AI-AC
Lab Assignment-7.3
2303A51511
BATCH – 27

Task 1: Fixing Syntax Errors

Scenario

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

Requirements:

Typical Python error message:

```javascript
SyntaxError: expected ':'
```

Output

3. Corrected Function (AI Fix)

The AI automatically corrects the issue by adding the missing colon.

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

4. AI Explanation of the Fix

The syntax error occurred because Python function definitions must end with a colon.
The colon tells Python that the indented block below belongs to the function.
Adding the colon resolves the syntax error and allows the function to run correctly.

Task 2: Debugging Logic Errors in Loops

Scenario

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

Requirements

1. Original Loop (With Logic Error)

This loop is intended to count **up to 5**, but it never stops.

```python
i = 1
while i <= 5:
    print(i)
    i -= 1
```

2. AI Identifies the Cause of Infinite Iteration

What the AI notices:

- The loop condition is `i <= 5`
- The variable `i` starts at `1`
- Inside the loop, `i` is **decremented** (`i -= 1`)
- This causes `i` to become `0, -1, -2, ...`
- The condition `i <= 5` is **always true**

👉 Result: the loop runs forever.

↓

3. AI Fixes the Loop Logic

Since the loop condition checks for `i <= 5`, the counter should **increase**, not decrease.

✅ **Corrected Loop**

```python
i = 1
while i <= 5:
    print(i)
    i += 1
```

4. Analysis of the Corrected Behavior

- `i` starts at `1`
- Each iteration increases `i` by `1`
- When `i` becomes `6`, the condition `i <= 5` becomes false
- The loop exits normally

Output

Output:

```
                                          Copy code
1

2

3

4

5
```

5. AI Explanation of the Logic Error

> The infinite loop occurred because the loop variable was moving in the wrong direction.
> Since the condition required `i` to eventually exceed `5`, decrementing `i` prevented the condition from ever becoming false.
> Changing the decrement to an increment allows the loop to terminate correctly.

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

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

Requriments

What Happens at Runtime

Calling:

```python
divide(10, 0)
```

Raises:

```vbnet
ZeroDivisionError: division by zero
```

2. AI Identifies the Runtime Error

AI Analysis:

- The function divides `a` by `b`
- When `b` is `0`, Python raises a `ZeroDivisionError`
- This is a **runtime error**, not a syntax or logic error
- The program crashes because the error is not handled

3. AI Adds try-except for Safe Execution

The AI fixes the issue by catching the exception using a `try-except` block.

✅ **Corrected Function (With Error Handling)**

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

4. Review of the Corrected Behavior

Normal case:

```python
divide(10, 2)
```

Output:

```
5.0
```

Error case:

```python
divide(10, 0)
```

Output:

```vbnet
Error: Division by zero is not allowed.
```

The function now:

- Executes safely
- Does not crash the program

Task 4: Debugging Class Definition Errors

Scenario

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

Example error:

```javascript
TypeError: __init__() takes 2 positional arguments but 3 were given
```

3. AI Corrects the Class Definition

The AI fixes the issue by:

- Adding `self` as the first parameter
- Using `self` to store instance variables

☑ **Corrected Class Definition**

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

Output

5. Review of Corrected Behavior

```python
p = Person("Alice", 25)
print(p.name)
print(p.age)
```

Output:

```nginx
Alice
25
```

Task 5: Resolving Index Errors in Lists

Scenario

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

Prompt Used

"Access an element from a list using an index that does not exist and fix the resulting error using safe list access techniques."

Problematic Code (Causes IndexError)

```python
numbers = [10, 20, 30]
print(numbers[5])  # Invalid index
```

Explanation

- The list `numbers` contains **3 elements**.

- Valid indices are `0`, `1`, and `2`.

- Attempting to access index `5` causes:

```
IndexError: list index out of range
```

Output

Output (After Fix)

```
Index out of range
```