AI ASSISTED CODING ASSIGNMENT-6.2

Task Description #1 (Documentation – Google-Style Docstrings for Python Functions)

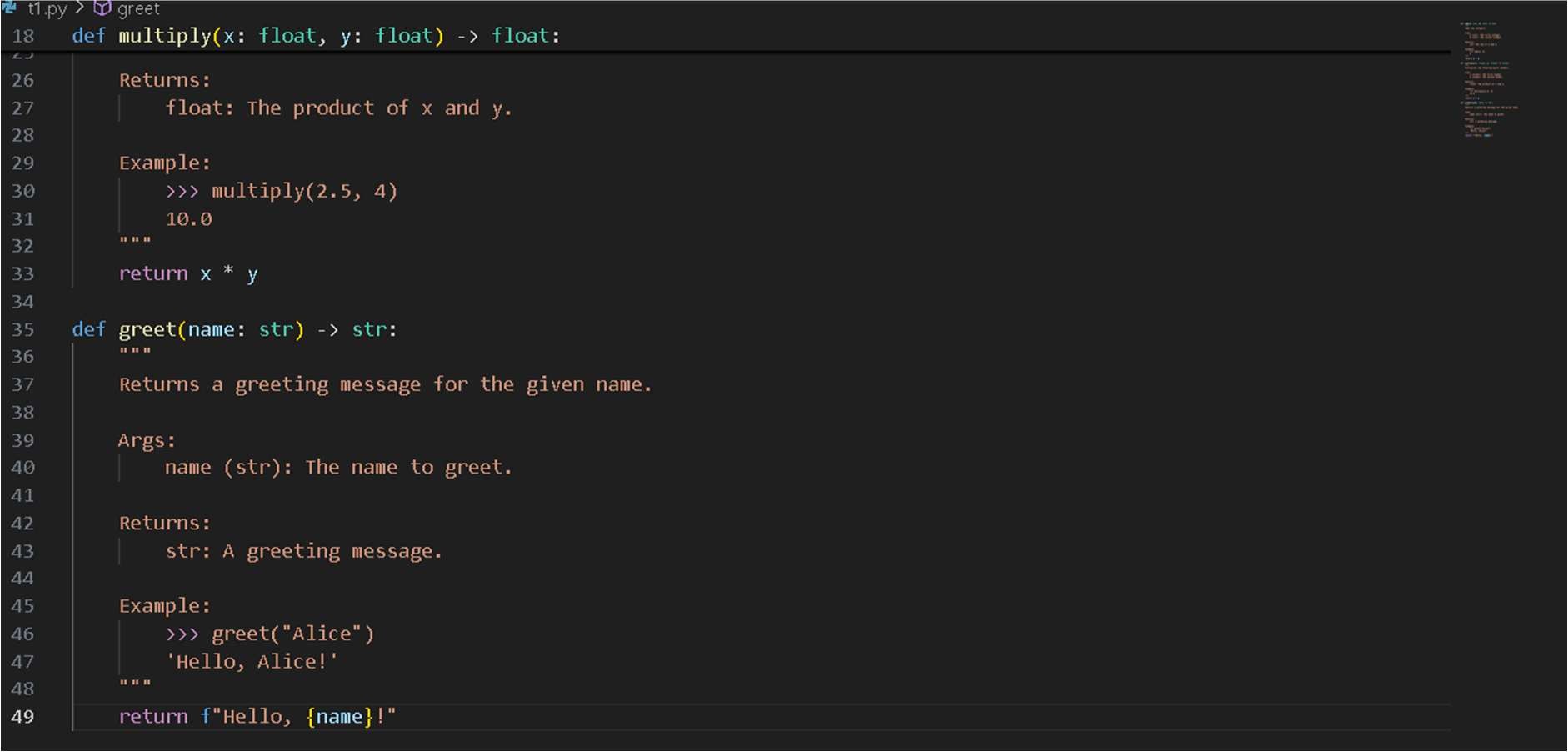
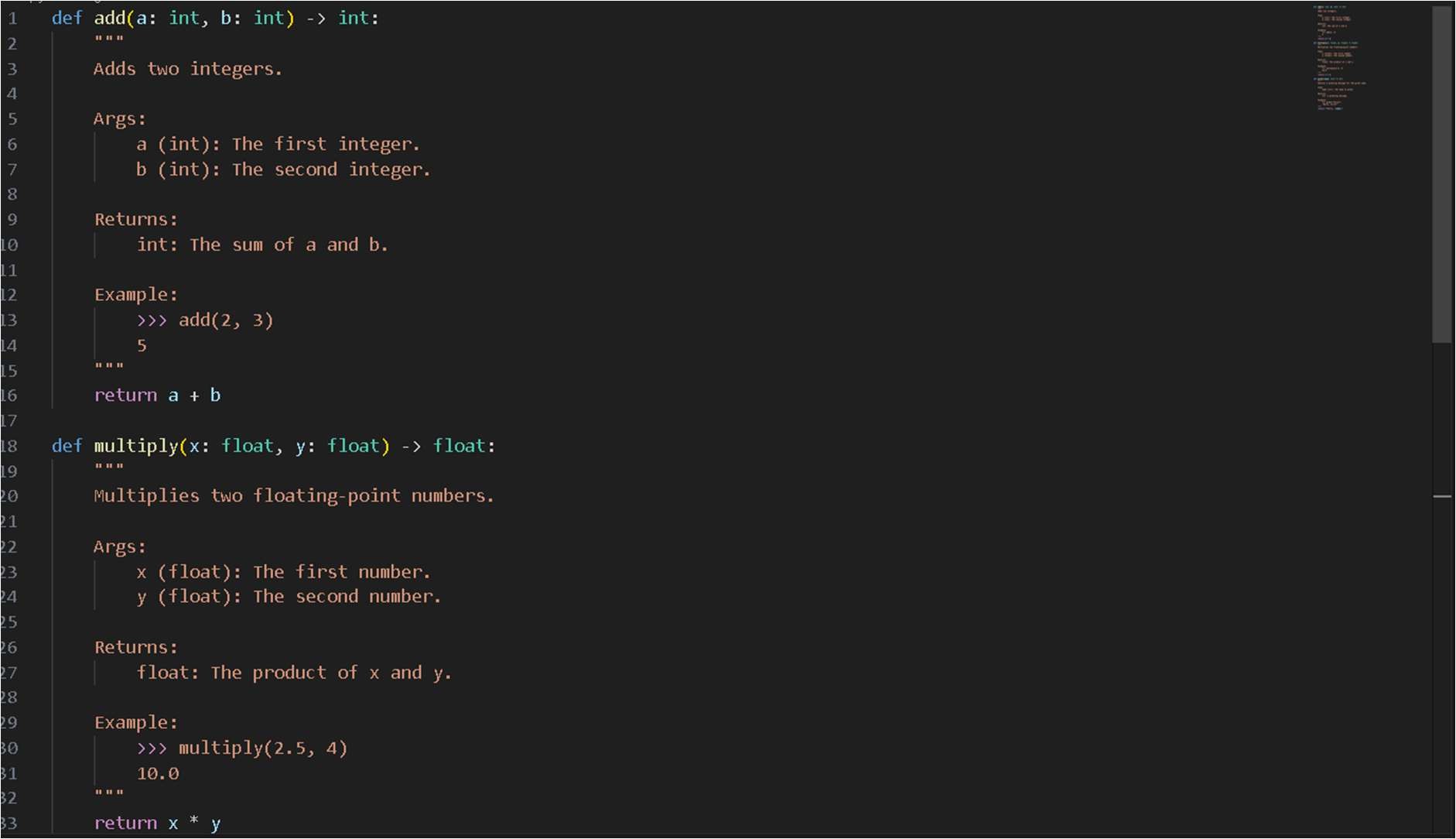
* Task: Use AI to add Google-style docstrings to all functions in a given Python script.
* Instructions:
* Prompt AI to generate docstrings without providing any input-output examples.
* Ensure each docstring includes:
* Function description
* Parameters with type hints
* Return values with type hints
* Example usage
* Review the generated docstrings for accuracy and formatting.
* Expected Output #1:
* A Python script with all functions documented using correctly formatted Google-style docstrings.

# PROMPT:

Task: Add Google-style docstrings to all functions in this file. Requirements:

* Include function description
* Parameters with type hints
* Return values with type hints
* Example usage
* Do not include input/output examples other than Example section.

# Code and Output:



Task Description #2 (Documentation – Inline Comments for Complex Logic)

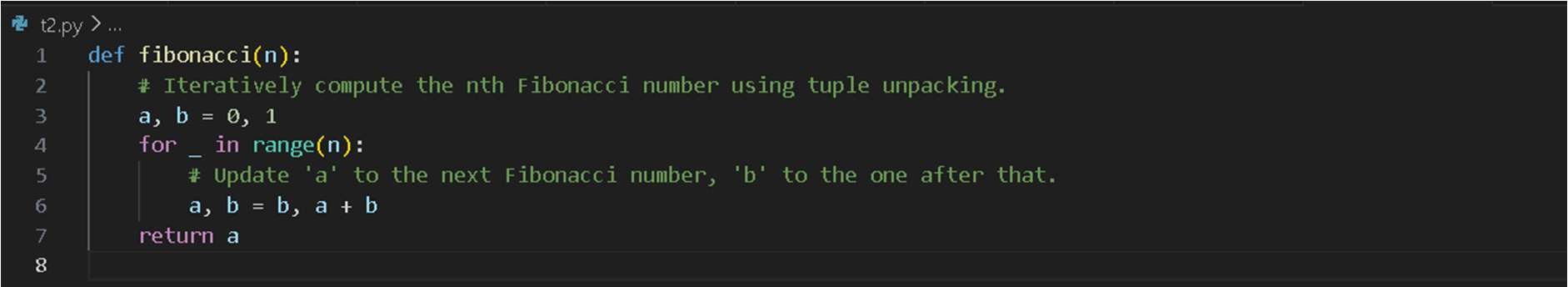
* Task: Use AI to add meaningful inline comments to a Python program explaining only complex logic parts.
* Instructions:
* Provide a Python script without comments to the AI.
* Instruct AI to skip obvious syntax explanations and focus only on tricky or non-intuitive code sections.
* Verify that comments improve code readability and maintainability.
* Expected Output #2:
* Python code with concise, context-aware inline comments for complex logic blocks.

# Prompt:

Task: Add meaningful inline comments to this Python program. Instructions for AI (Copilot):

* Focus on complex or non-intuitive logic blocks only.
* Skip obvious syntax explanations (like variable assignments or basic loops).
* Comments should improve readability and maintainability of the code.

# Code and Output:



Task Description #3 (Documentation – Module-Level Documentation)

* Task: Use AI to create a module-level docstring summarizing the purpose, dependencies, and main functions/classes of a Python file.
* Instructions:
* Supply the entire Python file to AI.
* Instruct AI to write a single multi-line docstring at the top of the file.
* Ensure the docstring clearly describes functionality and usage without rewriting the entire code.
* Expected Output #3:
* A complete, clear, and concise module-level docstring at the beginning of the file

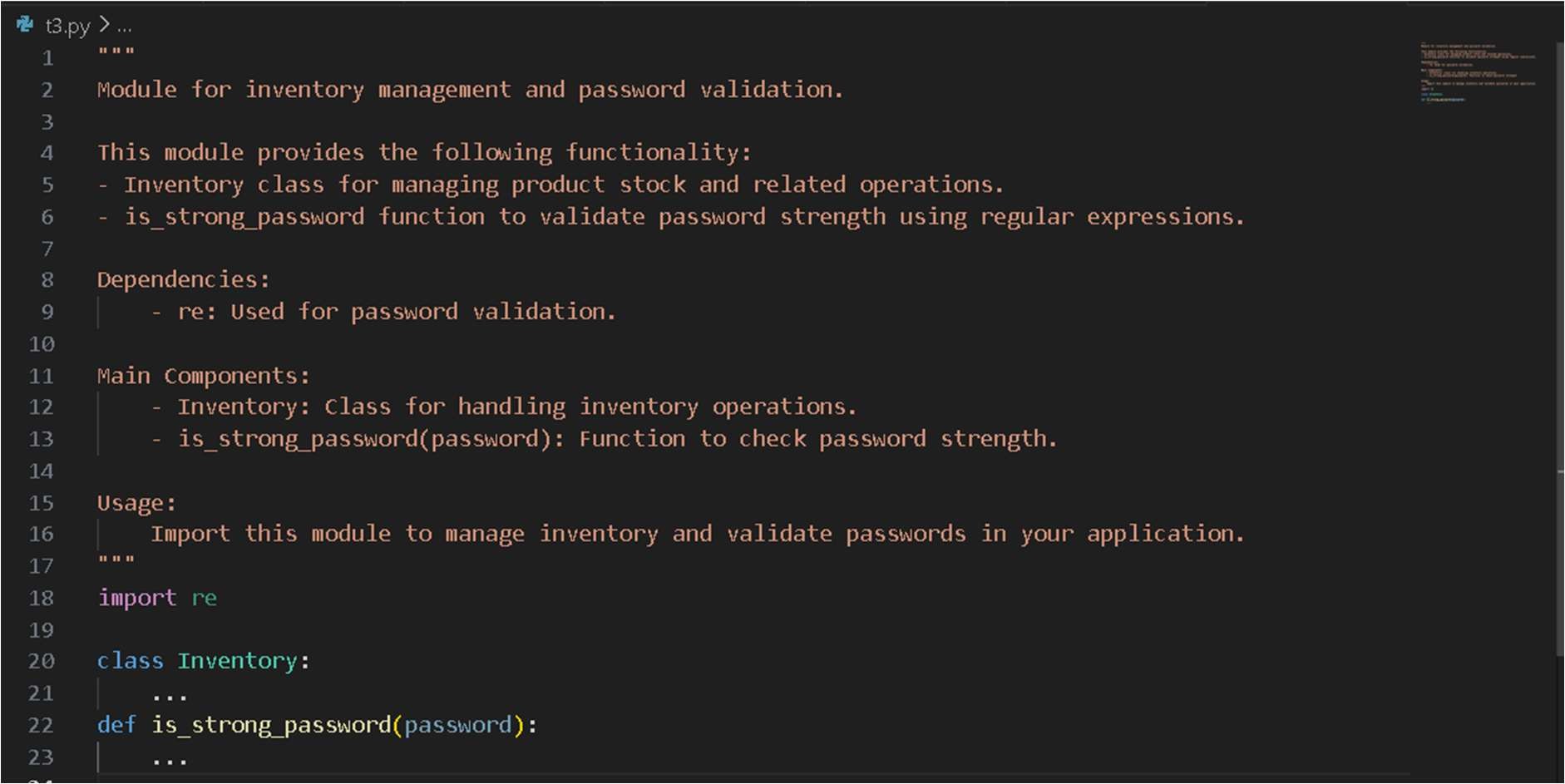
# Prompt:

Task: Create a module-level docstring summarizing the purpose, dependencies, and main functions/classes of this Python file.

Instructions for AI (Copilot):

* Write a single multi-line docstring at the very top of the file.
* Clearly describe the module’s functionality and usage.
* Include dependencies (imports), main functions, and classes.
* Do not rewrite or duplicate the entire code.

# Code and output:



Task Description #4 (Documentation – Convert Comments to Structured Docstrings)

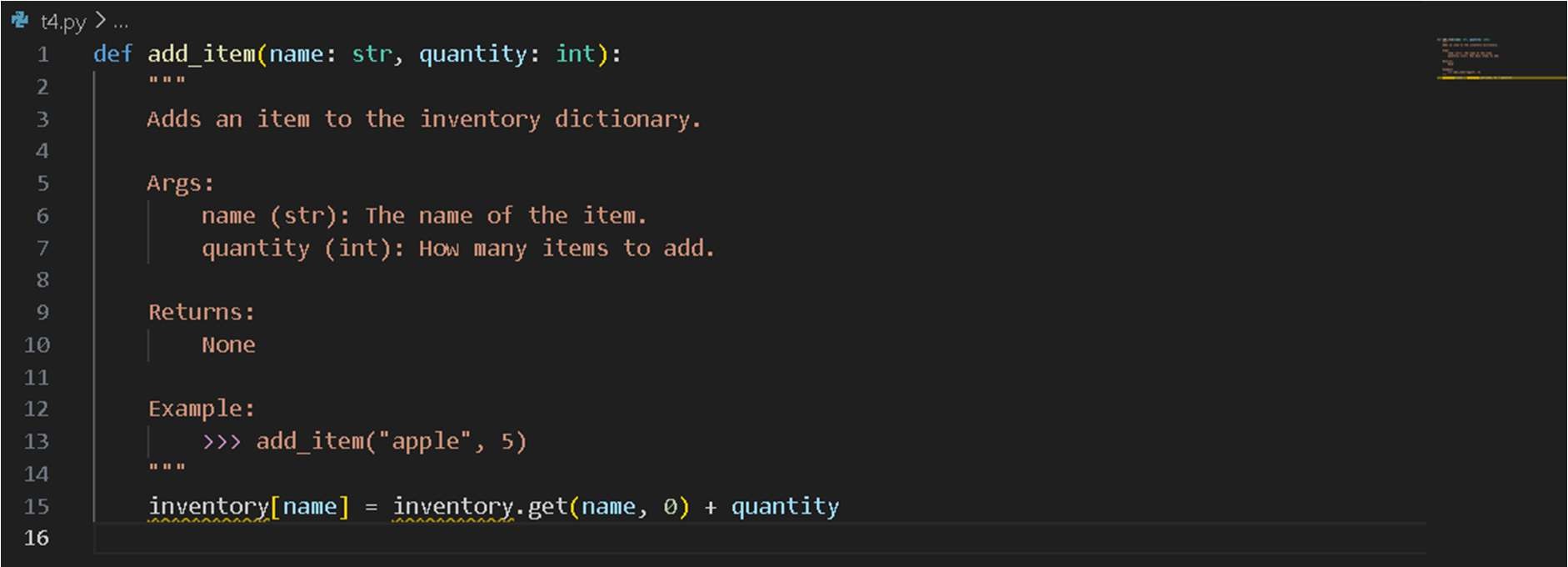
* Task: Use AI to transform existing inline comments into structured function docstrings following Google style.
* Instructions:
* Provide AI with Python code containing inline comments.
* Ask AI to move relevant details from comments into function docstrings.
* Verify that the new docstrings keep the meaning intact while improving structure.
* Expected Output #4:
* Python code with comments replaced by clear, standardized docstrings.

# Prompt:

Task: Transform existing inline comments into structured Google-style docstrings. Instructions for AI (Copilot):

* For each function, move meaningful inline comments into a docstring directly under the function definition.
* Use Google-style docstring format.
* Include function description, parameters (with type hints), returns (with type hints), and an Example section.
* Remove or minimize redundant inline comments once docstring is added.
* Keep the function logic unchanged.

# Code and output:



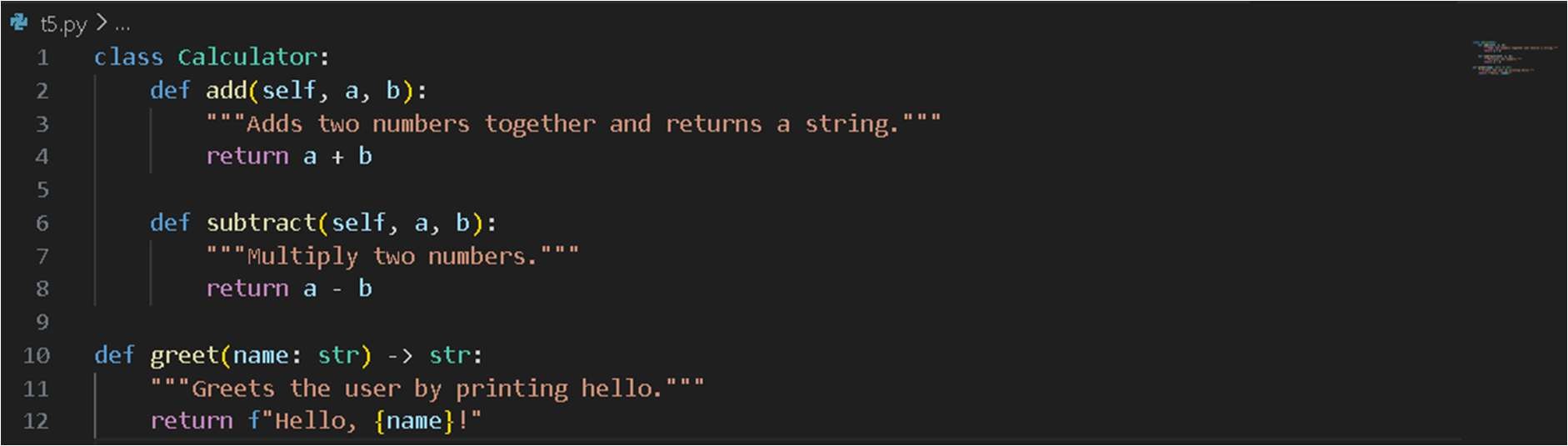
Task Description #5 (Documentation – Review and Correct Docstrings)

* Task: Use AI to identify and correct inaccuracies in existing docstrings.
* Instructions:
* Provide Python code with outdated or incorrect docstrings.
* Instruct AI to rewrite each docstring to match the current code behavior.
* Ensure corrections follow Google-style formatting.
* Expected Output #5:
* Python file with updated, accurate, and standardized docstrings.

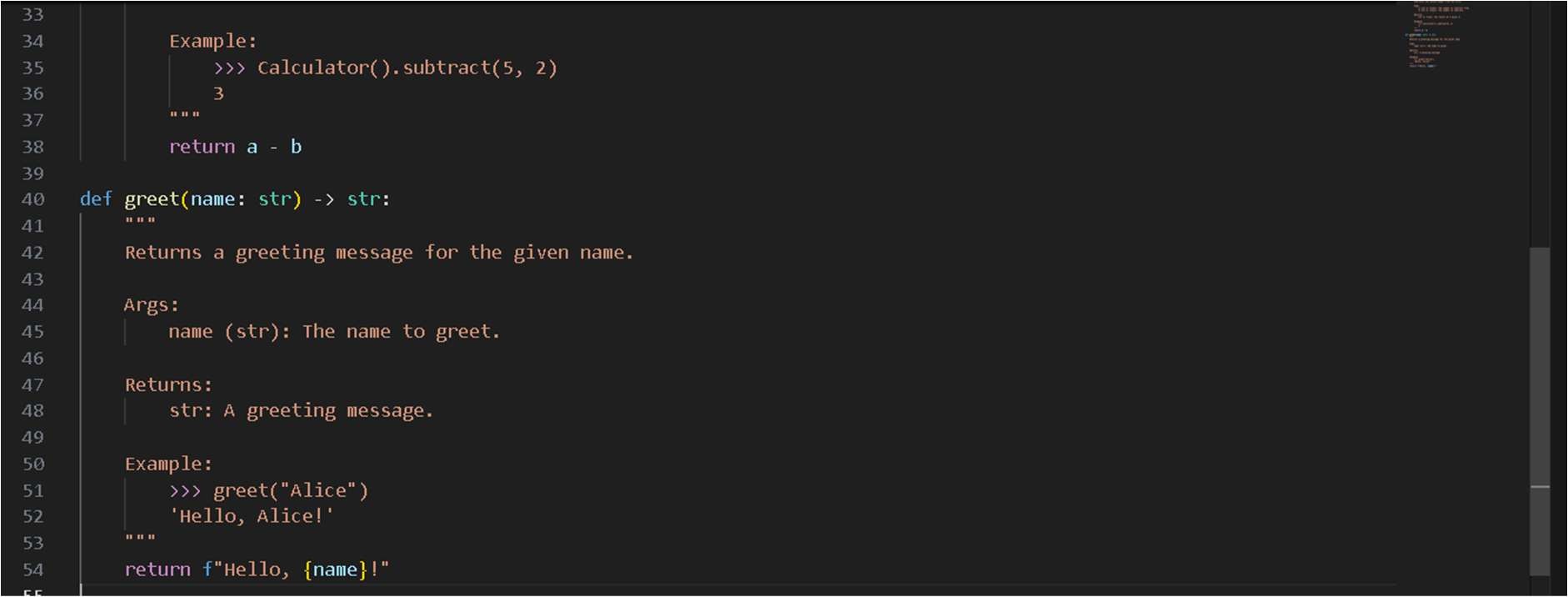
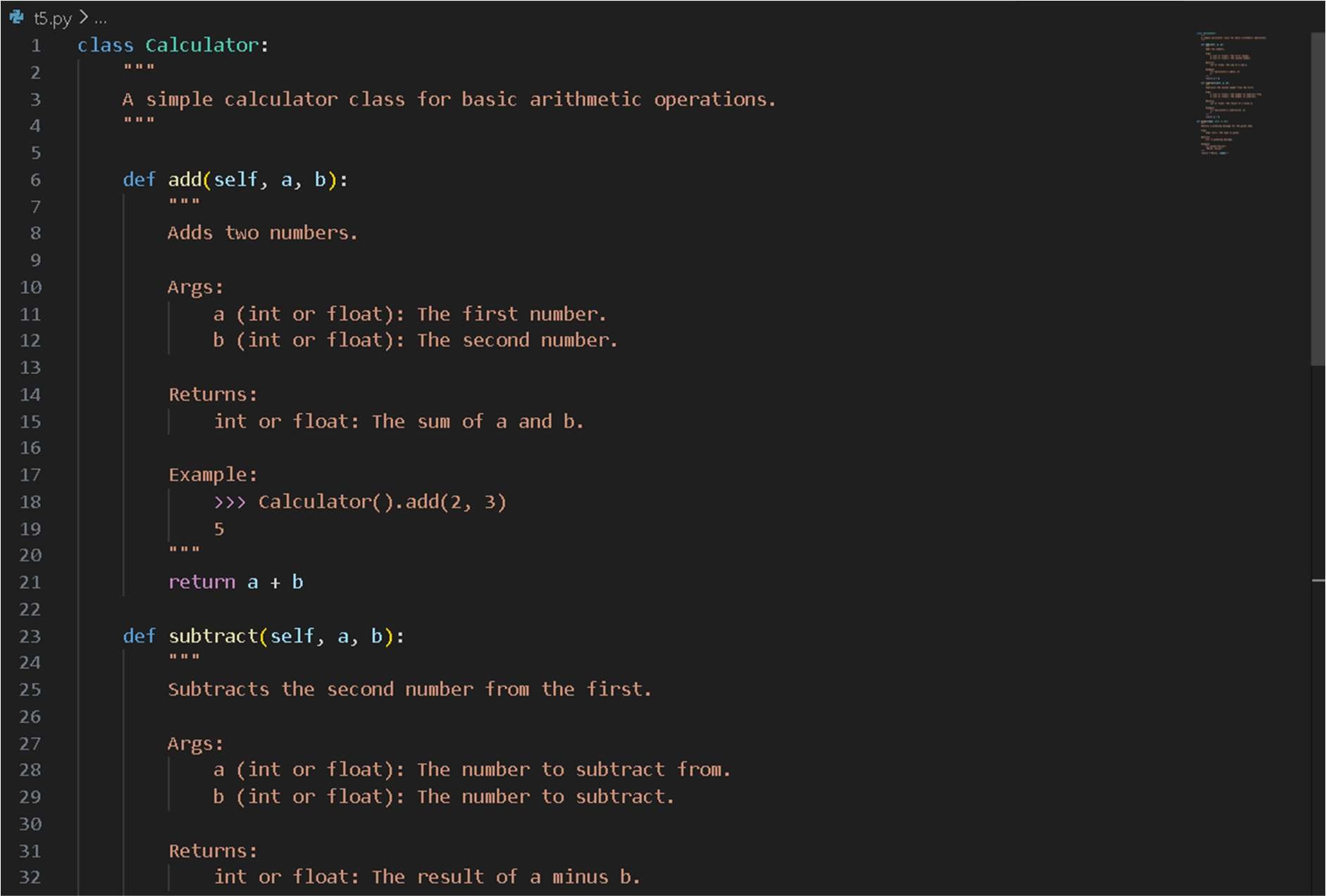
# Prompt:

I have Python code with outdated or incorrect docstrings. Please review the code, correct the docstrings to accurately describe the functions/classes, and rewrite them in Google-style formatting.

# Code with outdated docstrings :



Updated code:



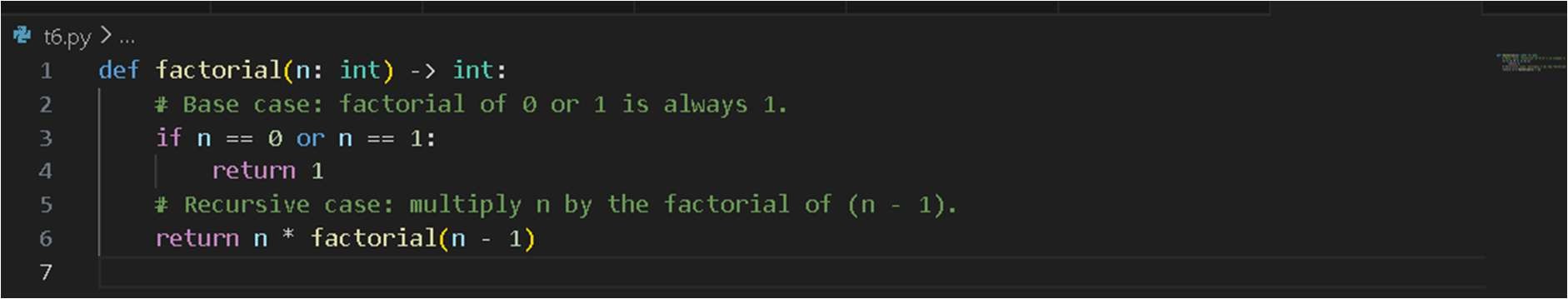
Task Description #6 (Documentation – Prompt Comparison Experiment)

* Task: Compare documentation output from a vague prompt and a detailed prompt for the same Python function.
* Instructions:
* Create two prompts: one simple (“Add comments to this function”) and one detailed (“Add Google-style docstrings with parameters, return types, and examples”).
* Use AI to process the same Python function with both prompts.
* Analyze and record differences in quality, accuracy, and completeness.
* Expected Output #6:
* A comparison table showing the results from both prompts with observations.

# Vague Prompt:

Add comments to this function.

# Code for vague prompt:



Detailed Prompt:

Add Google-style docstrings with parameters, return types, and examples.

Code for detailed prompt:

