Executing with an optional parameter as an input:

```
OptionalParameterDemo(8,6, c=66)
  8   6   66   0
```

Note that the optional parameter is explicitly defined in the function call using the *ParamName=Value* syntax. All optional parameters must come after any required function parameters.

Local Versus Global Variables

Numeric and string variables can be **local** or **global**. "Local" means that the variable can be accessed only from within the procedure in which it was created. "Global" means that the variable can be accessed from the command line or from within any procedure. The following table shows the characteristics of local and global variables:

Local Variables	Global Variables
Are part of a procedure.	Are part of the data folder hierarchy of an Igor experiment.
Created using Variable or String within a procedure.	Created using Variable or String or Variable/G or String/G from the command line or within a procedure.
Used to store temporary results while the procedure is executing.	Used to store values that are saved from one procedure invocation to the next.
Cease to exist when the procedure ends.	Exist until you use KillVariables, KillStrings, or KillDataFolder.
Can be accessed only from within the procedure in which they were created.	Can be accessed from the command line or from within any procedure.

Local variables are private to the function in which they are declared and cease to exist when the function exits. Global variables are public and persistent — they exist until you explicitly delete them.

If you write a function whose main purpose is to display a dialog to solicit input from the user, you may want to store the user's choices in global variables and use them to initialize the dialog the next time the user invokes it. This is an example of saving values from one invocation of a function to the next.

If you write a set of functions that loads, displays and analyzes experimentally acquired data, you may want to use global variables to store values that describe the conditions under which the data was acquired and to store the results from the analyses. These are examples of data accessed by many procedures.

With rare exceptions, you should not use global variables to pass information to procedures or from one procedure to another. Instead, use parameters. Using global variables for these purposes creates a web of hidden dependencies which makes understanding, reusing, and debugging procedures difficult.

Local Variables Used by Igor Operations

When invoked from a function, many Igor's operations return results via local variables. For example, the WaveStats operation creates a number of local variables with names such as V_avg, V_sigma, etc. The following function illustrates this point.

```
Function PrintWaveAverage(w)
   WAVE w

   WaveStats/Q w
   Print V_avg
End
```

When the Igor compiler compiles the WaveStats operation, it creates various local variables, V_avg among them. When the WaveStats operation runs, it stores results in these local variables.