## Introduction

This document explains why I chose to use dictionaries, lists, and tuples in my Library Management System. Each data structure was chosen for specific reasons to make the system work efficiently.

## Why I Used Dictionary for Books

I used a dictionary to store all books in the library.

The books dictionary uses ISBN as the key, and each book contains information like title, author, genre, total copies, and available copies.

1. **Fast Search** - When someone wants to borrow a book, I can quickly find it using the ISBN. Dictionary lookup is very fast.
2. **Easy to Use** - I can access any book directly using the ISBN as a key.
3. **No Duplicates** - Dictionary automatically prevents duplicate ISBNs because keys must be unique.
4. **Real Libraries Use ISBN** - In real libraries, every book has a unique ISBN number, so using it as a key makes sense.
5. **Easy to Update** - I can easily change book information like reducing available copies when someone borrows a book.

**Example:** When Alice wants to borrow a book with ISBN "111", the system finds it instantly instead of checking every book one by one.

## Why I Used List for Members

I used a list to store all library members.

The members list stores each member as a dictionary containing member ID, name, email, address, and a list of borrowed books.

1. **Simple to Add Members** - I can easily add new members using the append function.
2. **Easy to Loop Through** - When I need to find a member, I can check each one by looping through the list.
3. **Keeps Order** - Members are stored in the order they joined, which is nice for record keeping.
4. **Flexible Size** - The list grows automatically when I add more members.
5. **Can Store Complex Data** - Each member has multiple details like name, email, and borrowed books, and list handles this well.

**Example:** When Bob wants to borrow a book, I search through the members list to find his record, then add the book to his borrowed books list.

## Why I Used Tuple for Valid Genres

I used a tuple to store the allowed book genres.

The VALID_GENRES tuple contains eight fixed genres: Fiction, Non-Fiction, Sci-Fi, Mystery, Romance, History, Biography, and Fantasy.

1. **Cannot Be Changed** - Tuples are immutable. This means the genres cannot be accidentally changed while the program is running. This is important because we want fixed categories.
2. **Easy to Check** - I can easily check if a genre is valid by checking if it exists in the tuple.
3. **Shows Intent** - Using tuple tells other programmers "these values should not change".
4. **Saves Memory** - Tuples use less memory than lists.
5. **Prevents Mistakes** - If I accidentally try to change a genre, Python will give an error. This protects my data.

**Example:** When adding a new book, the system checks if the genre is in the VALID_GENRES tuple. If someone tries to add genre "Comedy", it will be rejected because it's not in the tuple.

## How They Work Together

All three data structures work together in the system:

1. **Adding a Book:**
    a. Check genre against VALID_GENRES tuple
    b. Add book to books dictionary
2. **Borrowing a Book:**
    a. Find member in members list
    b. Find book in books dictionary
    c. Add ISBN to member's borrowed books list
    d. Reduce available copies in books dictionary
3. **Searching:**
    a. Loop through books dictionary to find matches
    b. Loop through members list to find a specific member

## Alternatives I Considered

### Why Not Use List for Books?

- Lists are slower for searching by ISBN
- Would need to check every book to find the right ISBN
- Dictionary is much faster

### Why Not Use Dictionary for Members?

- Could work, but member lookup is less frequent
- List is simpler for this case
- We often need to show all members, and list is easier to loop through

### Why Not Use List for Genres?

- List can be changed accidentally
- Tuple is safer and shows these are fixed values

## Conclusion

I chose:

- **Dictionary for books** because ISBN is a unique key and we need fast lookup
- **List for members** because it's simple and we need to store multiple members
- **Tuple for genres** because genres are fixed and should not change

These choices make the system efficient, safe, and easy to understand.