Quick reference to *Mathematica*

Getting started

?Command - gives a fairly detailed description of a command.
e.g. ?Plot tells you all about this command. You can use * as a wildcard, for instance ?*Plot* gives a list of all commands with Plot in their name.

***** - Times command (2*3 gives 6)

^ - Power command (2^3 gives 8)

n! - factorial (3! gives 6)

{} - denotes a list eg {2,3,4}

() - Places variables together eg (1+x)/(1-x) takes 1+x over 1-x

[] - Generally used to denote a function of something else

f /. object1 -> object2
- tells Mathematica to make replace object1 with object2 in the function
eg **3*x^2 /. x -> 2*y+z** gives **3*(2*y + z)^2**

Out[%number] - This command allows you to reuse a function that you have already described, eg:

In[1]:= **3*x^2**
Out[1]= **3*x^2**
In[2]:= **D[Out[1],x]**
Out[2]= **6*x**

D[Out[1],x] takes the derivative of the first output (3*x^2) as a function of x.

Part[eqn,i] - Grabs the ith part of eqn eg Part[3x^2+x^3, 2] gives x^3

Functions and constants in *Mathematica* (A small fraction!)

Abs[x] -Takes the absolute value of x

E - The exponential constant 2.71838. **E^(x)** can be invoked using **Exp[x]**

I - The square root of negative 1.

Infinity - Self-explanatory.

Log[x] -Takes the natural log of x

Log[b,x] -Takes the log of x in base b

Pi - 3.14159...

Sin[x], Cos[x], Tan[x] - trigonometric functions
ArcSin[x], ArcCos[x], ArcTan[x] - inverse trigonometric functions

Sqrt[x] - The square root of x

Writing equations in *Mathematica*

x=y - Sets x to y immediately and from then on (use Clear[x] to unassign x)
eg **plot1 = Plot[x^2,{x,0,10}]** assigns the name **plot1** to a plot

x:=y - Does nothing until x is called, at which point x is assigned the value y

x==y - Tests whether x equals y BUT makes no assignment

f[x_]:= - This is how you define a function (called "f") of x
eg **f[x_] := x^2**

f[x] - This gives the function evaluated at x.
eg **f[3]** gives **9** in the above example

f[x_,y_,...]= - This is how you define a function of several variables

■ A list of helpful commands

Array[f, n] - makes a list by calling the function f n times
{f[1], f[2], ..,f[n]} where f is defined previously as f[x_]:= stuff.

Array[f, {n1, n2}] - makes a matrix (f must be a function of two variables).

Clear[symbol1, symbol2, ..] - clears variable or function definitions
e.g. Clear[x,y,pop1, f...]

D[f,x] - takes the partial derivative of f with respect to x - e.g. D[x^2+y Log[x], x]

D[f, {x, n}] - takes the nth derivative with respect to x - e.g. D[x^2+y Log[x],{x,2}]

DSolve[eqn, y[x],x] - solves differential equation for y as a function of x
(SYMBOLICALLY) e.g. DSolve[y'[x] == k y, y[x],x]

DSolve[eqns, {y1,y2,y3,..}, x] same as above but for a system of eqns
e.g. pred-prey equations DSolve[{y'[x] == k y-x, z'[x]==x+z}, {y[x],z[x]}, x]

NDSolve[eqns, y, {x, xmin,xmax}] - same as DSolve but seeks solution
NUMERICALLY - e.g. NDSolve[{y'[x] ==4 y[x], y[3]==62}, y[x], {x, 0,20}]

Expand[expr]- expands an expression e.g. Expand[(1+x)^2] gives 1+2x+x^2

Evaluate[object] - evaluates a symbolic object like interpolating functions

Factor[polynomial] - self explanatory - e.g. Factor[x^2 + 2 x + 1]

For[start,test,increment,body] - repeats procedure "body", starting from "start" until the "test" condition is met, adding "increment" each time.
e.g. For[i=1,i<10,i=i+1,Print[i]] prints out the integers 1 through 9.

FindRoot[eqn1==eqn2, {x, x0}] - searches for numerical root of eqn1==eqn2 starting at x0 e.g. FindRoot[Log[x] + x + Arctan[x] == 0, {x, 4}] tries to find x that satisfies this very ugly - impossible to solve by hand equation, starting at x=4.

Integrate[f,x] - finds indefinite integral of f with respect to x
e.g. Integrate[Log[x], x]

Integrate[f, {x, xmin, xmax}] - computes definite integral from xmin to xmax
e.g. Integrate[Log[x], {x, 1,6}]

ListPlot[list] - plots list versus integers - e.g. ListPlot[{2,4,3,5,4}]

ListPlot[list1, list2] - plots list1 versus list2 - e.g. ListPlot[{1,2,5},{2,1,0.5}]

N[f] - gives a numerical value for an expression - e.g. N[Pi] gives 3.14159

NestList[f, object, n] - applies f to object n times and makes a list
e.g. Nestlist[x^2, 2, 3] gives {4,16,256}

Plot[f,{x,xmin,xmax}] - plots f versus x on the interval [xmin,xmax]
e.g. Plot[x^2, {x,0,2}] NOTE: Plot has lots of options e.g. AxesLabel, Grid, AxesOrigin, etc. See the manual for a complete list and usage
eg Plot[x^2, {x,0,2}, Frame->True] puts a nice frame around your graph.

Show[graphics, options] - displays graphic objects using options e.g. Show[popplot1, PlotJoined->True]

Simplify[expr]- does its best to simplify expr e.g. Simplify[Out[42]] where Out[42] is something very ugly

Solve[eqns, vars] - tries to solve one or a system equations for the vars specified (SYMBOLICALLY)- e.g. Solve[{x+y ==1, x-y ==4}, {x,y}]

NSolve[eqns, vars] - does the same thing as Solve, but does it NUMERICALLY (See also FindRoot)

Sum[f, {i, imin, imax}] - sums f from imin to imax i.e. f[1] + f[2] + f[3] + ... (only really interesting if f depends on i) - e.g. Sum[i, {i, 1,4}] gives 10.

Table[f, {i, imin, imax}] - makes a table in list format of the function f from imin to imax - e.g. Table[i, {i, 1,4}] gives {1,2,3,4}.

Libraries

Mathematica has several libraries or packages that it does not load automatically. For instance, **SymbolicSum** is a list of summations that can be solved symbolically and **Graphics** allows you to make log plots.

The *Mathematica* book tells you how to load these libraries. Using the "Function Browser" is an easy way.