Table of contents

1 Python Tutorial Structure Plan

1.1 Tutorial Architecture

1.1.1 Target Audience

- Mixed levels (beginners to intermediate)
- Focus on Python 3.12+ features

1.1.2 Content Progression

1. **Environment Setup & Git** (Foundation)
    - Git/GitHub basics
    - uv (primary), poetry, miniforge (alternatives)
    - VS Code + ruff + pyright + error lens
    - Cross-platform installation (focus on Linux)
2. **Python Basics**
    - Syntax, data types, control flow
    - Functions, modules, packages
    - Code examples + execution output
    - Exercises and quizzes
3. **Object-Oriented Programming**
    - Classes, inheritance, polymorphism
    - Magic methods, decorators
    - Project: OOP-based application
4. **Advanced Topics**
    - Type hints and type checking
    - Async programming
    - Multiprocessing
    - Project: Performance comparison
5. **Applications** (Separate sections)

- Data Science (pandas, numpy, matplotlib)
- Automation (file handling, APIs, scheduling)
- General Programming (testing, debugging, packaging)
- Web Development
  - Frontend: Streamlit
  - Backend: FastAPI
- Final projects for each section

1.1.3 Technical Stack

- **Format**: Quarto (.qmd files) + Jupyter notebooks for examples
- **Environment**: uv primary, poetry/miniforge optional
- **Linting**: ruff (no black/flake8)
- **Type checking**: pyright for complex examples
- **Testing**: pytest (code must compile in Quarto)
- **IDE**: VS Code recommended
- **Hosting**: GitHub Pages with modern theme
- **Feedback**: GitHub Issues

1.1.4 Content Style

- Mix of formal and conversational
- Linear chapter progression
- Interactive code with execution output
- Exercises, quizzes, and section projects
- Git concepts woven throughout