Sistema de Gestión de Biblioteca

Este proyecto es una aplicación de consola y web desarrollada en **Python** que simula el funcionamiento de una biblioteca. Permite:

- Registrar y administrar libros físicos y digitales.
- Registrar miembros de la biblioteca.
- Prestar y devolver libros.
- Guardar y cargar datos desde archivos JSON.
- Ver información detallada de uso de memoria simulada.

Estructura de Archivos

Archivo	Descripción
`biblioteca.py`	Contiene las clases y lógica principal de la biblioteca (libros, miembros, operaciones).
`biblioteca_web.py`	Interfaz web construida con Flask para interactuar vía HTTP con la biblioteca.
`memory_management.py`	Módulo que simula la asignación y liberación de memoria dinámica.

Simulación de Memoria (`memory_management.py`)

Este módulo implementa una clase llamada `MemoryManagement` con dos contadores:

- `heap_allocations`: Suma las asignaciones simuladas (cuando se crean objetos como libros o miembros).
- `heap_deallocations`: Suma las liberaciones simuladas (cuando se eliminan objetos).

Ejemplo

```
memory_management.increment_heap_allocations(1)
memory_management.display_memory_usage()
```

Este sistema es puramente educativo y no afecta el rendimiento real de Python, ya que este lenguaje gestiona la memoria automáticamente.

Lógica de Biblioteca (`biblioteca.py`)

Clases Principales

`Genre`

Enumeración de géneros literarios disponibles.

```
Genre.FICTION     # Ficcion
Genre.HISTORY     # Historia
Genre.OTHER       # Otro
```

`Book`

Representa un libro físico.

Atributos:

- `id`, `title`, `author`, `publication_year`, `genre`, `quantity`

```
book = Book(1, "1984", "George Orwell", 1949, Genre.FICTION, 3)
```

`DigitalBook` (subclase de `Book`)

Extiende la clase `Book` añadiendo un atributo `file_format`.

```
digital = DigitalBook(2, "Python 101", "M. Author", 2020, Genre.SCIENCE, 5, "PDF")
```

`Member`

Representa a un usuario registrado en la biblioteca.

Atributos:

- `id`, `name`, `issued_books` (lista de IDs de libros prestados)

```
member = Member(1, "Juan Perez")
```

`Library`

Contiene la lógica del sistema:

- Lista de libros (`self.books`)
- Lista de miembros (`self.members`)

Funciones principales:

Método	Acción
`add_book(book)`	Agrega un libro
`add_member(member)`	Agrega un nuevo miembro

Método	Acción
`issue_book(book_id, member_id)`	Presta un libro
`return_book(book_id, member_id)`	Devuelve un libro
`save_library_to_file("library.json")`	Guarda libros en JSON
`load_library_from_file("library.json")`	Carga libros desde JSON
`display_books()`	Muestra libros
`display_members()`	Muestra miembros

💻 Interfaz Web (`biblioteca_web.py`)

Framework usado: **Flask**

El servidor web permite interacción mediante una API RESTful. Al iniciarse, carga los datos existentes desde archivos JSON (`library.json` y `members.json`).

Endpoints Principales

Ruta	Método	Acción
`/`	GET	Página principal (HTML)
`/books`	GET	Lista todos los libros
`/books`	POST	Crea un libro nuevo
`/members`	GET	Lista todos los miembros
`/members`	POST	Crea un miembro nuevo
`/issue_book`	POST	Presta un libro a un miembro
`/return_book`	POST	Devuelve un libro
`/save`	POST	Guarda datos en archivo
`/load`	POST	Carga datos desde archivo
`/genres`	GET	Devuelve lista de géneros disponibles

Ejemplo de POST `/books`

```
{
  "id": 10,
  "title": "Flask for Beginners",
  "author": "Jane Doe",
  "publication_year": 2022,
  "genre": "Ciencia",
  "quantity": 4,
```

```
  "is_digital": true,
  "file_format": "EPUB"
}
```

Interfaz de Consola (`main()` en `biblioteca.py`)

El programa también puede ejecutarse desde consola. Al hacerlo, muestra un menú interactivo:

```
Menu de sistema de manejo de biblioteca

1. Agregar un libro
2. Mostrar libros disponibles
3. Agregar un miembro
4. Prestar libro
5. Devolver libro
6. Mostrar miembros disponibles
7. Buscar miembro
8. Guardar y salir
```

Archivos de Datos

- `library.json`: Contiene los libros registrados.
- `members.json`: Contiene los miembros y sus libros prestados.

Estos archivos se generan automáticamente y pueden editarse manualmente si es necesario.

Requisitos

- Python 3.7+
- Flask (`pip install flask`)

Ejecución

Para usar como aplicación de consola:

```
python biblioteca.py
```

Para usar como servidor web:

```
python biblioteca_web.py
```

Luego accede en el navegador a:
`http://localhost:5000/`

📌 Notas Adicionales

- La gestión de memoria no es real (Python maneja su propio recolector de basura).
- Los datos se almacenan en archivos JSON en el mismo directorio.
- Ideal para proyectos escolares o para aprender programación orientada a objetos + desarrollo web en Python.
