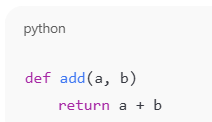
NAME: MANIDEEP.

ROLL.NO: 2403A52146.

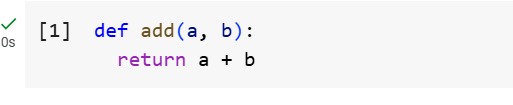
BATCH: 06.

TASK.1

* Paste a function with a missing colon (add(a, b)), and let AI fix the syntax error.



CODE:



EXPLANATION:

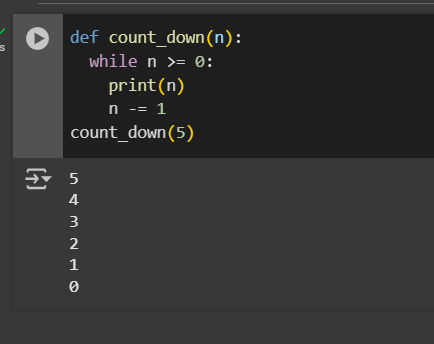
* It creates a small tool called **add**.
* This tool takes two things as input (you can think of them as boxes labeled **a** and **b**).
* It then gives you back the result of adding those two things together.

So if you give it 3 and 5, it will give you 8.

TASK.2

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

CODE:



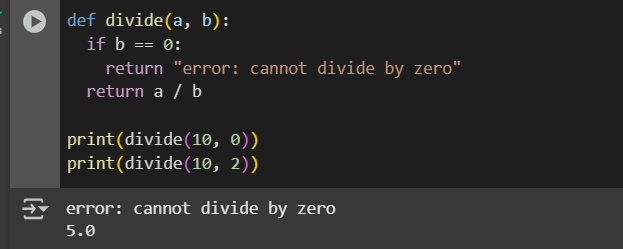
EXPLANATION:

* The code defines a function called count\_down that takes one input, a number n.
* Inside the function, there's a loop that continues as long as n is greater than or equal to 0.
* In each step of the loop, the current value of n is shown.
* Then, n is decreased by 1.
* The loop repeats until n becomes less than 0.
* Finally, the code calls the count\_down function with the number 5, starting the countdown from 5.

TASK:3

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

CODE:



EXPLANATION:

This code defines a function called divide that takes two numbers, let's call them 'a' and 'b'.

* First, it checks if the second number, 'b', is zero.
* If 'b' is zero, it knows that you can't divide by zero, so it gives back a message saying "error: cannot divide by zero".
* If 'b' is not zero, it goes ahead and divides 'a' by 'b' and gives you the result of that division.

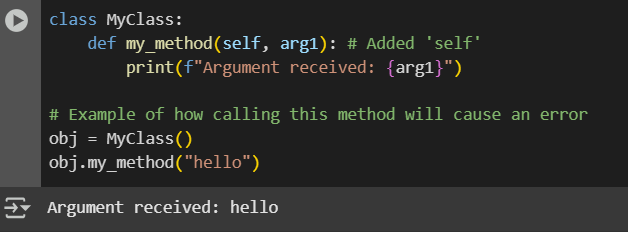
Then, the code shows two examples:

* It tries to divide 10 by 0, and because of the check, it will show the error message.
* It then tries to divide 10 by 2, and since 2 is not zero, it will perform the division and show the result, which is 5.0.

TASK:4

* Provide a faulty class definition (missing self in parameters). Let AI fix it

CODE:



EXPLANATION:

* The blueprint is called MyClass.
* Inside this blueprint, there's a specific action or task that objects made from this blueprint can do. This action is called my\_method.
* When you perform this action (my\_method), you also give it some information, which is called arg1 in this case.
* The important part is that my\_method needs to know which specific object is performing the action. This is why it has self as a special first piece of information – it refers to the object itself.
* When my\_method is performed, it simply shows you the information (arg1) that it received.

Below the blueprint, the code then creates an actual object from the MyClass blueprint. It's like building one of those toys from the blueprint.

* This object is given the name obj.
* Finally, it tells this obj object to perform the my\_method action and gives it the word "hello" as the information (arg1).
* So, the object named obj will perform its my\_method and show you the message "Argument received: hello".

0 / 2000

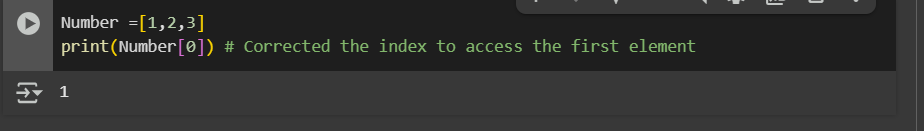
Gemini can make mistakes so double-check it and use code with caution. [Learn more](http://g.co/legal/generative-code)

b: Any, hint

TASK:5

* Access an invalid list index and use AI to resolve the Index Error.

CODE:



EXPLANATION:

* First, it creates a list of numbers called Number. Think of a list like a container holding items in a specific order. In this list, the items are the numbers 1, 2, and 3.
* Second, it asks to show you the item that is at the very beginning of the list. In programming, we count the positions of items starting from zero. So, the item at position 0 is the first item in the list.