**AI ASSISTED CODING**

**ASSIGNMENT-10.2**

**NAME:** G. Varshith Raju

**Roll No:** 2503A51L02

**BATCH:** 24BTCAICSB19

**Lab Objectives:**

* To understand the importance of code readability, maintainability, and quality.
* To explore how AI-assisted coding tools can review code and suggest improvements.
* To practice identifying code smells, redundant code, and poor naming conventions.
* To apply AI tools for refactoring and improving readability.
* To critically evaluate AI feedback and integrate it into real projects.

**Lab Outcomes (LOs):**

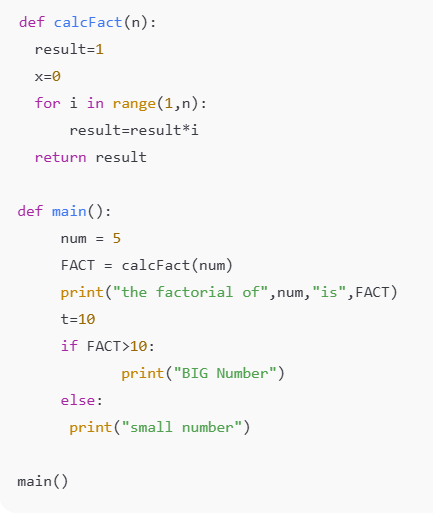
After completing this lab, students will be able to:

* Use AI-assisted tools (e.g., GitHub Copilot, Cursor AI) to review Python code.
* Identify and correct syntax issues, code smells, and inefficient logic.
* Improve readability by applying consistent formatting, naming, and comments.
* Refactor code with AI suggestions while ensuring functionality is preserved.
* Apply best practices for writing clean, maintainable, and professional code.

**Task Description#1 AI-Assisted Code Review (Basic Errors)**

•Write python program as shown below.

•Use an AI assistant to review and suggest corrections.



**Program:**

**A computer screen shot of a computer screen

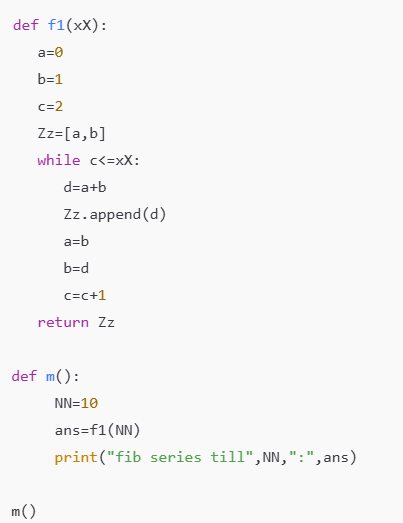
AI-generated content may be incorrect.**

**Output:**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

**Task Description#2 Automatic Inline Comments**

* Write the Python code for Fibonacci as shown below and execute.
* Ask AI to improve variable names, add comments, and apply PEP8 formatting (cleaned up).
* Students evaluate which suggestions improve readability most one.

**Program:**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**Output:**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**Task Description#3**

•Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).

•Incorporate manual docstring in code with NumPy Style

•Use AI assistance to generate a module-level docstring + individual function docstrings.

•Compare the AI-generated docstring with your manually written one.

**Manual Code:**

A screenshot of a computer

AI-generated content may be incorrect.

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Using AI assistance to generate a module-level docstring + individual function docstrings:**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Output:**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Docstring Comparison: Manual vs AI-Generated**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Manual Docstring** | **AI-Generated Docstring** |
| Length | Short and concise | More detailed and structured |
| Style | Informal and straightforward | Formal and follows NumPy style strictly |
| Purpose | Briefly states what the module does | Clearly states purpose, includes context (‘demonstrate the use of docstrings in N |
| Metadata | None | Includes author and date |
| Function Coverage | General description of all four functions | Mentions all functions with a clear description |
| Clarity | Clear, but less descriptive | Very clear, structured, and beginner-friendly |
| Extra Sections | None | Can include Examples, Raises, and Notes sections (more extensible |
| Professionalism | Basic, suitable for quick documentation | Professional, suitable for production-level documentation |