```
> restart;
```

You can make functions using the arrow "-->" assignment, like this:
f:=(x,y) -> x/y^2

However (for various reasons I won't go into now) I prefer to use the 'unapply' method, like this:
xy:=x/y^2; f:=unapply(fxy,(x,y));
or like this:
f:=unapply(x/y^2,(x,y));

```
> f:=unapply(x/y^2,(x,y));
```

$$f := (x, y) \rightarrow \frac{x}{y^2}$$

Now we want to compute and print f(x,y_i) for an array of y = (y_i) values (and for some fixed x). If the y_i are regularly spaced, we might express
y_i = a + b*i, for example. Alternatively, we can first form an array Y of y-values, then compute and print the f(x,Y[i]).
Below, I will take x = 2 and y = 1,1.5, 2, 2.5, 3.

```
> Y:=seq(1+.5*(i-1),i=1..5);
```

$$Y := 1.0, 1.5, 2.0, 2.5, 3.0$$

If we want a nicely formatted print, we can use the 'printf' command, which is essentially the same as in the C language.

```
> for i from 1 to 5 do
    if (i=1) then printf("%5s %5s %10s\n", "x", "y", "f(x,y)"):
    end if:
    printf("%5.1f %5.1f  %10.6f\n", 2.0, Y[i],f(2.0,Y[i])):
  end do:
    x     y     f(x,y)
  2.0   1.0   2.000000
  2.0   1.5   0.888889
  2.0   2.0   0.500000
  2.0   2.5   0.320000
  2.0   3.0   0.222222
```

```
> i:='i';
```

$$i := i$$

```
> 
```