AI ASSISTED CODING

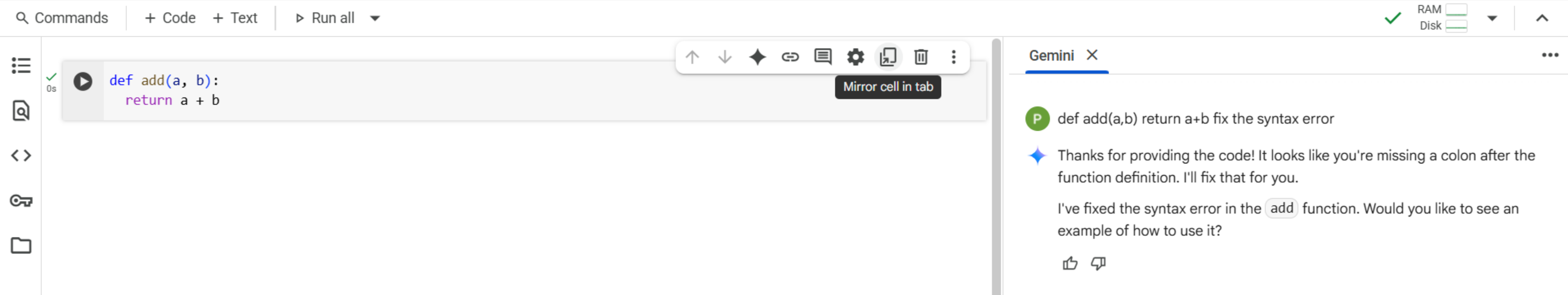
LAB 7

Task Description#1

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

python

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



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

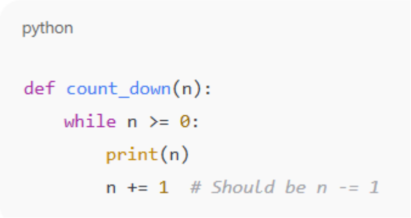
P def add(a,b) return a+b fix the syntax error

Thanks for providing the code! It looks like you're missing a colon after the function definition. I'll fix that for you.

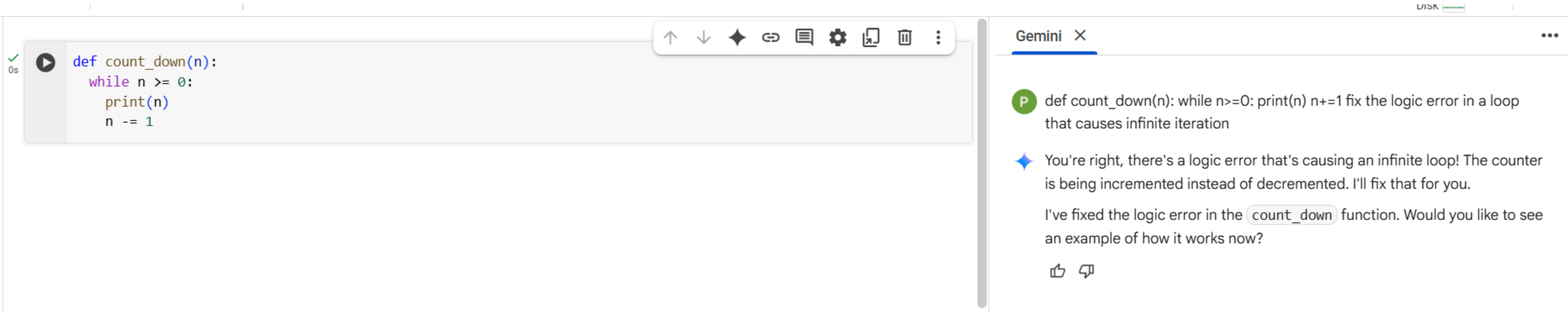
I've fixed the syntax error in the `add` function. Would you like to see an example of how to use it?

Task Description#2 (Loops)

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



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



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

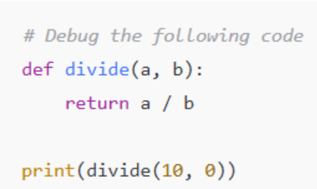
def count_down(n): while n>=0: print(n) n+=1 fix the logic error in a loop that causes infinite iteration

You're right, there's a logic error that's causing an infinite loop! The counter is being incremented instead of decremented. I'll fix that for you.

I've fixed the logic error in the `count_down` function. Would you like to see an example of how it works now?

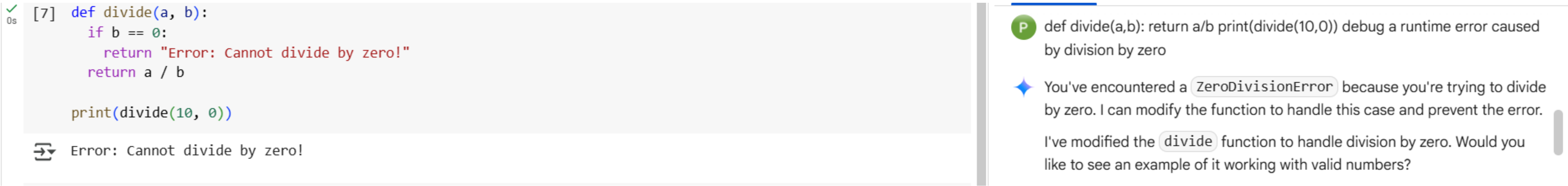
Task Description#3

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



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

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



```
[7] 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!
```

def divide(a,b): return a/b print(divide(10,0)) debug a runtime error caused by division by zero

You've encountered a `ZeroDivisionError` because you're trying to divide by zero. I can modify the function to handle this case and prevent the error.

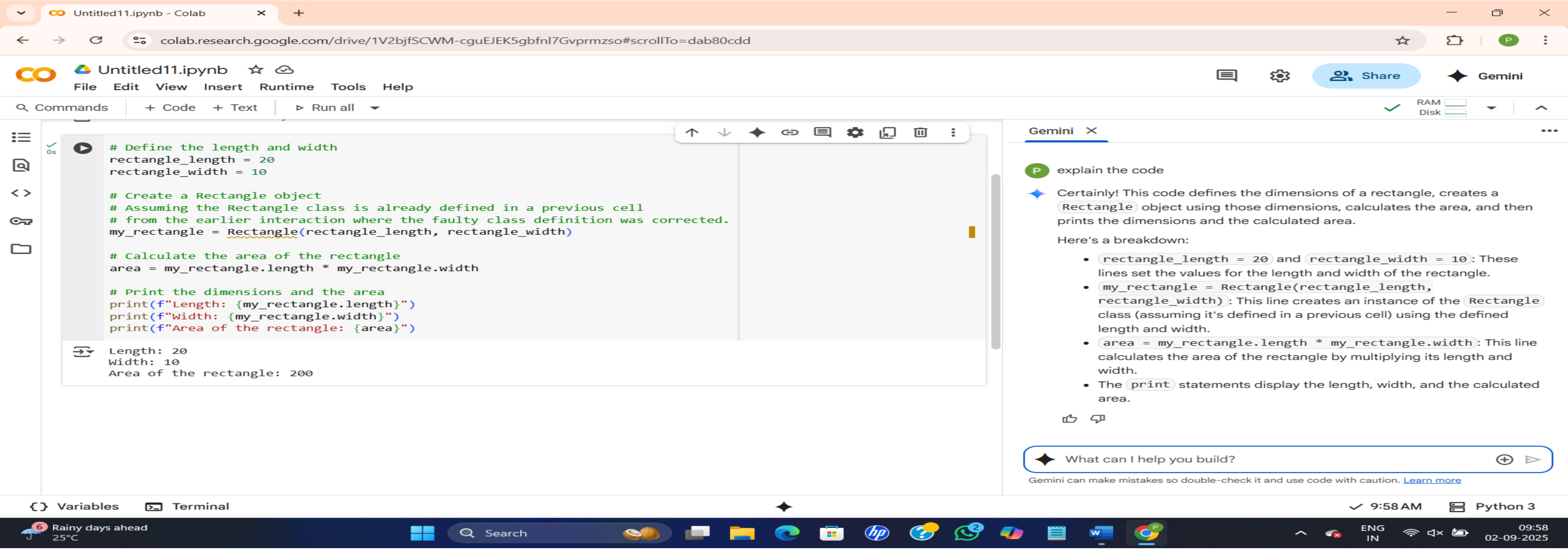
I've modified the `divide` function to handle division by zero. Would you like to see an example of it working with valid numbers?

Task Description#4

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

python

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



```python
# Define the length and width
rectangle_length = 20
rectangle_width = 10

# Create a Rectangle object
# Assuming the Rectangle class is already defined in a previous cell
# from the earlier interaction where the faulty class definition was corrected.
my_rectangle = Rectangle(rectangle_length, rectangle_width)

# Calculate the area of the rectangle
area = my_rectangle.length * my_rectangle.width

# Print the dimensions and the area
print(f"Length: {my_rectangle.length}")
print(f"Width: {my_rectangle.width}")
print(f"Area of the rectangle: {area}")
```

```
Length: 20
Width: 10
Area of the rectangle: 200
```

P explain the code

Certainly! This code defines the dimensions of a rectangle, creates a `Rectangle` object using those dimensions, calculates the area, and then prints the dimensions and the calculated area.

Here's a breakdown:

- `rectangle_length = 20` and `rectangle_width = 10`: These lines set the values for the length and width of the rectangle.
- `my_rectangle = Rectangle(rectangle_length, rectangle_width)`: This line creates an instance of the `Rectangle` class (assuming it's defined in a previous cell) using the defined length and width.
- `area = my_rectangle.length * my_rectangle.width`: This line calculates the area of the rectangle by multiplying its length and width.
- The `print` statements display the length, width, and the calculated area.

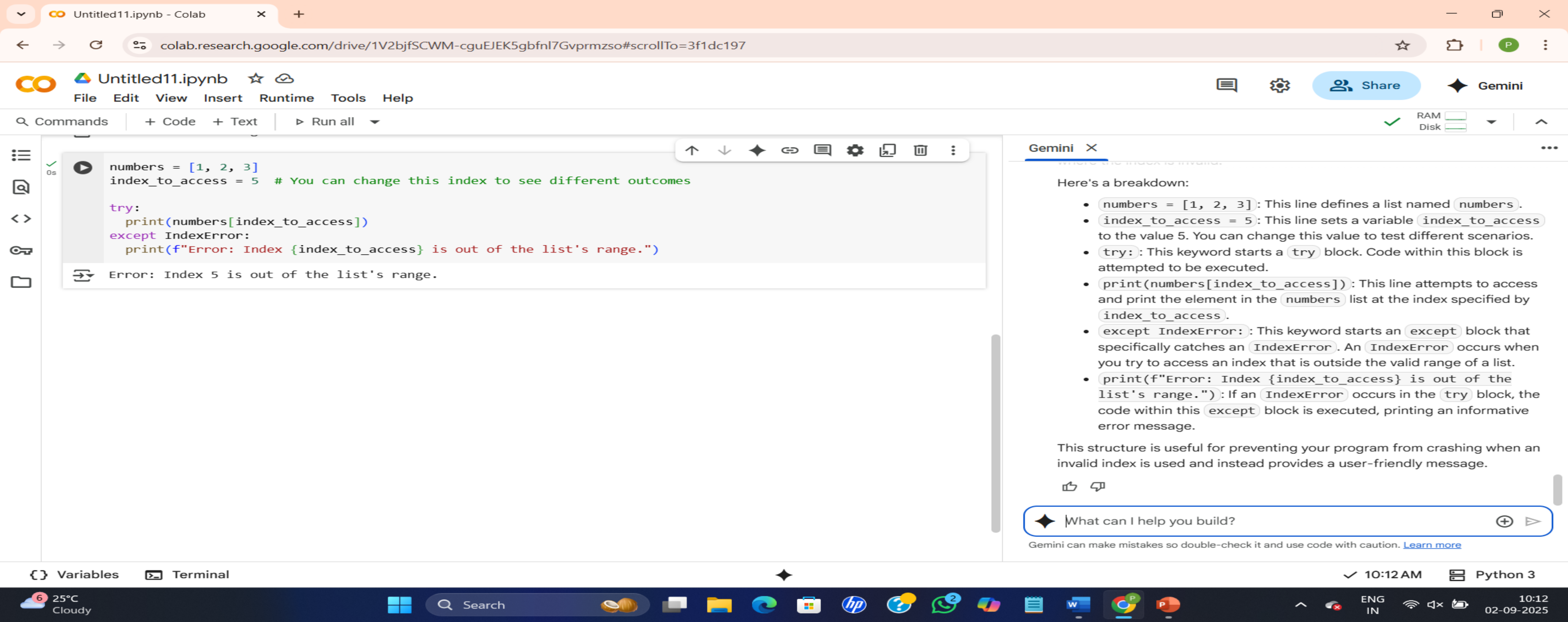
What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

Task Description#5

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

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


Untitled11.ipynb - Colab
colab.research.google.com/drive/1V2bjfSCWM-cguEJEK5gbfnl7Gvprmzso#scrollTo=3f1dc197
Untitled11.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
Variables
Terminal
10:12 AM
Python 3

```
numbers = [1, 2, 3]
index_to_access = 5  # You can change this index to see different outcomes

try:
  print(numbers[index_to_access])
except IndexError:
  print(f"Error: Index {index_to_access} is out of the list's range.")
```

```
Error: Index 5 is out of the list's range.
```

Gemini

Here's a breakdown:

- `numbers = [1, 2, 3]`: This line defines a list named `numbers`.
- `index_to_access = 5`: This line sets a variable `index_to_access` to the value 5. You can change this value to test different scenarios.
- `try:`: This keyword starts a `try` block. Code within this block is attempted to be executed.
- `print(numbers[index_to_access])`: This line attempts to access and print the element in the `numbers` list at the index specified by `index_to_access`.
- `except IndexError:`: This keyword starts an `except` block that specifically catches an `IndexError`. An `IndexError` occurs when you try to access an index that is outside the valid range of a list.
- `print(f"Error: Index {index_to_access} is out of the list's range.")`: If an `IndexError` occurs in the `try` block, the code within this `except` block is executed, printing an informative error message.

This structure is useful for preventing your program from crashing when an invalid index is used and instead provides a user-friendly message.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more