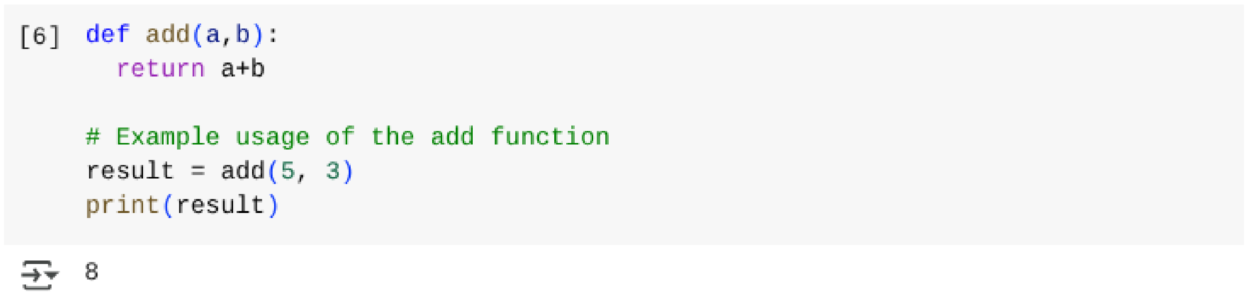
ASSIGNMENT-7

Task-1:

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

Please identify the error and correct it

Code and Output:

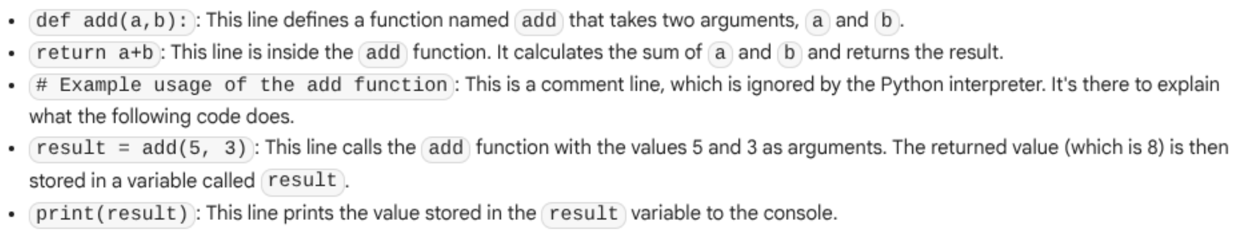


```
[6] def add(a,b):
      return a+b

    # Example usage of the add function
    result = add(5, 3)
    print(result)
```

```
8
```

Explanation:



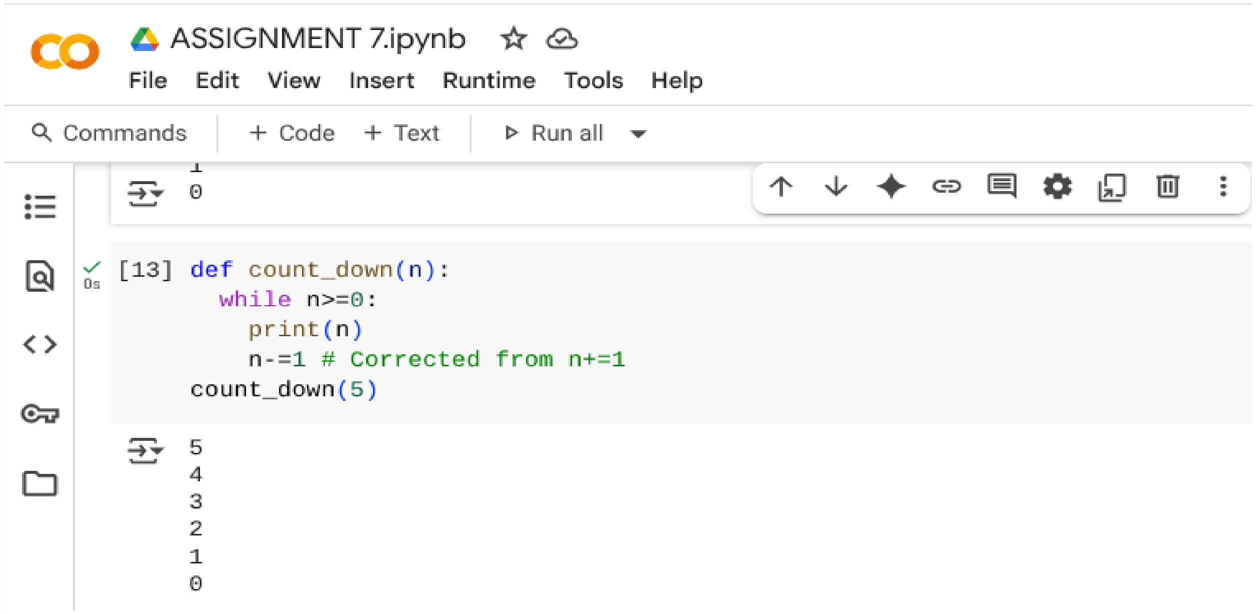
- `def add(a,b):`: This line defines a function named `add` that takes two arguments, `a` and `b`.
- `return a+b`: This line is inside the `add` function. It calculates the sum of `a` and `b` and returns the result.
- `# Example usage of the add function`: This is a comment line, which is ignored by the Python interpreter. It's there to explain what the following code does.
- `result = add(5, 3)`: This line calls the `add` function with the values 5 and 3 as arguments. The returned value (which is 8) is then stored in a variable called `result`.
- `print(result)`: This line prints the value stored in the `result` variable to the console.

Task-2:

```
def count_down(n):
while n>=0:
print(n)
n+=1
count_down(5)
```

Identify the error and correct it

Code and Output:


ASSIGNMENT 7.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```
1
0
```

```
[13] def count_down(n):
       while n>=0:
         print(n)
         n-=1 # Corrected from n+=1
     count_down(5)
```

```
5
4
3
2
1
0
```

Explanation:

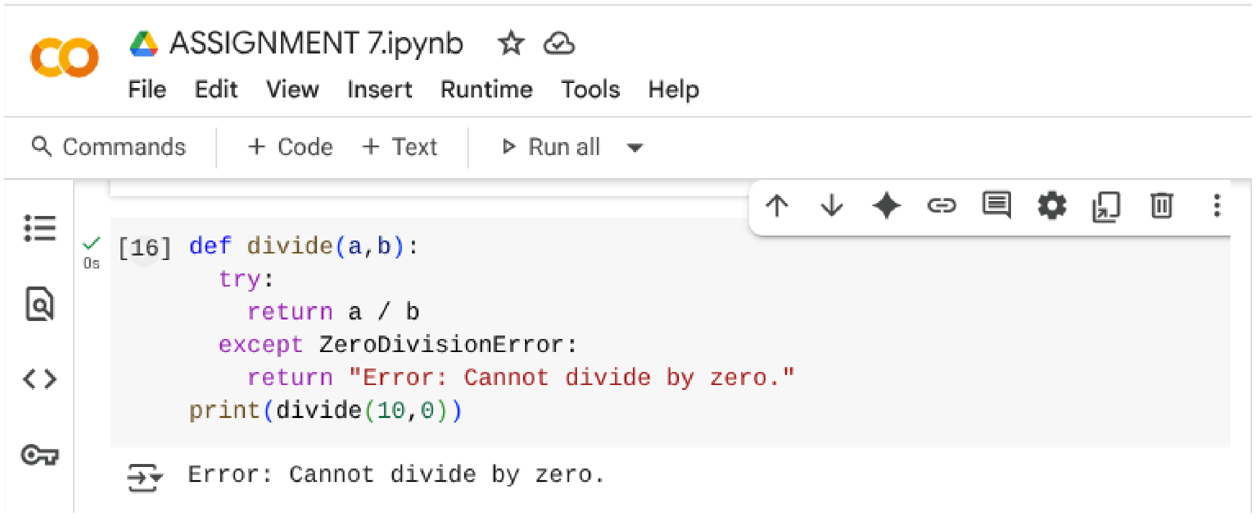
- `def count_down(n):` : This line defines a function named `count_down` that takes one argument, `n`.
- `while n>=0:` : This line starts a `while` loop. The code inside this loop will continue to execute as long as the value of `n` is greater than or equal to 0.
- `print(n)` : This line prints the current value of `n` to the console.
- `n-=1 # Corrected from n+=1` : This line subtracts 1 from the current value of `n` and updates `n` with the new value. This is the correction that was made to fix the infinite loop. The comment indicates what the original code was.
- `count_down(5)` : This line calls the `count_down` function with the value 5 as the argument, starting the countdown process from 5.

Task-3:

```
def divide(a,b):
return a / b
print(divide(10,0))
```

Identify the error and fix it

Code and Output:


ASSIGNMENT 7.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```
[16] def divide(a,b):
       try:
         return a / b
       except ZeroDivisionError:
         return "Error: Cannot divide by zero."
     print(divide(10,0))
```

```
Error: Cannot divide by zero.
```

Explanation:

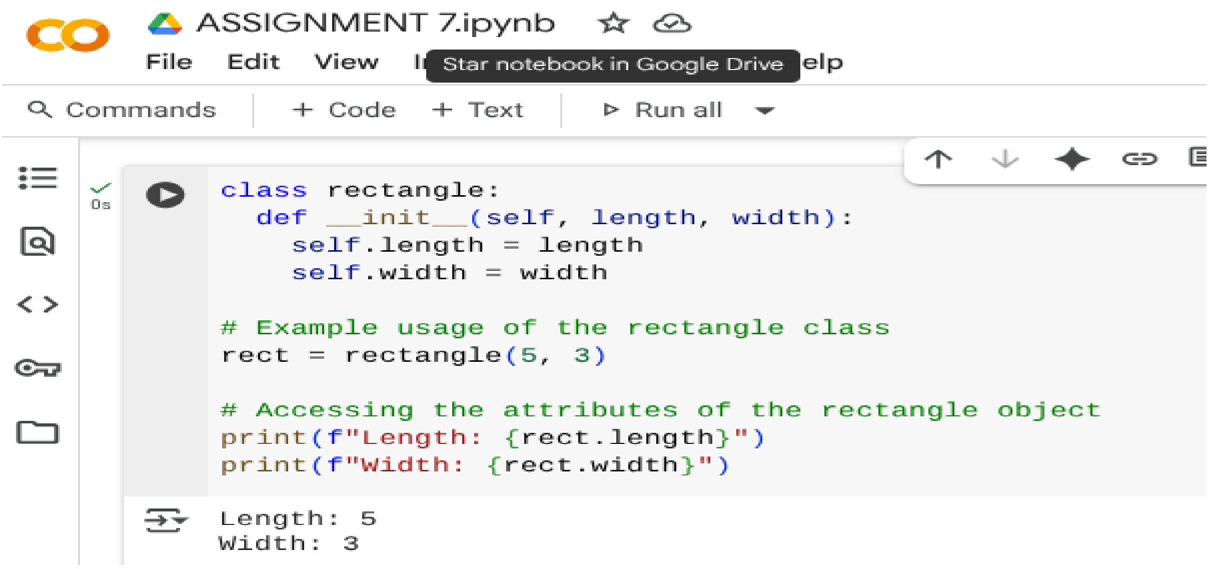
- `def divide(a,b):`: This line defines a function named `divide` that takes two arguments, `a` and `b`.
- `try:`: This keyword starts a `try` block. The code inside the `try` block is where you put the code that might cause an error.
- `return a / b`: This line is inside the `try` block. It attempts to divide `a` by `b` and return the result. This is the line that could potentially cause a `ZeroDivisionError` if `b` is 0.
- `except ZeroDivisionError:`: This keyword starts an `except` block. If a `ZeroDivisionError` occurs in the `try` block, the code inside this `except` block is executed.
- `return "Error: Cannot divide by zero."`: This line is inside the `except` block. If a `ZeroDivisionError` is caught, this line returns the string "Error: Cannot divide by zero." instead of crashing the program.
- `print(divide(10,0))`: This line calls the `divide` function with `a=10` and `b=0`. Since `b` is 0, a `ZeroDivisionError` will occur inside the `try` block, and the `except` block will be executed, causing the function to return the string "Error: Cannot divide by zero.", which is then printed to the console.

Task-4:

```
class rectangle:
def __init__(length, width):
self.length = length
self.width = width
rect = rectangle(5,3)
```

Identify the error and correct it

Code and Output:



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

# Example usage of the rectangle class
rect = rectangle(5, 3)

# Accessing the attributes of the rectangle object
print(f"Length: {rect.length}")
print(f"Width: {rect.width}")
```

```
Length: 5
Width: 3
```

Explanation:

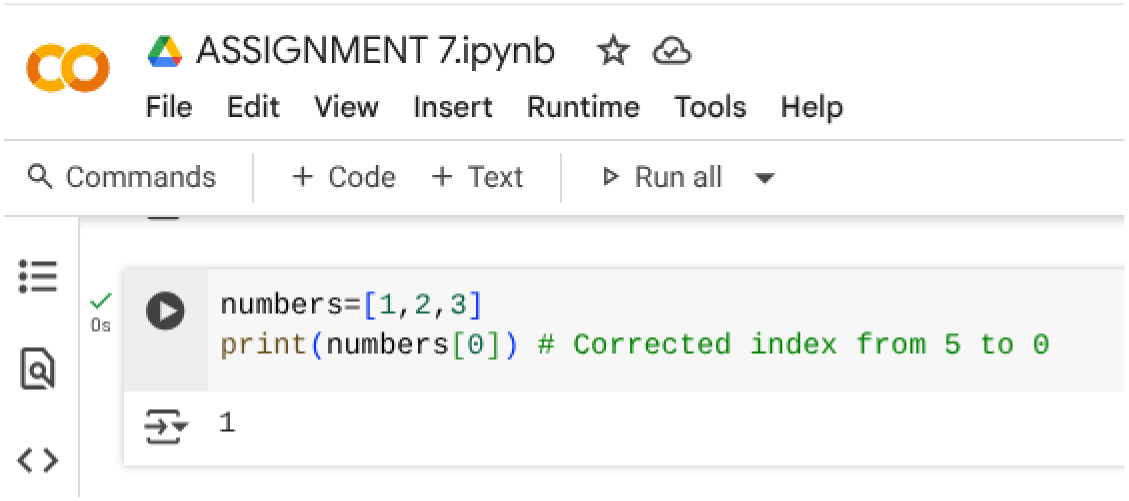
- `class rectangle:`: This line defines a new blueprint for creating objects, and we are naming this blueprint `rectangle`.
- `def __init__(self, length, width):`: This defines a special method that runs automatically when you create a new `rectangle` object. It takes the object itself (`self`), and the `length` and `width` you provide when creating the object.
- `self.length = length`: This line takes the `length` value you provided and stores it as a property of the `rectangle` object being created.
- `self.width = width`: This line takes the `width` value you provided and stores it as a property of the `rectangle` object.
- `rect = rectangle(5, 3)`: This line creates a specific instance of our `rectangle` blueprint. We're making a rectangle object with a length of 5 and a width of 3, and we're calling this object `rect`.
- `print(f"Length: {rect.length}")`: This line gets the `length` property from our `rect` object and prints it out in a sentence.
- `print(f"Width: {rect.width}")`: This line gets the `width` property from our `rect` object and prints it out in a sentence.

Task-5:

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

Identify the error and correct it

Code and Output:


ASSIGNMENT 7.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
0s

```
numbers=[1,2,3]
print(numbers[0]) # Corrected index from 5 to 0
```

```
1
```

Explanation:

- `numbers=[1,2,3]`: This line creates a list named `numbers` and initializes it with three integer values: 1, 2, and 3.
- `print(numbers[0]) # Corrected index from 5 to 0`: This line accesses an element from the `numbers` list using its index. In Python, list indices start from 0. `numbers[0]` refers to the first element in the list, which is 1. The comment `# Corrected index from 5 to 0` indicates that the original code attempted to access index 5, which is outside the bounds of this list and would cause an `IndexError`. This line now correctly accesses and prints the element at index 0.