

ASSIGNMENT 10.4

Task 1: AI-Assisted Syntax and Code Quality Review

Scenario

You join a development team and are asked to review a junior developer's Python script that fails to run correctly due to basic coding mistakes. Before deployment, the code must be corrected and standardized.

Task Description

You are given a Python script containing:

- Syntax errors
- Indentation issues
- Incorrect variable names
- Faulty function calls

Use an AI tool (GitHub Copilot / Cursor AI) to:

- Identify all syntactic and structural errors
- Correct them systematically
- Generate an explanation of each fix made

Expected Outcome

- Fully corrected and executable Python code
- AI-generated explanation describing:
 - Syntax fixes
 - Naming corrections
 - Structural improvements
- Clean, readable version of the script

TASK-1:

```
.github > 19.py > calculate_total
1 def calculate_total(price, tax
2     total = price + price * tax
3     print("Total is: " total)
4
5 itemPrice = 100
6 taxRate = 0.15
7
8 calculateTotal(itemPrice, taxrate)
9 ^

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

19.2026021801-win32-arm64\bundled\libs\debugpy\_vendored\pydevd\_pydevd_bundle\pydevd_runpy.py", line 309, in run_path
    code, fname = _get_code_from_file(run_name, path_name)
                                                               ^^^^^^^^^^^^^^^^^^
File "c:\Users\sriini\.vscode\extensions\ms-python.debugpy-2025.
19.2026021801-win32-arm64\bundled\libs\debugpy\_vendored\pydevd\_pydevd_bundle\pydevd_runpy.py", line 283, in _get_code_from_file
    code = compile(f.read(), fname, "exec")
File "C:\Users\sriini\OneDrive\Desktop\Blockchain\.github\19.py"
, line 1
    def calculate_total(price, tax
                           ^
SyntaxError: '(' was never closed
PS C:\Users\sriini\OneDrive\Desktop\Blockchain>
```

- ❑ Missing colon (:) after function definition
 - ❑ Incorrect indentation
 - ❑ Misspelled variable quantity
 - ❑ Incorrect print syntax
 - ❑ Function name mismatch (calculateTotal vs calculate_total)
 - ❑ return outside properly indented block

Prompt Used

“Review the following Python script. Identify syntax errors, indentation issues, naming problems, and structural mistakes. Correct the code and explain each fix according to PEP 8 standards.”

A screenshot of the Visual Studio Code interface. The top half shows a Python file named '1919.py' with code that defines a function 'calculate_total' and prints its result. The bottom half shows a terminal window with the command 'python 1919.py' and its output: 'Total is: 50'. The terminal tab is active, and other tabs like 'PROBLEMS' and 'OUTPUT' are visible.

```
.github > 1919.py > ...
1  def calculate_total(price, quantity):
2      """
3          Calculate the total price based on price and quantity.
4
5      Args:
6          price (float): Price of a single item.
7          quantity (int): Number of items.
8
9      Returns:
10         float: Total calculated price.
11     """
12     total = price * quantity
13     print(f"Total is: {total}")
14     return total
15
16
17 price = 10
18 quantity = 5
19
20 result = calculate_total(price, quantity)
21
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS + ⌂ ... [] X

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain> & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '64686' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\1919.py'
Total is: 50
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task 2: Performance-Oriented Code Review

Scenario

A data processing function works correctly but is inefficient and slows

down the system when large datasets are used.

Task Description

You are provided with a function that identifies duplicate values in a list

using inefficient nested loops.

Using AI-assisted code review:

- Analyze the logic for performance bottlenecks
- Refactor the code for better time complexity
- Preserve the correctness of the output

Ask the AI to explain:

- Why the original approach was inefficient
- How the optimized version improves performance

Expected Outcome

- Optimized duplicate-detection logic (e.g., using sets or hash-based structures)
- Improved time complexity
- AI explanation of performance improvement
- Clean, readable implementation

Original Inefficient Code

```
.github > Untitled-1.py > ...
1  def find_duplicates(lst):
2      duplicates = []
3      for i in range(len(lst)):
4          for j in range(i + 1, len(lst)):
5              if lst[i] == lst[j]:
6                  duplicates.append(lst[i])
7      return duplicates
8
```

The screenshot shows the VS Code interface with the code editor and terminal tabs active. The code editor contains a Python script named 'Untitled-1.py' with the following content:

```
def find_duplicates(lst):
    duplicates = []
    for i in range(len(lst)):
        for j in range(i + 1, len(lst)):
            if lst[i] == lst[j]:
                duplicates.append(lst[i])
    return duplicates
```

The terminal tab displays the following output:

```
-arm64\bundled\libs\debugpy\launcher' '64686' '--' 'C:\Users\sri
ni\OneDrive\Desktop\Blockchain\.github\1919.py'
Total is: 50
PS C:\Users\sri
ni\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\sri
ni\OneDrive\Desktop\Blockchain>
PS C:\Users\sri
ni\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\sri
ni\OneDrive\Desktop\Blockchain'; & 'c:\Users\sri
ni\AppData\Loc
al\Python\pythoncore-3.14-64\python.exe' 'c:\Users\sri
ni\.vscode\ext
ensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundl
ed\libs\debugpy\launcher' '53156' '--' 'C:\Users\sri
ni\OneDrive\Desktop\Blockchain\.github\Untitled-1.py'
PS C:\Users\sri
ni\OneDrive\Desktop\Blockchain>
```

Performance Issue

- **Uses nested loops → $O(n^2)$ time complexity**
- **Very slow for large datasets**

Optimized Version:

```
github > Untitled-1.py > ...
1  def find_duplicates(items):
2      """
3          Identify duplicate values in a list efficiently.
4
5      Args:
6          items (list): List of elements.
7
8      Returns:
9          list: List of duplicate elements.
10     """
11
12     seen = set()
13     duplicates = set()
14
15     for item in items:
16         if item in seen:
17             duplicates.add(item)
18         else:
19             seen.add(item)
```

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain> & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '64686' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain'
```

Task 3: Readability and Maintainability

Refactoring

Scenario

A working script exists in a project, but it is difficult to understand due to

poor naming, formatting, and structure. The team wants it rewritten for

long-term maintainability.

Task Description

You are given a poorly structured Python function with:

- Cryptic function names**
- Poor indentation**
- Unclear variable naming**
- No documentation**

Use AI-assisted review to:

- Refactor the code for clarity**
- Apply PEP 8 formatting standards**
- Improve naming conventions**
- Add meaningful documentation**

Expected Outcome

- Clean, well-structured code**
- Descriptive function and variable names**
- Proper indentation and formatting**
- Docstrings explaining the function purpose**
- AI explanation of readability improvements**

Poorly Written Code:

The screenshot shows a code editor with a Python file named `Untitled-1.py`. The code contains a function `f(x,y)` that prints "big" if `z > 100` and "small" otherwise. The terminal below shows the execution of the script, which results in the output "big".

```
.github > Untitled-1.py > ...
1  def f(x,y):
2      z=x+y
3      if(z>100):
4          print("big")
5      else:
6          print("small")
7      return z
8

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS + ▾

al\Python\pythoncore-3.14-64\python.exe' 'c:\Users\smini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '50819' '--' 'C:\Users\smini\OneDrive\Desktop\Blockchain\.github\Untitled-1.py'
PS C:\Users\smini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\smini\OneDrive\Desktop\Blockchain>
PS C:\Users\smini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\smini\OneDrive\Desktop\Blockchain'; & 'c:\Users\smini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\smini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '50839' '--' 'C:\Users\smini\OneDrive\Desktop\Blockchain\.github\Untitled-1.py'
PS C:\Users\smini\OneDrive\Desktop\Blockchain>
```

Issues

- **Cryptic function name**
- **Poor indentation**
- **No documentation**
- **Unclear variable names**

Refactored Version:

The screenshot shows a code editor in VS Code with a dark theme. A Python file named `def calculate_sum_and_check_limit(value_.py)` is open. The code defines a function `calculate_sum_and_check_limit` that calculates the sum of two numbers and checks if it exceeds 100. The function includes docstrings and type hints. Below the code editor is a terminal window showing PowerShell commands to run the script and its output.

```
.github > 🐍 def calculate_sum_and_check_limit(value_.py) > ...
1  def calculate_sum_and_check_limit(value_one, value_two):
2      """
3          Calculate the sum of two numbers and determine
4          whether the result exceeds 100.
5
6      Args:
7          value_one (int or float): First number.
8          value_two (int or float): Second number.
9
10     Returns:
11         int or float: The calculated sum.
12     """
13     total = value_one + value_two
14
15     if total > 100:
16         print("The total is large.")

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS + ▾

PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '50839' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\Untitled-1.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '60539' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\def calculate_sum_and_check_limit(value_.py)'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task 4: Secure Coding and Reliability Review

Scenario

A backend function retrieves user data from a database but has security vulnerabilities and poor error handling, making it unsafe for production deployment.

Task Description

You are given a Python script that:

- **Uses unsafe SQL query construction**
- **Has no input validation**
- **Lacks exception handling**

Use AI tools to:

- **Identify security vulnerabilities**
- **Refactor the code using safe coding practices**
- **Add proper exception handling**
- **Improve robustness and reliability**

Expected Outcome

- **Secure SQL queries using parameterized statements**
- **Input validation logic**
- **Try-except blocks for runtime safety**
- **AI-generated explanation of security improvements**
- **Production-ready code structure**

Unsafe Code

The screenshot shows a Python code editor with several tabs open at the top. The active tab contains the following code:

```
.github > import sqlite3.py > ...
1 import sqlite3
2
3 def get_user(username):
4     conn = sqlite3.connect("users.db")
5     cursor = conn.cursor()
6     query = "SELECT * FROM users WHERE username = '" + username +
7     cursor.execute(query)
8     result = cursor.fetchall()
9     return result
10
```

Below the code editor is a terminal window showing command-line history. The user has run several commands to navigate to a directory and execute Python files. The terminal output is as follows:

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '60539' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\def calculate_sum_and_check_limit(value_.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '57956' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\import sqlite3.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. A sidebar on the right shows some recent file icons.

Security Issues

- **SQL Injection vulnerability**

- **No input validation**
 - **No exception handling**
 - **Connection not properly closed**

Secure Version

Task 5: AI-Based Automated Code Review Report

Scenario

Your team uses AI tools to perform automated preliminary code reviews before human review, to improve code quality and consistency across projects.

Task Description

You are provided with a poorly written Python script.

Using AI-assisted review:

- Generate a structured code review report that evaluates:**
 - Code readability**
 - Naming conventions**
 - Formatting and style consistency**
 - Error handling**
 - Documentation quality**
 - Maintainability**

The task is not just to fix the code, but to analyze and report on quality issues.

Expected Outcome

- **AI-generated review report including:**
 - **Identified quality issues**
 - **Risk areas**
 - **Code smell detection**
 - **Improvement suggestions**
- **Optional improved version of the code**

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

- Code Editor:** The file `Untitled-2.py` contains the following Python code:

```
.github > Untitled-2.py > data
1 def data(x):
2     a=[]
3     for i in x:
4         if i not in a:
5             a.append(i)
6     return a
7
```
- Terminal:** The terminal window shows the command line history:

```
srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\sri... extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '64684' '--' 'C:\Users\sri... esktop\Blockchain\.github\II.py'
PS C:\Users\sri... Desktop\Blockchain> ^C
PS C:\Users\sri... OneDrive\Desktop\Blockchain>
PS C:\Users\sri... OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\sri... Blockchain'; & 'c:\Users\sri... extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '56364' '--' 'C:\Users\sri... OneDrive\Desktop\Blockchain\.github\Untitled-2.py'
PS C:\Users\sri... OneDrive\Desktop\Blockchain>
```

AI Code Review Report

1 Readability

- ✗ **Function name `data` is vague**
- ✗ **Variable `a` is meaningless**
- ✗ **No documentation**

2 Naming Conventions

- ✗ **Does not follow descriptive naming**
- ✗ **Violates clarity standards**

3 Formatting & Style

- ✗ **Poor indentation**
- ✗ **No spacing**
- ✗ **No docstring**

4 Error Handling

- ✗ **No input validation**
- ✗ **Assumes iterable input**

5 Maintainability

- ⚠ **Difficult to understand intent**
- ⚠ **No explanation of logic**

Improved Version:

```
.github > 📁 Untitled-3.py > ...
1  def remove_duplicates(items):
2      """
3          Remove duplicate elements from a list while preserving order.
4
5      Args:
6          items (list): List of elements.
7
8      Returns:
9          list: List without duplicates.
10     """
11     if not isinstance(items, list):
12         raise TypeError("Input must be a list.")
13
14     unique_items = []
15
16     for item in items:
17         if item not in unique_items:
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '56364' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\Untitled-2.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundle\libs\debugpy\launcher' '60172' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\Untitled-3.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```