Module 3 Programming Question: Library Management System

You are tasked with designing a **Library Management System** for a small library using Object-Oriented Programming (OOP) concepts in Python. The library allows users to borrow and return books. The system also keeps track of the available books and ensures that a book cannot be borrowed if it is not available.

Problem Statement:

Write a Python program to implement the following functionalities using OOP principles:

1. **Define a `Book` class**:
    - Attributes:
        - `title` : Title of the book
        - `author` : Author of the book
        - `copies` : Number of available copies
    - Methods:
        - `__str__` : To print the details of the book in a readable format.
        - `is_available` : Returns `True` if at least one copy of the book is available, otherwise `False` .
2. **Define a `Library` class**:
    - Attributes:
        - `books` : A list of `Book` objects available in the library.
    - Methods:
        - `add_book(book)` : Adds a `Book` object to the library's collection.
        - `borrow_book(title)` : Allows a user to borrow a book by its title. Reduces the number of available copies by 1 if the book is available. Prints an appropriate message if the book is not available or does not exist in the library.
        - `return_book(title)` : Allows a user to return a book by its title. Increases the number of available copies by 1 if the book exists in the library. Prints an appropriate message if the book does not exist in the library.

- `list_books()` : Prints all books in the library with their details.

3. **Simulate the System**:
   - Create a library and add at least three books to it.
   - Borrow a book, return a book, and list all books to demonstrate the functionality.

Evaluation Criteria:

1. Use of OOP principles like encapsulation, abstraction, and modularity.
2. Proper implementation of methods and attributes.
3. Code readability and use of appropriate naming conventions.
4. Use of Python-specific features like `__str__` .

Example Interaction:

```
# Create a Library
library = Library()

# Add Books
library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald", 3))
library.add_book(Book("1984", "George Orwell", 5))
library.add_book(Book("To Kill a Mockingbird", "Harper Lee", 2))

# List Books
library.list_books()

# Borrow a Book
library.borrow_book("1984")

# List Books Again
library.list_books()

# Return a Book
```

```
library.return_book("1984")

# List Books Again
library.list_books()
```

Output should show the library's status before and after borrowing/returning books.