Coding Standards

Commenting and Documentation

- Every .py file should begin with a header comment or describing the purpose of the file, the author, and the date.
- Every function and class (except simple main() scripts) must include a docstring explaining the purpose, arguments, return values, and possible exceptions.

Example:

```
"""
Gibberish description
    Args:
        var1: does somehthing
        var2: does nothing

    Returns:
        Return nothing that does something
"""
```

- Use inline comments only when the code is not self-explanatory but it's short.
- Keep comments short and meaningful.
- Do not over-comment obvious code.

Naming Conventions

- Use meaningful names for variables, functions, and classes, mustn't be too long.
- Variables and functions = snake_case
- Classes = PascalCase
- Constants = ALL_CAPS
- Single-letter names should be reserved for short-lived counters or mathematical formulas.
- Avoid obscure abbreviations unless they are widely understood (e.g., gpa, url).

Code Formatting

- Indentation must be 4 spaces per level (No tabs).
- Limit code lines to 79 characters. Break long expressions across lines.

- Use two blank lines between top-level functions and classes.
- Use one blank line between methods inside a class.
- Functions longer than ~60 lines should be refactored into smaller functions.

Imports and Dependencies

- All import statements must appear at the top of each file, following this order:
    - Standard Library Imports (eg. os, sys, datetime)
    - Third-party Libraries (eg. PyQt6, Pillow, sqlite3)
    - Local Package Imports (eg. From catalog import Catalog Manager)

Source File Organization

- Each source file should follow this structure in this order:
    - File Header Comment
    - Imports
    - Global Constants and Configuration Variables
    - Class Definitions (if any)
    - Helper Functions (private or utility functions)
    - Main Functional Code (core logic or main class implementation)

Coding Practices

- Use parentheses to clarify complex expressions even if operator precedence makes them optional.
- Avoid multiple break or continue statements in a single loop.
- Prefer simple, clear loop conditions over overly complex logic.
- Use exceptions for error handling, not error codes.
- Always catch specific exceptions rather than using bare except:.

Git Rules

- Always git pull before doing any coding
- Always commit and push to keep the commit head updated unless the code doesn't run. In that case, do not commit.