Informe Explicativo del Código – Sistema de Biblioteca

Presentado por:

Jhon Erick Puentes Bello

Presentado a:

Jesús Ariel González Bonilla

Ingeniería mecatrónica

Semestre III

Corporación universitaria del Huila

06 septiembre 2025

**Introducción**

Este proyecto consiste en un **sistema de biblioteca digital en consola (CLI)** desarrollado en Python. El sistema permite registrar usuarios, registrar materiales (libros o revistas), realizar préstamos y devoluciones, y generar reportes de préstamos activos.

El objetivo principal es **aprender los fundamentos de la programación orientada a objetos (POO)**, el manejo de estructuras de datos como listas y diccionarios, y la interacción con el usuario mediante menús en consola.

Este informe explicará el código paso a paso, mostrando fragmentos relevantes y explicando de manera clara y sencilla su funcionamiento.

Este proyecto tiene como objetivo enseñar, de manera práctica y sencilla, cómo aplicar los fundamentos de la **programación orientada a objetos (POO)** en Python para resolver un problema real: la gestión de una biblioteca digital.

El programa permitirá:

* Crear usuarios.
* Registrar materiales (libros y revistas).
* Prestar y devolver materiales.
* Generar reportes de préstamos.

El enfoque es **pedagógico**: cada parte del código está diseñada para que un estudiante novato entienda la lógica paso a paso.

**Paso a paso.**

**2. Paso 1 – Definir las bases del programa**

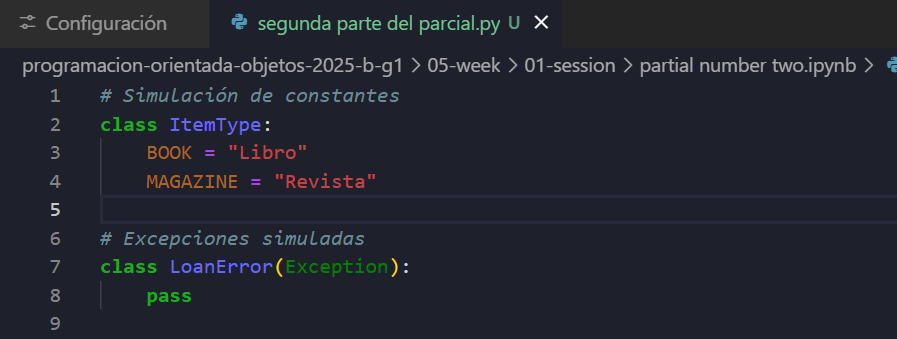
Antes de escribir el código, debemos pensar qué elementos tendrá una biblioteca:

1. **Usuarios**: personas que piden prestados libros.
2. **Materiales**: libros o revistas que pueden prestarse.
3. **Préstamos**: acciones de dar un material a un usuario.
4. **Reportes**: información sobre lo que está prestado o vencido.

Esto nos lleva a pensar en clases que organicen estos elementos.

**3. Paso 2 – Constantes y Excepciones**

Código:



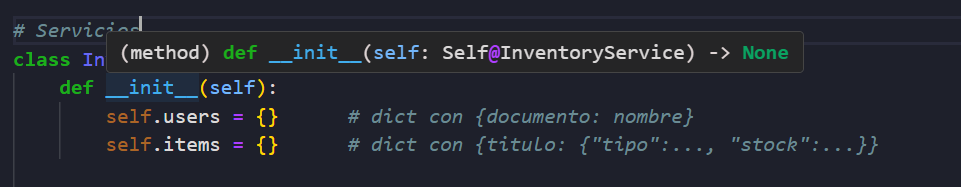
**EXPLICACION**

* ItemType: es una clase que **simula constantes**. En vez de escribir “Libro” o “Revista” en todas partes, usamos ItemType.BOOK o ItemType.MAGAZINE.
* LoanError: define un **tipo de error propio** para cuando algo falle en los préstamos (por ejemplo: usuario no existe).

Esto enseña a organizar mejor el código y dar mensajes más claros.

**4. Paso 3 – Clase InventoryService (Inventario)**

Código

**Explicación:**

* self.users: es un **diccionario** que guarda a los usuarios.
  + Clave = documento
  + Valor = nombre
  + Ejemplo: {"1010": "Ana"}
* self.items: es otro **diccionario** que guarda materiales.
  + Clave = título
  + Valor = tipo y stock
  + Ejemplo: {"Python Básico": {"tipo": "Libro", "stock": 2}}

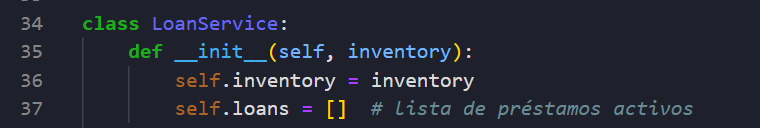
**Métodos:**

* create\_user: registra un usuario nuevo (y no permite documentos repetidos).
* list\_users: devuelve todos los usuarios registrados.
* create\_item: agrega un libro/revista con stock inicial.
* list\_items: lista los materiales disponibles.

Esta clase funciona como la **base de datos interna** del programa.

**5. Paso 4 – Clase LoanService (Préstamos)**

Código



**Explicación:**

* inventory: conecta esta clase con los usuarios y materiales ya creados.
* self.loans: es una **lista** que guarda préstamos activos.

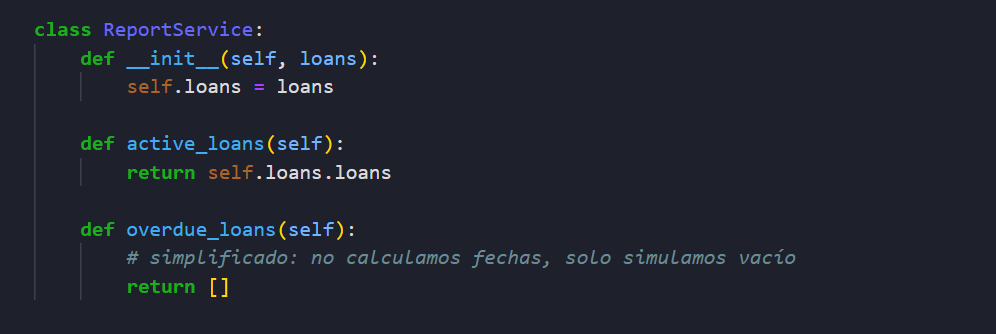
**Métodos:**

1. loan\_item:
   * Verifica que el usuario exista.
   * Verifica que el material exista.
   * Verifica que haya stock.
   * Si todo está bien:
     + Resta 1 al stock.
     + Registra el préstamo en la lista.
2. return\_item:
   * Busca el préstamo en la lista.
   * Si lo encuentra: lo elimina y devuelve el stock.
   * Retorna una “penalización ficticia” (siempre 0 en este ejemplo).

Aquí el estudiante aprende a **validar condiciones** antes de ejecutar una acción.

**6. Paso 5 – Clase ReportService (Reportes)**

Código:

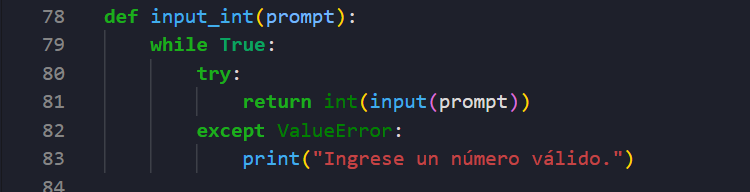


return [] # simplificado

* active\_loans: devuelve todos los préstamos activos.
* overdue\_loans: simula los vencidos (en un sistema real habría fechas).

Con esta clase mostramos cómo **separar responsabilidades**: el reporte no gestiona préstamos, solo los muestra.

**7. Paso 6 – Funciones Auxiliares**

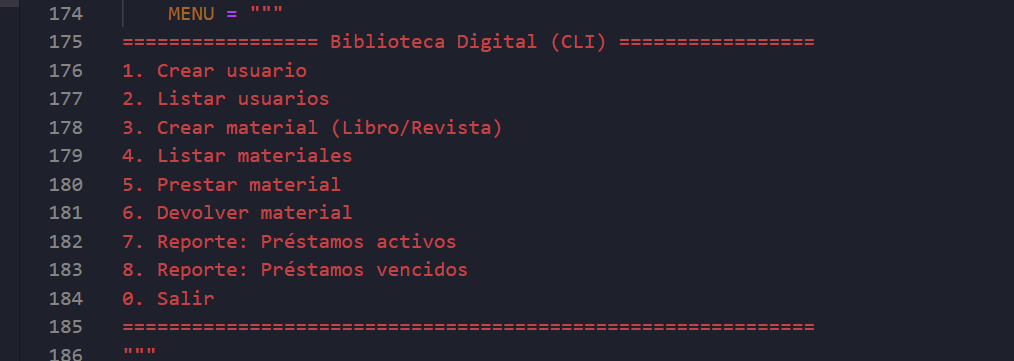
Código:

* input\_int: garantiza que el usuario ingrese un número válido.
* prompt: pide un texto y lo limpia de espacios.

Esto enseña **validación de entradas de usuario**, algo esencial en programación.

**8. Paso 7 – Menú de Opciones**

Código (fragmento):



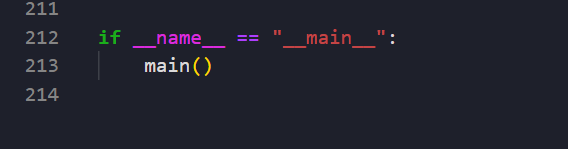
Este menú muestra al usuario todas las acciones posibles.

Cada número está conectado con una **función específica**, por ejemplo:

* Opción 1 → create\_user
* Opción 5 → loan\_item

**9. Paso 8 – Programa Principal**

Código:

* Garantiza que el programa se ejecute desde este archivo.
* Dentro de main() se crean los objetos (inventory, loans, reports) y se controla el flujo con el menú.

Aquí el estudiante aprende a estructurar un programa completo.

**10. Ejemplo de Ejecución**

1. Crear usuario:
   * Documento: 2025, Nombre: Juan.
2. Crear material:
   * Título: "Mecatrónica Avanzada", tipo Libro, stock: 3.
3. Prestar material:
   * Usuario Juan pide “Mecatrónica Avanzada”.
   * Stock pasa de 3 → 2.
4. Reportar activos:
   * “Juan tiene ‘Mecatrónica Avanzada’”.
5. Devolver material:
   * Stock vuelve a 3.

Esto permite a los estudiantes **probar y comprobar** el funcionamiento.

**11. Conclusiones para los Estudiantes**

* Se aplicaron los fundamentos de **POO en Python** (clases, objetos y métodos).
* Se usaron **diccionarios y listas** como estructuras de datos.
* Se practicó el manejo de **errores y validaciones**.
* Se construyó un sistema modular y entendible paso a paso.
* Es una base sólida para proyectos más grandes (ejemplo: conectar a una base de datos real o usar interfaces gráficas).

ANEXO LINK DEL VIDEO

https://drive.google.com/file/d/1xrA0AeVuD4JgumfCDfATXIWHqIV8inkk/view?usp=drive\_link