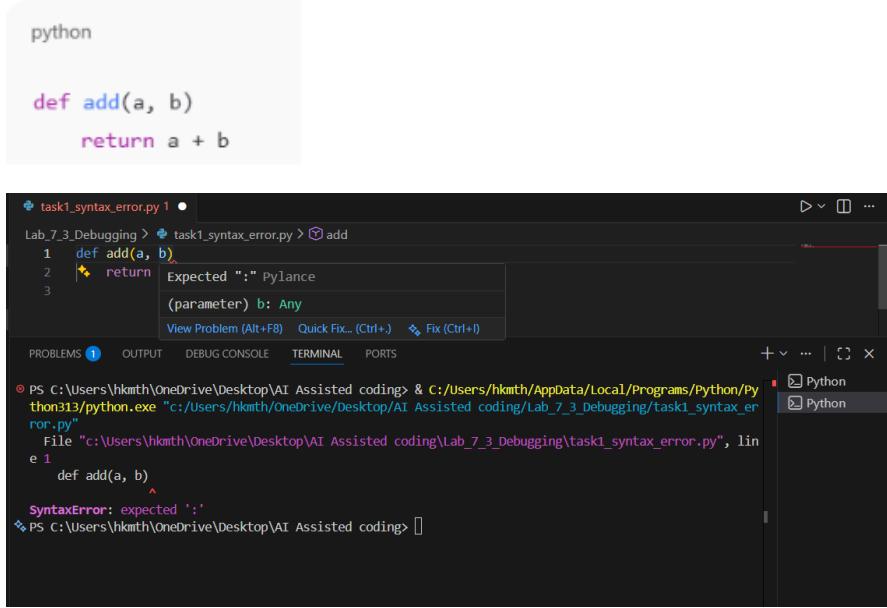


ASSIGNMENT – 7.3

Thumma Hasini | 2303A52076 | Batch – 37

Task 1: Fixing Syntax Errors

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



The screenshot shows a Python code editor with the following code:

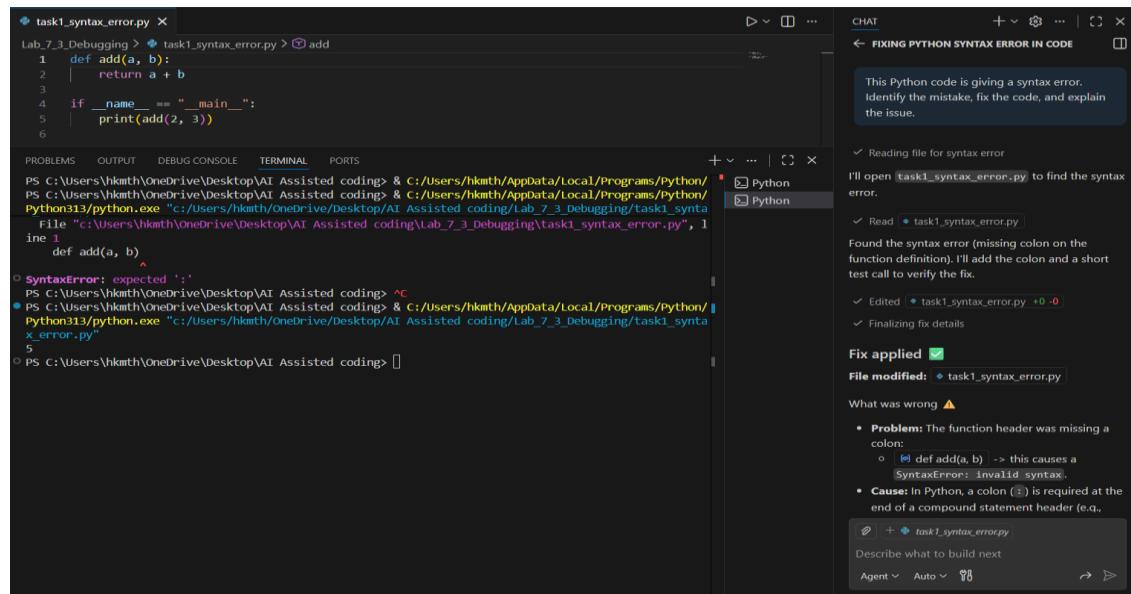
```
python

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

A tooltip is displayed over the opening parenthesis of the function definition, indicating a syntax error: "Expected ":" Pylance (parameter) b: Any". Below the editor, the terminal shows the command "python task1_syntax_error.py" and the resulting error message:

```
PS C:\Users\hkmtm\OneDrive\Desktop\AI Assisted coding> & C:/Users/hkmtm/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/hkmtm/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task1_syntax_error.py"
File "c:/Users/hkmtm/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task1_syntax_error.py", line 1
  def add(a, b)
          ^
SyntaxError: expected ':'
```

Prompt: This Python code is giving a syntax error. Identify the mistake, fix the code, and explain the issue.



The screenshot shows the same Python code after it has been fixed. The code now includes a colon at the end of the function header:

```
python

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

if __name__ == "__main__":
    print(add(2, 3))
```

The terminal shows the command "python task1_syntax_error.py" and the output "6". A "Fix applied" dialog is open on the right side of the interface, providing details about the fix:

- It reads the file for syntax errors.
- It finds the syntax error (missing colon on the function definition).
- It adds the colon and a short test call to verify the fix.
- It lists the file as modified.
- It asks what was wrong.
- It lists two problems:
 - The function header was missing a colon.
 - In Python, a colon (:) is required at the end of a compound statement header (e.g., if statements).

```

task1_syntax_error.py
Lab_7_3_Debugging > task1_syntax_error.py > add
1 def add(a, b):
2     return a + b
3
4 if __name__ == "__main__":
5     print(add(2, 3))
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hkmth\OneDrive\Desktop\AI Assisted coding> & C:/Users/hkmth/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/hkmth/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task1_syntax_error.py"
File "c:/Users/hkmth/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task1_syntax_error.py", line 1
ine 1
    def add(a, b)
      ^
SyntaxError: expected ':'
PS C:\Users\hkmth\OneDrive\Desktop\AI Assisted coding> ^
PS C:\Users\hkmth\OneDrive\Desktop\AI Assisted coding> & C:/Users/hkmth/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/hkmth/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task1_syntax_error.py"
5
PS C:\Users\hkmth\OneDrive\Desktop\AI Assisted coding> []

CHAT FIXING PYTHON SYNTAX ERROR IN CODE Fix applied File modified: task1_syntax_error.py What was wrong? Problem: The function header was missing a colon: def add(a, b) -> this causes a SyntaxError: invalid syntax. Cause: In Python, a colon (:) is required at the end of a compound statement header (e.g., def, if, for, while). Change made: Added the missing colon and a simple test call: def add(a, b):
    return a + b
if __name__ == "__main__":
    print(add(2, 3))

How to verify: Run the file: Expected output: 5 If you want, I can add unit tests or more examples. Agent Auto > >

```

Explanation: The buggy Python code is executed in VS Code to observe the syntax error. The error is then fixed using AI suggestions, and the corrected code runs successfully.

Task 2: Debugging Logic Errors in Loops

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

```

```

task2_infinite_loop.py
Lab_7_3_Debugging > task2_infinite_loop.py > ...
1 def count_down(n):
2     while n >= 0:
3         print(n)
4         n += 1 # logical error: should decrease n
5
6 count_down(3)
7 # The above code will result in an infinite loop because n is being incremented instead of decremented

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
47582
47583
47584
47585
47586
47587

```

Prompt : The following Python function runs infinitely due to a logic error in the loop. Identify the mistake, explain why the loop does not terminate, and correct the code so that the countdown works properly.

The screenshot shows a VS Code interface with the following details:

- Editor:** The file `task2_infinite_loop.py` is open, showing the following code:

```
1 def count_down(n):
2     while n >= 0:
3         print(n)
4         n -= 1 # decrement so the loop moves toward termination
5
6 if __name__ == "__main__":
7     count_down(3)
```
- Terminal:** The terminal shows the command being run: `python task2_infinite_loop.py`, which outputs the numbers 3, 2, 1, 0 sequentially.
- AI Chat:** A sidebar titled "FIXING PYTHON SYNTAX ERROR IN CODE" contains the following text:

This Python code is giving a syntax error. Identify the mistake, fix the code, and explain the issue.

✓ Reading file for syntax error
I'll open `task1_syntax_error.py` to find the syntax error.
✓ Read `task1_syntax_error.py`
Found the syntax error (missing colon on the function definition). I'll add the colon and a short test call to verify the fix.
✓ Edited `task1_syntax_error.py`
✓ Finalizing fix details
- Fix Applied:** The fix is applied to the code in the editor.
- File Modified:** The file `task1_syntax_error.py` is listed as modified.
- What was wrong:**
 - Problem:** The function header was missing a colon: `n += 1` → causes a `SyntaxError: invalid syntax`.
 - Cause:** In Python, a colon (`:`) is required at the end of a compound statement header (e.g., `def`, `if`, `for`, `while`).
- Change made:**
 - Added the missing colon and a simple test call:

```
+ task2_infinite_loop.py
```

The screenshot shows the same VS Code interface after the fix has been applied:

- Editor:** The code now includes the fixed colon at the end of the function header:

```
1 def count_down(n):
2     while n >= 0:
3         print(n)
4         n -= 1 # decrement so the loop moves toward termination
5
6 if __name__ == "__main__":
7     count_down(3)
```
- Terminal:** The terminal shows the corrected output: 3, 2, 1, 0.
- AI Chat:**

What was wrong ▲

 - Problem:** The loop used `n += 1` which increases `n`, so `n` never becomes negative and the `while n >= 0` condition remains true forever (infinite loop).
 - Why:** To reach termination the loop must move `n` toward a value that falsifies the condition; increasing `n` moves it away instead.

Change made ↴

 - Replaced `n += 1` with `n -= 1` so the counter decreases each iteration.
 - Added a safe runner:

```
if __name__ == "__main__":
    count_down(3)
```

Result ✅

 - Running the script now prints:

```
3
2
1
0
```

If you'd like, I can add an optional final message like "Liftoff!" or unit tests to validate behavior. 🚀

Explanation: The loop runs infinitely because the value of `n` is increased instead of decreased, so the loop condition always remains true. The AI identified this logical error and corrected the update statement. After fixing it, the loop properly counts down and terminates.

Task 3: Handling Runtime Errors (Division by Zero)

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

The screenshot shows a code editor with a Python file named `task3_runtime_error.py`. The code defines a `divide` function and calls it with arguments `10` and `0`. A red dot in the gutter indicates a breakpoint at the first line of the function. The terminal below shows the execution of the script, which results in a `ZeroDivisionError: division by zero` exception. The Python extension's status bar indicates two Python environments are active.

```
# Debug the following code
def divide(a, b):
    return a / b

print(divide(10, 0))

Lab_7_3_Debugging > task3_runtime_error.py > ...
1  def divide(a, b):
2  |      return a / b
3
4  print(divide(10, 0))
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hkmtm\OneDrive\Desktop\AI Assisted coding> & C:/Users/hkmtm/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/hkmtm/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task3_runtime_error.py"
    ~~~~~^~~~~~
File "c:/Users/hkmtm/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task3_runtime_error.py", line 2, in divide
    return a / b
          ~~~^~~
zeroDivisionError: division by zero
PS C:\Users\hkmtm\OneDrive\Desktop\AI Assisted coding>
```

Prompt: The following Python function causes a runtime error during execution. Identify the error, explain why it occurs, and modify the code using try-except to handle the error safely.

The screenshot shows the same Python file with the addition of a `try-except` block. The `divide` function now handles the `ZeroDivisionError` by printing a custom error message instead of crashing. The terminal shows the script running successfully without the error. The Python extension's status bar indicates two Python environments are active.

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        print("Error: division by zero")
        return None

if __name__ == "__main__":
    result = divide(10, 0)
    if result is not None:
        print(result)

print(divide(10, 0))
~~~~~^~~~~~
File "c:/Users/hkmtm/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task3_runtime_error.py", line 2, in divide
    return a / b
          ~~~^~~
zeroDivisionError: division by zero
PS C:\Users\hkmtm\OneDrive\Desktop\AI Assisted coding> & C:/Users/hkmtm/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/hkmtm/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task3_runtime_error.py"
Error: division by zero
PS C:\Users\hkmtm\OneDrive\Desktop\AI Assisted coding>
```

```

Lab_7_3_Debugging > task3_runtime_error.py ...
1 def divide(a, b):
2     try:
3         return a / b
4     except ZeroDivisionError:
5         print("Error: division by zero")
6         return None
7
8 if __name__ == "__main__":
9     result = divide(10, 0)
10    if result is not None:
11        print(result)
12
13
14 print(divide(10, 0))
~~~~~
File "c:/Users/hkmath/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task3_runtime_error.py", line 2, in divide
    return a / b
~~~~~
ZeroDivisionError: division by zero
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding> <C>
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding> & c:/Users/hkmath/AppData/Local/Programs/Python/Python313/python.exe "c:/users/hkmath/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task3_runtime_error.py"
Error: division by zero
Error: division by zero
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding>

```

Explanation: The program crashes at runtime because it attempts to divide a number by zero. This results in a Zero Division Error, which is a runtime error.

Task 4: Debugging Class Definition Errors

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

```



```

task4_class_error.py >
Lab_7_3_Debugging > task4_class_error.py ...
1 class Student:
2     def __init__(name):
3         name = name
4
5 s = Student("Hasini")
6
7
8 print(divide(10, 0))
~~~~~
File "c:/Users/hkmath/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task3_runtime_error.py", line 2, in divide
    return a / b
~~~~~
ZeroDivisionError: division by zero
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding> <C>
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding> & c:/Users/hkmath/AppData/Local/Programs/Python/Python313/python.exe "c:/users/hkmath/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task4_class_error.py"
Error: division by zero
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding>
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding> & c:/Users/hkmath/AppData/Local/Programs/Python/Python313/python.exe "c:/users/hkmath/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task4_class_error.py"
Traceback (most recent call last):
  File "c:/users/hkmath/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task4_class_error.py", line 5, in <module>
    s = Student("Hasini")
TypeError: student.__init__() takes 1 positional argument but 2 were given
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding>

```

Prompt: The following Python class is giving an error during object creation. Identify the issue in the constructor, correct the class definition, and explain why the error occurs.

The screenshot shows the VS Code interface with two tabs: 'task4_class_error.py' and 'task1_syntax_error.py'. The code in 'task4_class_error.py' is:

```

1 class Student:
2     def __init__(self, name):
3         self.name = name
4
5 if __name__ == "__main__":
6     s = Student("Hasini")
7     print(s.name)
8

```

The terminal shows the output of running the code, which results in a `TypeError` because the constructor was called without the `self` parameter. The AI interface on the right shows the problem identified as a missing colon in the constructor definition.

Explanation: In this task, a runtime error occurs because the `self` parameter is missing in the class constructor. Python automatically passes the object reference to methods, and without `self`, the arguments do not match. The AI fixes the issue by adding `self` to the constructor definition.

Task 5: Resolving Index Errors in Lists

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

```

python

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

```

The screenshot shows the VS Code interface with two tabs: 'task4_class_error.py' and 'task5_index_error.py'. The code in 'task5_index_error.py' is:

```

1 numbers = [10, 20, 30]
2
3 print(numbers[5])
4 # The above line will raise an IndexError because index 5 is out of range for

```

The terminal shows the output of running the code, which results in an `IndexError` because the list only has three elements. The AI interface on the right shows the problem identified as an index out of range.

Prompt: The following Python code crashes due to an index error while accessing a list. Identify the error, explain why it occurs, and modify the code to safely access the list using bounds checking or exception handling.

The screenshot shows the Visual Studio Code interface. On the left, there are two tabs: 'task4_class_error.py' and 'task5_index_error.py'. The 'task5_index_error.py' tab is active and contains the following Python code:

```
Lab_7_3_Debugging > task5_index_error.py > ...
1 if __name__ == "__main__":
2     numbers = [10, 20, 30]
3     index = 5
4
5     # Safe access using try/except to catch IndexError
6     try:
7         print(numbers[index])
8     except IndexError:
9         print(f"Error: index {index} out of range for list of length {len(numbers)}")
10
11     # Alternative: bounds-check before accessing
12     # if 0 <= index < len(numbers):
13     #     print(numbers[index])
14     # else:
15     #     print(f"Index {index} out of range (list length {len(numbers)})")
```

The terminal at the bottom shows the execution of the code and an error message:

```
File "c:/Users/hkmath/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task5_index_error.py", line 3, in <module>
    print(numbers[5])
           ^^^^^^
IndexError: list index out of range
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding> & C:/Users/hkmath/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/hkmath/OneDrive/Desktop/AI Assisted coding/Lab_7_3_Debugging/task5_index_error.py"
Error: index 5 out of range for list of length 3
PS C:\Users\hkmath\OneDrive\Desktop\AI Assisted coding> []
```

The right side of the interface features an AI-powered chat window titled 'FIXING PYTHON SYNTAX ERROR IN CODE' with the following message:

The following Python code crashes due to an index error while accessing a list. Identify the error, explain why it occurs, and modify the code to safely access the list using bounds checking or exception handling.

Below the chat window, a status bar indicates: 'Scanning task5_index_error.py to find where the index access goes out of bounds. I'll open it now.' A progress bar shows 'Revised code and planned updates' with a checkmark. A note says 'Found an out-of-range index access (index 5 on a 3-element list). I'll replace the direct access with a safe try/except and include a commented bounds-check example.' At the bottom of the status bar, there are 'Allow' and 'Skip' buttons.

Explanation: In this task, the program crashes because it tries to access a list index that does not exist. This causes an Index Error at runtime. The AI fixes the issue by handling the error safely, preventing the program from crashing.