Function Finder – Python Logic Hub with Gradio Interface

Project Description Document

■ Project Overview

The goal of this project is to consolidate Python logic-building skills by developing 50 independent functions (each solving a different logical problem) and integrating them into a searchable, interactive interface using Gradio. Students will design a small-scale application where users can search, view, and execute functions dynamically through a web-based UI.

■ Learning Objectives

• Apply modular programming concepts by writing reusable Python functions.
• Organize large codebases efficiently.
• Create a user interface using Gradio.
• Implement a search mechanism for function retrieval.
• Handle user input dynamically in a web app.

■ Project Requirements

1. Create 50 independent Python functions with proper naming, parameters, and return statements.
2. Each function must include a docstring (description, parameters, return type).
3. Use Gradio to build an interface with:
• Search bar
• Function viewer
• Execution area
• Output display
4. Implement a search logic that checks if a function exists and handles "not found" cases gracefully.

■ Suggested Project Structure

project/
■ functions.py – Contains all 50 functions
■ app.py – Gradio UI integration
■ README.md – Documentation and usage

■■ Technical Stack

• Language: Python 3.x
• Framework: Gradio
• IDE: VS Code / Jupyter Notebook
• Version Control: Git (optional)
Installation: pip install gradio

■ Implementation Roadmap

1. Build all 50 Python functions.
2. Add docstrings and test each individually.
3. Implement the search mechanism.
4. Develop the Gradio interface.
5. Test the entire system with multiple functions.
6. Optionally, enhance UI and add extra features.

■ Hints & Tips

- Use consistent lowercase function names with underscores (e.g., check_prime).
- Store functions in a dictionary for easier retrieval.
- Use the inspect module to display source code.
- Build and test UI for 2–3 functions first, then scale up.
- Implement graceful error handling for invalid inputs.

■ Optional Enhancements

- Add dropdown for quick function selection.
- Display syntax-highlighted code.
- Include short descriptions and example inputs.
- Add random 'function of the day' feature.

■ Evaluation Criteria

- Functionality – 40%
- Code quality & readability – 20%
- UI/UX design – 20%
- Documentation – 10%
- Creativity / Bonus features – 10%