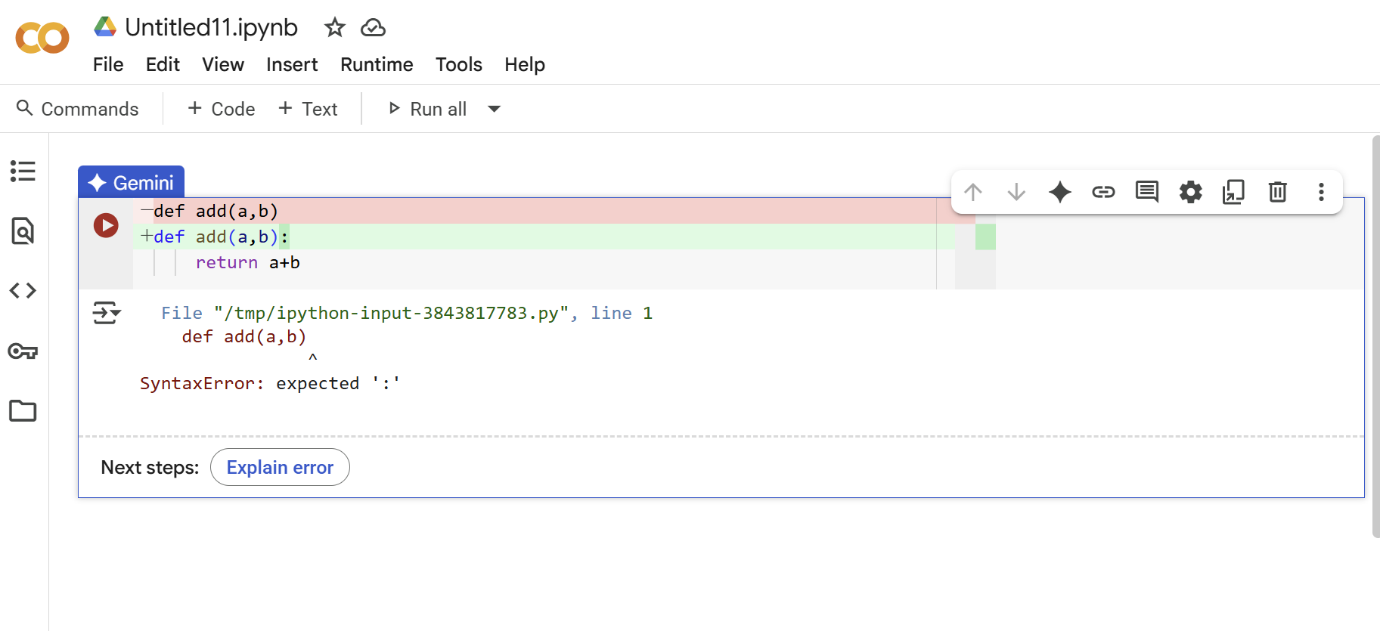
**HTNO:2403A52134**

**ASSIGNMENT-7**

**Task-1:**

**Gemini output:**



**Explanation:**

The original code in the cell with the SyntaxError was likely missing a colon after the function definition line:

def add(a,b) # Missing colon here  
    return a+b

***Error Explanation*:**

The error message SyntaxError: expected ':' means that the Python interpreter was expecting a colon (:) at the end of the def statement. In Python, function definitions must end with a colon before the indented code block that makes up the function's body.

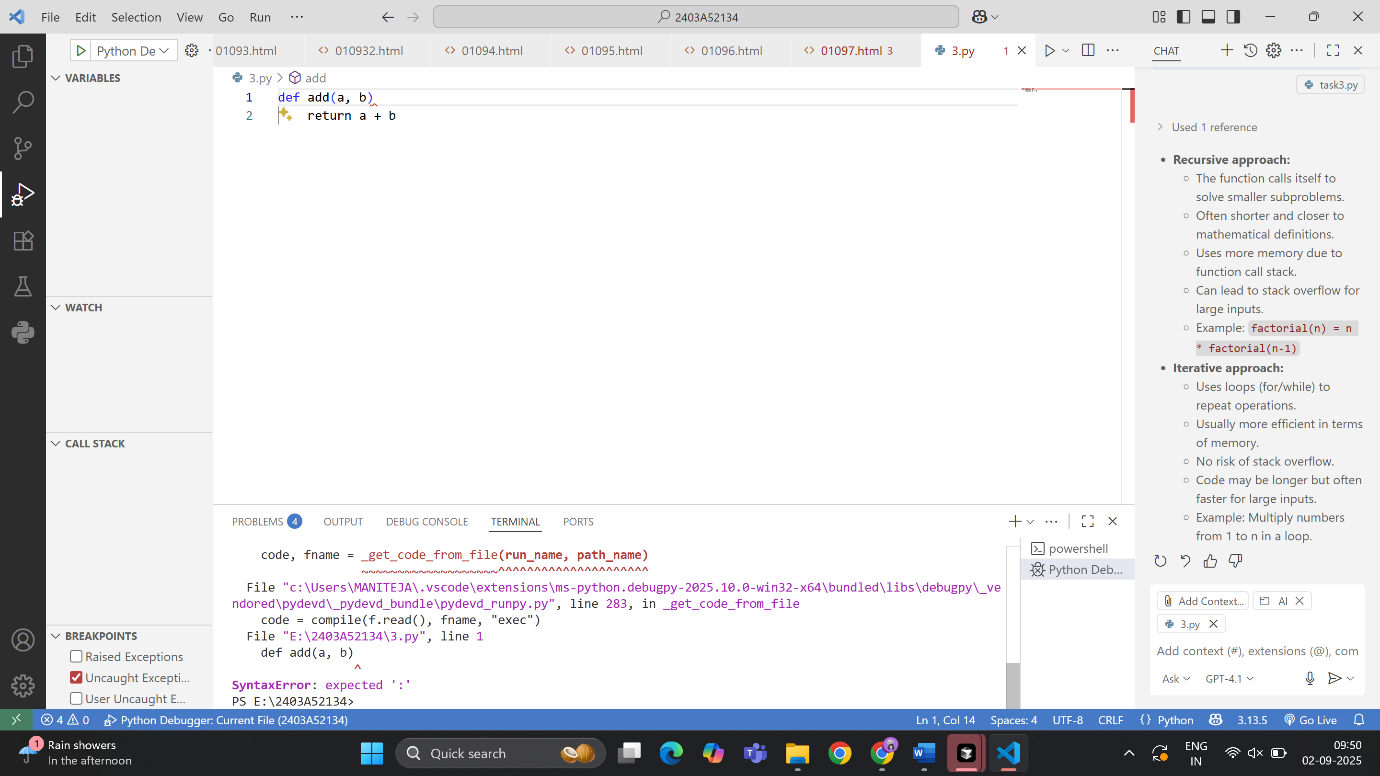
***How it was fixed:***

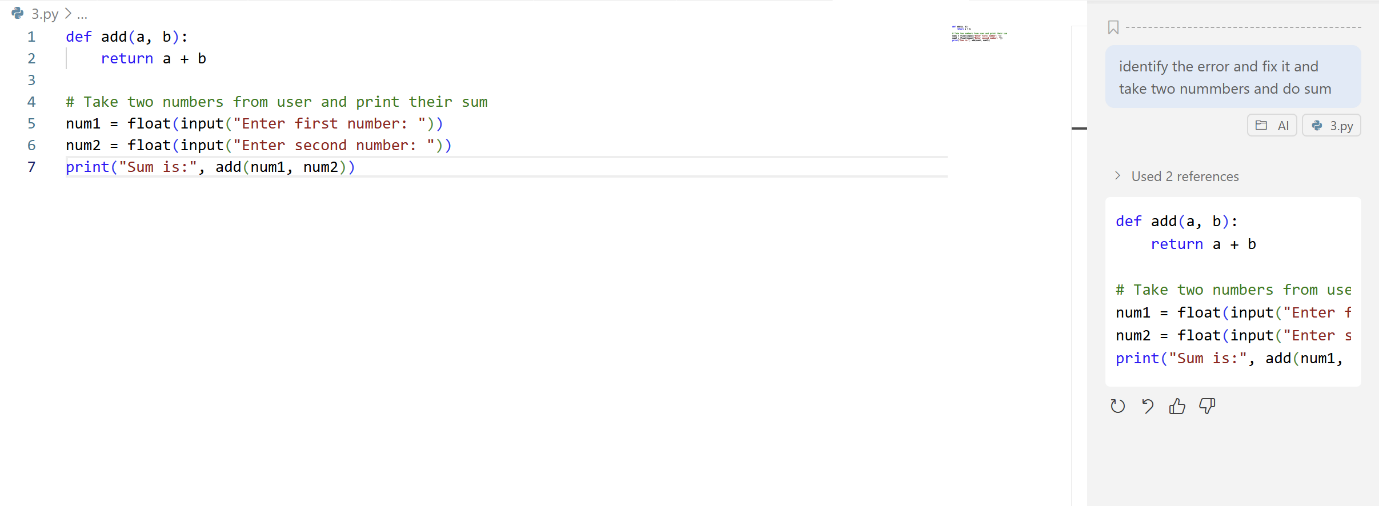
I modified the cell to add the missing colon after the parentheses in the def line.

def add(a,b): # Colon added here  
    return a+b

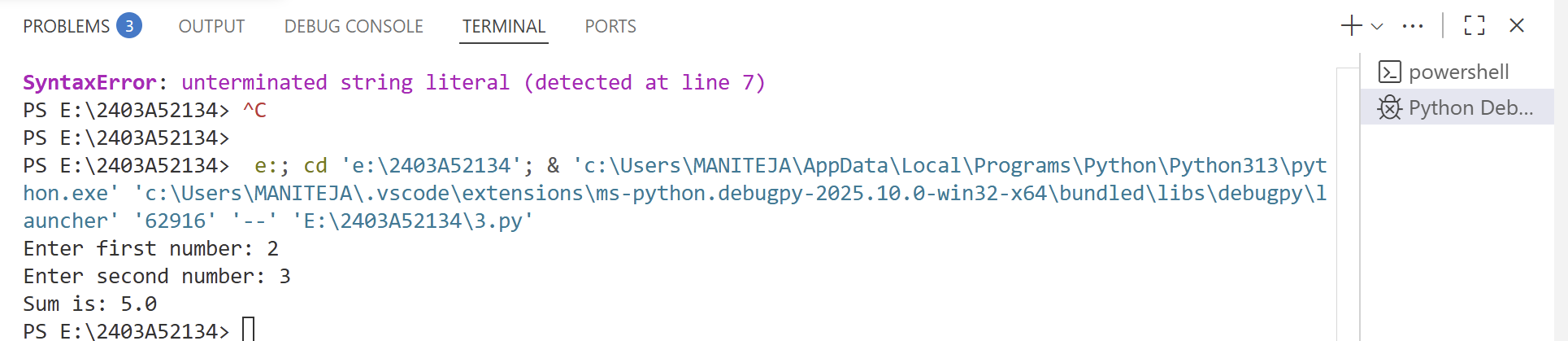
This correction makes the function definition syntactically correct, allowing the Python interpreter to understand and execute the code properly.

**GitHub Copilot output**:





**Output:**



**Explanation:**

1.

```python

def add(a, b):

return a + b

```

- This defines a function named `add` that takes two arguments, `a` and `b`, and returns their sum.

2.

```python

num1 = float(input("Enter first number: "))

```

- This line prompts the user to enter the first number. The input is converted from a string to a float and stored in `num1`.

3.

```python

num2 = float(input("Enter second number: "))

```

- This line prompts the user to enter the second number. The input is converted to a float and stored in `num2`.

4.

```python

print("Sum is:", add(num1, num2))

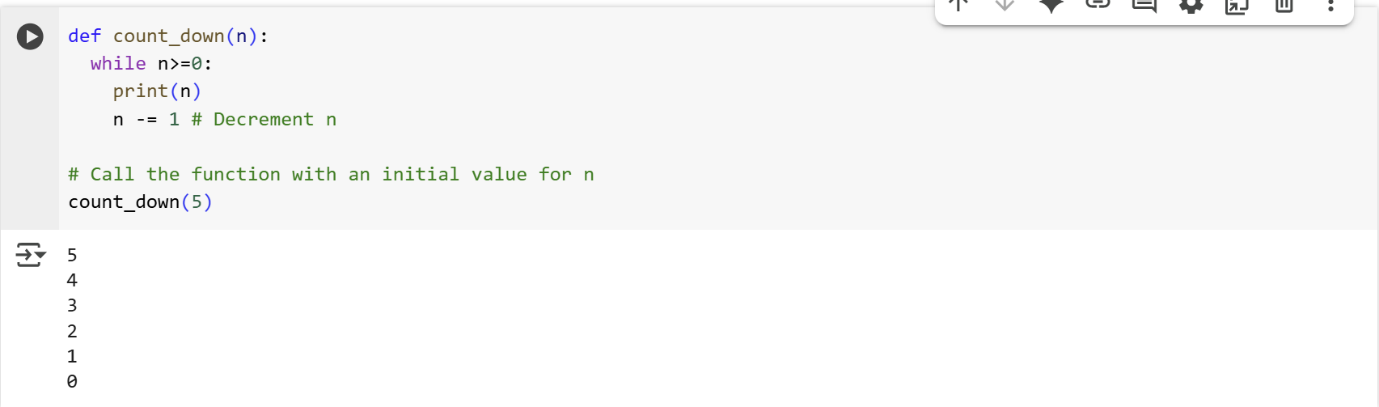
```

- This line calls the `add` function with `num1` and `num2` as arguments and prints the result.

\*\*Error Fix Explanation:\*\*

If there was an error previously, it was likely due to not converting the input to a numeric type (e.g., using `input()` without `float()` or `int()`), which would cause string concatenation instead of numeric addition. By converting the inputs to `float`, the code correctly adds the numbers.

**Task-2:**

**Gemini output**:

**Explanation:**

This code defines a function called count\_down that takes one argument, n, and prints numbers from n down to 0.

def count\_down(n):  
  while n>=0:  
    print(n)  
    n -= 1 # Decrement n

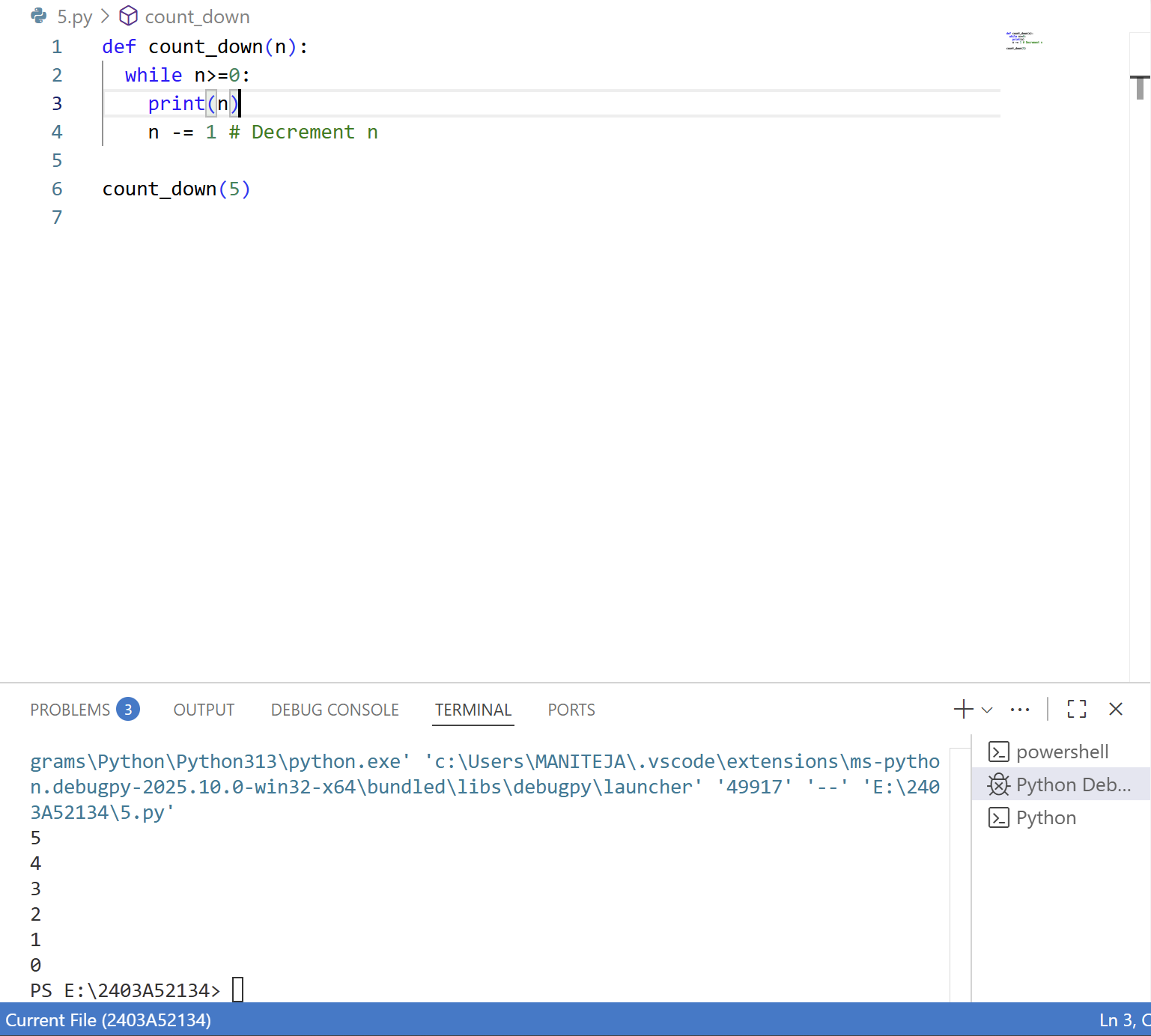
* def count\_down(n): defines a function named count\_down that accepts one parameter, n.
* while n>=0: starts a while loop that continues as long as the value of n is greater than or equal to 0.
* print(n): Inside the loop, the current value of n is printed.
* n -= 1: After printing, the value of n is decreased by 1. This is the decrement step that moves the count down towards 0.

# Call the function with an initial value for n  
count\_down(5)

* count\_down(5): This line calls the count\_down function with an initial value of 5 for n. The function will then print 5, 4, 3, 2, 1, and 0.

In summary, the count\_down function uses a while loop to repeatedly print the value of n and then decrease n by 1 until n becomes less than 0, at which point the loop terminates.

**GitHub copilot output:**



**Explanation:**

This code defines a function called count\_down that takes an integer n as input and prints numbers from n down to 0, one per line.

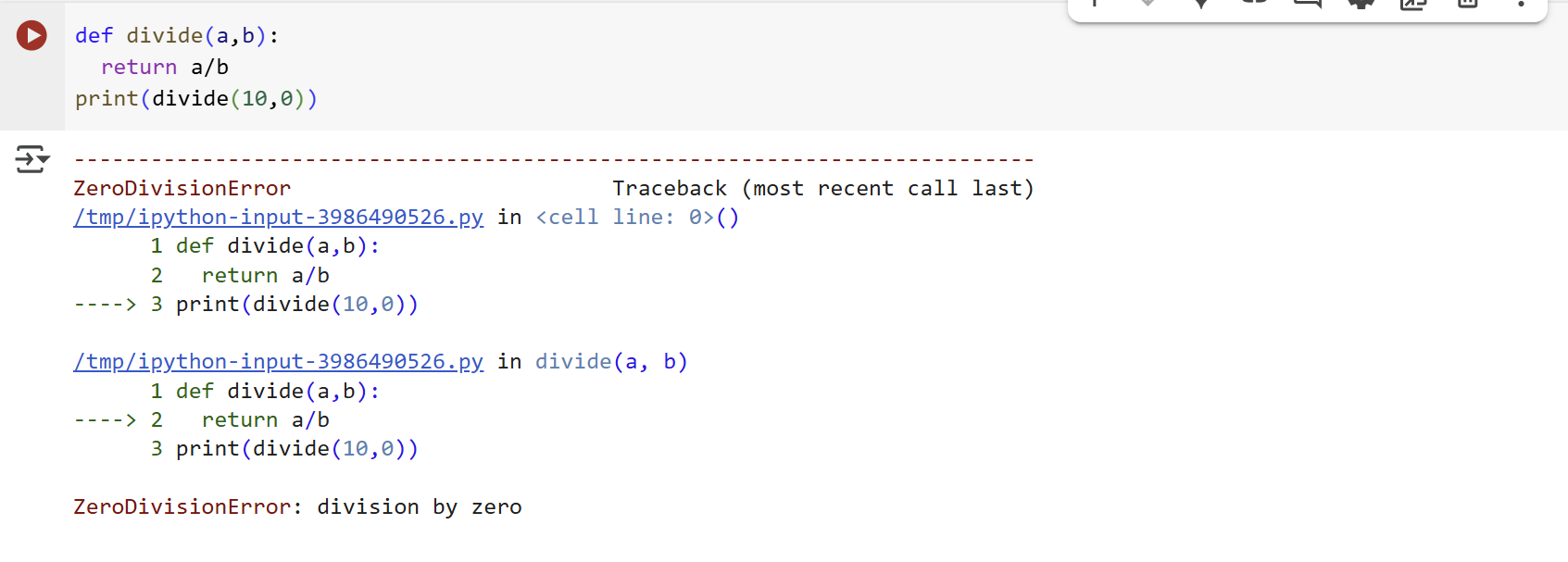
The while n>=0: loop continues as long as n is greater than or equal to 0.

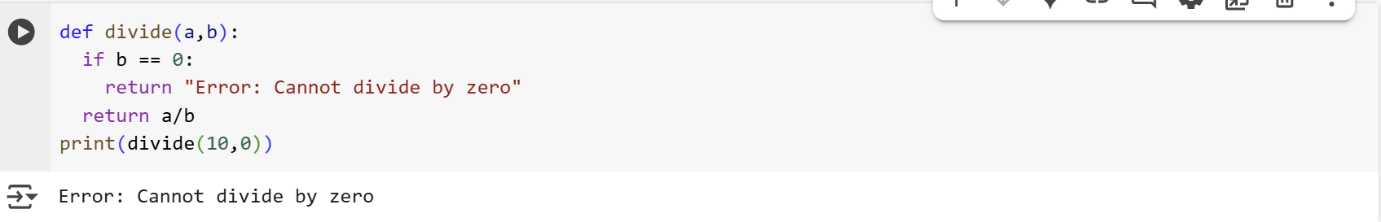
Inside the loop, it prints the current value of n and then decreases n by 1.

After defining the function, count\_down(5) is called, so the output will be:

**Task-3:**

**Gemini output:**





**Explanation:**

* def divide(a,b): defines a function named divide that accepts two parameters, a and b.
* if b == 0: checks if the value of b is equal to 0.
* return "Error: Cannot divide by zero": If b is 0, the function returns the string "Error: Cannot divide by zero". This prevents the division from happening.
* return a/b: If b is not 0, the function performs the division of a by b and returns the result.

print(divide(10,0))

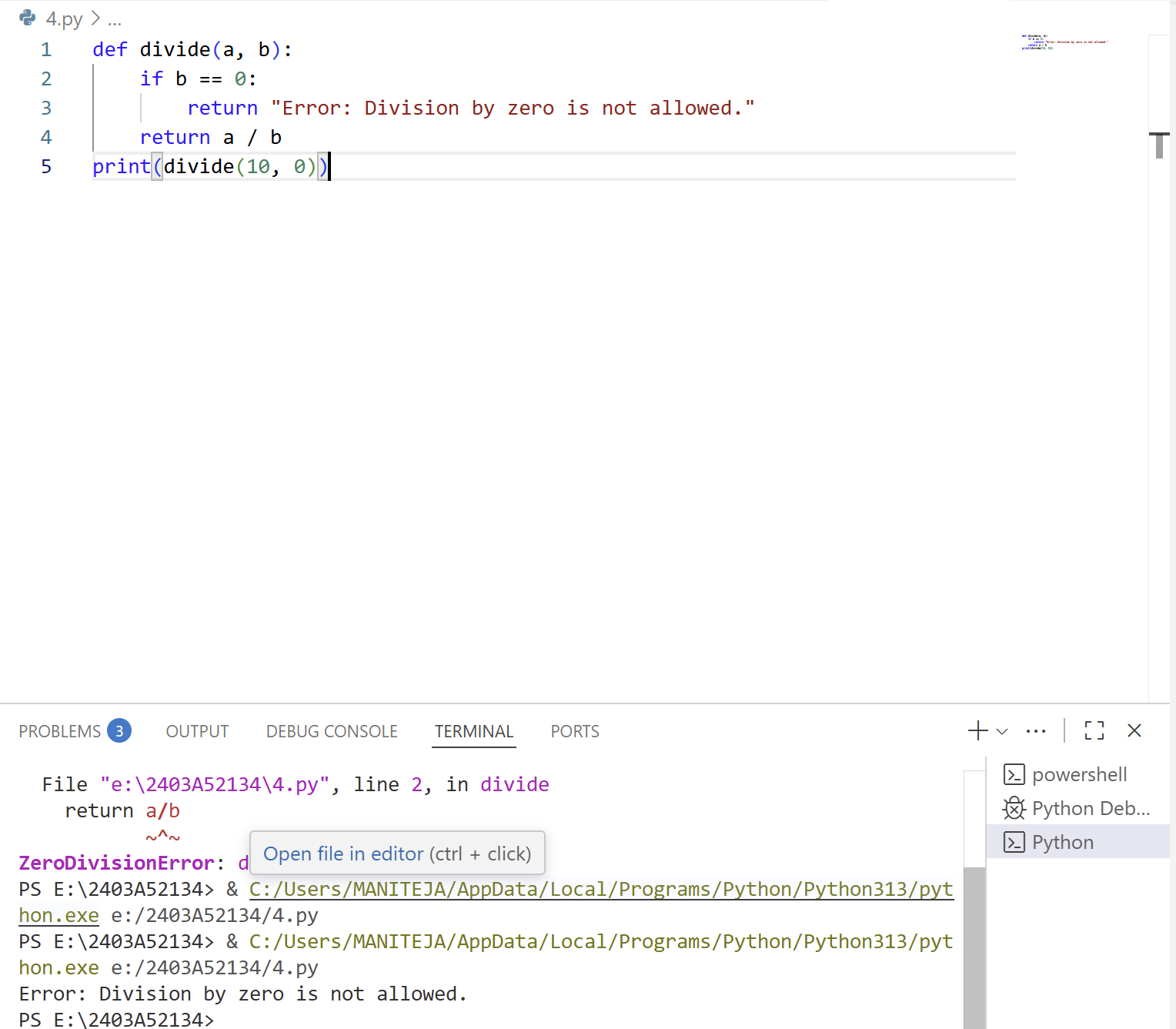
* print(divide(10,0)): This line calls the divide function with a=10 and b=0 and prints the returned value.

**Error Explanation and Fix:**

The original code likely did not have the if b == 0: check. When divide(10, 0) was called without this check, Python attempted to perform the division 10 / 0, which is mathematically undefined and results in a ZeroDivisionError.

The fix involved adding the if b == 0: condition. This check now intercepts the case where b is zero and returns an informative error message instead of attempting the division, thus preventing the ZeroDivisionError.

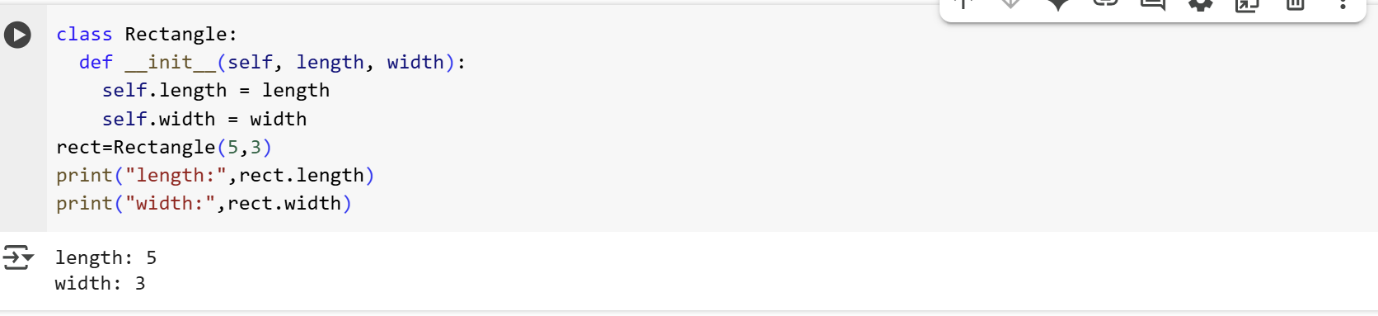
**GitHub copilot output**:



**Explanation:**

This code defines a function called divide that takes two arguments, a and b. Inside the function, it first checks if b is zero. If so, it returns an error message saying division by zero is not allowed. Otherwise, it returns the result of dividing a by b. The last line calls the divide function with arguments 10 and 0, so the function returns the error message instead of attempting the division. This prevents a runtime error that would occur if you tried to divide by zero.

**Task-4:**

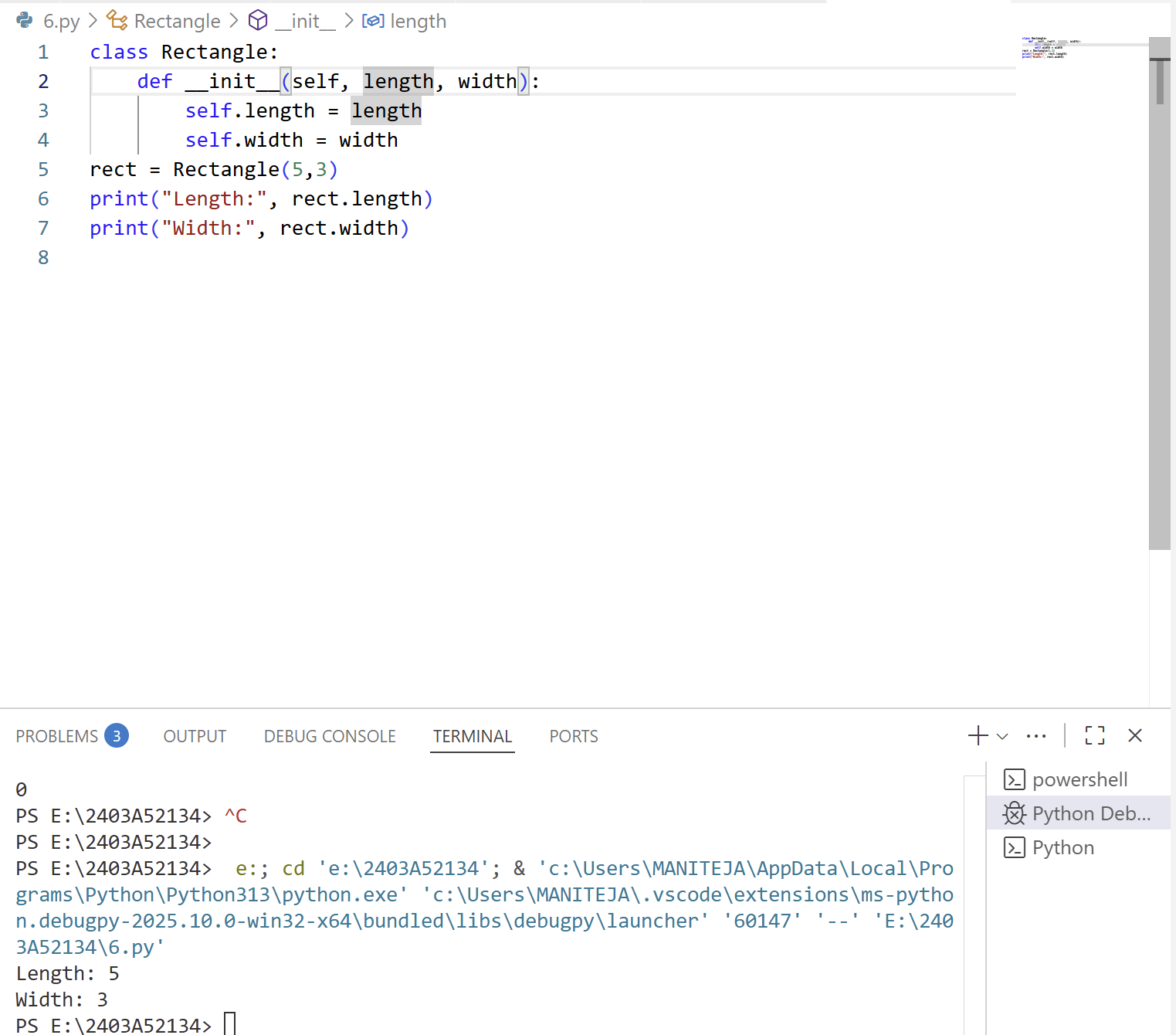
**Gemini output:**

**Explanation:**

* class Rectangle:: This line defines a new class named Rectangle. Classes are blueprints for creating objects (instances).
* def \_\_init\_\_(self, length, width):: This defines a special method called the constructor (\_\_init\_\_). This method is automatically called when you create a new object of the Rectangle class.
  + self: This is a reference to the instance of the class being created. It's the first parameter in instance methods.
  + length, width: These are parameters that will hold the values for the length and width of the rectangle when you create an object.
* self.length = length: This line takes the value passed for the length parameter and assigns it to an attribute of the object being created, also named length. self.length refers to the length attribute of the specific Rectangle object.
* self.width = width: Similarly, this line takes the value passed for the width parameter and assigns it to the width attribute of the object.
* rect = Rectangle(5, 3): This line creates a new instance (object) of the Rectangle class.
  + Rectangle(5, 3) calls the \_\_init\_\_ method of the Rectangle class.
  + 5 is passed as the value for the length parameter, and 3 is passed as the value for the width parameter.
  + The newly created Rectangle object is assigned to the variable rect.

After this code runs, the variable rect holds a Rectangle object with a length attribute of 5 and a width attribute of 3

**GitHub copilot output**:



**Explanation:**

```python

class Rectangle:

```

Defines a new class named `Rectangle`.

```python

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

```

Defines the constructor method (`\_\_init\_\_`) that runs when a new `Rectangle` object is created. It takes `length` and `width` as parameters.

```python

self.length = length

```

Assigns the value of `length` to the instance variable `self.length`.

```python

self.width = width

```

Assigns the value of `width` to the instance variable `self.width`.

```python

rect = Rectangle(5,3)

```

Creates a new `Rectangle` object named `rect` with `length` 5 and `width` 3.

```python

print("Length:", rect.length)

```

Prints the string "Length:" followed by the value of `rect.length` (which is 5).

```python

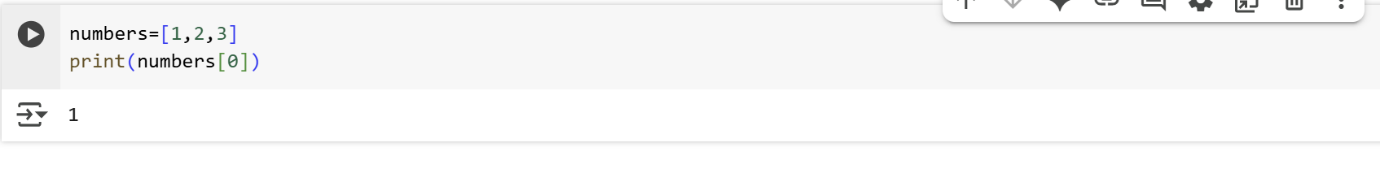
print("Width:", rect.width)

```

Prints the string "Width:" followed by the value of `rect.width` (which is 3).

**Task-5:**

**Gemini output**:



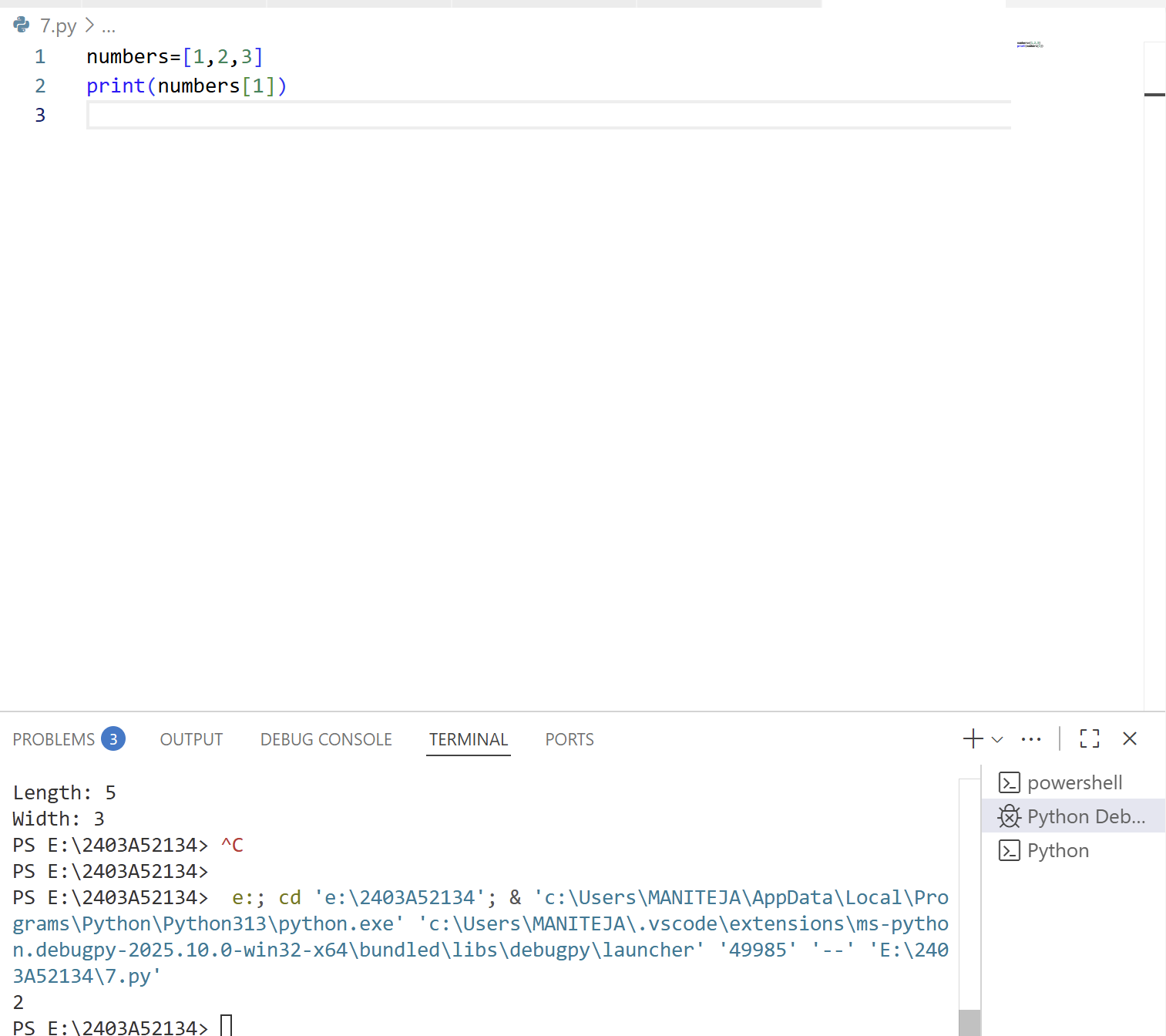
**Explanation:**

numbers=[1,2,3]  
print(numbers[0])

* numbers=[1,2,3]: This line creates a list named numbers and initializes it with three integer elements: 1, 2, and 3. Lists in Python are ordered collections of items, and they are indexed starting from 0. So, the element 1 is at index 0, 2 is at index 1, and 3 is at index 2.
* print(numbers[0]): This line accesses and prints an element from the numbers list.
  + numbers[0] uses square brackets [] with an index (in this case, 0) to access the element at that specific position within the list.
  + Since lists are zero-indexed, numbers[0] refers to the first element in the list, which is the number 1.
  + print() then displays the value of this element to the console.

In summary, this code creates a list of numbers and then prints the first element of that list by accessing it using its index.

**GitHub copilot output**:



**Explanation:**

This code creates a list called numbers containing three integers: 1, 2, and 3.

It then prints the value at index 1 of the list, which is 2.