Question 01: **File Analysis and Word Search**

Instructions:

1. Write a Python function named **analyze_file** that takes the name of a file as input and returns a dictionary containing the counts of characters, words, and lines in that file.
2. Write another Python function named **search_word** that takes the name of a file and a word as input and returns the count of occurrences of that word in the file.
3. Ensure that your functions handle file not found errors gracefully and return appropriate values in such cases.

Submission:

Additional Notes:

- Ensure that your functions are properly documented with clear descriptions of parameters, return values, and any assumptions made.
- Use appropriate error handling techniques to handle potential exceptions.
- Test your functions with different input files and scenarios to verify their correctness and robustness.

Question 2: **Library Management System using Python OOP**

Problem Statement:

Create a simplified Library Management System with Python classes that represent a Library, Books, and Users. Your goal is to design the system so that it enables basic functionalities, including adding and managing books, registering users, and borrowing/returning books.

Instructions:

1. **Create a base class named Book:**
   - Attributes:
     - title (string): The title of the book.
     - author (string): The author of the book.
     - isbn (string): A unique identifier for each book.
     - is_borrowed (boolean): Indicates if the book is currently borrowed.
   - Methods:
     - borrow(): Sets the is_borrowed attribute to True.
     - return_book(): Sets the is_borrowed attribute to False.
2. **Create a subclass named DigitalBook that inherits from Book:**
   - Additional Attributes:
     - file_format (string): Format of the digital book (e.g., PDF, EPUB).
   - Override the borrow() method to print an additional message indicating that the book can be accessed online.
3. **Create another subclass named AudioBook that inherits from Book:**
   - Additional Attributes:
     - duration (float): The length of the audiobook in hours.
   - Override the borrow() method to print an additional message indicating that the audiobook is available for streaming.

4. **Create a class named User:**
    - Attributes:
        - user_id (string): A unique identifier for each user.
        - name (string): The user's name.
        - borrowed_books (list): A list of Book objects borrowed by the user.
    - Methods:
        - borrow_book(book): Adds a book to borrowed_books if it is not already borrowed. Use encapsulation to ensure users cannot modify borrowed_books directly.
        - return_book(book): Removes the book from borrowed_books and marks it as returned.
5. **Create a class named Library:**
    - Attributes:
        - name (string): Name of the library.
        - books (list): A list of all Book objects in the library.
        - users (list): A list of all registered User objects.
    - Methods:
        - add_book(book): Adds a new book to the library.
        - register_user(user): Registers a new user.
        - lend_book(user_id, isbn): Allows a user to borrow a book if available.
        - receive_return(user_id, isbn): Allows a user to return a borrowed book.

Requirements:

- **Inheritance** should be evident in the DigitalBook and AudioBook subclasses that inherit from the Book base class.

- **Polymorphism** should be demonstrated in the overridden borrow() methods of DigitalBook and AudioBook.
- **Encapsulation** should be used in the User class to prevent direct modification of the borrowed_books attribute from outside the class.

Submission:

Submit your Python script containing all the classes (Book, DigitalBook, AudioBook, User, and Library) along with test cases demonstrating the following scenarios:

1. Adding books and users to the library.
2. Users borrowing and returning both digital and physical books.
3. Handling cases where users try to borrow books that are already borrowed.

Additional Notes:

- Document your code, explaining each class and method.
- Use error handling to manage potential issues, such as borrowing a book that's unavailable.
- Test the system by creating instances of DigitalBook and AudioBook, and by simulating