Explanation of Library Management System Code

Generated by ChatGPT (GPT-5)

November 3, 2025

Overview

This project implements a simple **Library Management System** in Python. It is composed of two main scripts:

1. `utils.py` – defines a helper function to generate unique IDs for books.
2. `main.py` – provides an interactive console-based menu that lets users manage the library's books.

The system can:

- Add new books to a collection.
- View all available books.
- Search for specific books.
- Delete unwanted books.
- Save data for future sessions.

1. utils.py – Utility Module

```python
import random
import string

def generate_id():
    return ''.join(random.choices(string.ascii_uppercase + string.
        digits, k=5))
```

Listing 1: utils.py

Explanation

The `generate_id()` function:

- Uses Python's `random` and `string` modules.
- Combines uppercase letters (`A-Z`) and digits (`0-9`).
- Randomly picks 5 characters to create an alphanumeric ID.

Purpose: Each book in the system can have a unique 5-character identifier such as "A7F2K". This helps in easily referencing and managing books without confusion.

2. Main Program – Library Management System

```python
from book_operations import add_book, view_books
from manage_books import search_book, delete_book
from storage import save_data, load_data

def main():
    books = load_data()

    while True:
        print("\n=== Library Management System ===")
        print("1. Add Book")
        print("2. View Books")
        print("3. Search Book")
        print("4. Delete Book")
        print("5. Save & Exit")

        choice = input("Enter your choice: ")

        if choice == ’1’:
            add_book(books)
        elif choice == ’2’:
            view_books(books)
        elif choice == ’3’:
            search_book(books)
        elif choice == ’4’:
            delete_book(books)
        elif choice == ’5’:
            save_data(books)
            print("    Data saved! Exiting...")
            break
        else:
            print("    Invalid choice, try again!")

if __name__ == "__main__":
    main()
```

Listing 2: main.py (core program)

Explanation

The main program controls the overall workflow:

- **load_data()**: Loads previously saved book data from a file.
- Displays a menu with 5 options for user interaction.
- Depending on user input:
  - Option 1 → Calls `add_book()` to insert new book entries.
  - Option 2 → Calls `view_books()` to display all stored books.
  - Option 3 → Calls `search_book()` to find specific titles or IDs.
  - Option 4 → Calls `delete_book()` to remove unwanted books.
  - Option 5 → Saves data using `save_data()` and exits safely.

Program Flow Summary

1. The script starts by loading book data.
2. A continuous loop presents menu options.
3. The user makes a choice.
4. The corresponding function executes.
5. When "Save & Exit" is chosen, all data is written to storage and the program ends.

3. Interaction Between Files

- `utils.py` provides the ID generation utility that other modules (like `book_operations.py`) can use when adding new books.
- The main file (`main.py`) integrates all operations: book handling, storage, and user interaction.

Conclusion

Together, these scripts form the backbone of a modular Library Management System. The structure allows easy maintenance, as new features (like editing books or sorting by author) can be added without modifying the core logic.