Class Exercise:

Decorators v2

Let's imagine you are working on a program for a library management system. In this exercise, you will create a class called Book that represents a book in the library. The Book class will have the following functionalities:

1. It should be able to track the number of times a book has been borrowed.
2. It should have a method called display that prints the book's title, author, and the number of times it has been borrowed.
3. It should have a method called borrow that increments the borrow count by 1 each time it is called.

To make things more interesting, you need to implement decorators to add additional functionalities to the Book class. Here are the decorators you need to implement:

1. **time\_taken\_decorator:** This decorator should calculate and print the time taken to execute any method of the Book class.
2. **authenticate\_decorator**: This decorator should prompt the user to enter a password before executing any method of the Book class. The password should be passed as an argument to the decorator.
3. **log\_decorator**: This decorator should log the name of the method being executed.

Your task is to implement the Book class and the decorators described above. Make sure to use lambda functions, composition of decorators, and passing arguments to decorators where necessary.

import time

def time\_taken\_decorator(func):

    def wrapper(\*args, \*\*kwargs):

# Your code here

    return wrapper

def authenticate\_decorator(password):

    def decorator(func):

        def wrapper(\*args, \*\*kwargs):

# Your code here

        return wrapper

    return decorator

def log\_decorator(func):

    def wrapper(\*args, \*\*kwargs):

# Your code here

    return wrapper

class Book:

    def \_\_init\_\_(self, title, author):

# Your code here

    @log\_decorator

    @authenticate\_decorator("secret")

    @time\_taken\_decorator

    def display(self):

# Your code here

    @log\_decorator

    @authenticate\_decorator("secret")

    @time\_taken\_decorator

    def borrow(self):

# Your code here

# Test the Book class and decorators

book = Book("The Great Gatsby", "F. Scott Fitzgerald")

book.display()

print()

book.borrow()

book.borrow()

Expected output:

Enter password: secret

Executing method: display

Title: The Great Gatsby

Author: F. Scott Fitzgerald

Borrow count: 0

Time taken to execute display: 0.001234 seconds

Enter password: secret

Executing method: borrow

The Great Gatsby has been borrowed 1 times.

Time taken to execute borrow: 0.000567 seconds

Enter password: secret

Executing method: borrow

The Great Gatsby has been borrowed 2 times.

Time taken to execute borrow: 0.000678 seconds