Task Description#1

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

```
#Paste a function with a missing colon (add(a, b)), and let AI fix the syntax error.
def add(a, b):
    return a + b
# Example usage
result = add(5, 3)
print(f"The sum of 5 and 3 is: {result}")
#The AI has added the missing colon at the end of the function definition line.
# The AI has added the missing colon at the end of the function definition line.

```

The AI has added the missing colon at the end of the function definition line.

The AI has added the missing colon at the end of the function definition line.

Task Description#2 (Loops)

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

```
#Identify and fix a logic error in a loop that causes infinite iteration.
def countdown(n):
    while n > 0:
        print(n)
        n -= 1  # Fixed logic error: decrement n to avoid infinite loop
    print("Countdown finished!")
# Example usage
countdown(5)
# Explanation:
# The function countdown takes an integer n and prints numbers from n down to 1.

```

The function countdown takes an integer n and prints numbers from n down to 1.

Task Description#3

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

```
#Debug a runtime error caused by division by zero. Let AI insert try-except.
def divide_numbers(num1, num2):
    try:
        result = num1 / num2
        return result
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."
# Example usage
num1 = 10
num2 = 0
output = divide_numbers(num1, num2)
print(f"Dividing {num1} by {num2} gives: {output}")
# Example usage
num1 = 10
num2 = 2
output = divide_numbers(num1, num2)
print(f"Dividing {num1} by {num2} gives: {output}")
# The AI has inserted a try-except block to handle division by zero errors

```

The AI has inserted a try-except block to handle division by zero errors

Task Description#4

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

```
#Provide a faulty class definition (missing self in parameters). Let AI fix it
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def display_info(self):
        print(f"Car Make: {self.make}")
        print(f"Car Model: {self.model}")
        print(f"Car Year: {self.year}")
# Example usage
car1 = Car("Toyota", "Camry", 2020)
car1.display_info()
car2 = Car("Honda", "Civic", 2019)
car2.display_info()
car3 = Car("Ford", "Mustang", 2021)
car3.display_info()
#observation: The AI has added 'self' to the method parameters to fix the class definition.
```

observation: The AI has added 'self' to the method parameters to fix the class definition.

Task Description#5

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

```
#Access an invalid list index and use AI to resolve the Index Error.
def access_list_element(lst, index):
    try:
        return lst[index]
    except IndexError:
        return "Error: Index out of range."
# Example usage
my_list = [10, 20, 30, 40, 50]
index = 10  # Invalid index
element = access_list_element(my_list, index)
print(f"Accessing index {index} gives: {element}")
# Example usage with a valid index
index = 2  # Valid index
element = access_list_element(my_list, index)
print(f"Accessing index {index} gives: {element}")
# The AI has added a try-except block to handle IndexError exceptions.

```

The AI has added a try-except block to handle IndexError exceptions.