How can I write functions in Maple?

How can I write functions in Maple?

Mathematical functions of one or more variables may be defined as follows:

```
f:=(x_1,x_2...x_n) -> function definition
```

For example:

```
f:=(x,y,z) -> x^2+sin(y*z);
```

defines f as a function of the 3 variables x,y,z.

To evaluate the function at a given point, we just call the function as we normally would passing argument values:

```
f(2,1,3);
```

Operations such as differentiation are treated differently from those of ordinary algebraic expressions.

For example, to differentiate the expression d:=x^2*y+z*sin(y*z), we can use diff as follows:

```
diff(d,z)
```

which gives us

```
sin(y*z)+z*y*cos(y*z)
```

However if we had the function

```
f:=(x,y,z) -> x^2*y+z*sin(y*z)
```

then using diff(f,z) gives us zero. In order to differentiate the function, we have to specify the arguments since Maple would treat f as a symbol in and of itself and return 0.

Thus the proper expression would be

```
diff(f(x,y,z),z)
```

A common problem among many users who use Maple to do differentiation of mathematical functions is in the evaluation of the derivatives of functions at points.

Say for example, we wished to evaluate the derivative of f at the points (x1,y1,z1). If we tried to define a new function in terms of the diff function, viz

```
r:=(x,y,z) -> diff(f(x,y,z),z)
```

we would get the result

```
r:=(x,y,z) -> diff(f(x,y,z),z)
```

and evaluating at the point (Pi,Pi,Pi) we would get

```
3*Pi^2+sin(Pi^2)+2*Pi^2*cos(Pi^2)
```

which is clearly wrong. This is due to the fact that in defining r as above results in delayed evaluation of the expression and substitues the values of the args before differentiation which results in Maple treating Pi as a symbolic variable with which to differentiate r with respect to.

The way to solve this would be to use the **unapply** operator, viz

```
r:=unapply(diff(f(x,y,z),z),x,y,z);
```

and evaulating r(Pi,Pi,Pi), we obtain

```
sin(Pi^2)+Pi^2*cos(Pi^2)
```

which is indeed correct.

A few notes on unapply, adapted (and corrected) from the ?unapply help page:

- The result of unapply(expr,x) is a functional operator. Applying this operator to x gives the original expression.

  ```
  unapply(expr,x)(x) ==> eval(expr)
  ```

- Typically, for a function f(x):

  ```
  unapply(f(x),x) ==> f
  ```

  although this depends somewhat on the evaluation of f(x). A safer statement would be:

  ```
  unapply('f'(x),x) ==> f
  ```

- unapply should be used whenever it is desired to construct an operator using contents of variables or evaluated expressions.
- unapply implements (almost) the lambda-expressions of lambda calculus. The scoping behaviour of unbound names is not the same in the lambda calculus.

See Also

- Maple Landing Page