# Assignment-9.4

Name: Arjun Manoj

H. No:2303A52134

Batch:44

Task 1: Auto-Generating Function Documentation in a Shared Codebase

Ask the AI to automatically generate Google-style function

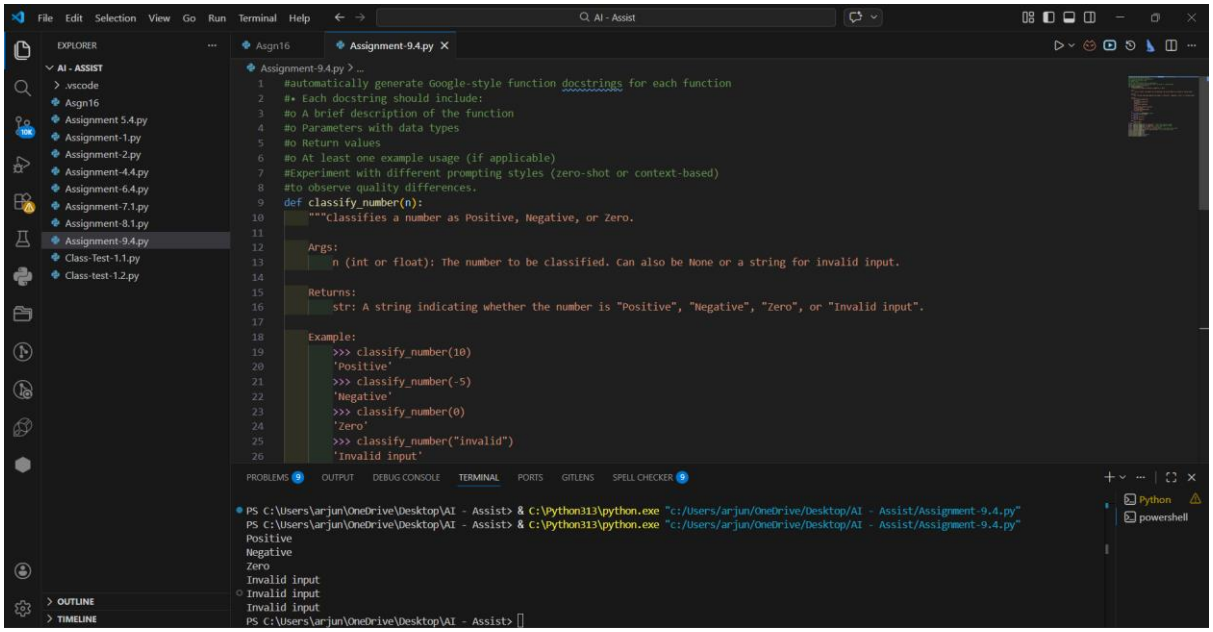
docstrings for each function

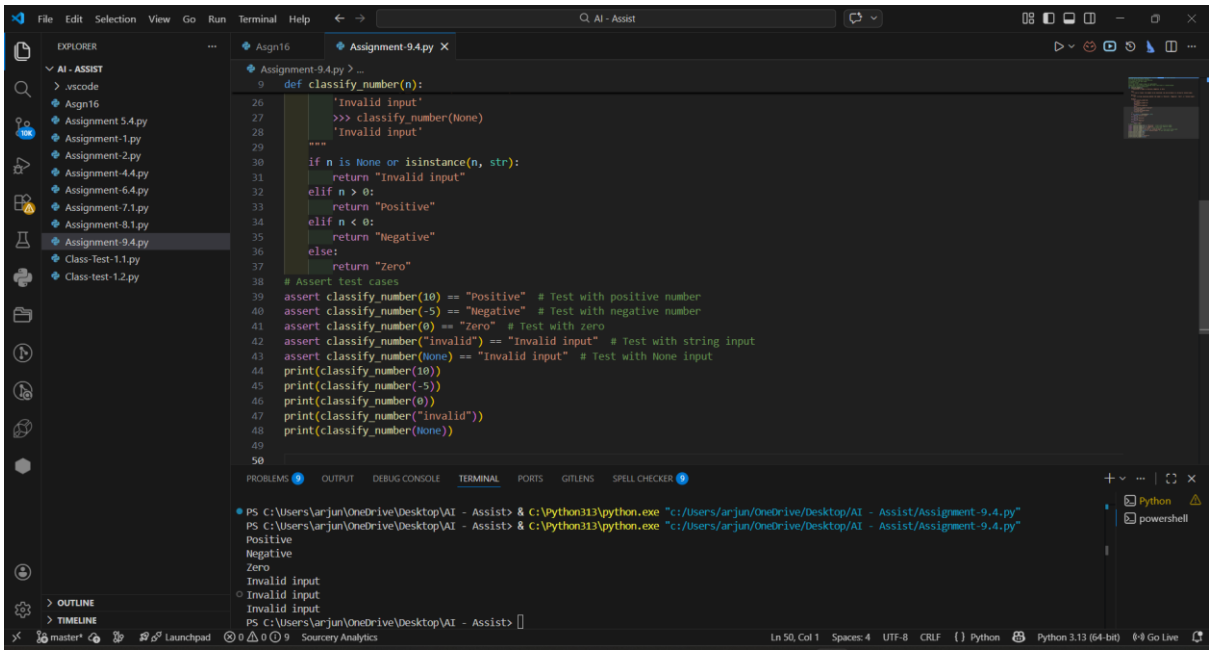
- Each docstring should include:
- A brief description of the function
- Parameters with data types
- Return values
- At least one example usage

## Prompt:

"Generate Google-style docstrings for the following Python function."

## Code:




# Observation:

- The AI generated a basic docstring.
- Description and parameters were included.
- Examples were sometimes missing or minimal.
- Formatting was correct but less detailed.

# Task 2: Enhancing Readability Through AI-Generated Inline Comments

You are provided with a Python script containing:

• Loops

• Conditional logic

• Algorithms (such as Fibonacci sequence, sorting, or searching)

Use AI assistance to:

• Automatically insert inline comments only for complex or non-obvious logic

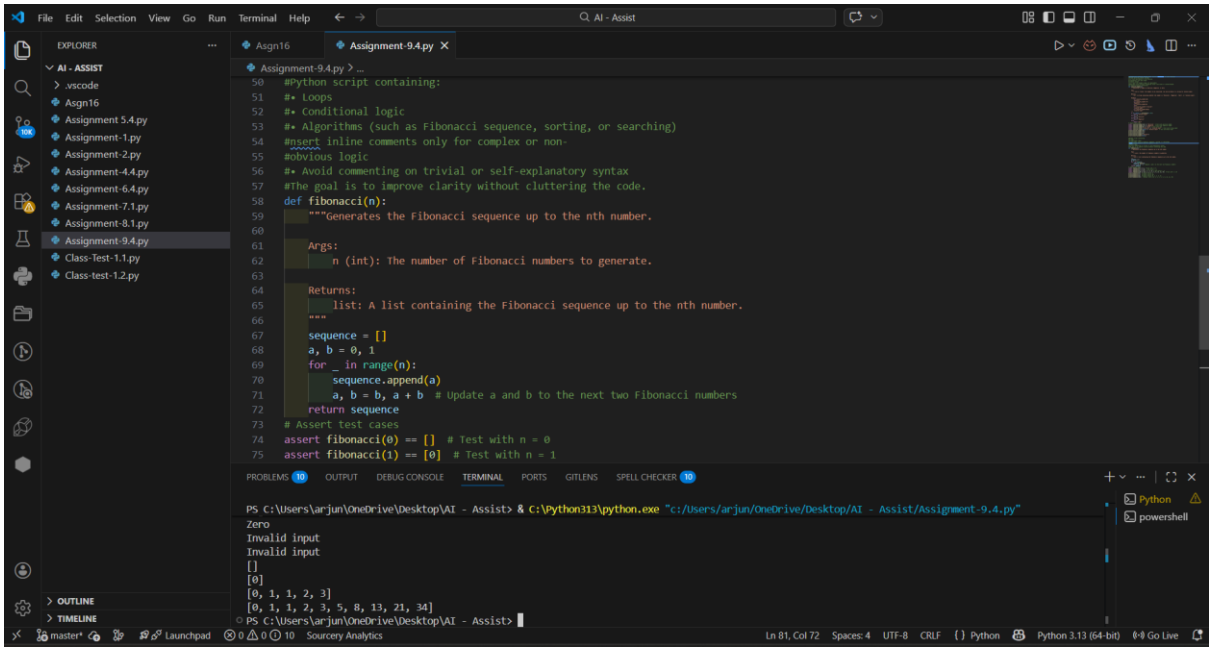
• Avoid commenting on trivial or self-explanatory syntax

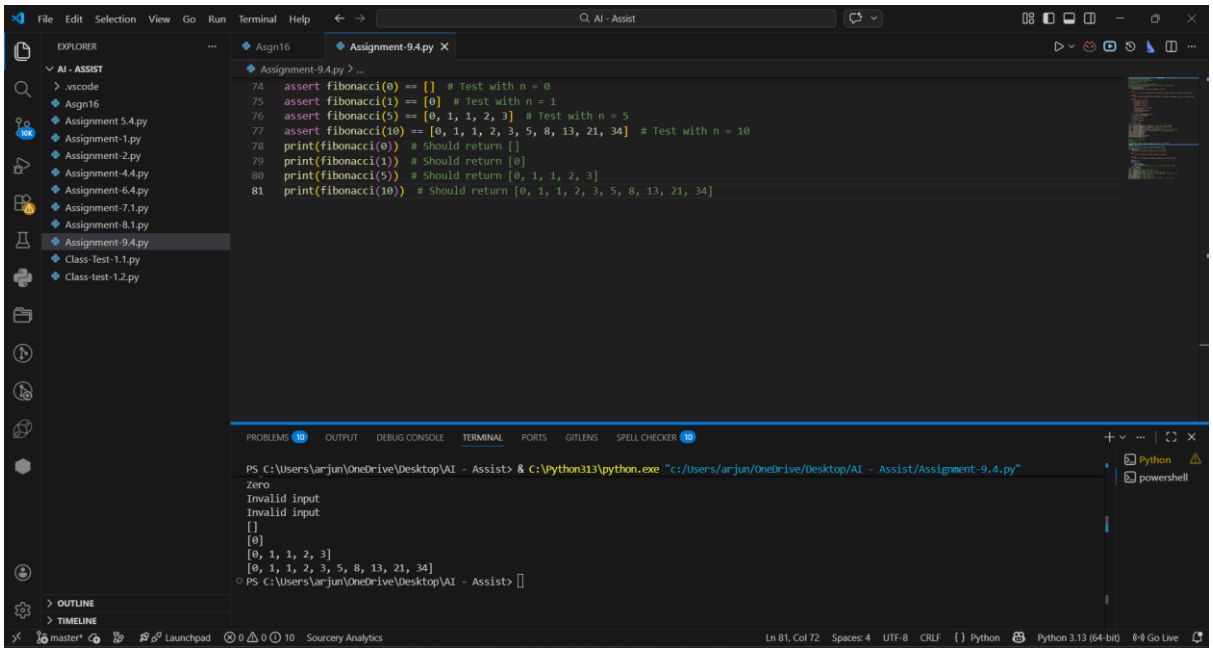
The goal is to improve clarity without cluttering the code.

## Prompt:

Insert meaningful inline comments in the following Python code. Comment only complex or non-obvious logic and avoid explaining basic Python syntax

## Code:




# Observation

- The AI added comments only where the logic was not immediately obvious.
- The tuple update in the Fibonacci algorithm was clearly explained.
- Trivial lines like variable initialization and loop syntax were not commented.
- Code readability improved without adding unnecessary clutter.

## Task 3: Generating Module-Level Documentation for a Python Package

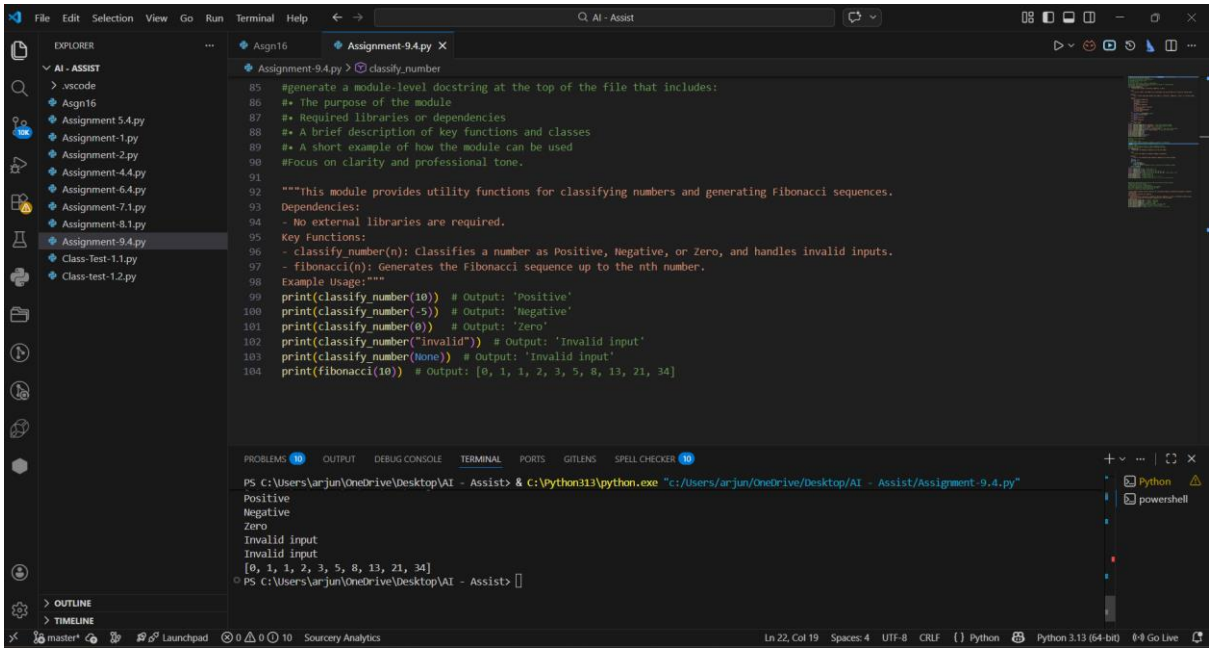
Provide a complete Python module to an AI tool and instruct it to automatically generate a module-level docstring at the top of the file that includes:

- The purpose of the module
- Required libraries or dependencies
- A brief description of key functions and classes
- A short example of how the module can be used

## Prompt:

Generate a professional module-level docstring for this Python module. Include the purpose, dependencies, key functions, and example usage

## Code:




# Observation

- The AI generated a structured multi-line docstring at the top of the file.
- The documentation clearly explained the module's purpose and functionality.
- Dependencies and key functions were summarized concisely.
- Example usage improved clarity for new users of the module.
- The tone was professional and suitable for real-world repositories.

## Task 4: Converting Developer Comments into Structured Docstrings

You are given a Python script where functions contain detailed inline comments explaining their logic.

Use AI to:

• Automatically convert these comments into structured Google-

style or NumPy-style docstrings

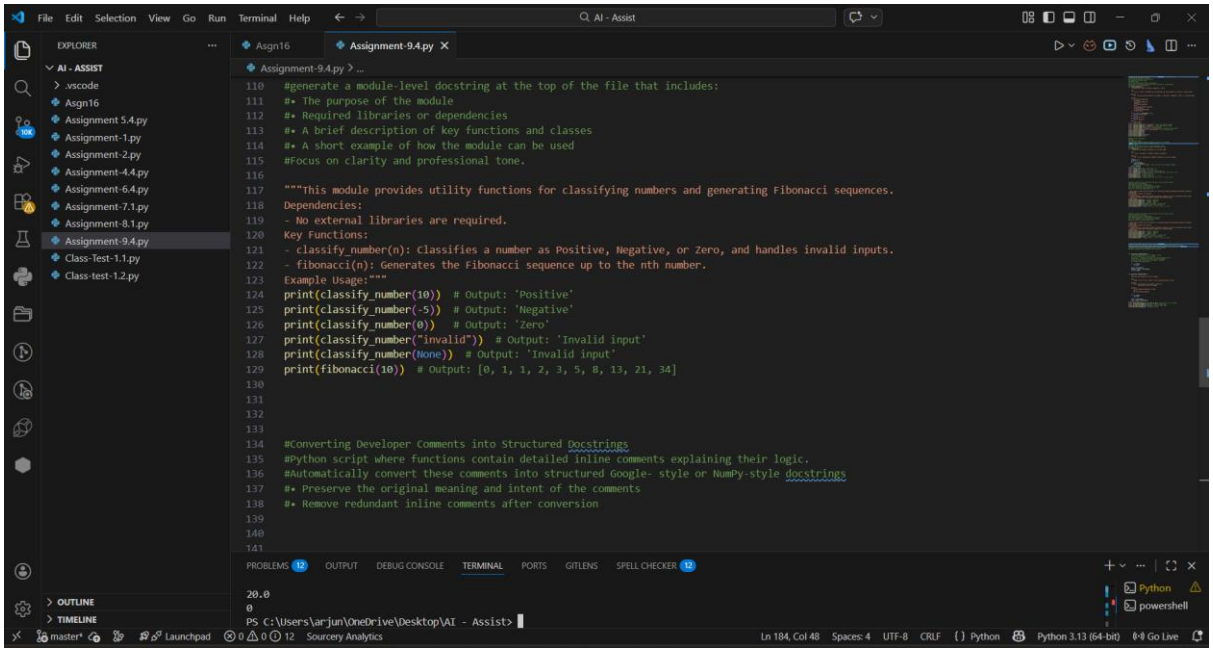
• Preserve the original meaning and intent of the comments

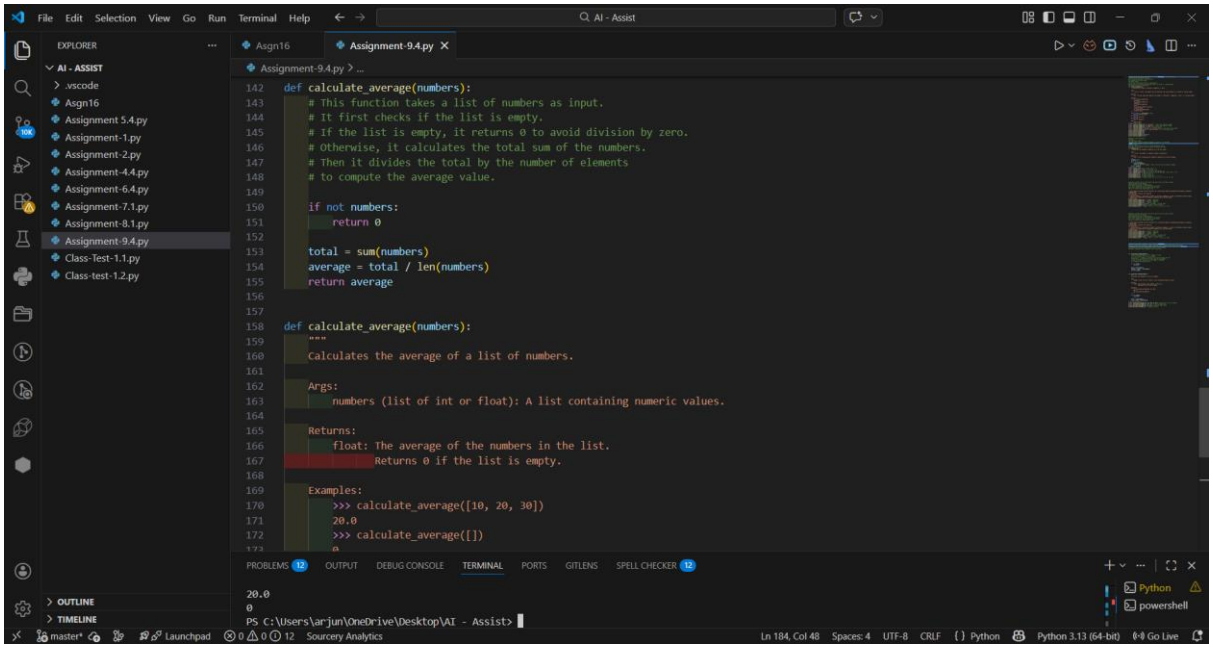
• Remove redundant inline comments after conversion

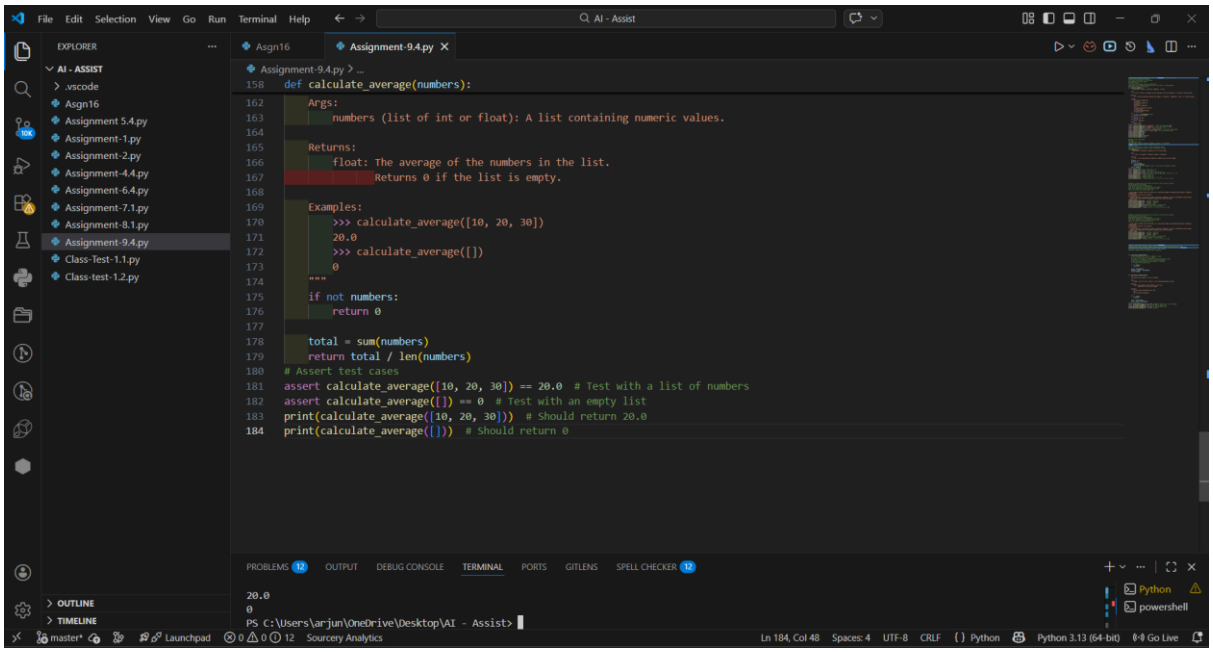
## Prompt:

Convert the following inline explanatory comments into a structured Google-style docstring. Preserve the original meaning and remove redundant inline comments

## Code:






# Observation:

- The AI successfully converted descriptive inline comments into a structured docstring.
- The original meaning of the logic was preserved.
- Redundant inline comments were removed.
- The function body became cleaner and more readable.

- Documentation format became consistent with professional standards.

## Task 5: Building a Mini Automatic Documentation Generator

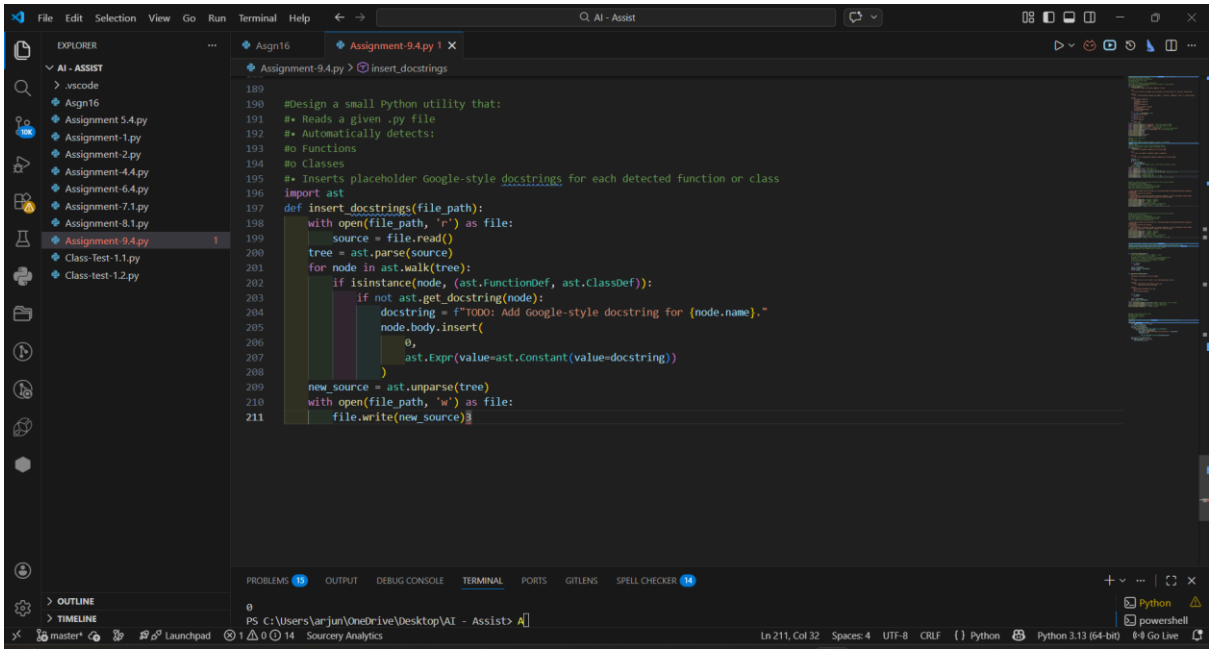
Design a small Python utility that:

- Reads a given .py file
- Automatically detects:
- Functions
- Classes
- Inserts placeholder Google-style docstrings for each detected function or class

## Prompt:

Help me build a Python utility that reads a .py file, detects functions and classes using AST, and inserts placeholder Google-style docstrings where missing

## Code:




# Observation:

- AI suggested using the ast module to parse and modify the syntax tree.
- Functions and classes were automatically detected.
- Placeholder docstrings were inserted only where missing.
- The tool successfully scaffolded documentation without affecting existing docstrings.
- Demonstrates automation of documentation setup.