Debugging & Traceability

```
import logging


# Configure the logging settings

logging.basicConfig(filename='video_debug.log', level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')


def main():

    try:

        # Simulate a video processing task

        process_video()

    except Exception as e:

        # Handle and log any exceptions

        logging.exception("Error in video processing: %s", e)


def process_video():

    try:

        # Simulate video processing

        result = 1 / 0  # Intentionally raise a division by zero error for demonstration

    except ZeroDivisionError as e:

        # Handle and log specific exceptions

        logging.error("Error in video processing: Division by zero - %s", e)

    except Exception as e:

        # Handle and log any other exceptions

        logging.error("Error in video processing: %s", e)


if __name__ == "__main__":

    main()
```

In this program:

We set up a logging configuration that writes log messages to a file named “video_debug.log” and logs messages with a minimum level of DEBUG. We also specify the log message format to include timestamps.

The main function simulates a video processing task and catches any exceptions that might occur during video processing. It logs these exceptions with a stack trace.

The process_video function demonstrates handling and logging specific exceptions, such as a division by zero error.

When you run this program, it will perform the video processing task and log any errors, exceptions, or debug information to the “video_debug.log” file. This log file will help you trace and debug issues in your video processing application.

You can extend this program by adding more sophisticated debugging and traceability features, such as custom log messages and advanced exception handling, depending on the specific needs of your video application. Additionally, consider integrating external tools or services for enhanced traceability, such as issue tracking systems or specialized debugging tools.