

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)
```

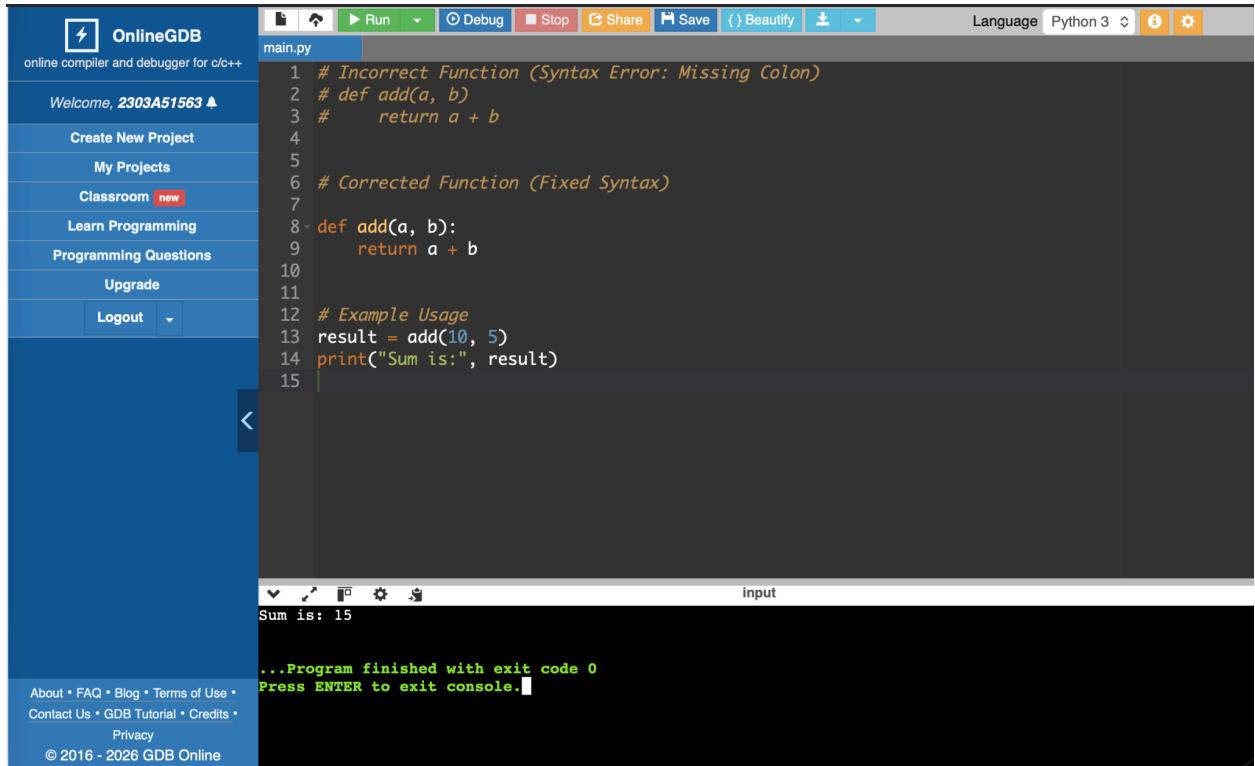
```
#     return a + b
```

```
# Corrected Function (Fixed Syntax)
```

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

```
# Example Usage
```

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



The screenshot shows the OnlineGDB interface. On the left, there's a sidebar with navigation links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area has tabs for 'main.py' and 'main.cpp'. The 'main.py' tab contains the following code:

```
1 # Incorrect Function (Syntax Error: Missing Colon)
2 # def add(a, b)
3 #     return a + b
4
5
6 # Corrected Function (Fixed Syntax)
7
8 def add(a, b):
9     return a + b
10
11
12 # Example Usage
13 result = add(10, 5)
14 print("Sum is:", result)
15
```

Below the code editor is a terminal window showing the output:

```
Sum is: 15
...
...Program finished with exit code 0
Press ENTER to exit console.
```

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)
```

The screenshot shows the OnlineGDB interface. On the left, there's a sidebar with options like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area has tabs for 'main.py' and 'main.cpp'. The 'main.py' tab contains the provided Python code. The code editor shows two versions of the 'add' function: one with a syntax error ('# Incorrect Function (Syntax Error: Missing Colon)') and one corrected ('# Corrected Function (Fixed Syntax)'). The corrected version runs successfully, outputting 'Sum is: 15'. Below the code editor is a terminal window showing the command 'input' and the output 'Sum is: 15'. At the bottom, there's a footer with links to 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact Us', 'GDB Tutorial', 'Credits', and 'Privacy', along with a copyright notice: '© 2016 - 2026 GDB Online'.

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

```

The screenshot shows the OnlineGDB interface. On the left is a sidebar with navigation links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', 'Logout', and footer links for 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact Us', 'GDB Tutorial', 'Credits', 'Privacy', and copyright information. The main area has tabs for 'main.py' and 'input'. The code in 'main.py' is:

```

# 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

```

The 'input' tab shows the output of the program:

```

5.0
Error: Division by zero is not allowed!

...Program finished with exit code 0
Press ENTER to exit console.

```

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)

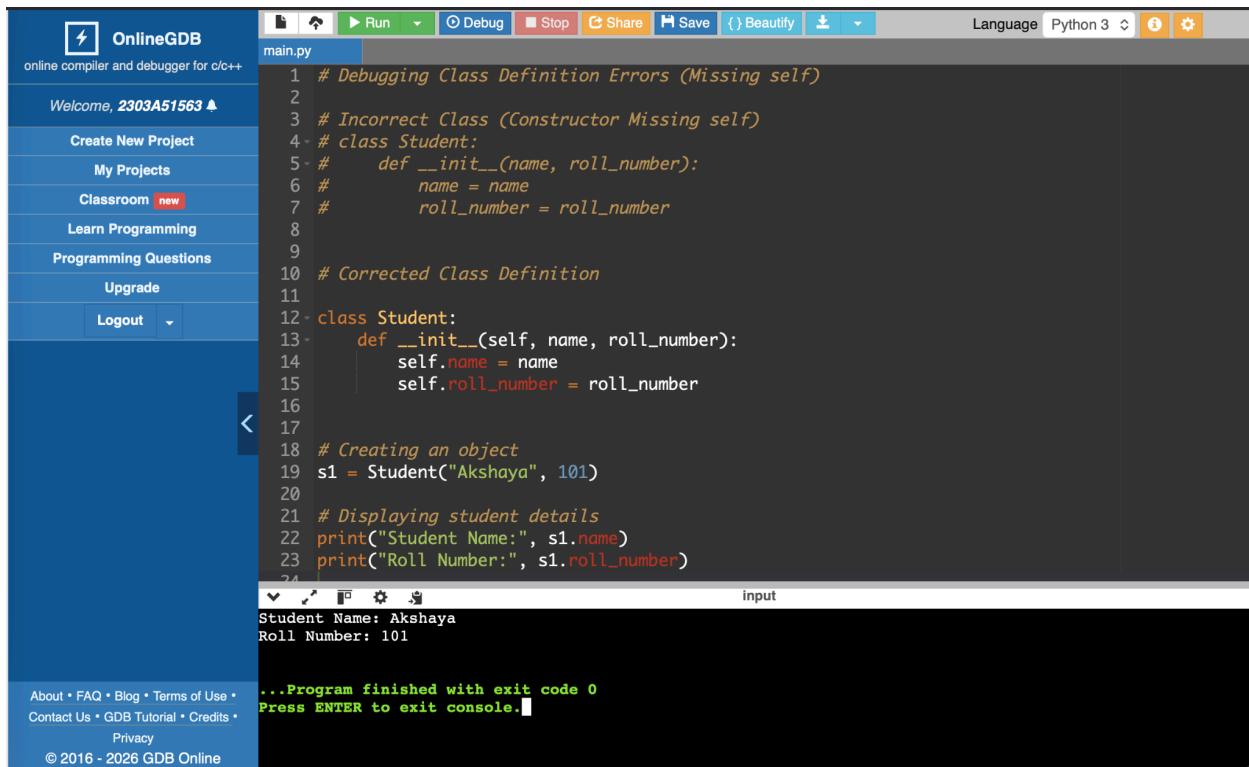
```
# Incorrect Class (Constructor Missing self)
# class Student:
#     def __init__(name, roll_number):
#         name = name
#         roll_number = roll_number
```

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)
```



The screenshot shows the OnlineGDB web interface. On the left, there's a sidebar with navigation links like 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area has a toolbar at the top with 'Run', 'Debug', 'Stop', 'Save', 'Beautify', and other icons. The code editor window is titled 'main.py' and contains the following Python code:

```
1 # Debugging Class Definition Errors (Missing self)
2
3 # Incorrect Class (Constructor Missing self)
4 # class Student:
5 #     def __init__(name, roll_number):
6 #         name = name
7 #         roll_number = roll_number
8
9
10 # Corrected Class Definition
11
12 class Student:
13     def __init__(self, name, roll_number):
14         self.name = name
15         self.roll_number = roll_number
16
17
18 # Creating an object
19 s1 = Student("Akshaya", 101)
20
21 # Displaying student details
22 print("Student Name:", s1.name)
23 print("Roll Number:", s1.roll_number)
24
```

Below the code editor is a terminal window showing the output of the program:

```
Student Name: Akshaya
Roll Number: 101
...Program finished with exit code 0
Press ENTER to exit console.
```

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!")
```

The screenshot shows the OnlineGDB interface. On the left, there's a sidebar with navigation links: Welcome, Create New Project, My Projects, Classroom (new), Learn Programming, Programming Questions, Upgrade, and Logout. The main area has tabs for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The Language is set to Python 3. The code editor contains a file named 'main.py' with the following content:

```
1 # Resolving Index Errors in Lists
2
3 # List of numbers
4 numbers = [10, 20, 30]
5
6 # Out-of-range index
7 index = 5
8
9 # Safe List Access Using Length Check
10 if index < len(numbers):
11     print("Element:", numbers[index])
12 else:
13     print("Error: Index is out of range!")
14
15
16 # Safe List Access Using try-except
17 try:
18     print("Element:", numbers[5])
19 except IndexError:
20     print("Error: Index is out of range!")
21
```

The output window below shows the results of running the code. It displays two error messages: "Error: Index is out of range!" followed by "...Program finished with exit code 0 Press ENTER to exit console.".