Variables

A variable is a holding place for the value of an expression. There are several types of variables. They differ in how they are created and how long they last. This chapter will explore temporary and workspace variables.

Workspace Variables

Every workspace starts off with three variables. They are named a, b & c. You can see these variables in the top part of the workspace. The value that each variable is currently holding is also displayed on the right. Initially, workspace variables hold the value ??? (pronounced ‘unknown’).

The value of a variable can be changed by the assignment symbol ‘:=’.

If you execute the following two expressions, you’ll see the value of two of the workspace variables change.

```
a := 3 + 4.

b := 1 / 5.
```

Notice that, even if you execute other expressions, unless you assign a different value to a workspace variable, it will hold onto its value.

You can use a variable in an expression. The value of the variable at the time the expression is executed is used. Executing these expressions in turn changes the values of a and b:

Expression	:: Value in a	:: Value in b
a := 3.	:: 3	:: ???
b := 2 * a.	:: 3	:: 6
a := "Hello".	:: "Hello"	:: 6
a := a && "kitty".	:: "Hello kitty"	:: 6
b := b * b / 2.	:: "Hello kitty"	:: 18

»
The results of executing expressions in a workspace appear in the list of the workspace next to the word result. Result is just another workspace variable and can be used in expressions.

Local Variables

A local variable lasts for just a short period of time. It can be used as a temporary holding place for values during the execution of a script that you won't need after the script is done. After the script is executed, local variables disappear.

You have to tell the system the names of the local variables you will use. You do this by declaring them with the '$' syntax:

```
$ x.

x := "cha".

a := x & x & x.
```

After this script is run, a will have the value "chachacha".

!!
Be sure you selected all three lines of the above expression to execute. If you select and execute just the first line, the system will create the local variable x, and then destroy it. If you then select the second line and execute it, you'll get an error because the variable x no longer exists. Local variables last only as long as a single execution.

Variable Names

Local variables can have almost any name:

```
bob

q-factor

pre-1886-shipments
```

The rules are simple: it can be any combination of letters, digits, and dashes so long as it starts with a letter. Spaces and punctuation cannot be part of a variable name. Upper/Lower case distinctions do not matter; however the system will convert everything to lower case.

More than one local can be declared at a time:

```
$ the-price, the-tax.

the-price := 13.29.

the-tax := the-price * 0.07.

the-price + the-tax.
```

The result of this is 14.22. While it is running, the local variables are set to 13.29 and 0.93. Then the sum of them is returned as the result.

Unlike workspace variables, local variables don't start off with any value. When they are declared, their value is undefined. If you execute an expression with a local variable and haven't assigned the variable a value, the system will give the error "using an undefined value".

»
You must declare all the local variables you are going to use before any expressions that use them. However, you can declare them with one or more declaration statements:

$ game-1, game-2.

is the same as

$ game-1.

$ game-2.

!!

A common mistake is to forget the space between the $ and the variable name. If you do this you'll see get one of two error messages: either "nothing more expected" or "unused expression". The error messages are a little confusing because a $ with letters following it without a space actually means something else to the system: it is a symbol.