Function definitions

The syntax for defining functions is

$$\textit{function-name}\ (\ \textit{arg-list}\ ) = \textit{expr}$$

where *arg-list* is a comma separated list of zero to nine symbols that receive arguments. Unlike symbol definitions, *expr* is not evaluated when *function-name* is defined. Instead, *expr* is evaluated when *function-name* is used in a subsequent computation. The scope of function arguments is the function definition *expr*.

Function definitions cannot be nested. In other words, function definition *expr* cannot contain another function definition.

The following example defines a sinc function and evaluates it at $\pi/2$.

```
f(x) = sin(x)/x
f(pi/2)
```

$$\frac{2}{\pi}$$

After a user function is defined, *expr* can be recalled using the `binding` function.

```
binding(f)
```

$$\frac{\sin(x)}{x}$$

To define a local symbol for use inside *expr*, extend the argument list. In the following example, argument y is used as a local symbol. Note that function L is called without supplying an argument for y.

```
L(f,n,y) = eval(exp(y) / n! d(exp(-y) y^n, y, n), y, f)
L(cos(x),2)
```

$$\tfrac{1}{2}\cos(x)^2 - 2\cos(x) + 1$$

Sometimes it is necessary to evauate an argument at a particular value. Use `eval` to evaluate function arguments inside *expr*.

```
h(f,x,a) = abs(eval(f,x,a))
h(cos(y),y,0)
```

1