PYTHON

Assignment

1. **Basic Decorator Function:**
   - **Write a decorator that prints "Hello" before executing a function and "Goodbye" after executing it.**
   - **Apply this decorator to a function that prints "World".**
2. **Timing Function Execution:**
   - **Create a decorator that measures the time a function takes to execute.**
   - **Apply this decorator to a function that sums numbers from 1 to 1,000,000.**
3. **Repeat Execution:**
   - **Write a decorator that executes a function three times.**
   - **Test this decorator on a function that prints a given message.**
4. **Access Control Decorator:**
   - **Create a decorator that checks if a user is an admin before allowing the execution of a function.**
   - **If the user is not an admin, print "Access Denied".**
   - **Test with a simple function that returns a protected message.**
5. **Decorator with Arguments:**
   - **Write a decorator that accepts a string argument and prints it before executing the function.**
   - **Apply it to a function that returns a greeting message.**

6. **Chaining Decorators:**
   - **Implement two simple decorators (e.g., one that converts the result to uppercase and another that doubles the result).**
   - **Apply both decorators to a single function to demonstrate chaining.**
7. **Context Management with Decorators:**
   - **Write a decorator that acts as a context manager, ensuring that a resource (e.g., a file) is opened before the function runs and closed afterward.**
   - **Use it to read and print the contents of a file.**
8. **Retry Decorator:**
   - **Write a decorator that retries a function up to 3 times if it raises an exception.**
   - **Test it on a function that raises an exception with a probability of 50%.**
9. **Decorator for Type Checking:**
   - **Create a decorator that checks the types of the arguments passed to a function.**
   - **If the types don't match the expected ones, raise a TypeError.**
   - **Test it on a function that adds two integers.**
10. **Decorator for Flattening Nested Lists:**
    - **Write a decorator that flattens nested lists passed to a function.**
    - **Apply it to a function that sums all the numbers in a list, including nested ones.**