# **AI ASSISTED CODING**

**ASSIGNMENT-9.2**

Name:S.Vrindha Reddy

Hall Number:2403A51255

Batch:11

**Task-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: Write a Python script with 4–5 simple functions (like add, subtract, multiply, divide, factorial). Add Google-style docstrings for every function. Each docstring should have: description, Args, Returns, raises (if needed), and Example. At the end, add a main block with print statements to show sample outputs.

**Code:**

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer program

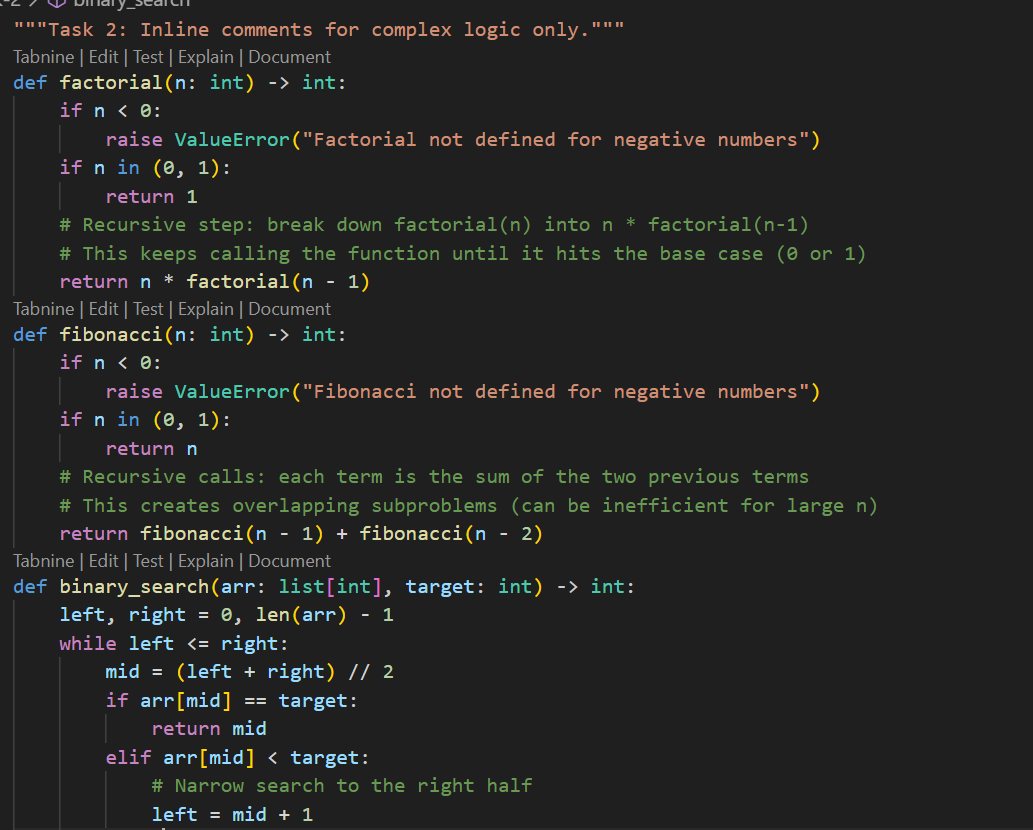
AI-generated content may be incorrect.

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

**Prompt:** Take this Python script and add inline comments only for the complex or non-obvious logic. Do not explain simple syntax like variable assignments or loops. Focus on tricky sections, such as conditional branches, recursion, or algorithm steps. Make comments concise and clear so they improve readability and maintainability.

**Code:**



A computer screen shot of a program code

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

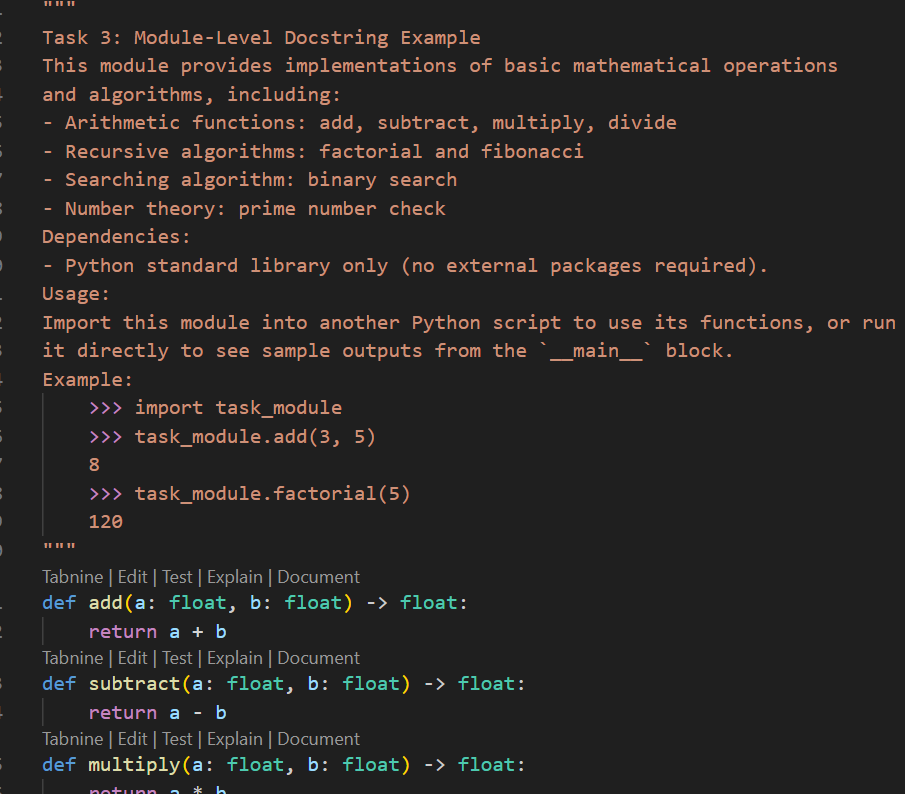
**Task-3:(Documentation – Module-Level Documentation)**

Task: Use AI to create a module-level docstring summarizing the  
purpose, dependencies, and main functions/classes of a Python  
file.

• Instructions:  
o Supply the entire Python file to AI.  
o Instruct AI to write a single multi-line docstring at the top  
of the file.  
o Ensure the docstring clearly describes functionality and  
usage without rewriting the entire code.  
**• Expected Output -3:**o A complete, clear, and concise module-level docstring at  
the beginning of the file.

**Prompt:** Here is my Python file. Write a module-level docstring at the top of the file. The docstring should summarize the purpose of the module, list any dependencies, and describe the main functions or classes. It should also include a short example of how to use the module. Do not rewrite the code, just add the docstring.

**Code:**



A computer screen shot of a program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

**Output:**

A computer screen with text and numbers

AI-generated content may be incorrect.

**Task-4: (Documentation – Convert Comments to Structured Docstrings)**

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

**Prompt:** Here is a Python script with inline comments. Transform those comments into Google-style docstrings for each function. Keep the meaning intact but remove the inline comments once they are moved into docstrings. Ensure each docstring includes a description, Args, Returns, and Raises (if relevant).

**Code:**

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

**Task-5: (Documentation – Review and Correct Docstrings)**

Task: Use AI to identify and correct inaccuracies in existing  
docstrings.  
• Instructions:  
o Provide Python code with outdated or incorrect docstrings.

o Instruct AI to rewrite each docstring to match the current  
code behaviour.  
o Ensure corrections follow Google-style formatting.  
**• Expected Output -5:**o Python file with updated, accurate, and standardized  
docstrings

**Prompt:** Here is my Python code with outdated or incorrect docstrings. Rewrite each docstring so it accurately matches the current code behaviour. Use Google-style formatting with clear descriptions, Args, Returns, and Raises where relevant.

**Code:**

A computer screen shot of text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

**Output:**

A computer screen shot of a black screen

AI-generated content may be incorrect.

**Task-6: (Documentation – Prompt Comparison Experiment)**

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

**Prompt:** For this Python function:

def factorial(n):

if n < 0:

raise ValueError ("Factorial not defined for negative numbers")

if n in (0, 1):

return 1

return n \* factorial (n - 1)

1. Add minimal inline comments (vague prompt).

2. Add a detailed Google-style docstring with parameters, return type, exceptions, and example (detailed prompt).

3. Compare both results in a table and summarize differences in clarity, completeness, and accuracy.

**Code:**

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.