Overview

When we first created Igor, some time in the last millennium, it supported automation through macros. The idea of the macro was to allow users to collect commands into conveniently callable routines. Igor interpreted and executed each command in a macro as if it were entered in the command line.

WaveMetrics soon realized the need for a faster, more robust technology that would support full-blown programming. This led to the addition of user-defined functions. Because functions are compiled, they execute much more quickly. Also, compilation allows Igor to catch syntactic errors sooner. Functions have a richer set of flow control capabilities and support many other programming refinements.

Over time, the role of macros has diminished in favor of functions. With rare exceptions, new programming should be done using user-defined functions.

Macros are still supported and there are still a few uses in which they are preferred. When you close a graph, table, page layout, Gizmo plot, or control panel, Igor offers to automatically create a window recreation macro which you can later run to recreate the window. You can also ask Igor to automatically create a window style macro using the Window Control dialog. The vast majority of programming, however, should be done using functions.

The syntax and behavior of macros are similar to the syntax and behavior of functions, but the differences can be a source of confusion for someone first learning Igor programming. If you are just starting, you can safely defer reading the rest of this chapter until you need to know more about macros, if that time ever comes.

Comparing Macros and Functions

Like functions, macros are created by entering text in procedure windows. Each macro has a name, a parameter list, parameter declarations, and a body. Unlike functions, a macro has no return value.

Macros and functions use similar syntax. Here is an example of each. To follow along, open the Procedure window (Windows menu) and type in the macro and function definitions.

```
Macro MacSayHello(name)
   String name

   Print "Hello "+name
End

Function FuncSayHello(name)
   String name

   Print "Hello "+name
End
```

Now click in the command window to bring it to the front.

If you execute the following on the command line

```
MacSayHello("John"); FuncSayHello("Sue")
```

you will see the following output printed in the history area:

```
Hello John
Hello Sue
```

This example may lead you to believe that macros and functions are nearly identical. In fact, there are a lot of differences. The most important differences are:

- Macros automatically appear in the Macros menu. Functions must be explicitly added to a menu, if desired, using a menu definition.
- Most errors in functions are detected when procedures are compiled. Most errors in macros are detected when the macro is executed.

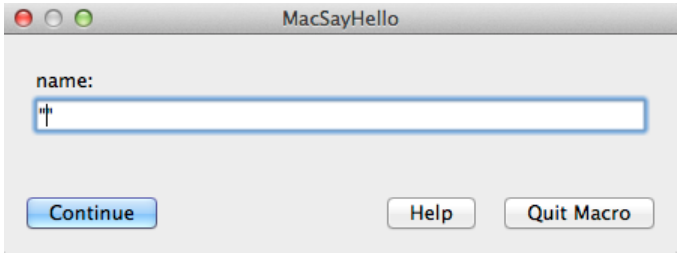
- Functions run a lot faster than macros.
- Functions support wave parameters, for loops, switches, structures, multithreading, and many other features not available in macros.
- Functions have a richer syntax.

If you look in the Macros menu, you will see MacSayHello but not FuncSayHello.

If you execute `FuncSayHello()` on the command line you will see an error dialog. This is because you must supply a parameter. You can execute, for example:

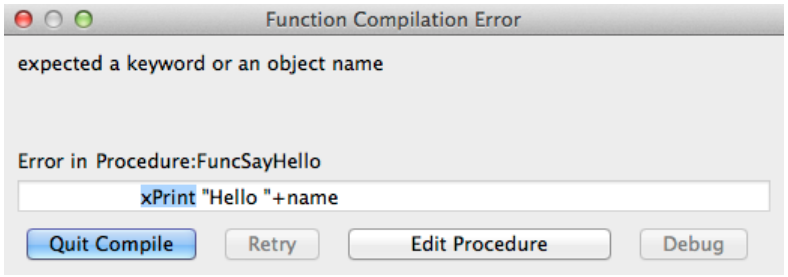
```
FuncSayHello("Sam")
```

On the other hand, if you run MacSayHello from the Macros menu or if you execute `MacSayHello()` on the command line, you will see a dialog that you use to enter the name before continuing:



This is called the "missing parameter dialog". It is described under **The Missing Parameter Dialog** on page IV-121. Functions can display a similar dialog, called a simple input dialog, with a bit of additional programming.

Now try this: In both procedures, change "Print" to "xPrint". Then click in the command window. You will see an error dialog complaining about the xPrint in the function:



Click the Edit Procedure button and change "xPrint" back to "Print" in the function but not in the macro. Then click in the command window.

Notice that no error was reported once you fixed the error in the function. This is because only functions are compiled and thus only functions have their syntax completely checked at compile time. Macros are interpreted and most errors are found only when the line in the procedure window in which they occur is executed.

To see this, run the macro by executing "`MacSayHello("Sam")`" on the command line. Igor displays an error dialog: