Assignment-7-3

Task 1:

I have a Python function with a syntax error. The function add(a, b) is written without a colon. First detect what the error is. Then correct the function definition. Explain clearly what the mistake is and why it causes a syntax error. Give the correct code and a simple explanation of the fix.

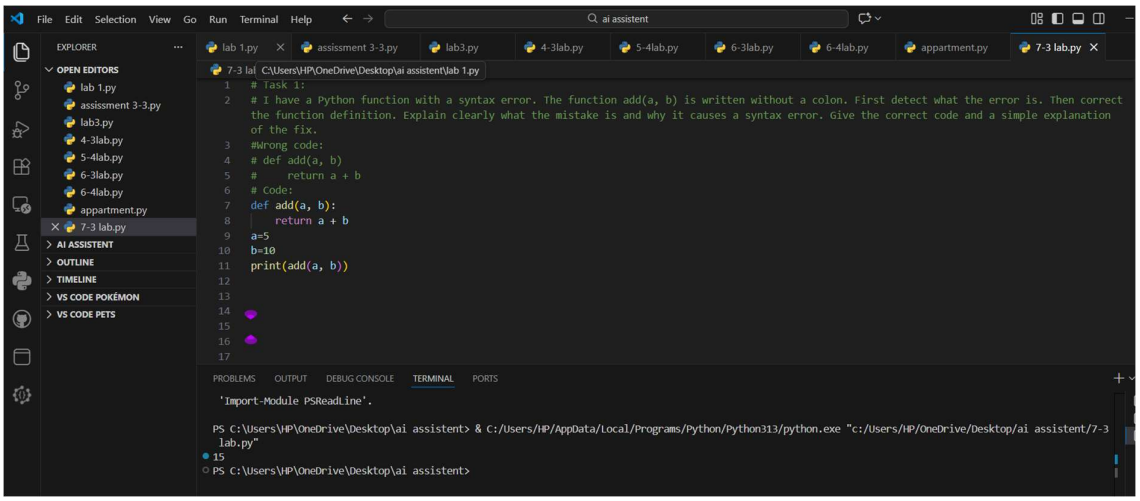
Wrong code:

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

Code:

```
def add(a, b):
    return a + b
a=5
b=10
print(add(a, b))
```

Output:



Analysis:

By adding the colon at the end of the function header, we correctly indicate that the indented block of code that follows is the body of the function. This allows Python to parse the function definition correctly and execute the code without any syntax errors.

Task 2:

I have a Python loop that runs infinitely due to a logic mistake. Find the error, fix it, and explain simply why the loop was infinite and how the correction solves it.

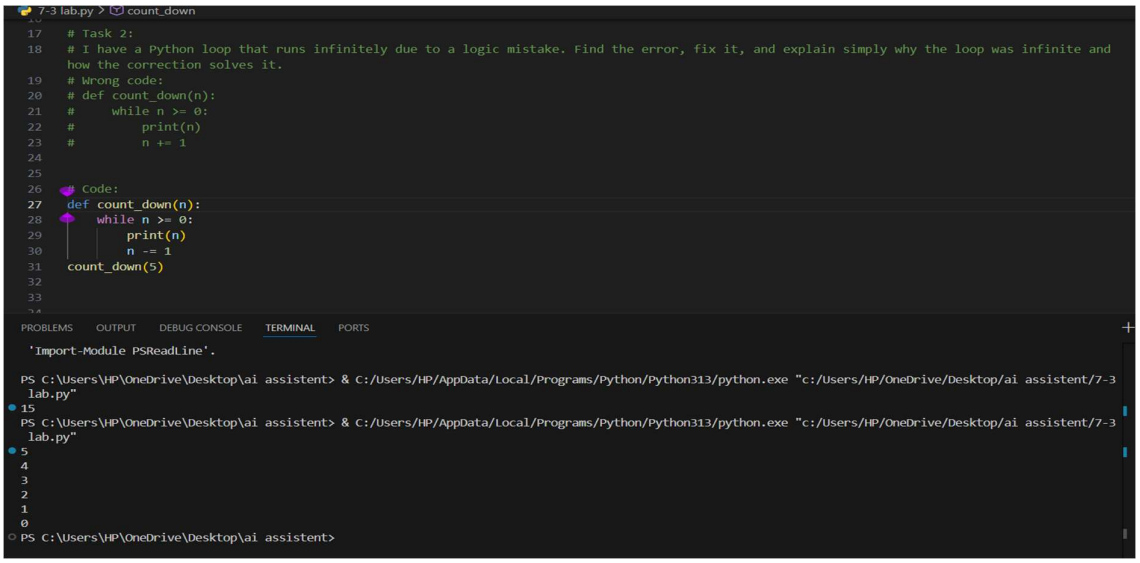
Wrong code:

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

Code:

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

Output:



Analysis:

The original code was incrementing n (n += 1) instead of decrementing it (n -= 1). This caused the value of n to increase indefinitely, resulting in an infinite loop. By changing it to n -= 1, we ensure that n decreases with each iteration, eventually reaching a point where n is less than 0, which will terminate the loop.

Task 3:

I have a Python function that performs division but it crashes at runtime. Identify the runtime error, fix it using try-except, and explain simply how the error is handled.

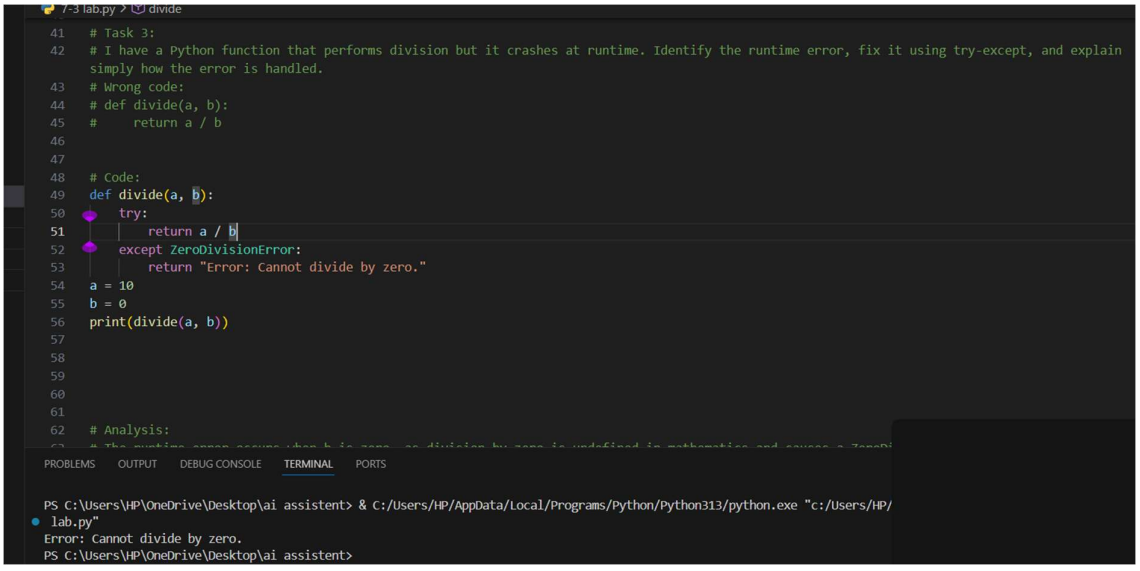
Wrong code:

```
def divide(a, b):
    return a / b
```

Code:

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

Output:



Analysis:

The runtime error occurs when b is zero, as division by zero is undefined in mathematics and causes a ZeroDivisionError in Python. By using a try-except block, we can catch this specific error and return a user-friendly message instead of crashing the program. This way, if the user tries to divide by zero, they will receive an informative message rather than an unhandled exception.

Task 4:

I have a Python class where the constructor is defined incorrectly. Find the error in the __init__() method, fix it by adding the missing parameter, and explain simply why self is required.

Wrong code:

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

Code:

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width
object1 = Rectangle(5, 10)
print(f"Length: {object1.length}, Width: {object1.width}")
```

Output:

Analysis:

The error in the original code is that the __init__() method is missing the 'self' parameter, which is required to refer to the instance of the class being created. The 'self' parameter allows us to access and modify the attributes of the instance. By adding 'self' as the first parameter, we can properly initialize the length and width attributes for each instance of the Rectangle class.

Task 5:

I have a Python program that crashes due to accessing an invalid list index. Identify the error, fix it using safe access (bounds checking or try-except), and explain simply how the error is handled.

Wrong code:

numbers = [1, 2, 3]

print(numbers[5])

Code:

numbers = [1, 2, 3]

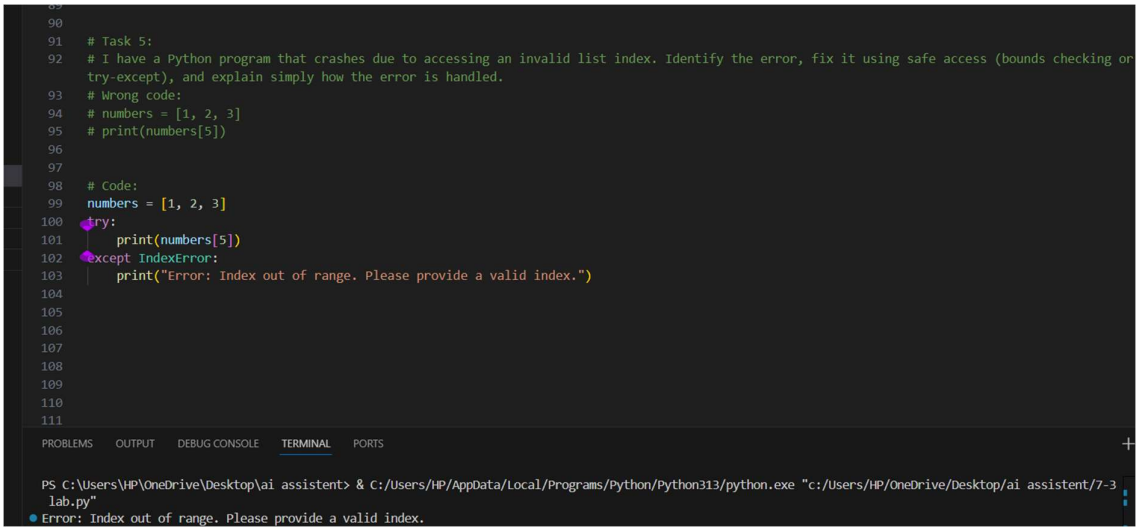
try:

    print(numbers[5])

except IndexError:

    print("Error: Index out of range. Please provide a valid index.")

Output:



Analysis:

The error occurs because we are trying to access an index (5) that does not exist in the list 'numbers', which only has indices 0, 1, and 2. This results in an IndexError. By using a try-except block, we can catch this specific error and provide a user-friendly message instead of crashing the program. This way, if the user tries to access an invalid index, they will receive an informative message rather than an unhandled exception.