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))
```

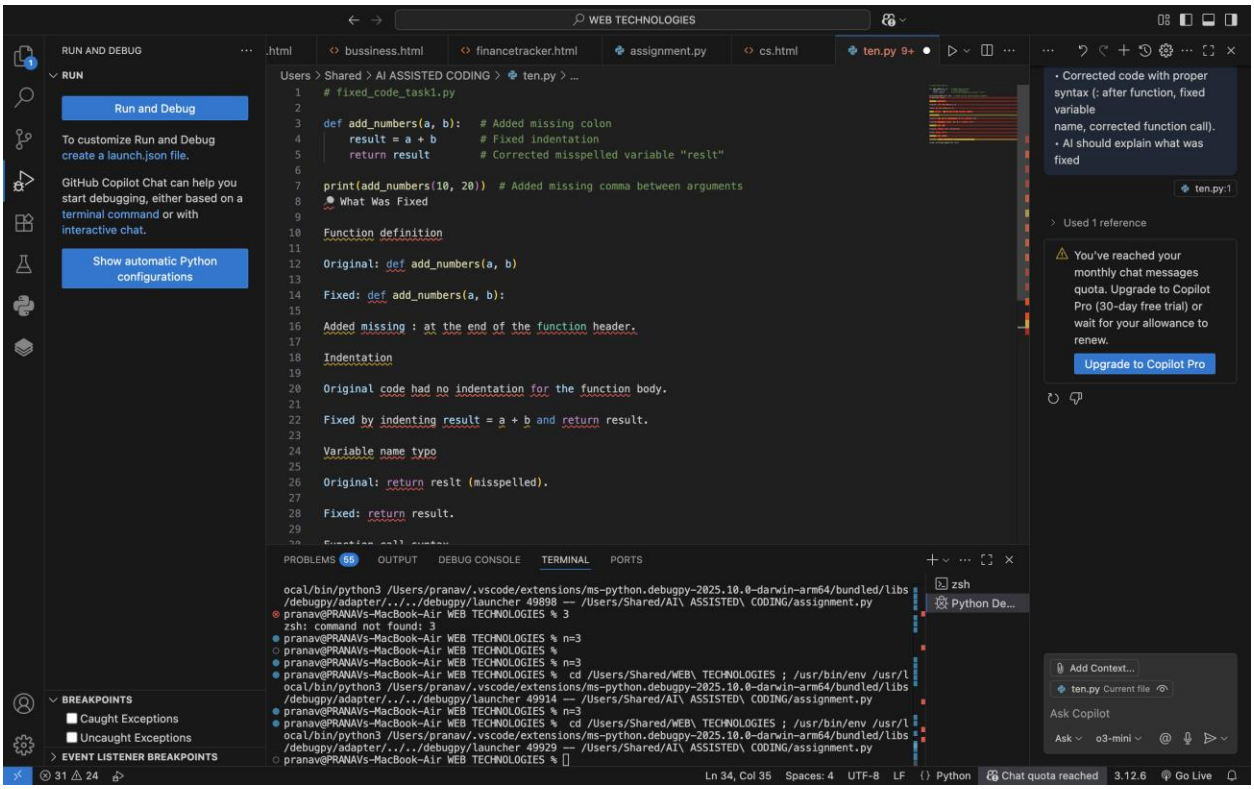
Expected Output:

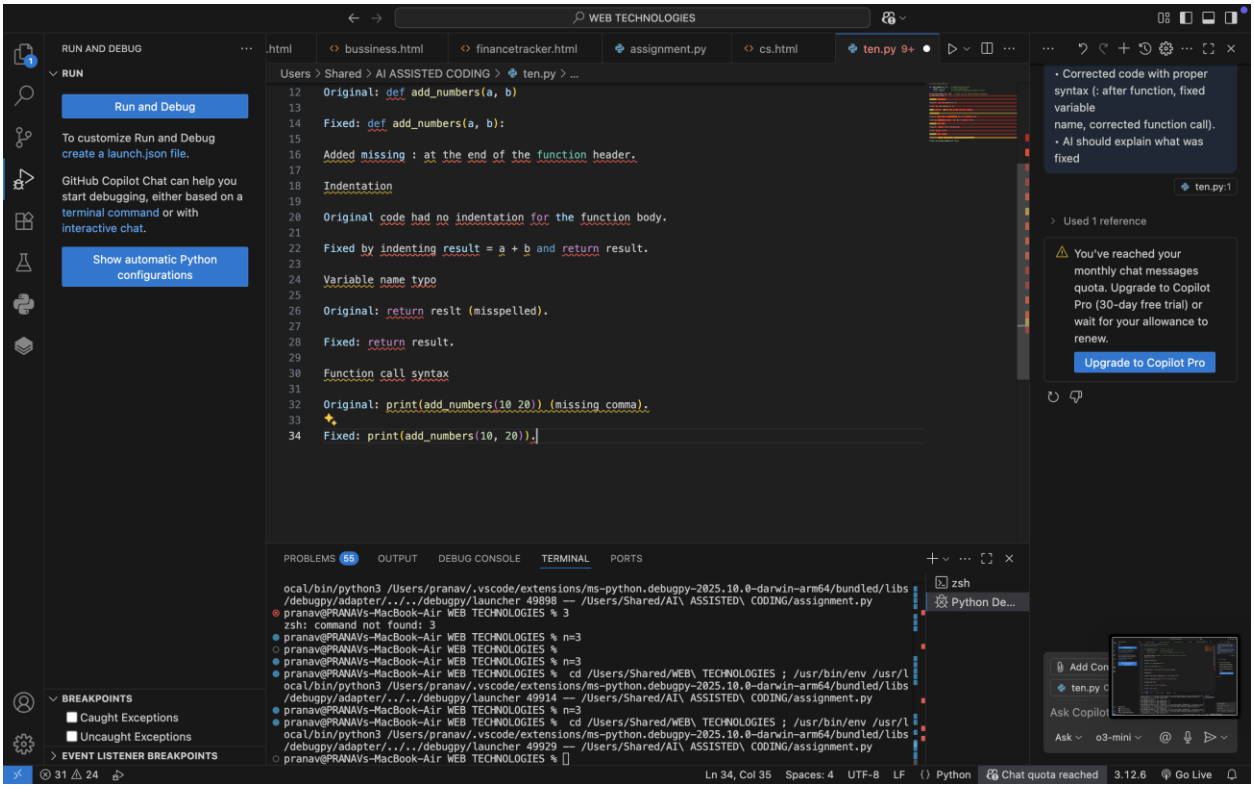
• Corrected code with proper syntax (: after function, fixed variable name, corrected function call).

• AI should explain what was fixed

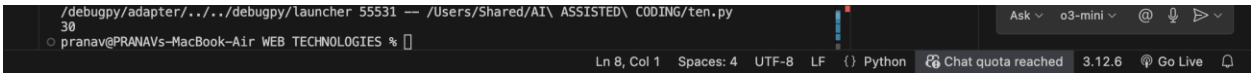
PROMPT :

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


WEB TECHNOLOGIES
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
GitHub Copilot Chat can help you start debugging, either based on a terminal command or with interactive chat.
Show automatic Python configurations
BREAKPOINTS
Caught Exceptions
Uncaught Exceptions
EVENT LISTENER BREAKPOINTS
.html
bussiness.html
financetracker.html
assignment.py
cs.html
ten.py 9+
Users > Shared > AI ASSISTED CODING > ten.py > ...
fixed_code_task1.py
def add_numbers(a, b): # Added missing colon
result = a + b # Fixed indentation
return result # Corrected misspelled variable "reslt"
print(add_numbers(10, 20)) # Added missing comma between arguments
What Was Fixed
Function definition
Original: def add_numbers(a, b)
Fixed: def add_numbers(a, b):
Added missing : at the end of the function header.
Indentation
Original code had no indentation for the function body.
Fixed by indenting result = a + b and return result.
Variable name typo
Original: return reslt (misspelled).
Fixed: return result.
PROBLEMS 55
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
ocal/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs /debugpy/adapter/../../debugpy/launcher 49898 -- /Users/Shared/AI\ ASSISTED\ CODING/assignment.py
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % 3
zsh: command not found: 3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % n=3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES %
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % n=3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB\ TECHNOLOGIES ; /usr/bin/env /usr/l
ocal/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs /debugpy/adapter/../../debugpy/launcher 49914 -- /Users/Shared/AI\ ASSISTED\ CODING/assignment.py
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % n=3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB\ TECHNOLOGIES ; /usr/bin/env /usr/l
ocal/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 %
zsh
Python De...
• Corrected code with proper syntax (: after function, fixed variable name, corrected function call).
• AI should explain what was fixed
ten.py:1
Used 1 reference
You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.
Upgrade to Copilot Pro
Add Context...
ten.py Current file
Ask Copilot
Ask
o3-mini
31 24
Ln 34, Col 35
Spaces: 4
UTF-8
LF
Python
Chat quota reached
3.12.6
Go Live


WEB TECHNOLOGIES
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
GitHub Copilot Chat can help you start debugging, either based on a terminal command or with interactive chat.
Show automatic Python configurations
BREAKPOINTS
Caught Exceptions
Uncaught Exceptions
EVENT LISTENER BREAKPOINTS
.html
bussiness.html
financetracker.html
assignment.py
cs.html
ten.py 9+
Users > Shared > AI ASSISTED CODING > ten.py > ...
Original: def add_numbers(a, b)
Fixed: def add_numbers(a, b):
Added missing : at the end of the function header.
Indentation
Original code had no indentation for the function body.
Fixed by indenting result = a + b and return result.
Variable name typo
Original: return reslt (misspelled).
Fixed: return result.
Function call syntax
Original: print(add_numbers(10 20)) (missing comma).
Fixed: print(add_numbers(10, 20)).
PROBLEMS 55
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
ocal/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs /debugpy/adapter/../../debugpy/launcher 49898 -- /Users/Shared/AI\ ASSISTED\ CODING/assignment.py
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % 3
zsh: command not found: 3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % n=3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES %
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % n=3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB\ TECHNOLOGIES ; /usr/bin/env /usr/l
ocal/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs /debugpy/adapter/../../debugpy/launcher 49914 -- /Users/Shared/AI\ ASSISTED\ CODING/assignment.py
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % n=3
pranav@PRANAVs-MacBook-Air WEB TECHNOLOGIES % cd /Users/Shared/WEB\ TECHNOLOGIES ; /usr/bin/env /usr/l
ocal/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 %
zsh
Python De...
• Corrected code with proper syntax (: after function, fixed variable name, corrected function call).
• AI should explain what was fixed
ten.py:1
Used 1 reference
You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.
Upgrade to Copilot Pro
Ask Copilot
Ask
o3-mini
31 24
Ln 34, Col 35
Spaces: 4
UTF-8
LF
Python
Chat quota reached
3.12.6
Go Live

OUTPUT :



CHANGES :

1. **Function definition**
    a. Original: def add_numbers(a, b)
    b. Fixed: def add_numbers(a, b):
    c. Added missing : at the end of the function header.
2. **Indentation**
    a. Original code had no indentation for the function body.
    b. Fixed by indenting result = a + b and return result.
3. **Variable name typo**
    a. Original: return reslt (misspelled).
    b. Fixed: return result.
4. **Function call syntax**
    a. Original: print(add_numbers(10 20)) (missing comma).
    b. Fixed: print(add_numbers(10, 20)).

Task 2: Logical and Performance Issue Review
Task: Optimize inefficient logic while keeping the result correct.

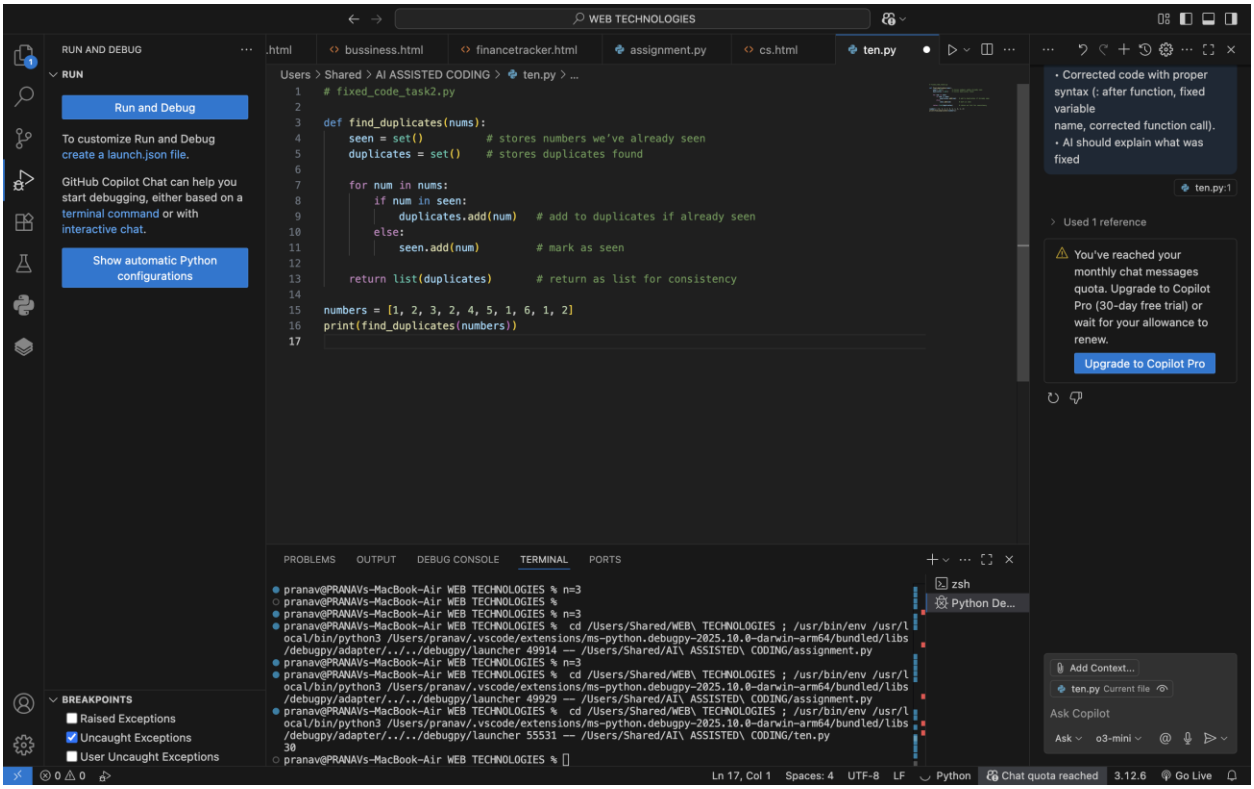
```
# buggy_code_task2.py
def find_duplicates(nums):
duplicates = []
for i in range(len(nums)):
for j in range(len(nums)):
if i != j and nums[i] == nums[j] and nums[i] not in duplicates:
duplicates.append(nums[i])
return duplicates
numbers = [1,2,3,2,4,5,1,6,1,2]
print(find_duplicates(numbers))
```

Expected Output:
• More efficient duplicate detection (e.g., using sets).
• AI should explain the optimization.

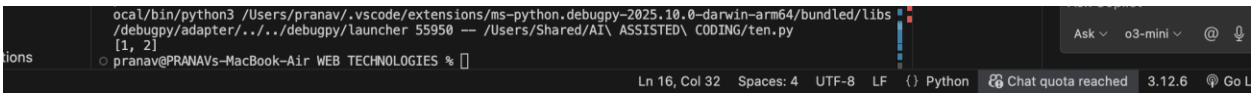
PROMPT USED :

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

CODE GENERATED :



OUTPUT:



CHANGES MADE :

1. **Original Logic**
    a. Used **two nested loops** → O(n^2) time complexity.
    b. Checked each element against all others.
2. **New Logic**
    a. Uses **two sets (seen, duplicates)** → only **one loop** → O(n) time complexity.
    b. 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).

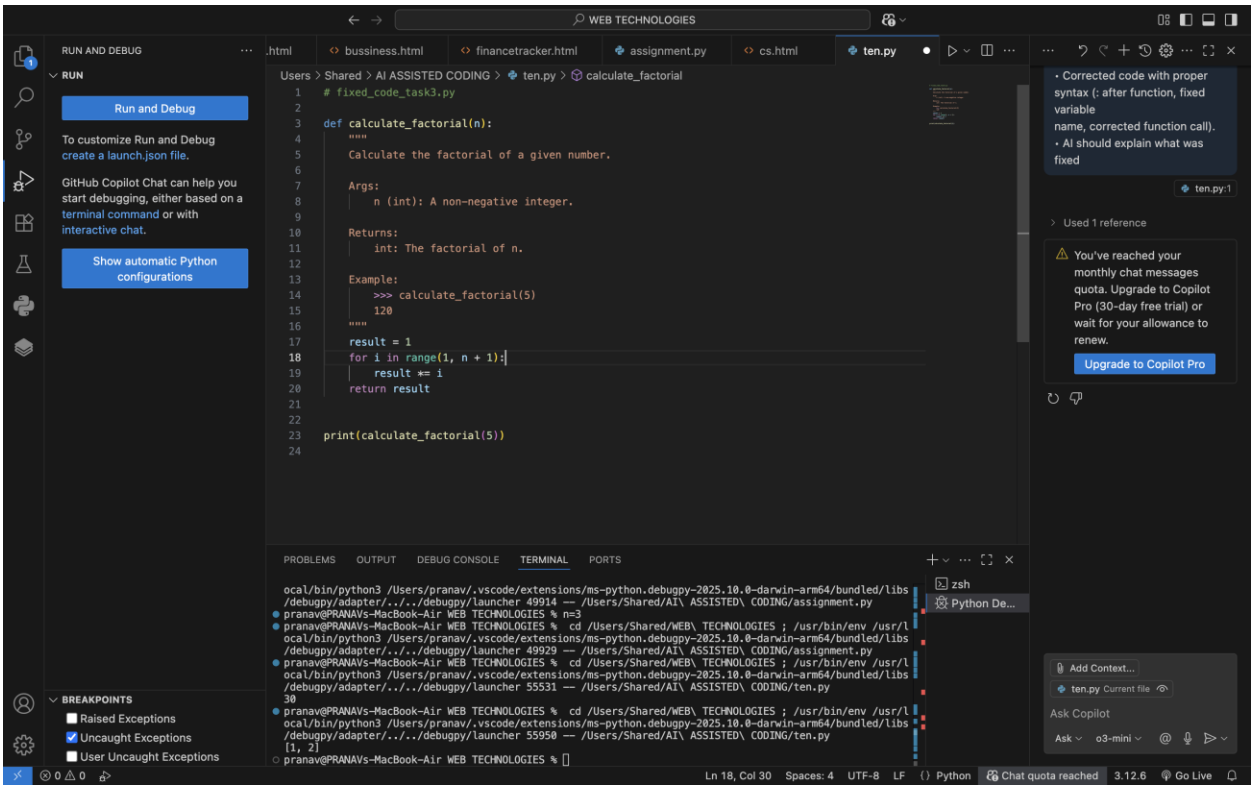
Task 3: Code Refactoring for Readability
Task: Refactor messy code into clean, PEP 8–compliant, well-structured code.
buggy_code_task3.py

```
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 :



OUTPUT GENERATED :

Improvements Made

1. **Function name**
    a. Original: c → unclear.
    b. Fixed: calculate_factorial → descriptive and meaningful.
2. **Variable naming**
    a. Original: x → vague.
    b. 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).

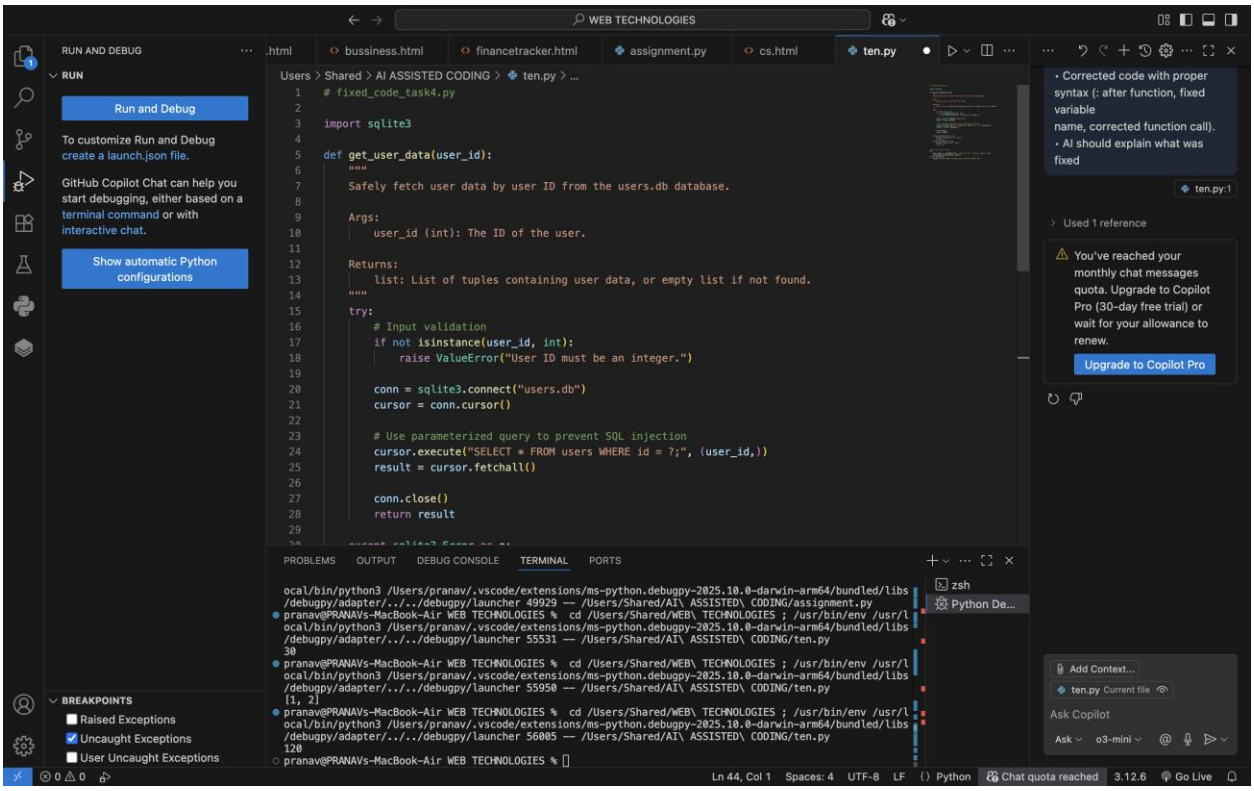
Try-except block for database errors.

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 :



OUTPUT GENERATED :

CHANGES MADE :

1. **SQL Injection Protection**
    a. Replaced string interpolation (f"SELECT ... {user_id}") with **parameterized queries** (? placeholders).
2. **Input Validation**
    a. Ensures user_id is an integer before running the query.
3. **Error Handling**
    a. Added try-except for sqlite3.Error (database issues).
    b. 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.

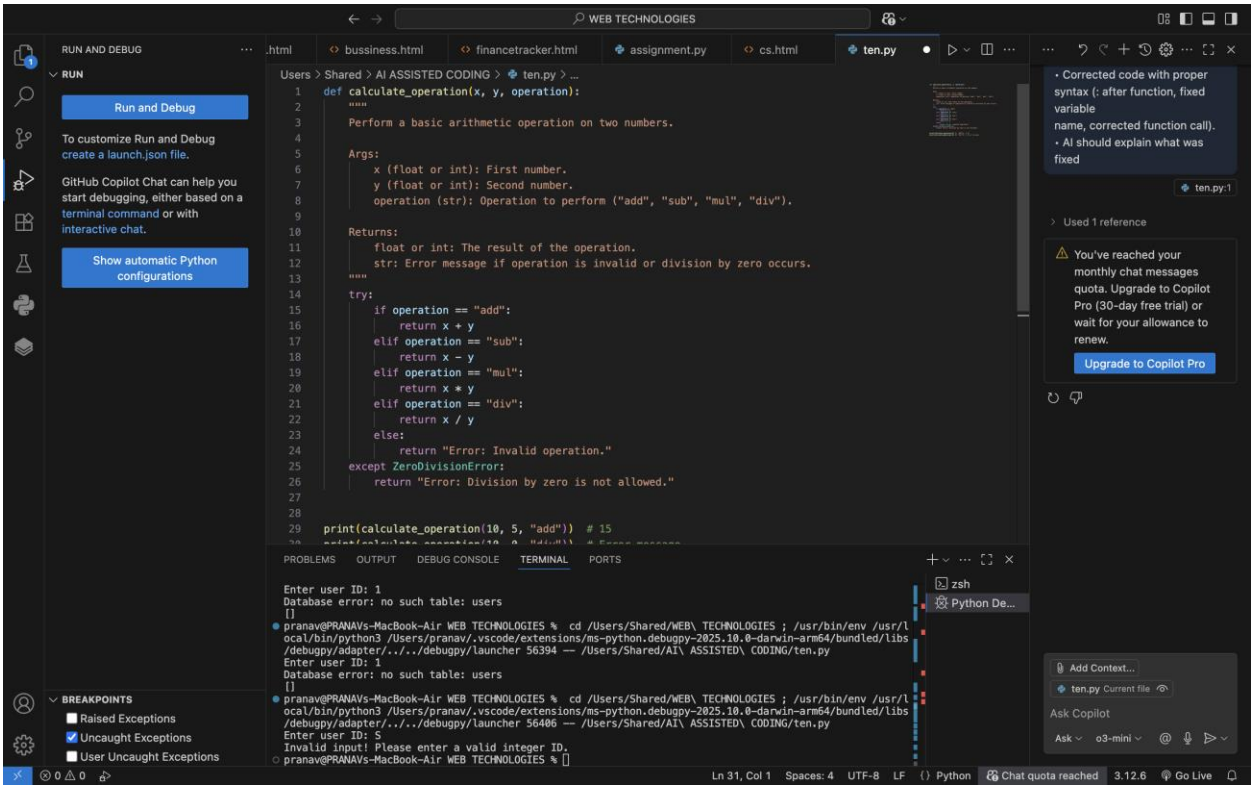
Task 5: Automated Code Review Report Generation
Task: Generate a review report for this messy code.
buggy_code_task5.py

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

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

REGENERATED CODE :



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.