

AI ASSISTED CODING LAB-7.3

Name:M.Shashidhar

Rollno:2303a52291 batch:42

Task 1: Fixing Syntax Errors

Scenario

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

Code with syntax error

```
25.py > add
 1 def add(a, b)      Expected ":" 
 2   return a + b    "return" can be used only within a function
```

Error Identified by AI

- **Type of Error:** SyntaxError
- **Reason:** Python requires a colon (:) at the end of a function definition line.

Code debugged by ai

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

Output:

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SONARQUBE 2
PS C:\Users\s9409\Downloads\aiassitantcoding> & C:/Users/s9409/AppData/Local/Programs/Python/Python313/python.exe c:/Users/s9409/Downloads/aiassitantcoding/25.py
5
PS C:\Users\s9409\Downloads\aiassitantcoding>
```

Explanation by AI

The AI detected a syntax error because the function declaration line did not end with a colon. In Python, the colon indicates the start of an indented code block. Without it, the interpreter cannot parse the function body.

- AI-generated explanation of the fix

Task 2: Debugging Logic Errors in Loops

Scenario

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

Requirements

Code with errors

```
i = 1
while i <= 5:
    print(i)    Expected indented block
    i = i - 1
```

Error Identified by AI

Type of Error: Logical Error

Reason: The loop variable is decremented instead of incremented, so the condition $i \leq 5$ is always true.

Code debugged by ai

```
i = 1
while i <= 5:
    print(i)    Expected indented block
    ↵ print(i)
    i = i + 1
```

Output:

```
IndentationError: expected an indented block after 'while' statement on line 5
PS C:\Users\s9409\Downloads\aiassistantcoding> & C:/Users/s9409/AppData/Local/Programs/Python/Python313/python.exe c:/Users/s9409/Downloads/aiassistantcoding/25.py
● 5
1
2
3
4
5
○ PS C:\Users\s9409\Downloads\aiassistantcoding> []
```

Explanation by AI

The loop condition depends on *i* increasing toward a stopping point. Decrementing *i* causes it to move away from the termination condition, resulting in an infinite loop.

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

A function crashes due to division by zero.

Buggy code

```
def divide(a, b):
    return a / b      Expected indented block

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

Error Identified by AI

- **Type of Error:** Runtime Error (ZeroDivisionError)
- **Reason:** Division by zero is mathematically undefined.

Debugged code

```
def divide(a, b):
    return a / b
try:
    print(divide(10, 0))
except ZeroDivisionError:
    return "Error: Division by zero is not allowed."
```

Output:

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SONARQUBE 3
PS C:\Users\s9409\Downloads\aiassitantcoding> & C:/Users/s9409/AppData/Local/Programs/Python/Python313/python.exe c:/Users/s9409/Downloads/aiassitantcoding/25.py
Error: Division by zero is not allowed.
PS C:\Users\s9409\Downloads\aiassitantcoding>

Explanation by AI

The AI added exception handling using try-except to prevent program termination. This ensures graceful handling of invalid input instead of crashing the program.

Task 4: Debugging Class Definition Errors

Scenario

A Python class constructor is incorrectly defined without the self parameter.

Buggy Code

```
def __init__(name, age):    Expected indented block
    name = name    Expected indented block
    age = age      "age" is not defined
```

Error Identified by AI

- Type of Error:** TypeError / Object-Oriented Error
- Reason:** The self parameter is mandatory for instance variable access.

Debugged code

```
class Student:
    def __init__(name, age):    Expected indented block
        name = (parameter) age: Any
        age = age
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

Output:

```
PS C:\Users\s9409\Downloads\aiassitantcoding> & C:/Users/s9409/AppData/Local/Programs/Python/Python  
n313/python.exe c:/Users/s9409/Downloads/aiassitantcoding/25.py  
Alice  
20  
PS C:\Users\s9409\Downloads\aiassitantcoding> []
```

Explanation by AI

The self parameter represents the current object instance. Without it, instance variables cannot be properly assigned or accessed.

Task 5: Resolving Index Errors in Lists

Scenario

A program crashes when accessing an invalid list index.

Buggy Code

```
numbers = [10, 20, 30]  
print(numbers[5])
```

```
Traceback (most recent call last):  
  File "c:/Users/s9409/Downloads/aiassitantcoding/25.py", line 27, in <module>  
    print(numbers[5])  
           ~~~~~^~  
IndexError: list index out of range  
PS C:\Users\s9409\Downloads\aiassitantcoding> []
```

Error Identified by AI

- **Type of Error:** IndexError
- **Reason:** Index is outside the valid range of the list.

Debugged code

```
numbers = [10, 20, 30]  
try:  
    print(numbers[5])  
except IndexError:  
    print("Error: List index out of range.")
```

Explanation by AI

The AI suggested two safe approaches: checking the index against list length or handling the exception using try-except. Both prevent program crashes.