Syntax Highlighting Editor

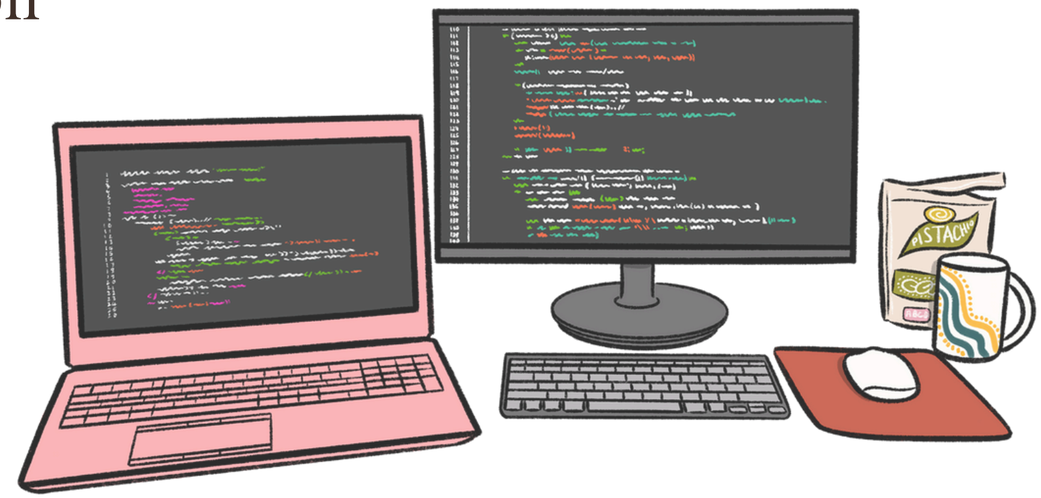
By:

Yateeka Goyal

Tool used: Python, Tkinter, Regular Expressions

Agenda

- Problem Statement
- Project Objectives
- System Design & Architecture
- Core Features & Implementation
- Live Demo
- Challenges & Solutions
- Future Work
- Q&A

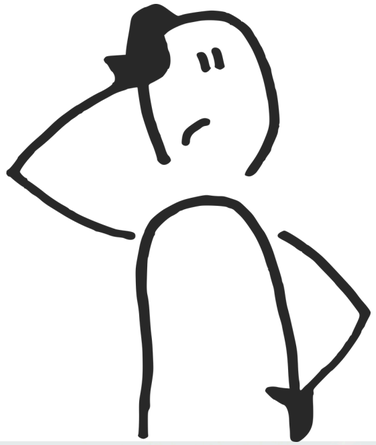


Problem Statement

- Text editors are essential tools for developers, but many beginner-level editors lack real-time syntax highlighting, dynamic line numbers, and customizable themes.
- Existing editors like VS Code are powerful but heavy, and not ideal for learning how syntax parsing works under the hood.

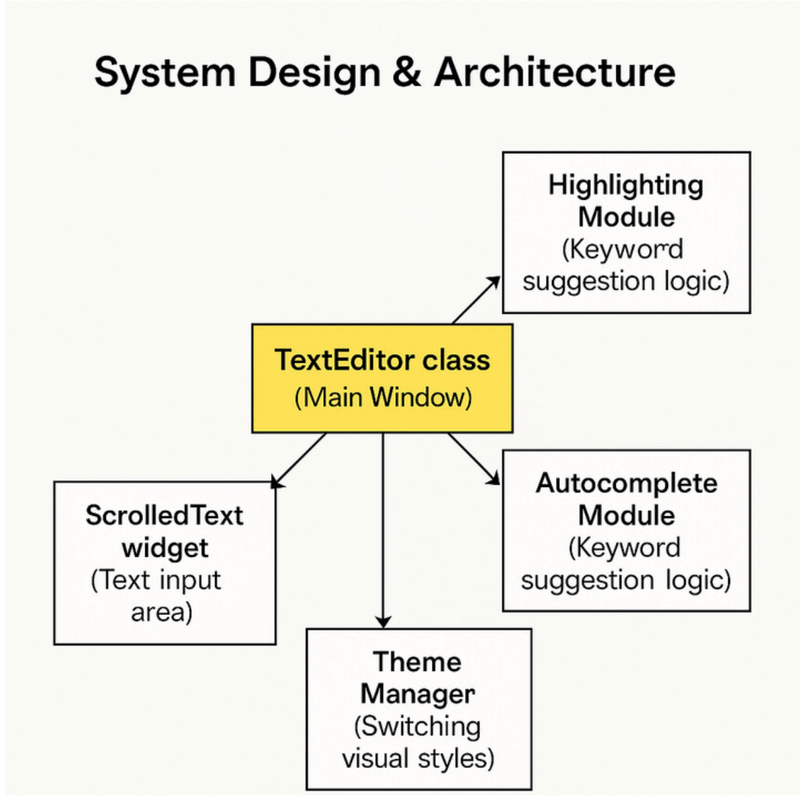
Project Objectives

- Build a GUI-based text editor using Tkinter
- Implement syntax highlighting for Python (keywords, strings, numbers, comments, etc.)
- Add built-in keyword suggestions (autocomplete)
- Provide theme switching for accessibility
- Support bracket matching, line numbers, and status tracking
- Enable basic file I/O operations



System Design & Architecture

- TextEditor class (Main Window)
  - ScrolledText widget
  - LineNumberCanvas
  - Highlighting Module
  - AutocompleteModule
  - Theme Manager



Core Features

- Syntax Highlighting:
  - Uses re.finditer() and text.tag_add() with regex patterns to detect keywords, strings, etc.
- Autocomplete:
  - Matches prefix of current word against Python keyword list.
- Line Numbers:
  - Auto-updates on scroll and key release.

Core Features

- Bracket Matching:
  - Inserts closing brackets for () [] {}.
- Themes:
  - Six built-in themes: Light, Dark, Monokai, Solarized (light & dark), Dracula, Nord.
- Status Bar:
  - Shows real-time line/column info.

Demo Time

Challenges & Fixes

- Challenge
  - Real-time performance drop
  - Indexing errors with regex matches
  - Theme refresh bugs
  - Scroll sync for line numbers
- Solution
  - Used after_idle() for optimized updates
  - Wrapped in try/except for stability
  - Forced config refresh on toggle
  - Bound scroll events to both text & line widget

Future Work

- Support for more languages (e.g., JS, HTML)
- Integrated terminal / Python execution
- Tabs for multiple file editing
- AI-powered code suggestions

Conclusion

- Demonstrated a custom Python-based text editor
- Implemented smart features for coding comfort
- Lightweight yet powerful — perfect for learners and tinkerers

Thank you!!