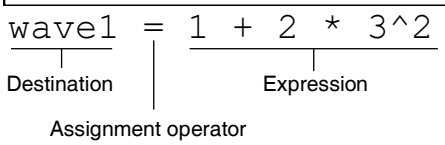

This assigns 19 to every point in wave1.

The spaces in the above example are not required. You could write:

```
wave1=1+2*3^2
```

See **Waveform Arithmetic and Assignments** on page II-74 for details on wave assignment statements.

In the following examples, str1 is a string variable, created by the String operation, var1 is a numeric variable, created by the Variable operation, and wave1 is a wave, created by the Make operation.

```
str1 = "Today is " + date()                 // string assignment

str1 += ", and the time is " + time()       // string concatenation

var1 = strlen(str1)                         // variable assignment

var1 = pnt2x(wave1,numpnts(wave1)/2)        // variable assignment

wave1 = 1.2*exp(-0.2*(x-var1)^2)            // wave assignment

wave1[3] = 5                                // wave assignment

wave1[0,;3]= wave2[p/3] *exp(-0.2*x)        // wave assignment
```

These all operate on objects in the current data folder. To operate on an object in another data folder, you need to use a data folder path:

```
root:'run 1':wave1[3] = 5        // wave assignment
```

(This syntax applies to the command line and macros only, not to user-defined functions in which you must use **Wave References** to read and write waves.)

See Chapter II-8, **Data Folders**, for further details.

If you use liberal wave names (see **Object Names** on page III-501), you must use quotes:

```
'wave 1' = 'wave 2'        // Right

wave 1 = wave 2            // Wrong
```

(This syntax applies to the command line and macros only, not to user-defined functions in which you must use **Wave References** to read and write waves.)

Assignment Operators

The assignment operator determines the way in which the expression is combined with the destination. Igor supports the following assignment operators:

Operator	Assignment Action
=	Destination contents are set to the value of the expression.
+=	Expression is added to the destination.
-=	Expression is subtracted from the destination.
*=	Destination is multiplied by the expression.
/=	Destination is divided by the expression.
:=	Destination is dynamically updated to the value of the expression whenever the value of any part of the expression changes. The := operator is said to establish a “dependency” of the destination on the expression.