

AI ASSISTANT CODING

ASSIGNMENT-7.3

Name:V.Sruthi

Ht.no:2303A51308

Batch:05

Task-01: Fixing Syntax Errors

Identification of error:

The screenshot shows a code editor interface with a Python script. The code is as follows:

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

An error message is displayed below the code:

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

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

Fixing Error:

The screenshot shows a code editor interface with a Python script. The code is identical to the one in the previous screenshot:

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

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

Explanation of Error:

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 a code editor window titled "Gemini". The code in the editor is:

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

The line `n+=1` is highlighted in red, indicating an error. The line `n-=1` is highlighted in green, suggesting a fix. The status bar at the top right shows navigation icons.

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`.

Fixing Error:

The screenshot shows the same code editor window after the error was fixed. The code now uses `n -= 1` instead of `n += 1`. The output below the code shows the numbers from 10 down to 0, indicating the loop has terminated correctly.

```
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
```

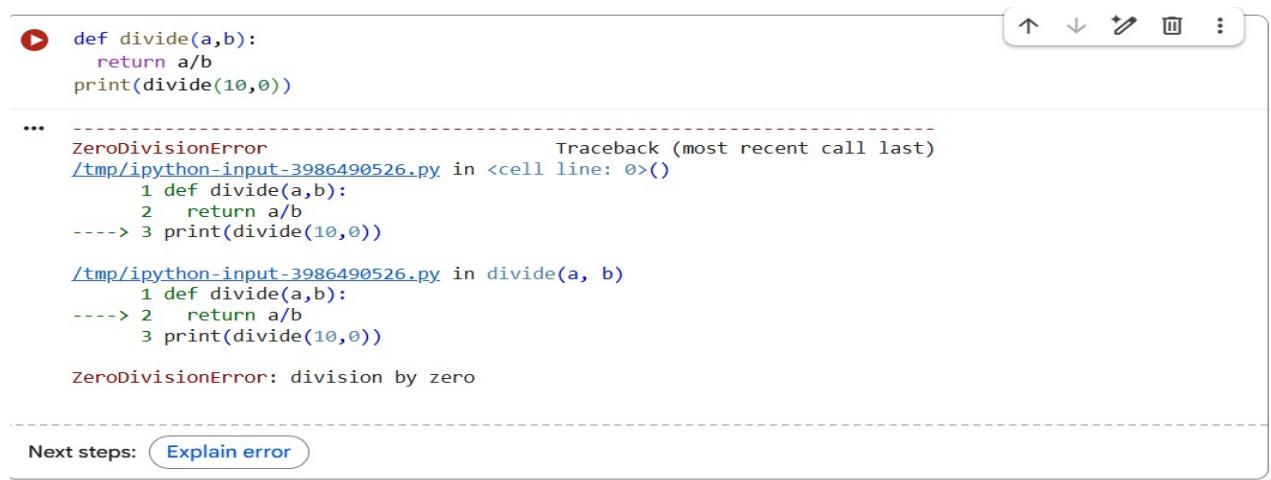
Explanation of Code:

- 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:



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

```
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:

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.



Fixing Error:

The screenshot shows a code editor window titled "Gemini". The code defines a function "divide" that attempts to divide two numbers. It includes a check for division by zero and prints the result. A runtime error message is displayed below the code area.

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

... 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 Jupyter Notebook cell 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 `TypeError` 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 with the option "Explain error".

Explanation of Error:

The screenshot shows a Jupyter Notebook cell with the same code as before. The error message is identical:

```
*** -----  
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
```

To the right of the notebook, the Gemini AI interface is open, showing the following release notes:

Release notes Gemini X

① `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

Fixing Error:

The screenshot shows the Gemini AI interface with a code editor window. The title bar says "Gemini". The code in the editor is:

```
s   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())
...
... 30
```

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:

```
▶ 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
```

Next steps: [Explain error](#)

Explanation of Error:

```
◆ 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
```

Next steps: [Explain error](#)

Fixing Error:

The screenshot shows the Gemini AI interface. At the top, there's a header with a play button icon and the word "Gemini". To the right is a toolbar with icons for up, down, edit, delete, and more. Below the header, the code editor contains 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 a list containing [1, 2, 3]. Since lists are zero-indexed, this is out of range. A try-except block catches the IndexError and prints an error message. The AI tool has analyzed this and identified the error.

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.

