Python project: Library Management

Objective: Create a Python console application that simulates library management using advanced concepts: **inheritance**, **abstraction**, **enums**, and **exception handling**.

Tasks:

1. Create an abstract class `Document` that represents the base element of your library. It must contain:
   - a class attribute `nb_document` (int)
   - the property `title` (String)
   - the property `year_of publication` (Int)
   - an abstract method `show_informations()` that will be overridden to display specific details
   - a class method `show_nbr_documents()` that displays the number of instantiated documents
2. Create an enum class `Genre` used to classify books by category.
   - Example: NOVEL, SCIENCE FICTION, FANTASY
3. From `Document`, create two classes:
   - A `Book` (Book) class containing:
     - a property `author`: String
     - a property `nbr_pages`: Int
     - a property `type`: Type (enum)
     - a **secondary constructor** (static method) that only takes `title`, `author`, and `type` (`nbr_pages` defaults to 100 and `year_of publication` defaults to 0)
     - a static method `Pages_counter()` that takes a list of books and returns the total number of pages
   - A `Magazine` class containing:
     - a property `number`: Int
4. Create an abstract class `Borrowable` inherited only by `Book`, containing:
   - a property `is_borrowed`: Boolean
   - an abstract method `borrow()`
   - an abstract method `give_back()`
5. Create an abstract class `Consultable` implemented by both `Book` and `Magazine`, containing:
   - an abstract method `consult()` that displays: **"You are consulting this document."**
6. Create two custom exceptions:
   - `DocumentAlreadyBorrowedException`
   - `DocumentNotBorrowedException`
7. Implement the methods `borrow()` and `give_back()` in `Book`, checking the boolean `is_borrowed`:
   - If the book can be borrowed or returned, display a success message
   - Otherwise, raise the corresponding exception

8. In the main function:
    - Create several books (of different genres) and magazines
    - Store them in a list
    - Display the full list of documents using `show_informations()`
    - Simulate several actions:
        - consulting a book or magazine
        - borrowing a book
        - attempting to borrow a book that is already borrowed
        - attempting to return a book that was not borrowed
        - returning a borrowed book

Bonus:

- Each class should be in a separate file and imported into the main script
- Create a user interface allowing the user to perform all previous actions until they exit the program:

```
====== LIBRARY MANAGEMENT ======
1. Consult
2. Borrow
3. Give back
0. Exit
```

- You may also add options: `Add Document` and `Remove Document`

Example:

```
--- List of documents ---
Book: "The Witcher", Andrzej Sapkowski, 1993, 320 pages, Genre: FANTASY
Magazine: "Canard PC", No. 420, 2023
Book: "Harry Potter", J.K. Rowling, 0000, 100 pages, Genre: ROMAN

Consulting the magazine "Canard PC"...
You are consulting this document.

Attempting to borrow the book "The Witcher"...
Borrow successful!

Attempting to borrow the book "The Witcher" again...
ERROR: This book is already borrowed!

Attempting to return the book "Harry Potter" without borrowing it...
ERROR: This document has not been borrowed!

Returning the book "The Witcher"...
The book is now available.
```