Logging module questions

EASY LEVEL :

1. Write a Python program that logs a message **"Program started"** using the default logging configuration.
2. Configure the logging module to display logs of **level INFO and above**.
3. Write a program that logs messages at all five standard levels:
   - DEBUG
   - INFO
   - WARNING
   - ERROR
   - CRITICAL
4. Change the logging format to include:
   - Log level
   - Log message
     Example output:
5. INFO: This is an info message
6. Write a program that saves log messages to a file named app.log.
7. Log an error message when a **division by zero** occurs.
8. Configure logging to show the **current date and time** with each log message.

MEDIUM LEVEL :

1. Create a logger that writes:
    - INFO logs to the console
    - ERROR logs to a file called error.log
2. Write a function that accepts two numbers and logs:
    - INFO when division is successful
    - ERROR when division fails
3. Configure a logger with a custom format that includes:
    - Timestamp
    - Logger name
    - Log level
    - Message
4. Create a Python script with two modules where both modules use the **same logger configuration**.

    Log a full stack trace when an exception occurs using the logging module.
5. Set different logging levels for:
    - Root logger
    - Custom logger named "my_app"
6. Write a program that logs user login attempts and warns if a user fails to log in more than **3 times**.

HARD LEVEL :

1. Configure **RotatingFileHandler** to:
   - Create app.log
   - Rotate the log file when it exceeds **1 MB**
   - Keep **5 backup files**
2. Implement **TimedRotatingFileHandler** to rotate logs **daily at midnight**.
3. Create a decorator that logs:
   - Function name
   - Arguments
   - Return value
   - Execution time
4. Design a logging system for a **web application** that logs:
   - Requests (INFO)
   - Client errors (WARNING)
   - Server errors (ERROR)
5. Write a multi-threaded program where each thread logs its name and task progress safely.
6. Create a configuration-based logging system using a **logging.config.dictConfig()** setup that supports:
   - Console logging
   - File logging
   - Different log levels for different modules