**ASSIGNMENT-7**

**NAME:**V.VIVEK VARDHAN

**HT NO:**2403A52097

**BATCH:** 24BTCAIAIB05

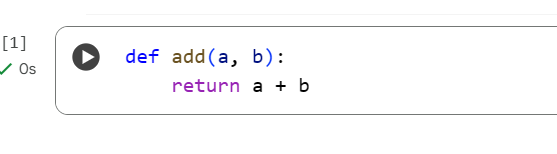
**TASK-1:**

Prompt:

def add(a, b)

return a + b  
Fix the syntax error

**CODE AND OUTPUT:**



**EXPLANATION:**

* **Original Error:** The original code had a SyntaxError.
* **Reason for Error:** This error occurred because the line defining the function (def add(a, b)) was missing a colon at the end.
* **Python Requirement:** In Python, function definition lines using def must end with a colon (:).
* **The Fix:** The fix involved adding the missing colon to the end of the function definition line.
* **Corrected Code:** The corrected line is def add(a, b):.
* **Result:** Adding the colon made the syntax valid and resolved the error.

**TASK-2:**

Prompt:

def count\_down(n):

while n >= 0:

print(n)

n += 1

Identify and fix a logic error in a loop that causes infinite iteration.

**CODE AND OUTPUT:**

****

**EXPLANATION:**

* **Original Error:** The original code had a logic error causing an infinite loop.
* **Reason for Error:** Inside the while loop, the line n += 1 was incrementing the value of n.
* **Intended Behavior:** A countdown function should *decrease* the value of n.
* **The Fix:** The line n += 1 was changed to n -= 1.
* **Result:** Decrementing n in each iteration ensures that the loop condition n >= 0 eventually becomes false, terminating the loop and correctly performing a countdown.

**TASK-3:**

Prompt:

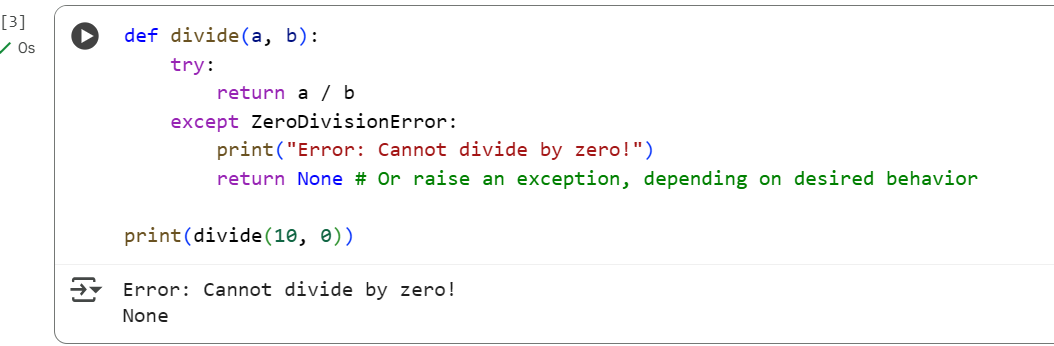
def divide(a,b):

return a/b

print(divide(10,0))

Debug a runtime error caused by division by zero and insert try-except**.**

**CODE AND OUPUT**:

****

**EXPLANATION:**

1. **def divide(a, b):**: This line defines a function named divide that takes two arguments, a and b.
2. **try:**: This keyword starts a "try block". Code inside this block is executed, and if an error occurs, Python looks for a matching except block to handle it.
3. **return a / b**: This line attempts to perform the division of a by b. This is the operation that can potentially cause a ZeroDivisionError if b is 0.
4. **except ZeroDivisionError:**: This line starts an "except block". This block is executed only if a ZeroDivisionError occurs in the preceding try block.
5. **print("Error: Cannot divide by zero!")**: If a ZeroDivisionError occurs, this line prints an informative error message to the console.
6. **return None**: Also within the except block, this line returns the value None if a division by zero occurs. This is one way to handle the error; you could also choose to raise a different exception or handle it in another way.
7. **print(divide(10, 0))**: This line calls the divide function with the arguments 10 and 0. Since the second argument is 0, this will trigger the ZeroDivisionError, causing the code inside the except block to execute. The function will return None, which will then be printed to the console.

In essence, this code defines a division function that gracefully handles the specific case of trying to divide by zero, preventing the program from crashing and instead providing a user-friendly error message and a defined return value (None).

**TASK-4:**

Prompt:

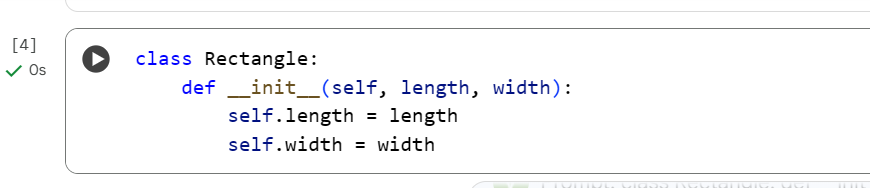
class Rectangle:

def \_\_init\_\_(length, width):

self.length = length

self.width = width  
Fix the given code.

**CODE AND OUTPUT:**

****

**EXPLANATION:**

1. **class Rectangle:**: This line declares a class named Rectangle. A class is a blueprint for creating objects (instances). It defines the properties (attributes) and behaviors (methods) that objects of this class will have.
2. **def \_\_init\_\_(self, length, width):**: This defines a special method called \_\_init\_\_. This is the constructor method, and it's automatically called when you create a new object (an instance) of the Rectangle class.
   * self: This is the first parameter and refers to the instance of the class being created. It's a convention in Python, and you'll see it in all instance methods.
   * length and width: These are additional parameters that you provide when creating a Rectangle object (e.g., Rectangle(10, 5)). They represent the dimensions of the rectangle.
3. **self.length = length**: This line takes the value passed for the length parameter when creating the object and assigns it to an attribute of the object called length. self.length means "the length attribute of this specific instance."
4. **self.width = width**: Similarly, this line takes the value passed for the width parameter and assigns it to the width attribute of the object (self.width).

In summary, this code defines a blueprint for a Rectangle. When you create a Rectangle object, the \_\_init\_\_ method is called to set up the object's initial state by storing the provided length and width values as attributes of that object.

**TASK-5:**

Prompt:

numbers = [1, 2, 3]

print(numbers[5])  
Fix the given code.

**CODE AND OUPUT:**



**EXPLANATION:**

* **Original Error:** The code produced an IndexError: list index out of range.
* **Reason for Error:** This happened because the code tried to access the element at index 5 in the list numbers = [1, 2, 3].
* **Valid Indices:** For a list with 3 elements, the valid indices are 0, 1, and 2. Index 5 is outside this range.
* **The Fix:** The index used to access the list was changed from 5 to 0.
* **Corrected Code:** The line print(numbers[5]) was changed to print(numbers[0]).
* **Result:** Accessing index 0 is valid for the list, resolving the IndexError.

0 / 2000