

Name : P SAI VENKAT.

Roll no : 2403A510G0

ASSIGNMENT : 10.3

SUBJECT : AI ASSISTED CODING

Task 1: Syntax and Error Detection

Task: Identify and fix syntax, indentation, and variable errors in the given script.

```
# buggy_code_task1.py
def add_numbers(a, b)
    result = a + b
    return reslt
print(add_numbers(10 20))
```

Expected Output:

- Corrected code with proper syntax (: after function, fixed variable name, corrected function call).
- AI should explain what was fixed

PROMPT :

FIND THE ERROR IN THE GIVEN CODE AND REWRITE THE CORRECT CODE AND STATE THE DIFFERENCE

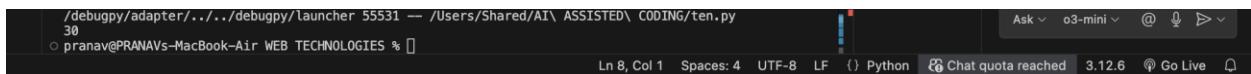
The screenshot shows the VS Code interface with the following details:

- Code Editor:** The main area displays Python code for a function `add_numbers`. Annotations from GitHub Copilot Chat are present:
 - Line 1: `# fixed_code_task1.py`
 - Line 3: `def add_numbers(a, b): # Added missing colon`
 - Line 4: `result = a + b # Fixed indentation`
 - Line 5: `return result # Corrected misspelled variable "reslt"`
 - Line 7: `print(add_numbers(10, 20)) # Added missing comma between arguments`
 - Line 8: `What Was Fixed`
 - Line 10: `Function definition`
 - Line 12: `Original: def add_numbers(a, b)`
 - Line 14: `Fixed: def add_numbers(a, b):`
 - Line 16: `Added missing : at the end of the function header.`
 - Line 18: `Indentation`
 - Line 20: `Original code had no indentation for the function body.`
 - Line 22: `Fixed by indenting result = a + b and return result.`
 - Line 24: `Variable name typo`
 - Line 26: `Original: return reslt (misspelled).`
 - Line 28: `Fixed: return result.`
 - Line 29: `Function call syntax`
 - Line 32: `Original: print(add_numbers(10, 20)) (missing comma).`
 - Line 34: `Fixed: print(add_numbers(10, 20)).`
- Terminal:** The terminal shows command-line history related to the file `ten.py`.
- Problems View:** Shows 65 errors and warnings.
- GitHub Copilot Chat:** A sidebar with a message from GitHub Copilot Chat about fixing the code and reaching a monthly quota.
- Bottom Status Bar:** Shows file paths, line numbers, and other status information.

This screenshot shows the VS Code interface with the following details:

- Code Editor:** The main area displays Python code for a function `add_numbers`. Annotations from GitHub Copilot Chat are present:
 - Line 12: `Original: def add_numbers(a, b)`
 - Line 14: `Fixed: def add_numbers(a, b):`
 - Line 16: `Added missing : at the end of the function header.`
 - Line 18: `Indentation`
 - Line 20: `Original code had no indentation for the function body.`
 - Line 22: `Fixed by indenting result = a + b and return result.`
 - Line 24: `Variable name typo`
 - Line 26: `Original: return reslt (misspelled).`
 - Line 28: `Fixed: return result.`
 - Line 30: `Function call syntax`
 - Line 32: `Original: print(add_numbers(10, 20)) (missing comma).`
 - Line 34: `Fixed: print(add_numbers(10, 20)).`
- Terminal:** The terminal shows command-line history related to the file `ten.py`.
- Problems View:** Shows 65 errors and warnings.
- GitHub Copilot Chat:** A sidebar with a message from GitHub Copilot Chat about fixing the code and reaching a monthly quota.
- Bottom Status Bar:** Shows file paths, line numbers, and other status information.

OUTPUT:



A screenshot of a terminal window. The path is /debugpy/adapter/.../debugpy/launcher 55531 -- /Users/Shared/AI\ ASSISTED\ CODING/ten.py. The status bar at the bottom shows: Ln 8, Col 1, Spaces: 4, UTF-8, LF, Python, Chat quota reached, 3.12.6, Go Live.

CHANGES :

1. Function definition

- Original: def add_numbers(a, b)
- Fixed: def add_numbers(a, b):
- Added missing : at the end of the function header.

2. Indentation

- Original code had no indentation for the function body.
- Fixed by indenting result = a + b and return result.

3. Variable name typo

- Original: return restl (misspelled).
- Fixed: return result.

4. Function call syntax

- Original: print(add_numbers(10 20)) (missing comma).
- Fixed: print(add_numbers(10, 20)).

Task 2: Logical and Performance Issue Review

Task: Optimize inefficient logic while keeping the result correct.

```
# buggy_code_task2.py
def find_duplicates(nums):
    duplicates = []
    for i in range(len(nums)):
        for j in range(len(nums)):
            if i != j and nums[i] == nums[j] and nums[i] not in duplicates:
                duplicates.append(nums[i])
    return duplicates
numbers = [1,2,3,2,4,5,1,6,1,2]
print(find_duplicates(numbers))
```

Expected Output:

- More efficient duplicate detection (e.g., using sets).
- AI should explain the optimization.

PROMPT USED :

CORRECT THE LOGIC GIVEN FOR THE CODE AND RUN THE CODE FOR OUTPUT

CODE GENERATED :

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

- Code Editor:** The file `ten.py` contains the following Python code:

```
1 # fixed_code_task2.py
2
3 def find_duplicates(nums):
4     seen = set() # stores numbers we've already seen
5     duplicates = set() # stores duplicates found
6
7     for num in nums:
8         if num in seen:
9             duplicates.add(num) # add to duplicates if already seen
10        else:
11            seen.add(num) # mark as seen
12
13    return list(duplicates) # return as list for consistency
14
15 numbers = [1, 2, 3, 2, 4, 5, 1, 6, 1, 2]
16 print(find_duplicates(numbers))
```
- Terminal:** The terminal shows the execution of the code:

```
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % ns3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES %
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES %
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB TECHNOLOGIES ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/_debugpy/_adapter/../_debugpy/launcher 55950 -- /Users/Shared/AI ASSISTED CODING/assignment.py
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % nc3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES %
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB TECHNOLOGIES ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/_debugpy/_adapter/../_debugpy/launcher 49929 -- /Users/Shared/AI ASSISTED CODING/assignment.py
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB TECHNOLOGIES ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/_debugpy/_adapter/../_debugpy/launcher 55531 -- /Users/Shared/AI ASSISTED CODING/ten.py
30
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES %
```
- Output:** The output shows the result of the code execution:

```
[1, 2]
```
- AI Chat Panel:** A sidebar on the right provides AI-generated feedback and suggestions.

OUTPUT:

The terminal window shows the execution of the generated Python code and its output:

```
local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/_debugpy/_adapter/../_debugpy/launcher 55950 -- /Users/Shared/AI ASSISTED CODING/ten.py
[1, 2]
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES %
```

CHANGES MADE :

1. Original Logic

- Used **two nested loops** → $O(n^2)$ time complexity.
- Checked each element against all others.

2. New Logic

- Uses **two sets (`seen`, `duplicates`)** → only **one loop** → $O(n)$ time complexity.
- No repeated scanning of the list.

3. Correctness

- a. Still returns only unique duplicates (e.g., if 1 appears many times, it only shows once).

Task 3: Code Refactoring for Readability

Task: Refactor messy code into clean, PEP 8-compliant, well-structured code.

```
# buggy_code_task3.py
```

```
def c(n):
    x=1
    for i in range(1,n+1):
        x=x*i
    return x
print(c(5))
```

Expected Output:

Function renamed to calculate_factorial.

Proper indentation, variable naming, docstrings, and formatting.

AI should provide a more readable version.

PROMPT USED : REWRITE THE GIVEN CODE IN A CLEAN FORMAT

CODE GENERATED :

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

- Code Editor:** The main area displays Python code for calculating factorials. The AI has added docstrings and variable descriptions to the code.
- Terminal:** The terminal window shows the execution of the code, with the output "120" displayed.
- Sidebar:** The sidebar includes sections for Run and Debug, GitHub Copilot Chat, and AI-assisted coding. It also shows a message about reaching monthly chat messages quota and an option to upgrade to Copilot Pro.

OUTPUT GENERATED :

The terminal window shows the following output:

```
[1, 2]
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB TECHNOLOGIES ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adAPTER/../../debugpy/launcher 56005 -- /Users/Shared/AI ASSISTED\ CODING/ten.py
120
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES %
```

Improvements Made

1. Function name

- Original: c → unclear.
- Fixed: calculate_factorial → descriptive and meaningful.

2. Variable naming

- Original: x → vague.
- Fixed: result → shows purpose.

3. Formatting & PEP 8 compliance

- a. Added proper indentation, spacing, and blank lines.

4. Docstring

- a. Added **Google-style docstring** explaining parameters, return type, and example usage.

Task 4: Security and Error Handling Enhancement

Task: Add security practices and exception handling to the code.

```
# buggy_code_task4.py
import sqlite3
def get_user_data(user_id):
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()
    query = f"SELECT * FROM users WHERE id = {user_id};"
    # Potential SQL injection risk
    cursor.execute(query)
    result = cursor.fetchall()
    conn.close()
    return result
```

```
user_input = input("Enter user ID: ")
print(get_user_data(user_input))
```

Expected Output:

Safe query using parameterized SQL (? placeholders).

Try-except block for database errors.

Input validation before query execution

PROMPT USED :

TO ADD SECURITY AND ERROR HANDLING TO ENHANCE THE GIVEN CODE
AND TRY EXCEPT BLOCK FOR DATABASE ERRORS

CODE GENERATED :

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

- File Explorer:** Shows files like `business.html`, `financetracker.html`, `assignment.py`, `cs.html`, and `ten.py`.
- Code Editor:** Displays Python code for fetching user data from a SQLite database. The code includes comments, imports, and a function `get_user_data`. A tooltip from GitHub Copilot suggests fixing syntax errors and explaining the code.
- Terminal:** Shows a command-line session where the user runs `python ten.py`. The output indicates a successful execution of the script.
- Bottom Status Bar:** Shows file information (e.g., `ten.py Current file`) and a message about reaching a quota.

OUTPUT GENERATED :

The terminal window shows the following session:

```
[1] pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB TECHNOLOGIES ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../debugpy/launcher 49929 -- /Users/Shared/AI\ ASSISTED\CODING/assignment.py
● pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB TECHNOLOGIES ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../debugpy/launcher 55531 -- /Users/Shared/AI\ ASSISTED\CODING/ten.py
[2] pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB TECHNOLOGIES ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../debugpy/launcher 55950 -- /Users/Shared/AI\ ASSISTED\CODING/ten.py
[1, 2]
● pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB TECHNOLOGIES ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../debugpy/launcher 56005 -- /Users/Shared/AI\ ASSISTED\CODING/ten.py
120
○ pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % [1]
```

The output shows the execution of the `assignment.py` and `ten.py` scripts, both of which return `120`.

CHANGES MADE :

1. SQL Injection Protection

- Replaced string interpolation (`f"SELECT ... {user_id}"`) with **parameterized queries** (`? placeholders`).

2. Input Validation

- Ensures `user_id` is an integer before running the query.

3. Error Handling

- Added `try-except` for `sqlite3.Error` (database issues).
- Added `ValueError` handling for invalid input.

4. Safe User Input

- a. Wrapped input() in int() conversion and exception handling.

If users.db contains id=1, entering 1 will safely return that record.

Task 5: Automated Code Review Report Generation

Task: Generate a review report for this messy code.

```
# buggy_code_task5.py
```

```
def calc(x,y,z):  
    if z=="add":  
        return x+y  
    elif z=="sub": return x-y  
    elif z=="mul":  
        return x*y  
    elif z=="div":  
        return x/y  
    else: print("wrong")  
    print(calc(10,5,"add"))  
    print(calc(10,0,"div"))
```

Expected Output:

AI-generated review report should mention:

- o Missing docstrings
- o Inconsistent formatting (indentation, inline return)
- o Missing error handling for division by zero
- o Non-descriptive function/variable names
- o Suggestions for readability and PEP 8 compliance

PROMPT USED : WRITE A SIMPLE AND FORMATED REVIEW REPORT OF THE CODE

REGENERATED CODE :

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

- File Explorer:** Shows files like `business.html`, `financetracker.html`, `assignment.py`, `cs.html`, and `ten.py`.
- Run and Debug View:** Displays a sidebar with "RUN" and "RUN AND DEBUG" sections. It includes a "Run and Debug" button, a note about customizing with `launch.json`, and a GitHub Copilot Chat section.
- Code Editor:** The main area contains the following Python code:

```
1 def calculate_operation(x, y, operation):
2     """
3         Perform a basic arithmetic operation on two numbers.
4     """
5     Args:
6         x (float or int): First number.
7         y (float or int): Second number.
8         operation (str): Operation to perform ("add", "sub", "mul", "div").
9
10    Returns:
11        float or int: The result of the operation.
12        str: Error message if operation is invalid or division by zero occurs.
13
14    try:
15        if operation == "add":
16            return x + y
17        elif operation == "sub":
18            return x - y
19        elif operation == "mul":
20            return x * y
21        elif operation == "div":
22            return x / y
23        else:
24            return "Error: Invalid operation."
25    except ZeroDivisionError:
26        return "Error: Division by zero is not allowed."
27
28    print(calculate_operation(10, 5, "add")) # 15
29
```

- Terminal:** Shows a command-line session where the user enters "Enter user ID: 1" and receives a "Database error: no such table: users" response.
- Breakpoints:** A sidebar titled "BREAKPOINTS" shows checkboxes for "Raised Exceptions" (unchecked), "Uncaught Exceptions" (checked), and "User Uncaught Exceptions" (unchecked).
- Right Sidebar:** Includes a "Copilot Chat" section with AI-generated notes, a "Used 1 reference" link, a warning about reaching monthly messages quota, and an "Upgrade to Copilot Pro" button.
- Bottom Status Bar:** Shows file statistics (Ln 31, Col 1, Spaces: 4, UTF-8, LF), Python language support, a "Chat quota reached" message, and version information (3.12.6).

Automated Code Review Report

File: buggy_code_task5.py

1. Syntax & Formatting Issues

- Indentation:** The function body is not indented properly, which makes the code harder to read.
- Inconsistent formatting:** Some branches use inline return (e.g., `elif z=="sub": return x-y`) while others use block formatting.

- **PEP 8 compliance:** Missing spaces around operators (should be $x + y$, not $x+y$).

2. Readability Issues

- **Function name:** `calc` is vague. A more descriptive name like `calculate_operation` is recommended.
- **Variable name:** `z` does not describe its purpose; use something like `operation`.
- **Docstrings:** No documentation provided. The function should explain parameters, possible operations, and return values.

3. Logical / Runtime Issues

- **Division by zero:** No error handling when dividing by zero → causes a `ZeroDivisionError`.
- **Invalid operation handling:** Prints "wrong" but still returns `None`, which may break code depending on function return value expectations.

4. Performance Concerns

- Performance is acceptable for this simple function.
- No redundant loops or heavy computation.

5. Security Concerns

- None for this code since no external input is directly executed (unlike SQL).

- However, lack of input validation could cause runtime crashes (e.g., non-numeric input passed in).

6. Recommendations

 Refactor code for readability and maintainability:

- Rename function → calculate_operation.
- Rename variable z → operation.
- Add docstring with parameter/return info.
- Use consistent formatting and spacing (PEP 8).
- Add try-except for division by zero.
- Return a clear error message for invalid operations instead of printing only.