Collected *Mathematica*® commands (by topic)

Constants:

- `E` – The base of the natural logs, or exponential, $e = 2.71..$
- `Pi` – The most famous irrational number, $\pi = 3.14156...$
- `I` – The square root of -1, $i = \sqrt{-1}$ (and not j the way engineers label it!)
- `Infinity` – ∞
- `Degree` – $\pi/180$ which can be used as the *degrees to radians* conversion factor.

Operations:

- `+` – addition
- `-` – subtraction
- `*` – multiplication
- `\` – division
- `^` – exponentiation, as in `2^3`.
- `Sqrt[ ]` – square root
- `[n]Sqrt[ ]` – n-th root
- `!` – factorial, as in `5!`.

Functions:

- `Exp[z]` – exponential
- `Log[z]` – natural log, log base e
- `Log[b,z]` – log of base b
- `Sin[z]` – sin (in radians)
- `Cos[z]` – cos (in radians)
- `Tan[z]` – tan (in radians)
- `Csc[z]` – csc (in radians)
- `Sec[z]` – sec (in radians)
- `Cot[z]` – cot (in radians)

- `ArcSin[z]` – inverse trig function (and similarly for all of the rest)
- `Sinh[z]` – sinh (hyperbolic)
- `Cosh[z]` – cosh (hyperbolic, and so forth)

Defining new functions:

- `f[x_] := expr` – defines the function `f[x]` to be given by whatever `expr` is. (Note the `:=` type equality sign and the underscore of the argument(s).
- `f[x_,y_,..] := expr` – defines a multivariable function.
- `f[x_] = expr` is sometimes the right way to do it, if `expr` has already been defined in another way. We'll see examples of this.

Special functions:

- `AiryAi[z]` – $Ai(z)$ (the well-behaved one)
- `AiryBi[z]` – $Bi(z)$ (the poorly behaved one)
- `BesselJ[n,z]`– Bessel function $J_n(z)$ (the well-behaved ones)
- `BesselY[n,z]`– Bessel function $Y_n(z)$ (the poorly-behaved ones)
- `Gamma[z]` – Euler gamma function $\Gamma(z)$.
- `LegendreP[n,x]` – Legendre polynomials, $P_n(x)$.
- `LegendreP[n,m,x]` – Associate Legendre polynomials, $P_n^m(x)$.
- `SphericalHarmonic[l,m,theta,phi]` – spherical harmonics, the $Y_{l,m}(\theta, \phi)$.
- `HermiteH[n,x]` – Hermite polynomials, $H_n(x)$.
- `LaguerreL[n,x]` – Laguerre polynomials, $L_n(x)$.
- `LaguerreL[n,a,x]` – generalized Laguerre polynomials, $L_n^a(x)$.
- `Beta[a,b]` – The Euler Beta function, $B(a, b)$.
- `Erf[z]` – The *error function*.
- `EllipticK[m]` – Complete elliptic integral of the first kind.

Plotting commands:

- `Plot[f,{x,xmin,xmax}]` – Plots the function $f(x)$ over the limits shown.
- `Plot[f,{x,xmin,xmax},PlotRange->All]` – Plots the function $f(x)$ over the limits shown, showing all y values.

- `Plot[f,{x,xmin,xmax},PlotRange->{ymin,ymax}]` – Plots the function $f(x)$ over the limits shown, showing y values in the range shown.
- `Plot[f,{x,xmin,xmax},PlotStyle->Dashing[{0.05}]]` – Plots with dashed lines with the same length dashes and spaces between the dashes.
- `Plot[f,{x,xmin,xmax},PlotStyle->Dashing[{0.05,0.1}]]` – Same as above, but with different length dashes and spaces (second length is the space)
- `Plot[f,{x,xmin,xmax},PlotStyle->Dashing[{0.01,0.05,0.05,0.05}]]` – Same as above but with alternating lines and spaces...this gives a dot-dash line.
- `Plot3D[f,{x,xmin,xmax},{y,ymin,ymax}]` – Plots $f(x, y)$ over the x, y ranges shown.
- `Show[g1,g2,...]` – shows the graphic elements `g1,g2,...` all on the same plot (with the same horizontal and vertical ranges.)
- `Show[GraphicsArray[{plot1,plot2,...}]]` – shows `plot1,plot2,...` side-by-side.
- `Show[GraphicsArray[{{plot1,plot2},{plot3,plot4}}]]` – shows a 2×2 array of plots. This can be easily generalized to show an $n \times m$ array.
- `ParametricPlot[{fx,fy},{t,tmin,tmax}]` – makes a parametric plot of $(fx(t), fy(t))$ in 2D space, as a function of time t over the range indicated.
- `ParametricPlot3D[{fx,fy,fz},{t,tmin,tmax}]` – does the same thing, but in 3D.

Programming/line entry commands:

- `%` – Show (or otherwise use) the line directly above
- `%%` – Show (or otherwise use) two lines ago (and so forth)
- `;` – Namely a semi-colon. If you put this at the end of a line, it suppresses output. For example, if you entered `Expand[(x+y)^(10)]`, you'd get a long line of output, but if you wrote `Expand[(x+y)^(10)];` it would still do the `Expand`, but it would not display the output. You could then type `%` to see the line above and it would show it.
- `(* anything *)` – Comment line for programs, `anything` will be not part of the program, and so can be used for comments. Note that it needs both the left and right parentheses **and** the stars.
- `Quit` – ends the Mathematica session (in some versions.)

Symbolic manipulations:

- `Integrate[f[x],x]` – integrates the function `f[x]` w.r.t. `x` .
- `Differentiate[f[x],x]` – differentiates the function `f[x]` w.r.t. `x` .
- `Differentiate[f[x],{x,n}]` – differentiates the function `f[x]` w.r.t. `x` n times.

- `D[f,x]` – partial derivative of `f[x]` w.r.t. `x` .
- `D[f,{x,y,...}]` – partial derivative of `f[x]` w.r.t. `x,y,z` .
- `Sum[f,{i,imin,imax}]` – does the summation $\sum_{i=imin}^{imax} f$.
- `Sum[f,{i,imin,imax,di}]` – does the summation $\sum_{i=imin}^{imax} f$ but in increments of `di` . Note that `imin` and `imax` don't have to be integers.
- `Product[f,{i,imin,imax}]` – does the product $\Pi_{i=imin}^{i=imax} f$.
- `Series[f,{x,x0,n}]` – Does a power series expansion of $f(x)$ about the point $x0$ to order n.
- `Limit[f,x->x0]` – does the limit $\lim_{x \to x0} f$.

Algebraic manipulations:

- `Expand[poly]` – Expand out products and powers.
- `Factor[poly]` – Factor polynomials.
- `Collect[poly,x]` – arrange a polynomial as a sum of powers of `x` .
- `Collect[poly,{x,y,...}]` – arrange a polynomial as a sum of powers of `x,y,...` .
- `Length[poly]` – total number of terms in `poly` .
- `Exponent[poly,x]` – maximum exponent of `x` which appears in `poly` .
- `ExpandNumerator[expr]` – expand numerator of rational expression.
- `ExpandDenominator[expr]` – expand denominator of rational expression.
- `Together[expr]` – combine terms over a common denominator.
- `TrigReduce[expr]` – can often simplify combinations of trig functions (for example, implementing identities such as $\cos^2(\theta) + \sin^2(\theta) = 1$.)

Solving equations:

- `Solve[lhs == rhs, x]` – solves an equation for x.
- `Solve[{lhs1 == rhs1, lhs2 == rhs2, ...},{x,y,z}]` – solves a set of simultaneous equations for $x, y, ...$
- `DSolve[eqns,y[x],x]` – solves a differential equation for `y[x]` with `x` as the independent variable.

Numerical operations:

- `N[expr]` – numerical values of `expr`

- `N[expr,20]` – evaluates `expr` to 20 digit accuracy.
- `expr //N` – also evaluates `expr` numerically.
- `NIntegrate[f,{x,xmin,xmax}]` – numerical approximation to $\int_{xmin}^{xmax} f(x)\,dx$.
- `NSum[f,{i,imin,Infinity}]` – numerical approximation to $\sum_{imin}^{\infty} f(i)$.
- `Fit[data,funs,vars]` – fits the list of `data` with the functions `funs` over the variables `vars` . An example of a fit to a quadratic in x would be `Fit[data,{1,x,x^2},x]` .
- `Exp[Fit[Log[data],{1,x},x]]` – fits to an exponential of the form e^{a+bx}.
- `FindRoot[lhs == rhs, {x,x0}]` – looks for a solution of the equation `lhs = rhs` starting with $x \sim x0$.
- `FindRoot[lhs == rhs, {x,{x0,x1}}]` – looks for a solution of the equation `lhs = rhs` starting with $x \sim x0, x1$ as initial guesses.
- `NDSolve[{eqn1, eqn2,...},y,{x,xmin,xmax}]` – finds numerical solutions of `y[x]` as a function of `x` in the range shown using the various differential equations labeled `eqns1,eqns2,...` . We'll use this a lot.

Vectors, matrices, and arrays:

- `{a,b,c}` – 3D vector with elements (a, b, c). Easily generalized to any dimension.
- `{{a,b},{c,d}}` – 2×2 matrix of the form

  $$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

  Also easily generalized to any dimension.
- `MatrixForm[m]` – Writes the matrix `m` in the form as shown above.
- `Table[f,{i,n}]` – builds a vector of length n by evaluating f with $i = 1,..,i = n$.
- `Length[list]` – gives length of vector array (number of elements)
- `Array[a,{m,n}]` – builds an $m \times n$ matrix with (i, j) element given by $a(i, j)$.
- `Flatten[list]` – Removes **ALL** curly brackets from embedded lists.
- `Flatten[list,1]` – Removes the outermost set of curly brackets from embedded lists.
- `DiagonalMatrix[list]` – diagonal matrix (zeros for $i \neq j$) with the values in `list` used in the (i, i) places.
- `IdentityMatrix[n]` – gives the $n \times n$ unit matrix.
- `v[[i]]` – gives the i-th element of the vector `v` .

- `m[[i]][[j]]` – gives the (i, j)-th element of the matrix `m` .
- `c*m` – multiply the matrix `m` by the scalar `c`.
- `a.b` – multiplies two matrices, or two vectors. Recall that the dimensions of the two entries must match appropriately.
- `m.a` – multiplies a vector by a matrix (or vice versa).
- `Det[m]` – determinant of a matrix.
- `Eigenvalues[m]` – eigenvalues of a square matrix.
- `Eigenvectors[m]` – eigenvectors of a square matrix.
- `Inverse[m]` – gives the inverse of a square matrix `m` .
- `Transpose[m]` – gives transpose of matrix `m` .
- `Sum[m[[i,j]],{i,Length[m]}]` – gives the trace (sum of diagonal elements) of matrix `m` .
- `Sort[list]` – Sorts a 1D list, smallest to largest.
- `AppendTo[list,elem]` – adds `elem` to the list `list` and resets `list` to the resulting larger list.
- `Part[expr,n]` – Gives various pieces of `expr`.

Relational and logical operators:

- To check if two numbers or expressions are related (equal to, greater to, less than, etc.) one uses the following relational operators:
- `x==y` – Is $x = y$?
- `x>y` – Is $x > y$?
- `x>=y` – Is $x \geq y$?
- `x<y` – Is $x < y$?
- `x<=y` – Is $x \leq y$?
- These return values of `True` or `False` which can then be used in `If` and similar structures.
- `p && q && k` – p **and** q **and** k.
- `If[p, then, else]` – gives `then` if p is True, and `else` is False. If you don't want anything to happen if p is False, then you can put nothing in the last spot.

- `Which[test1,value1,test2,value2,...]` allows for testing of multiple possibilities, like an extended `If[ ]`.
- `Do[expr,{i,imin,imax}]` – Evaluates `expr` from the lower to upper values of `i` .
- `While[test,body]` – Evaluates `test`, then evaluates `body`, over and over, until `test` is no longer True.

Other input/output commands:

- `Save[‘‘file_name’’,f,g,...]` – saves the definitions of the variables `f,g,...` .
- `<<file.name` – read in a plain text file with *Mathematica*® commands
- The best way of importing files in a modern notebook version of Mathematica is with the Import command from the pull-down menu.
- The best way of saving files is in a notebook is using the various Save options from the pull-down menus.

Animation package:

- `<<Graphics‘Animation‘` – loads animation packages
- `Needs[‘‘Graphics‘Animation’’]` does the same thing but a bit more cleanly.
- `Animate[plot,{t,tmin,tmax}]` – generates `plot` at the various values of t and shows resulting animation.
- `ShowAnimation[{g1,g2,,...]` – produces animation of string of graphics objects.

Statistical analysis package:

- `<<Statistics‘DescriptiveStatistics‘` – loads statistical analysis package.
- `Needs[‘‘Statistics‘DescriptiveStatistics’’]` does the same thing.
- `Mean[data]` – finds mean value.
- `StandardDeviation[data]` – finds standard deviation.

Getting help:

- `?Command` – Gives a short description of what `Command` does.
- `??Command` – Gives a more complete description of what `Command` does.
- `Options[Command]` – Lists the default options for `Command`.
- And most importantly, you can use the on-line help incorporated into Mathematica.

Random stuff:

- If you want to turn ’text commands’ into more Math-typeset type format, highlight the line of code you want changed and press *shift+cntl+t* all at once.