

Assignment-7.3

Task 1: Fixing Syntax Errors

Scenario

You are reviewing a Python program where a basic function definition contains a syntax error.

```
python

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

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

```
1 # Assume the python code and identify syntax error
2 # explain it and correct the code
3 #
4 #def (a,b)
5 #    return a + b
6 #
7 #The provided code has a syntax error in the function definition. The keyword "def" is used to define a function, but it is missing the function name. Additionally, the function body should be indented properly.
8 #Here is the corrected code:
9 def add(a, b):
10     return a + b
```

Task 2: Debugging Logic Errors in Loops

Scenario

You are debugging a loop that runs infinitely due to a logical mistake.

```
python

def count_down(n):
    while n >= 0:
        print(n)
        n += 1 # Should be n -= 1
```

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

```
1 #Observe the python code and identify logical error that causes the loop to run infinitely.
2 #Explain why the infinite loop occurs,correct the loop logic and describe how the corrected loop behaves
3
4 def count_down(n):
5     while n >= 0:
6         print(n)
7         n += 1
8
9 #Explain:
10 #The logical error in the code is that the variable "n" is being incremented ("n += 1") instead of decremented. This causes the loop to run indefinitely because "n" will always be greater than or equal to 0, and it will keep increasing indefinitely.
11 #To correct the loop logic, we should change the increment operation to a decrement operation. The corrected code would look like this:
12 #Corrected code:
13 def count_down(n):
14     while n >= 0:
15         print(n)
16         n -= 1
```

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

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

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

print(divide(10, 0))
```

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

```
File Today7.ipynb
1 #Analyze the following python code and identify run time error that occurs during execution .
2 #explain why the error happens ,then modify the code using try-except blocks to handle the error safely.
3 #Also explain how the error-handling solution works
4
5 def divide(a,b):
6     #return a/b
7     #print(divide(10,0))
8
9 #Explanation of the error:
10 #The code provided will raise a 'ZeroDivisionError' when it attempts to divide 10 by 0. This error occurs because division by zero is undefined in mathematics, and Python does not allow it.
11 #To handle this error safely, we can use a try-except block to catch the exception and provide a user-friendly message instead of crashing the program. Here's the modified code:
12
13 def divide(a, b):
14     try:
15         return a / b
16     except ZeroDivisionError:
17         return "Error: Cannot divide by zero."
18     print(divide(10, 0))
19
20 #In this modified code, the 'try' block contains the code that may raise an exception (the division operation). If a 'ZeroDivisionError' occurs, the control is transferred to the 'except' block,
21 #which catches the exception and returns a user-friendly error message instead of allowing the program to crash. This way, we can handle the error gracefully and provide feedback to the user without interrupting the flow of the program.
```

Task 4: Debugging Class Definition Errors

Scenario

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

python

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

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

The screenshot shows a terminal window with the following content:

```
PS C:\Users\ghgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/ghgu/AppData/Local/Programs/Python/Python312/python.exe "C:/Users/ghgu/OneDrive/Documents/Desktop/Assisted coding/7.3(4).py"
1 #Analyze the following python class definition and identify the error in the constructor method.
2 ##Explain why the error occurs ,correct the class definition ,and explain why the self parameter is required in python classes.
3
4 class Rectangle:
5     def __init__(length, width):
6         self.length = length
7         self.width = width
8
9 from copyreg import constructor
10
11 #Explanation of the error:
12 #The error in the constructor method of the Rectangle class is that the 'self' parameter is missing from the '__init__' method definition. In Python,
13 #the first parameter of any instance method (including the constructor) must be 'self', which refers to the instance of the class being created or manipulated.
14 #Corrected class definition:
15 class Rectangle:
16     def __init__(self, length, width):
17         self.length = length
18         self.width = width
19
20 #The 'self' parameter is required in Python classes because it allows the method to access and modify the instance's attributes. When you create an instance of a class, 'self' refers to that specific instance,
21 #enabling you to store and retrieve data unique to that instance. Without 'self', the method would not know which instance's attributes to access or modify, leading to errors when trying to use the class.
22 rect = Rectangle(5, 3)
23 print(rect.length) # Output: 5
24 print(rect.width) # Output: 3
25
```

Below the code, the terminal shows the following output:

```
PS C:\Users\ghgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/ghgu/AppData/Local/Programs/Python/Python312/python.exe "C:/Users/ghgu/OneDrive/Documents/Desktop/Assisted coding/7.3(4).py"
1
2
3
4
```

Task 5: Resolving Index Errors in Lists

Scenario

A program crashes when accessing an invalid index in a list.

```
python

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

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

The screenshot shows a code editor window with a Python script named `7.3(5).py`. The code is designed to demonstrate how to handle an index out of range error. It starts by importing the `os` module and defining a list `numbers` with elements 1, 2, and 3. A comment explains that the code will attempt to print the element at index 5. The code then uses a try-except block to catch the `IndexError`. Inside the try block, it prints the element at the specified index. In the except block, it catches the `IndexError` and prints a user-friendly message: "Index out of range. Please provide a valid index.". The code editor interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The bottom status bar shows the file path and the Python command-line interface (PS).

```
#!/usr/bin/python3
# Analyze the following python code and identify the error that occurs when accessing the list.
# Explain why the error happens, then modify the code to safely access the list using bounds checking or exception handling.
# Also explain how the safe access logic works.
# numbers = [1, 2, 3]
# print(numbers[5])

#Explanation of the error:
#The error that occurs when accessing the list is an "IndexError: list index out of range". This happens because the code is trying to access an index (5) that does not exist in the list (which only has indices 0, 1, and 2).
#To safely access the list, we can use bounds checking to ensure that the index we are trying to access is within the valid range of indices for the list. Alternatively, we can use exception handling to catch the error if it occurs.

numbers = [1, 2, 3]
index_to_access = 5
if index_to_access < len(numbers):
    print(numbers[index_to_access])
else:
    print("Index out of range. Please provide a valid index.")

#In this code, we first check if the index we want to access (5) is less than the length of the list (3). If it is, we proceed to access the element at that index.
#If it is not, we print a message indicating that the index is out of range. This way, we prevent the program from crashing due to an IndexError and provide a user-friendly message instead.

#Another approach is to use exception handling:
try:
    print(numbers[index_to_access])
except IndexError:
    print("Index out of range. Please provide a valid index.")

#In this code, we attempt to access the element at the specified index (5) within a try block. If the index is out of range, an IndexError will be raised, which we catch in the except block.
#We then print a message indicating that the index is out of range. This approach allows us to handle the error gracefully without crashing the program.

# . The cloud file provider is not running.
# At line 1 char 3
# File "C:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/7.3(5).py"
#   +-----+
#   | CategoryInfo : ObjectNotFound: (String[], CommandNotFoundException)
#   +-----+
#   + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding & C:\Users\gbsgu\AppData\Local\Programs\Python\Python312\python.exe "C:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/7.3(5).py"
Index out of range. Please provide a valid index.
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding
```

