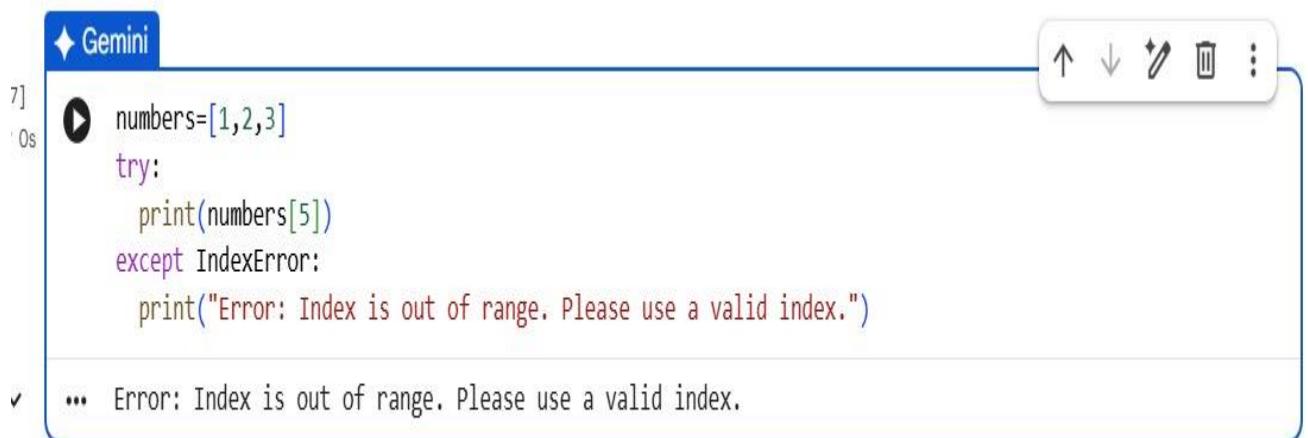
IndexError: list index out of range
```

On the right side of the screen, there is a sidebar titled "Release notes" with the following content:

- M** use exception handling
  - I've added a `try-except` block to handle potential `IndexError`s. Now, if you try to access an index that is out of range, it will gracefully print an error message instead of crashing.

Below the code, there is a button labeled "Next steps: Explain error".

## Fixing Error:



The screenshot shows the Gemini AI interface. At the top, it says "Gemini". Below that, there's a code editor window with the following Python code:

```
7] 0s
▶ numbers=[1,2,3]
try:
    print(numbers[5])
except IndexError:
    print("Error: Index is out of range. Please use a valid index.")

✓ ... Error: Index is out of range. Please use a valid index.
```

The code attempts to print the element at index 5 of the list `numbers`, which contains only three elements. This results in an `IndexError` being raised and caught by the `try-except` block, which then prints an error message. The AI tool has detected this error and highlighted the problematic line of code in red.

## EXPLANATION:

- A Python program attempts to access a list element using an index that is out of range.
- When the program is executed, Python raises an `IndexError`.
- The AI tool analyzes the code and identifies the invalid index access as the cause of the error.
- The AI suggests using safe access methods such as bounds checking or exception handling.
- Bounds checking ensures the index is within the valid range before accessing the list.
- Alternatively, a `try-except` block is used to catch the `IndexError`.
- After applying safe access logic, the program executes without crashing.
- The index error is successfully resolved, and list access becomes safe.