Name:M.Akash H.No:2303A51820 Batch:26

| | | | |
|---|---|---|---|
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | |
| **Program Name:**B. Tech | **Assignment Type: Lab** | | **Academic Year:**2025-2026 |
| **Course Coordinator Name** | Dr. Rishabh Mittal | | |
| **Instructor(s)Name** | Mr. S Naresh Kumar<br>Ms. B. Swathi<br>Dr. Sasanko Shekhar Gantayat<br>Mr. Md Sallauddin<br>Dr. Mathivanan<br>Mr. Y Srikanth<br>Ms. N Shilpa<br>Dr. Rishabh Mittal (Coordinator)<br>Dr. R. Prashant Kumar<br>Mr. Ankushavali MD<br>Mr. B Viswanath<br>Ms. Sujitha Reddy<br>Ms. A. Anitha<br>Ms. M.Madhuri<br>Ms. Katherashala Swetha<br>Ms. Velpula sumalatha<br>Mr. Bingi Raju<br>Mr. G. Kranthi | | |
| **Course Code** | 23CS002PC304 | **Course Title** | AI Assisted Coding |
| **Year/Sem** | III/I | **Regulation** | R23 |
| **Date and Day of Assignment** | Week 5 - Thursday | **Time(s)** | 23CSBTB01 To 23CSBTB52 |
| **Duration** | 2 Hours | **Applicable to Batches** | All Batches |
| **AssignmentNumber:10.4** (Present assignment number)/**24**(Total number of assignments) | | | |

| **Q.No.** | **Question** | ***ExpectedTime to complete*** |
|---|---|---|
| 1 | **Lab 9 – Code Review and Quality: Using AI to Improve Code Quality and** | Week 5 |

**Readability**

## Lab Objectives

- Use AI for automated code review and quality enhancement.
- Identify and fix syntax, logical, performance, and security issues in Python code.
- Improve readability and maintainability through structured refactoring and comments.
- Apply prompt engineering for targeted improvements.
- Evaluate AI-generated suggestions against PEP 8 standards and software engineering best practices

## 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:

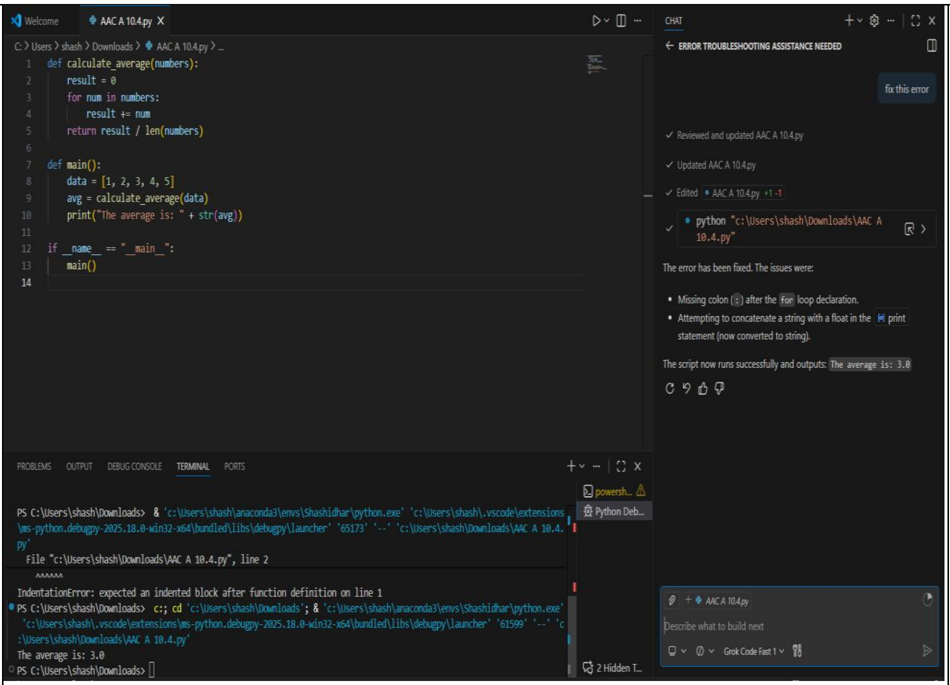
- Syntax errors
- Indentation issues
- Incorrect variable names
- Faulty function calls

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



```python
def calculate_average(numbers):
    result = 0
    for num in numbers:
        result += num
    return result / len(numbers)

def main():
    data = [1, 2, 3, 4, 5]
    avg = calculate_average(data)
    print("The average is: " + str(avg))

if __name__ == "__main__":
    main()
```

```
PS C:\Users\shash\Downloads>  & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '65173' '--' 'c:\Users\shash\Downloads\AAC A 10.4.py'
  File "c:\Users\shash\Downloads\AAC A 10.4.py", line 2
    ^^^^^^
IndentationError: expected an indented block after function definition on line 1
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61599' '--' 'c:\Users\shash\Downloads\AAC A 10.4.py'
The average is: 3.0
PS C:\Users\shash\Downloads>
```

**ERROR TROUBLESHOOTING ASSISTANCE NEEDED**

fix this error

✓ Reviewed and updated AAC A 10.4.py

✓ Updated AAC A 10.4.py

✓ Edited AAC A 10.4.py +1 -1

✓ `python "c:\Users\shash\Downloads\AAC A 10.4.py"`

The error has been fixed. The issues were:

- Missing colon (`:`) after the `for` loop declaration.
- Attempting to concatenate a string with a float in the print statement (now converted to string).

The script now runs successfully and outputs: `The average is: 3.0`

## 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:

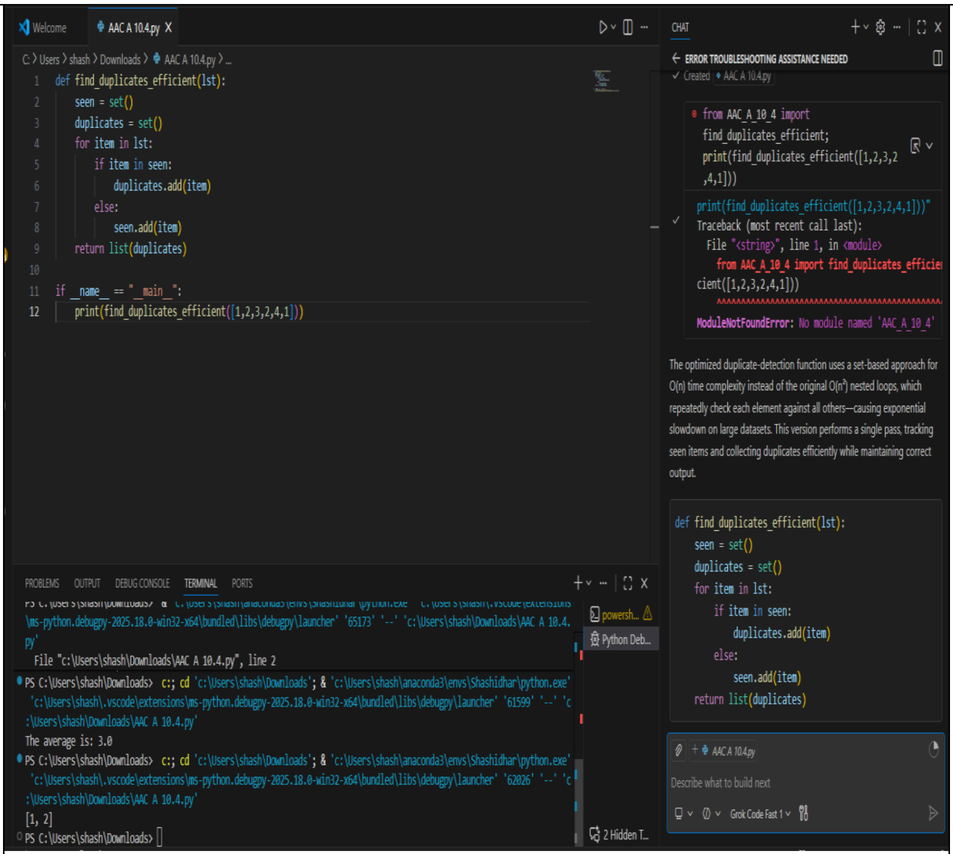
- Analyze the logic for performance bottlenecks
- Refactor the code for better time complexity
- Preserve the correctness of the output

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



```python
def find_duplicates_efficient(lst):
    seen = set()
    duplicates = set()
    for item in lst:
        if item in seen:
            duplicates.add(item)
        else:
            seen.add(item)
    return list(duplicates)

if __name__ == "__main__":
    print(find_duplicates_efficient([1,2,3,2,4,1]))
```

```
PS C:\Users\shash\Downloads> & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\extensions
\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '65173' '--' 'c:\Users\shash\Downloads\AAC A 10.4.
py'
  File "c:\Users\shash\Downloads\AAC A 10.4.py", line 2
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61599' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
The average is: 3.0
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62026' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
[1, 2]
PS C:\Users\shash\Downloads>
```

```
from AAC_A_10_4 import
find_duplicates_efficient;
print(find_duplicates_efficient([1,2,3,2
,4,1]))
```

```
print(find_duplicates_efficient([1,2,3,2,4,1]))"
Traceback (most recent call last):
  File "<string>", line 1, in <module>
    from AAC_A_10_4 import find_duplicates_efficie
cient([1,2,3,2,4,1]))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ModuleNotFoundError: No module named 'AAC_A_10_4'
```

The optimized duplicate-detection function uses a set-based approach for O(n) time complexity instead of the original $O(n^2)$ nested loops, which repeatedly check each element against all others—causing exponential slowdown on large datasets. This version performs a single pass, tracking seen items and collecting duplicates efficiently while maintaining correct output.

```python
def find_duplicates_efficient(lst):
    seen = set()
    duplicates = set()
    for item in lst:
        if item in seen:
            duplicates.add(item)
        else:
            seen.add(item)
    return list(duplicates)
```

---

# 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:

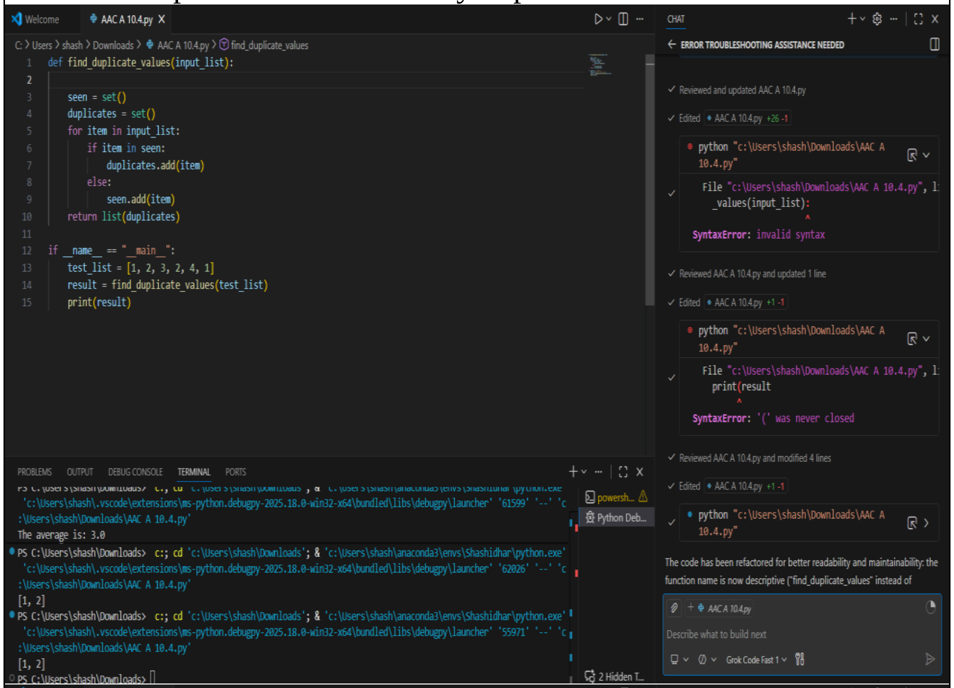
- Cryptic function names
- Poor indentation
- Unclear variable naming
- No documentation

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


Welcome
AAC A 10.4.py
C: › Users › shash › Downloads › AAC A 10.4.py › find_duplicate_values
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powersh...
Python Deb...
2 Hidden T...
CHAT
ERROR TROUBLESHOOTING ASSISTANCE NEEDED
AAC A 10.4.py
Describe what to build next
Grok Code Fast 1

```python
def find_duplicate_values(input_list):

    seen = set()
    duplicates = set()
    for item in input_list:
        if item in seen:
            duplicates.add(item)
        else:
            seen.add(item)
    return list(duplicates)

if __name__ == "__main__":
    test_list = [1, 2, 3, 2, 4, 1]
    result = find_duplicate_values(test_list)
    print(result)
```

```
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61599' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
The average is: 3.0
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62026' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
[1, 2]
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55971' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
[1, 2]
PS C:\Users\shash\Downloads> 
```

✓ Reviewed and updated AAC A 10.4.py

✓ Edited AAC A 10.4.py +26 -1

```
python "c:\Users\shash\Downloads\AAC A 10.4.py"
  File "c:\Users\shash\Downloads\AAC A 10.4.py", l
    _values(input_list):
                       ^
SyntaxError: invalid syntax
```

✓ Reviewed AAC A 10.4.py and updated 1 line

✓ Edited AAC A 10.4.py +1 -1

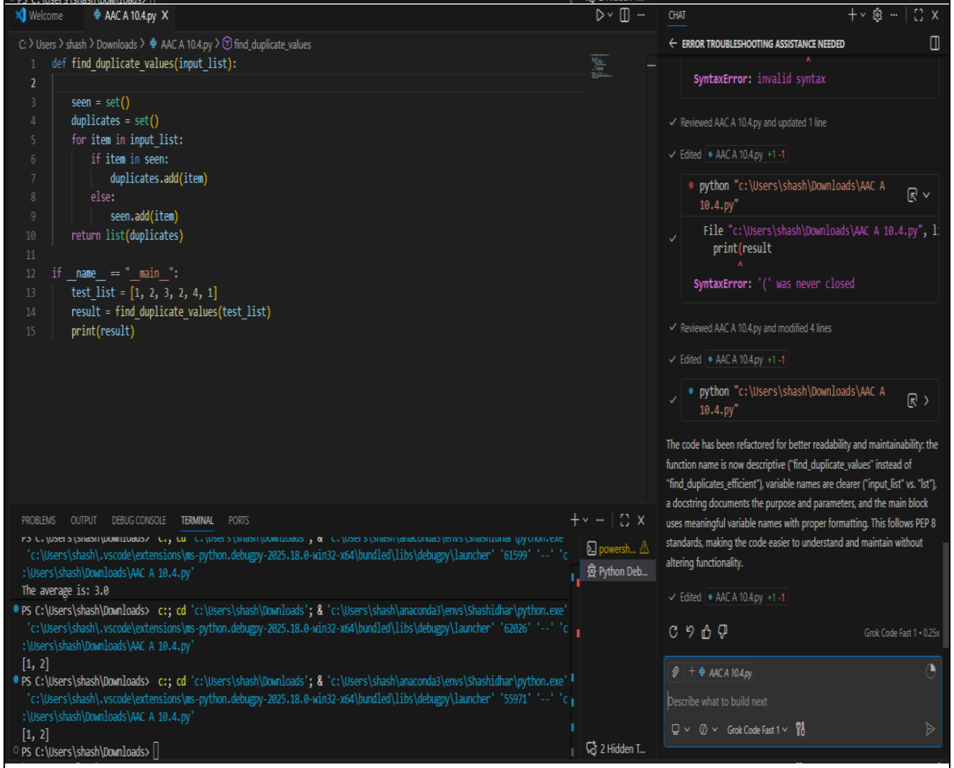
```
python "c:\Users\shash\Downloads\AAC A 10.4.py"
  File "c:\Users\shash\Downloads\AAC A 10.4.py", l
    print(result
         ^
SyntaxError: '(' was never closed
```

✓ Reviewed AAC A 10.4.py and modified 4 lines

✓ Edited AAC A 10.4.py +1 -1

```
python "c:\Users\shash\Downloads\AAC A 10.4.py"
```

The code has been refactored for better readability and maintainability: the function name is now descriptive ("find_duplicate_values" instead of



```python
def find_duplicate_values(input_list):

    seen = set()
    duplicates = set()
    for item in input_list:
        if item in seen:
            duplicates.add(item)
        else:
            seen.add(item)
    return list(duplicates)

if __name__ == "__main__":
    test_list = [1, 2, 3, 2, 4, 1]
    result = find_duplicate_values(test_list)
    print(result)
```

```
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61599' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
The average is: 3.0
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62026' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
[1, 2]
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55971' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
[1, 2]
PS C:\Users\shash\Downloads> 
```

```
                       ^
SyntaxError: invalid syntax
```

✓ Reviewed AAC A 10.4.py and updated 1 line

✓ Edited AAC A 10.4.py +1 -1

```
python "c:\Users\shash\Downloads\AAC A 10.4.py"
  File "c:\Users\shash\Downloads\AAC A 10.4.py", l
    print(result
         ^
SyntaxError: '(' was never closed
```

✓ Reviewed AAC A 10.4.py and modified 4 lines

✓ Edited AAC A 10.4.py +1 -1

```
python "c:\Users\shash\Downloads\AAC A 10.4.py"
```

The code has been refactored for better readability and maintainability: the function name is now descriptive ("find_duplicate_values" instead of "find_duplicates_efficient"), variable names are clearer ("input_list" vs. "lst"), a docstring documents the purpose and parameters, and the main block uses meaningful variable names with proper formatting. This follows PEP 8 standards, making the code easier to understand and maintain without altering functionality.

✓ Edited AAC A 10.4.py +1 -1

## 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:

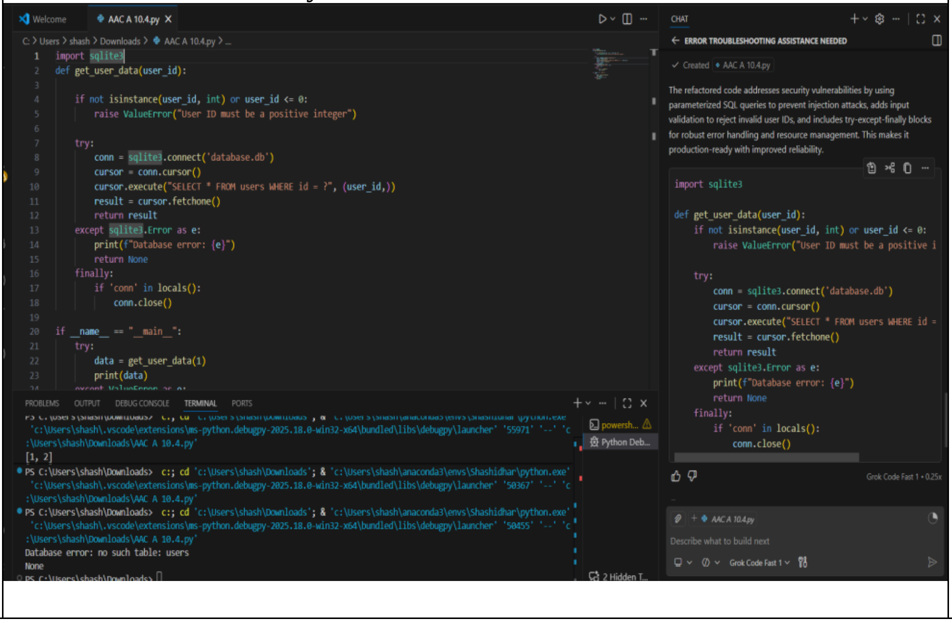
- Uses unsafe SQL query construction
- Has no input validation
- Lacks exception handling

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



```python
import sqlite3
def get_user_data(user_id):

    if not isinstance(user_id, int) or user_id <= 0:
        raise ValueError("User ID must be a positive integer")

    try:
        conn = sqlite3.connect('database.db')
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE id = ?", (user_id,))
        result = cursor.fetchone()
        return result
    except sqlite3.Error as e:
        print(f"Database error: {e}")
        return None
    finally:
        if 'conn' in locals():
            conn.close()

if __name__ == "__main__":
    try:
        data = get_user_data(1)
        print(data)
```

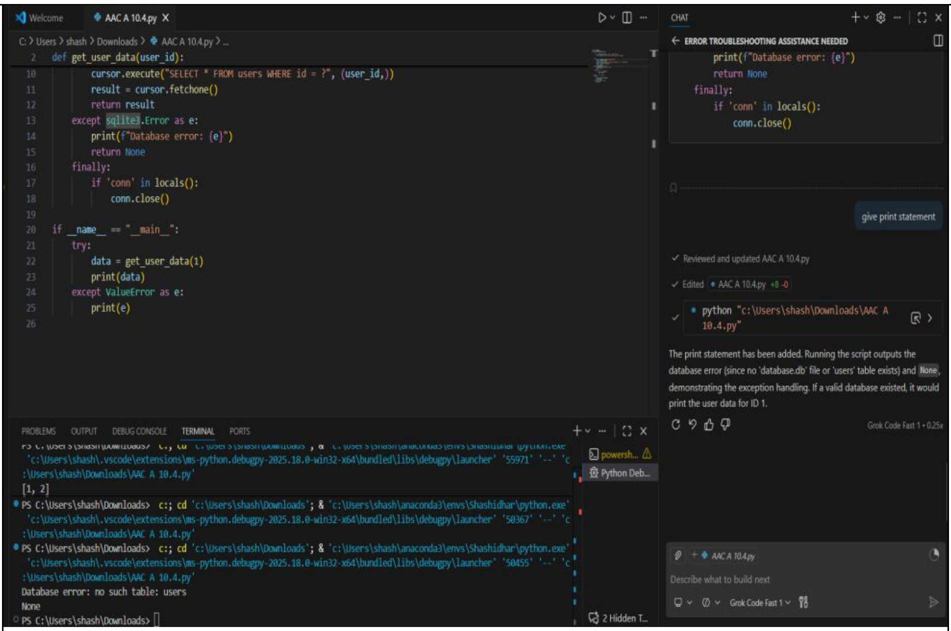
```
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55971' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
[1, 2]
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50367' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50455' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
Database error: no such table: users
None
PS C:\Users\shash\Downloads>
```

The refactored code addresses security vulnerabilities by using parameterized SQL queries to prevent injection attacks, adds input validation to reject invalid user IDs, and includes try-except-finally blocks for robust error handling and resource management. This makes it production-ready with improved reliability.

```python
import sqlite3

def get_user_data(user_id):
    if not isinstance(user_id, int) or user_id <= 0:
        raise ValueError("User ID must be a positive i

    try:
        conn = sqlite3.connect('database.db')
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE id =
        result = cursor.fetchone()
        return result
    except sqlite3.Error as e:
        print(f"Database error: {e}")
        return None
    finally:
        if 'conn' in locals():
            conn.close()
```



```python
def get_user_data(user_id):
        cursor.execute("SELECT * FROM users WHERE id = ?", (user_id,))
        result = cursor.fetchone()
        return result
    except sqlite3.Error as e:
        print(f"Database error: {e}")
        return None
    finally:
        if 'conn' in locals():
            conn.close()

if __name__ == "__main__":
    try:
        data = get_user_data(1)
        print(data)
    except ValueError as e:
        print(e)

```

```
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55971' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
[1, 2]
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50367' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50455' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
Database error: no such table: users
None
PS C:\Users\shash\Downloads> 
```

```python
        print(f"Database error: {e}")
        return None
    finally:
        if 'conn' in locals():
            conn.close()
```

give print statement

✓ Reviewed and updated AAC A 10.4.py

✓ Edited AAC A 10.4.py +8 -0

```
python "c:\Users\shash\Downloads\AAC A 10.4.py"
```

The print statement has been added. Running the script outputs the database error (since no 'database.db' file or 'users' table exists) and `None`, demonstrating the exception handling. If a valid database existed, it would print the user data for ID 1.

## 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.

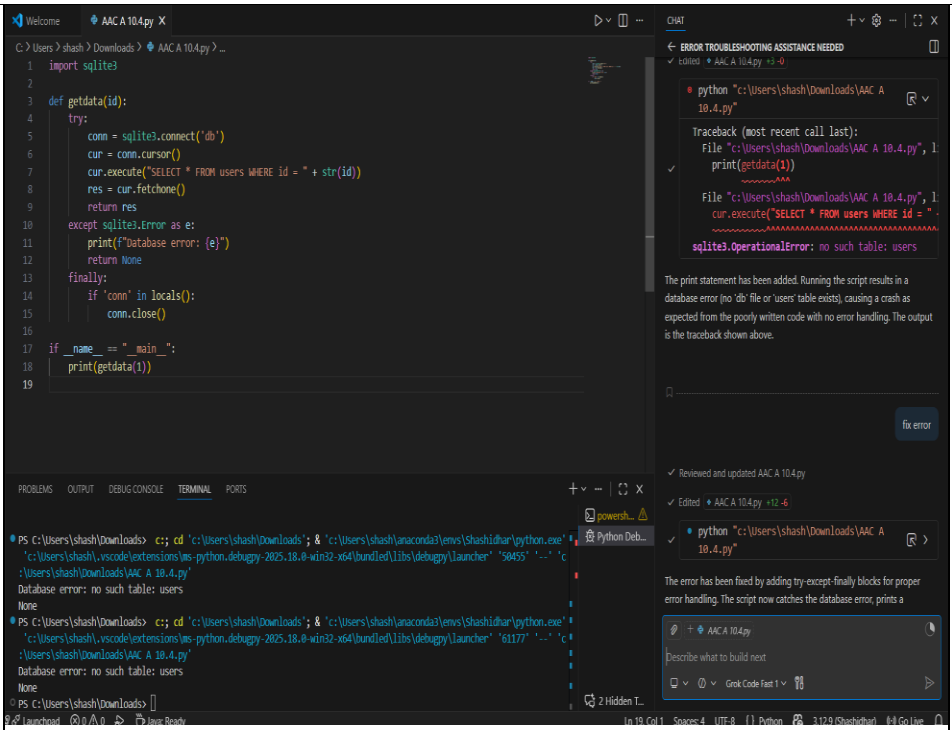
Using AI-assisted review:

- 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
- Demonstration of AI as a **code reviewer**, not just a code generator



```python
import sqlite3

def getdata(id):
    try:
        conn = sqlite3.connect('db')
        cur = conn.cursor()
        cur.execute("SELECT * FROM users WHERE id = " + str(id))
        res = cur.fetchone()
        return res
    except sqlite3.Error as e:
        print(f"Database error: {e}")
        return None
    finally:
        if 'conn' in locals():
            conn.close()

if __name__ == "__main__":
    print(getdata(1))

```

```
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50455' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
Database error: no such table: users
None
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61177' '--' 'c
:\Users\shash\Downloads\AAC A 10.4.py'
Database error: no such table: users
None
PS C:\Users\shash\Downloads> 
```

✓ Edited AAC A 10.4.py +3 -0

```
python "c:\Users\shash\Downloads\AAC A
10.4.py"

Traceback (most recent call last):
  File "c:\Users\shash\Downloads\AAC A 10.4.py", l
    print(getdata(1))
          ~~~~~~~^^^
  File "c:\Users\shash\Downloads\AAC A 10.4.py", l
    cur.execute("SELECT * FROM users WHERE id = " 
    ~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
sqlite3.OperationalError: no such table: users
```

The print statement has been added. Running the script results in a database error (no 'db' file or 'users' table exists), causing a crash as expected from the poorly written code with no error handling. The output is the traceback shown above.

fix error

✓ Reviewed and updated AAC A 10.4.py

✓ Edited AAC A 10.4.py +12 -6

```
python "c:\Users\shash\Downloads\AAC A
10.4.py"
```

The error has been fixed by adding try-except-finally blocks for proper error handling. The script now catches the database error, prints a

AAC A 10.4.py

Describe what to build next

**Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**