2303A51563
batch=10
Task 1: Fixing Syntax Errors
Scenario
You are reviewing a Python program where a basic function definition contains a syntax error.
Requirements
• Provide a Python function add(a, b) with a missing colon
• Use an AI tool to detect the syntax error
• Allow AI to correct the function definition
• Observe how AI explains the syntax issue
Expected Output
• Corrected function with proper syntax
• Syntax error resolved successfully
• AI-generated explanation of the fix

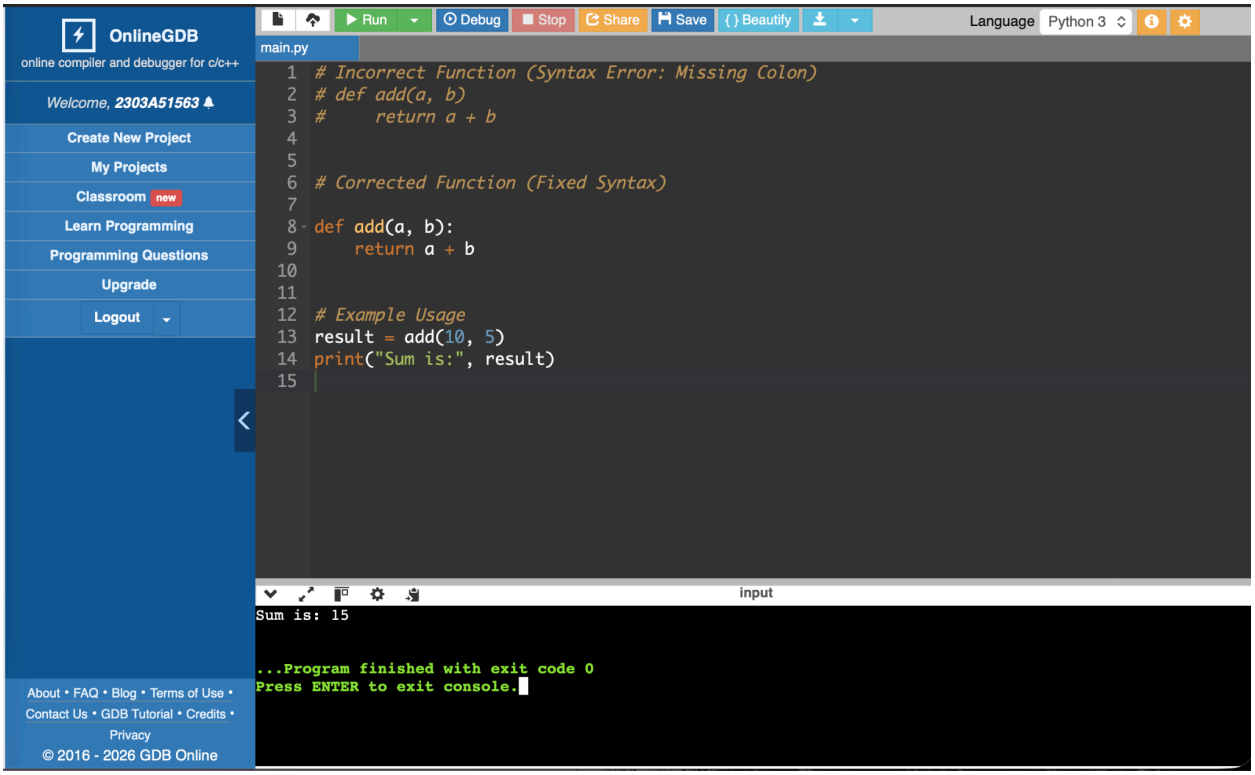
```
# Incorrect Function (Syntax Error: Missing Colon)
# def add(a, b)
#     return a + b


# Corrected Function (Fixed Syntax)

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


# Example Usage
```

result = add(10, 5)
print("Sum is:", result)

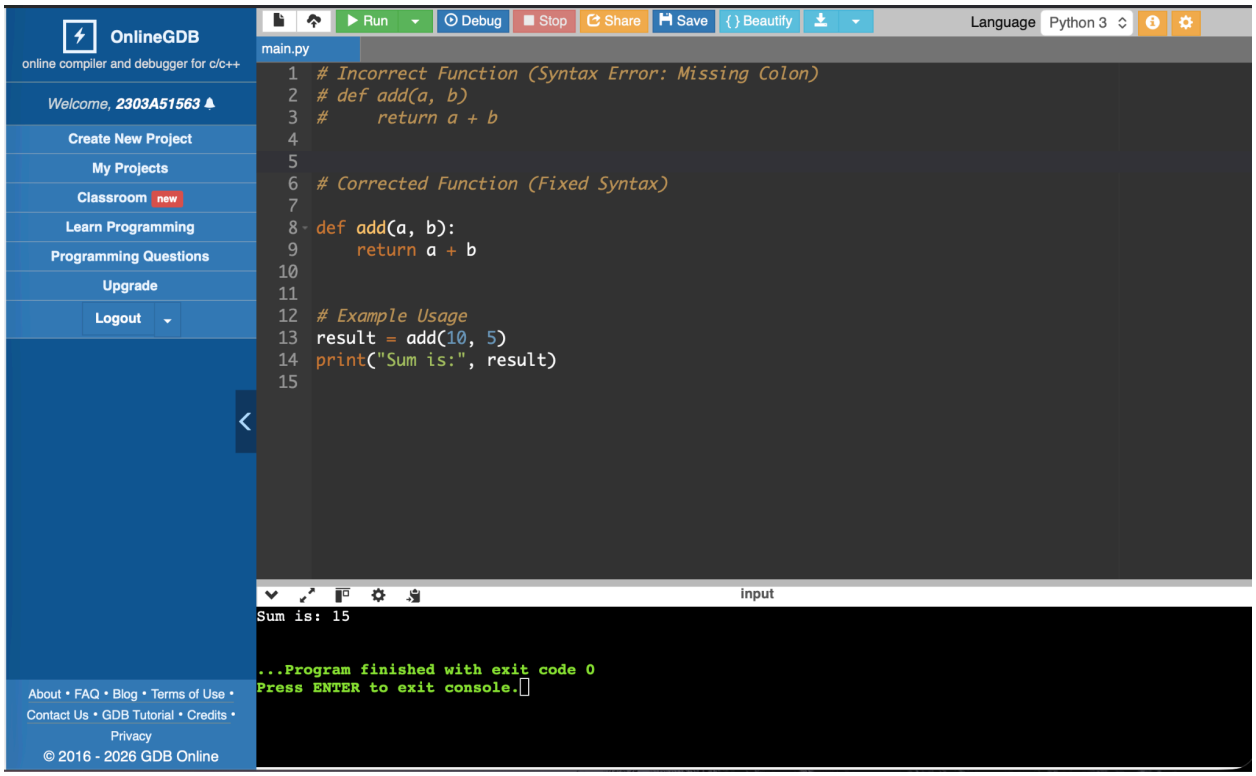


Task 2: Debugging Logic Errors in Loops
Scenario
You are debugging a loop that runs infinitely due to a logical mistake.
Requirements
• Provide a loop with an increment or decrement error
• Use AI to identify the cause of infinite iteration
• Let AI fix the loop logic
• Analyze the corrected loop behavior
Expected Output
• Infinite loop issue resolved
• Correct increment/decrement logic applied
• AI explanation of the logic error
Code
Incorrect Function (Syntax Error: Missing Colon)
def add(a, b)
return a + b

Corrected Function (Fixed Syntax)

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


# Example Usage
result = add(10, 5)
print("Sum is:", result)
```



Task 3: Handling Runtime Errors (Division by Zero)

Scenario

A Python function crashes during execution due to a division by zero error.

Requirements

• Provide a function that performs division without validation
• Use AI to identify the runtime error
• Let AI add try-except blocks for safe execution
• Review AI’s error-handling approach

Expected Output

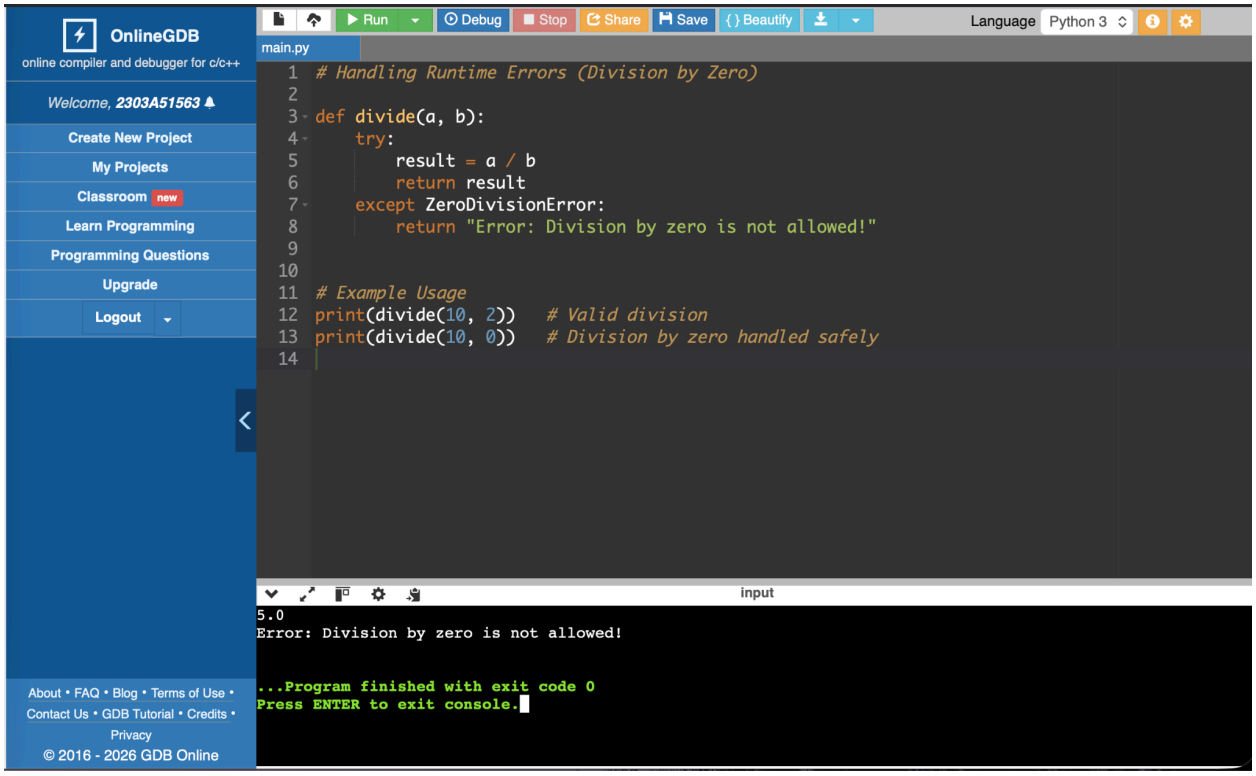
• Function executes safely without crashing
• Division by zero handled using try-except
• Clear AI-generated explanation of runtime error handling

```
# Handling Runtime Errors (Division by Zero)

def divide(a, b):
```

```
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed!"


# Example Usage
print(divide(10, 2))   # Valid division
print(divide(10, 0))   # Division by zero handled safely
```



ask 4: Debugging Class Definition Errors

Scenario

You are given a faulty Python class where the constructor is incorrectly defined.

Requirements

• Provide a class definition with missing self-parameter
• Use AI to identify the issue in the __init__() method
• Allow AI to correct the class definition
• Understand why self is required

Expected Output

• Corrected __init__() method
• Proper use of self in class definition
• AI explanation of object-oriented error

```
# Debugging Class Definition Errors (Missing self)
```

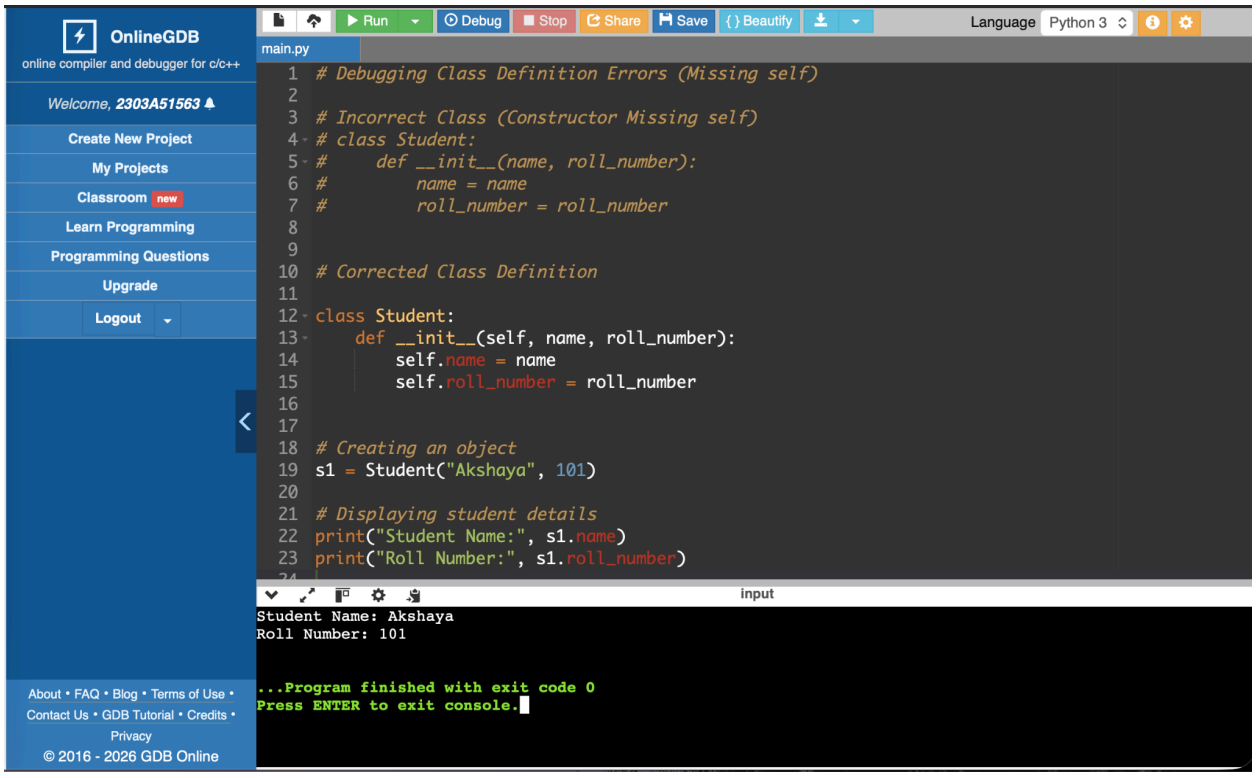
```
# Incorrect Class (Constructor Missing self)
# class Student:
#     def __init__(name, roll_number):
#         name = name
#         roll_number = roll_number


# Corrected Class Definition

class Student:
    def __init__(self, name, roll_number):
        self.name = name
        self.roll_number = roll_number


# Creating an object
s1 = Student("Akshaya", 101)

# Displaying student details
print("Student Name:", s1.name)
print("Roll Number:", s1.roll_number)
```



ask 5: Resolving Index Errors in Lists

Scenario
A program crashes when accessing an invalid index in a list.
Requirements
• Provide code that accesses an out-of-range list index
• Use AI to identify the Index Error
• Let AI suggest safe access methods
• Apply bounds checking or exception handling
Expected Output
• Index error resolved
• Safe list access logic implemented
• AI suggestion using length checks or exception handling
Code

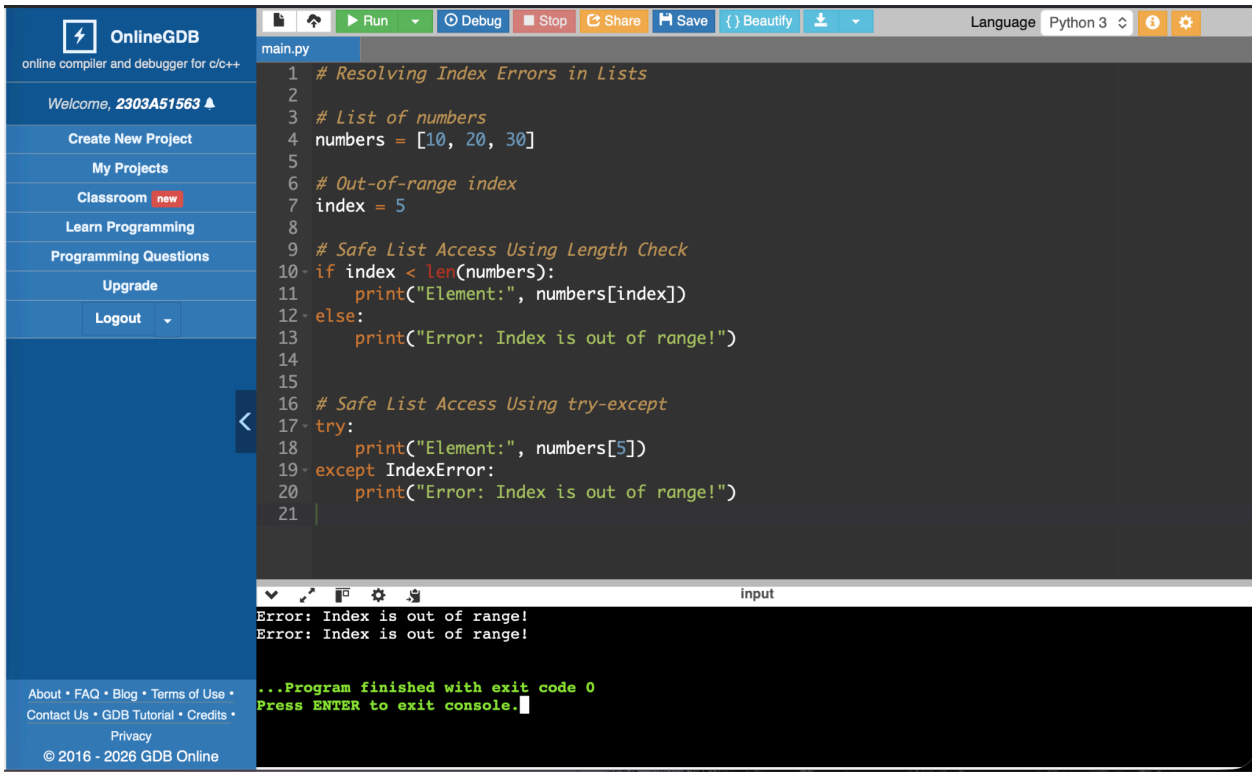
```
# Resolving Index Errors in Lists

# List of numbers
numbers = [10, 20, 30]

# Out-of-range index
index = 5

# Safe List Access Using Length Check
if index < len(numbers):
    print("Element:", numbers[index])
else:
    print("Error: Index is out of range!")


# Safe List Access Using try-except
try:
    print("Element:", numbers[5])
except IndexError:
    print("Error: Index is out of range!")
```


OnlineGDB
online compiler and debugger for c/c++
Welcome, 2303A51563
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2026 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
# Resolving Index Errors in Lists
# List of numbers
numbers = [10, 20, 30]
# Out-of-range index
index = 5
# Safe List Access Using Length Check
if index < len(numbers):
print("Element:", numbers[index])
else:
print("Error: Index is out of range!")
# Safe List Access Using try-except
try:
print("Element:", numbers[5])
except IndexError:
print("Error: Index is out of range!")
input
Error: Index is out of range!
Error: Index is out of range!
...Program finished with exit code 0
Press ENTER to exit console.