Personal Library Management System — Project Guidelines

SkillShikshya

Prashant Karna—Python Django Class 04

February 23, 2025

1 Important Dates

- **Submission Deadline:** 03 March 2025 Monday

2 Project Overview

Design and implement a command-line application that allows users to manage a personal library. The project will:

- Utilize Python basics (variables, loops, conditionals, functions, file I/O)
- Apply object-oriented programming (OOP) principles (classes, objects, methods, encapsulation)
- Implement file persistence by saving data to a JSON file

3 Project Requirements

- **Python Concepts:** Use loops, conditionals, functions, and error handling.
- **OOP Concepts:**
  - Create a `Book` class to represent individual books.
  - Create a `Library` class to manage a collection of books.
- **File Persistence:** Save and load data using JSON, ensuring that the library's contents persist between sessions.
- **Modular Design:** Organize your code into separate files for clarity and maintainability.

4 Project Structure

It is recommended to structure your project as follows:

```
library_management/
 main.py          # Main program and user interface.
 book.py          # Contains the Book class.
 library.py       # Contains the Library class with file persistence.
```

5 Implementation Steps

5.1 Step 1: Planning & Design

- **Design Classes:**
  - **Book Class:** Decide on key attributes such as title, author, year, and ISBN.
  - **Library Class:** Plan methods to add, view, search, update, and delete books. Also, design methods for file I/O: `save_to_file` and `load_from_file`.
- **Define File Format:** Use JSON to store book data. Determine how each `Book` object will be converted to and from a dictionary.

5.2 Step 2: Implement the `Book` Class

- Create `book.py` and define the `Book` class.
- **Attributes:** Include title, author, year, and ISBN.
- **Methods:**
  - Implement the constructor (`__init__`) to initialize attributes.
  - Override the `__str__` method to return a human-readable string of the book's details.

5.3 Step 3: Implement the `Library` Class

- Create `library.py` and define the `Library` class.
- **Data Management:** Use a list to store `Book` objects.
- Implement methods:
  - `add_book`: Add a new book to the collection.
  - `view_books`: Display all books.
  - `search_books`: Search for books by title or author.
  - `update_book`: Update book details.
  - `delete_book`: Remove a book.
- **File I/O for Persistence:**
  - `save_to_file`: Convert each `Book` object to a dictionary and write the list to a JSON file.
  - `load_from_file`: Read the JSON file and reconstruct `Book` objects.

5.4 Step 4: Build the Main Application

- Create `main.py` for the command-line interface.
- **User Interface:**
  - Present a menu with options (Add Book, View Books, Search Books, Update Book, Delete Book, Exit).
  - Ensure that after any modification (add, update, delete), the library data is saved automatically.
- **Input Handling:** Prompt for user input and handle invalid entries using error handling techniques.

5.5 Step 5: Testing & Debugging

- **Functionality Testing:**
  - Test each function (adding, viewing, searching, updating, deleting).
  - Verify that file persistence works correctly by closing and reopening the program.
- **Error Handling:** Test for common errors, such as invalid inputs or file read/write errors.
- **Code Review:** Add comments throughout your code to explain the functionality and logic. Prepare a README file explaining how to set up and run your project.

6 Documentation & Deliverables

- **Source Code:** Submit your project folder containing all Python files (`book.py`, `library.py`, `main.py`).
- **README File:** Include a README with setup instructions, project description, and usage guidelines.
- **Optional PDF Report:** Prepare a PDF summarizing your design decisions, implementation process, and any challenges encountered.

7 Additional Enhancements (Optional and only try if you want to explore yourself)

- **Advanced Search:** Add options to search by additional criteria.
- **Data Validation:** Enhance error handling to check for duplicate ISBNs or invalid data entries.
- **User Interface Improvements:** Consider adding a graphical user interface (GUI) if you wish to extend the project further.

8 Helpful Resources

- Python Official Documentation
- JSON Module Documentation
- FPDF for Python (if generating PDF reports)