Maple syntax

The most common kind of mistake to make when using Maple is a syntax error. This section reviews the basics of Maple syntax and demonstrates several common syntax errors and how to recover from them.

Don't forget the semi-colon!

Rule 1, of course, is that every Maple statement must end with a semi-colon or a colon. If you forget to end a statement with a colon, Maple will warn you about this and let you go back and put the semicolon in.

```
> int(x^2,x)
Warning, incomplete statement or missing semicolon
```

At this point you could go back and insert the semicolon.

Match your parentheses! (Brackets and braces,too!)

In Maple statements, there must be a right parenthesis for every left parenthesis, a right bracket for every left bracket and a right brace for every left one. For instance:

```
> plot(x^2,x=-2..2;
Syntax error, `;` unexpected
```

Maple gives you an error message and puts the cursor back on the statement at the point at which the statement ceased to be syntactically correct. In this example, this point happens to be where the right parenthesis is needed. Sometimes, however, the cursor may not be in the place where the correction needs to be made. For instance, suppose you are trying to assign the expression
`(x-3)*sin(x)` to the variable `y`, but you mistakenly type:

```
> y:=(x-3*sin(x);
Syntax error, `;` unexpected
```

The statement could have made sense all the way up to the semi-colon (but because of the unbalanced parentheses, the statement cannot end there); Maple indicates that something is wrong, but it does not tell (indeed, there is no way to tell) where the missing parenthesis should go.

Don't forget the * for multiplication...

A common mistake is to forget the asterisk for multiplication. The `*` is required whenever multiplication is to be performed. It is especially easy to forget this when writing Maple for what in standard notation looks like sin(2x) or something similar. Maple requires you to write `sin(2*x)` :

```
> sin(2x);
Syntax error, missing operator or `;`
```

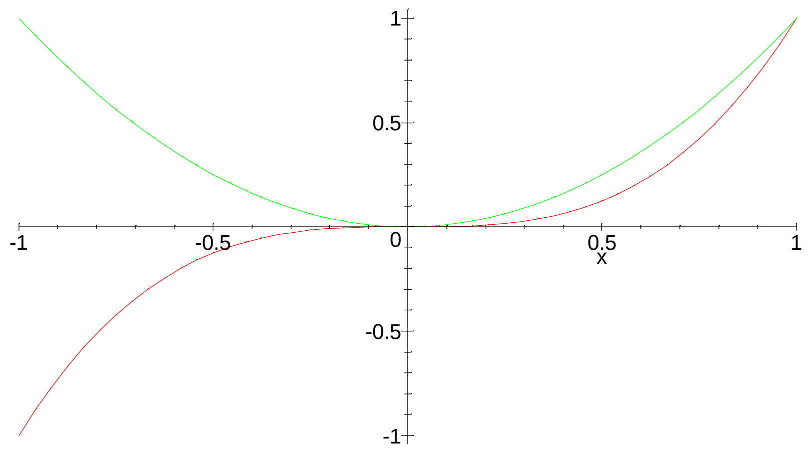
Group lists of things to plot or equations to solve with braces...

A common error is to forget the braces when plotting several curves in the same picture, or solving several equations simultaneously:

```
> plot(x^2,x^3,x=-1..1);
Error, (in plot) invalid arguments
```

This means that Maple has construed the three arguments to plot as being "`x^2`", "`x^3`" and "`x=-1..1`". The second of these, "`x^3`" is not a valid second argument, because the second argument to plot must be a range (like "`x=-1..1`") or some other indication of how to do the plot. What to plot must be indicated by the first argument only. Therefore, to plot several things at once, you must group them together with braces so that the first argument becomes a set of things to plot:

```
> plot({x^2,x^3},x=-1..1);
```



That's better.

The same applies to solve:

```
> solve(x+y=1,x-y=2);
Error, (in solve) invalid arguments
```

This is the same problem -- the second argument to solve (which is optional) is the list of variables to solve for. The equations must be in the first argument:

```
> solve({x+y=1,x-y=2});
```

$$\{x = \frac{3}{2}, y = \frac{-1}{2}\}$$

It is important to remember that the values of the variables are not assigned to them by the solve statement.

Misspellings

Usually, if you spell the name of a command incorrectly, no syntax error will result. Maple will just assume that you meant to use a command that has yet to be specified:

```
> solbe(x^2+x=4);
```

$$\text{solbe}(x^2 + x = 4)$$

You can see that Maple just parrots back the input, because it has no way of interpreting it. When this happens, it is ok simply to retype the statement correctly, or even to place the cursor back on the incorrect statement, correct the error, and try again.

```
> solve(x^2+x=4);
```

$$-\frac{1}{2}+\frac{1}{2}\sqrt{17}, -\frac{1}{2}-\frac{1}{2}\sqrt{17}$$

Commands contained in special libraries...

Some of the commands we will use are contained in special libraries that are not read in automatically when you start Maple (because there are too many of them). For example, the commands "`leftbox`", "`leftsum`", etc.. for illustrating the idea of integration are contained in the "`student`" library. To read the commands into the computer's memory, you must use a "`with`" statement. If you forget to do this, Maple will respond as though you misspelled a command:

```
> leftbox(x^2,x=1..4,10);
```

$$\text{leftbox}(x^2, x = 1 .. 4, 10)$$

When Maple parrots like this, and you have ascertained that you spelled the command correctly, then consider whether you need to read the command in from the library. For example, to get the `leftbox` command:

```
> with(student,leftbox);
```

$$[\mathit{leftbox}]$$

The syntax here is

`with(` "name of library"`,` "name(s) of command(s) separated by commas if there are more than one"`);`

If you want to read in the entire contents of a library (we do this occasionally with the `student` library), then `with(student);` will do this.

Here is a (reasonably) complete list of the commands we will be using from various libraries in Math 140-141 and 150-151:

`student` library: `leftbox, leftsum, rightbox, rightsum, value, isolate, changevar, middlebox, middlesum, makeproc, simpson, trapezoid`

`plots` library: `animate, display, display3d, implicitplot, textplot, tubeplot`

`linalg` library (Math 151 only): `addrow, augment, backsub, det, evalm, gausselim, gaussjord, inverse, matrix, mulrow, multiply, rank, rref, scalarmul, swaprow, trace, vector`

`DEtools` library: `DEplot`

`stats` library (Math 151 only): `describe[mean], describe[variance], describe[standarddeviation], statevalf[...]`

`simplex` library (Math 151 only): `maximize, minimize`