# Forward

## About the tutorial

Every useful library will, sooner or later, encounter an abnormal condition, like a missing file, or divide by zero, that the library itself can’t handle. The usual reaction of the program is often to somehow display an error message. Simply printing unstructured text out stderr is inconvenient for library users who may need to handle the condition programmatically, translate the message, or display the message elsewhere, like a GUI or web page.

This tutorial is about handling such conditions properly, especially in libraries, in SWI-Prolog, using a system borrowed from Quintus Prolog and enshrined in the ISO Prolog standard. It is particularly important for authors of packs to follow this method, and not simply use format/3. Doing so ensures that libraries can be embedded in diverse environments.

It should take most programmers well under an hour.

This tutorial should be accessible to any programmer who understands the basics of Prolog. It would also be useful to understand the basic idea of an execution stack, and to have seen the common throw/catch mechanism in some computer language.

# Simplified pipeline

## Handle the exception (semantic term)

## Produce list of tokens ("lines")

## Hooks to output or print\_message\_lines

# Producing a message term

## Semantic messages

### Describe semantic message and why it's better

### Format of semantic message term

### ISO exception format

### The ISO exceptions

### The format/2 format - and why not to use it

### The debug/2 format - its for debug/3

## Producing messages

### Directly calling

### Debug library (discussed later)

### Uncaught exceptions

## Exception Handling

### Basics of exceptions and catching

### Rethrowing

### Decorating the exception with prolog\_exception\_hook

### Uncaught exceptions - the top level exception handler

### Transition to print\_message/2

## Review

### Use the ISO messages

### Make your own terms

# Converting messages to lines of strings

## Print\_message/2

## prolog:message//1

## The 'lines' format

# Producing output

## print\_message\_lines/3

## message\_hook

## thread\_message\_hook

## message\_to\_string/2

# The debug library

## Debug messages with debug/3

## The debug message window

## Prolog:debug\_print\_hook