Mathematica Quick Reference

Run Mathematica

`mathematica` (evoke Mathematica frontend) `Quit[]` or `Exit[]` (exit Mathematica)

Arithmetic

`+`, `-`, `*` (or a space ␣), `/`, `^` (power), e.g. `a*x^2 + 2 x - 3/5` ($ax^2 + 2x - 3/5$) *n*`^^`*digits* base *n* number

Mathematical Functions

`Sqrt[`x`]`	($\sqrt{x}$)	`Exp[`x`]` or `E^`x	(e^x)	`Log[`x`]`	($\ln x$)	`Log[`b`,`x`]`	($\log_b x$)
`Sin[`x`]`	($\sin x$)	`Cos[`x`]`	($\cos x$)	`Tan[`x`]`	($\tan x$)	`ArcSin[`x`]`	($\sin^{-1} x$)
`ArcCos[`x`]`	($\cos^{-1} x$)	`ArcTan[`x`]`	($\tan^{-1} x$)	`Sinh[`x`]`	($\sinh x$)	`Cosh[`x`]`	($\cosh x$)
`Tanh[`x`]`	($\tanh x$)	`ArcSinh[`x`]`	($\sinh^{-1} x$)	n`!`	(factorial)	`Abs[`x`]`	($\lvert x \rvert$)
`Mod[`n`,`m`]`	($n \bmod m$)	`Max[`x`,`y`,...]`	(max)	`Min[`x`,`y`,...]`	(min)	`Binomial[`n`,`m`]`	(C_m^n)
`Re[`z`]`	(real part)	`Im[`z`]`	(imaginery part)	`Conjugate[`z`]`	($\bar{z}$)	`Arg[`z`]`	(argument)
`LegendreP[`n`,`x`]`	($P_n(x)$)	`HermiteH[`n`,`x`]`	($H_n(x)$)	`LaguerreL[`n`,`a`,`x`]`	($L_n^a(x)$)	`BesselJ[`n`,`z`]`	($J_n(z)$)
`Erf[`x`]`	($\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2} dt$)	`Gamma[`z`]`	(Euler $\Gamma(z)$)	`Zeta[`s`]`	(Riemann $\zeta(s)$)	`Random[]`	(number)

Mathematical Constants

`Pi`	($\pi \approx 3.14159$)	`E`	($e \approx 2.71828$)	`I`	($i = \sqrt{-1}$)	`Infinity`	(∞)

Data Objects

`123` (integer) `3/7` (rational) `1.0` (real) `2 + 8I` (complex) `"`*text*`"` (string)

`{`a`,` b`,` ... `}` (list with elements a, b, ...) h`[`a`,` b`,` ...`]` (expression/w head h, elements a, b, ...)

expr`[[`i`]]` (i-th element of the expression *expr*) a`[`i`]` (indexed object)

Arbitrary-Precision Calculation

expr`//N` or `N[`*expr*`]` (numerical value of *expr*) `N[`*expr*`,` n`]` (value of *expr* with n-digit precision)

`Rationalize[`x`]` (rational number approximation) `Precision[`x`]` (significant decimal digits in x)

Defining Variables, Functions, and Rules

x `=` *value*	(assign a value to symbol x)	*expr* `/.` x `->` a	(replace x by a in *expr*)
x `:=` *expr*	(delayed assignment)	*expr* `//.` x `->` a	(replace repeatedly)
x `= .` or `Clear[`x`]`	(remove value assigned to x)	*lhs* `:>` *rhs* `/;` *test*	(apply rule if *test* is `True`)
f`[`x`_] :=` *expr*	(define a function $f(x)$)	*lhs* `:=` *rhs* `/;` *test*	(assign if *test* is `True`)

Algebraic Calculations

`Expand[`*expr*`]`	(multiply out products)	`Factor[`*expr*`]`	(reduce to a product of factors)
`Together[`*expr*`]`	(common denominator)	`Cancel[`*expr*`]`	(cancel common factors)
`Expand[`*expr*`,Trig->True]`	($\sin^2 x \to \sin 2x$, etc.)	`Simplify[`*expr*`]`	(find simplist form)
`Factor[`*expr*`,Trig->True]`	($\sin 2x \to \sin^2 x$, etc.)	`Apart[`*expr*`]`	(write *expr* as a sum of terms)
`Coefficient[`*expr*`,`x`]`	(coefficient of x in *expr*)	`Exponent[`*expr*`,`x`]`	(max power of x in *expr*)
`Solve[`*lhs*`==`*rhs*`,` x`]`	(solve algebraic equation for x)	`DSolve[`*eqn*`,`y`[`x`],`x`]`	(solve differential equation)
`Reduce[`*eqn*`,` x`]`	(reduce equations)	`Eliminate[`*eqn*`,` e`]`	(eliminate variable e)

Linear Algebra

`{`a`,` b`,` c `}`	(vector)	`{{`a`,` b`}, {`c`,` d`}}`	(2×2 matrix)	n`.`m	(matrix multiply)
`Inverse[`m`]`	(inverse of matrix)	`Transpose[`m`]`	(transpose)	`Det[`m`]`	(determinant)
`MatrixPower[`m`,` n`]`	(m^n)	`MatrixExp[`m`]`	(e^m)	`LinearSolve[`m`,` b`]`	(solve $mx = b$)
`Eigenvalues[`m`]`	(eigenvalues)	`Eigenvectors[`m`]`	(eigenvectors)	`Eigensystem[`m`]`	(value & vector)

Calculus

`D[`f`,` x`]`	($\partial f/\partial x$)	`Integrate[`f`,` x`]`	($\int f\,dx$)	`Series[`f`,{`x`,`a`,`n`}]`	(expand at a)
`D[`f`, {`x`,` n`}]`	($\partial^n f/\partial x^n$)	`Integrate[`f`, {`x`,` a`,` b`}]`	($\int_a^b f\,dx$)	`Limit[`f`,` x`->`a`]`	($\lim_{x\to a} f$)
`Dt[`f`]`	(df)	`Sum[`f`, {`i`,` m`,` n`}]`	($\sum_{i=m}^{n} f$)	`Product[`f`, {`i`,` m`,` n`}]`	($\prod_{i=n}^{m} f$)

`NIntegrate, NSum, NProduct, NSolve, NDSolve, FindRoot` (numerical integration, summation, etc.)

Graphics

`Plot[`f`,` `{`x`,` a`,` b`},` *option*`->`*value*`]` (plot f as a function of x from a to b)

`Show[`*plot*$_1$`,` *plot*$_2$`,` ...`]` (redraw plots)

`Plot3D[`f`, {`x`,` a`,` b`}, {`y`,` c`,` d`}]` (three-dimensional plot of f as a function of x and y)

`ListPlot[{{`x_1`,` y_1`},{`x_2`,` y_2`},` ... `}]` (plot points (x_1, y_1) ...)

`ParametricPlot[{`x`,` y`}, {`t`,` a`,` b`}]` (plot curve $\big(x(t),\, y(t)\big)$)

Plot options:

`AspectRatio` (height-to-width ratio) `AxesLabel->{"`x`","`y`"}` (add labels) `Frame -> True` (draw frame)

Programming

`Module[{`a`, `b`, ... }, `$expr_1$`; `$expr_2$`; ... ]`	(a procedure/w local variables a, b, ..., return value of last $expr$)
`Table[`$expr$`,{`i`, `max`}]`	(make a list of values of $expr$ with i from 1 to max)
`Do[`$expr$`,{`i`, `min`, `max`, `di`}]`	(evaluate $expr$ with i run from min to max in steps of di)
`While[`$test$`, `$body$`]`	(evaluate $body$ repetitively, so long as $test$ is `True`)
`For[`$start$`, `$test$`, `inc`, `$body$`]`	(evaluate $start$, then repetitively evaluate $body$ and inc, until $test$ fails)
`If[`$test$`, `$then$`, `$else$`]`	(evaluate $then$ if $test$ is `True`, and $else$ if it is `False`)
`Which[`$test_1$`, `$value_1$`, `$test_2$`, ... ]`	(give the value associated with the first $test$ that is `True`)
`Switch[`$expr$`, `$form_1$`, `$value_1$`, `$form_2$`, ... ]`	(give the value associated with first $form$ matching $expr$)
`Function[`x`, `$body$`]` or $body$ `&`	(specify a pure function)
`Nest[`f`, `x`, `n`]`	(apply the function f nested n times to x)
`NestList[`f`, `x`, `n`]`	(generate list $\{x, f(x), f(f(x)), \ldots\}$ nested up to n deep)
`Fold[`f`, `x`, {`a`, `b`, `c`}]`	(produce $f\big(f(f(x,a),b),c\big)$)
`FixedPoint[`f`, `x`]`	(apply the function f repeated until the result no longer changes)
`Apply[`f`, {`a`, `b`, `c`}]` or `f @@` $expr$	(produce $f(a,b,c)$)
`Map[`f`, {`a`, `b`, `c`}]` or `f /@` $expr$	(apply f to each elements, $\{f(a), f(b), f(c)\}$)

i`++`	(post-increment)	i`--`	(post-decrement)	i `+=` di	(add di to i)	x `*=` c	(multiply x by c)
`++`i	(pre-increment)	`--`i	(pre-decrement)	i `-=` di	(subtract di)	x `/=` c	(divide x by c)
`(*` $text$ `*)`	(comment)	f`::usage="`$text$`"`	(info)	`Timing[`$expr$`]`	(time)	`MemoryInUse[]`	(space)

Iterators in `Do`, `Table`, `Sum`, etc.

`{`max`}`	(iterate max times)	`{`i`, `max`}`	(i from 1 to max in steps of 1)
`{`i`, `min`, `max`}`	(i from min to max in steps of 1)	`{`i`, `min`, `max`, `di`}`	(i from min to max in steps of di)

Logical Operators

`==`	(equal)	`!=`	(unequal)	`===`	(identical)	`=!=`	(not identical)
`<`	(less than)	`<=`	(less or equal)	p `||` q	(or)	p `&&` q	(and)
`>`	(greater than)	`>=`	(greater or equal)	`!`p	(not)	`Xor[`p`,`q`]`	(exclusive or)

List Manipulation

`Part[`t`, `i`]` or t`[[`i`]]`	(i-th sublist)	`Position[`t`, `$form$`]`	(the position $form$ occur)
`Take[`t`, `n`]`	(first n elements)	`Last[`t`]`	(last element in t)
`Drop[`t`, `n`]`	(drop first n elements)	`First[`t`]`	(first element in t)
`Count[`t`, `$form$`]`	(number of times $form$ occur)	`MemberQ[`t`, `$form$`]`	(test whether $form$ is in t)
`Prepend[`t`, `e`]`	(add e at the beginning)	`Insert[`t`, `e`, `i`]`	(insert e at position i)
`Append[`t`, `e`]`	(add e at the end)	`Delete[`t`, `i`]`	(delete element at position i)
`ReplacePart[`t`, `e`, `i`]`	(replace with e at position i)	`Join[`t_1`, `t_2`, ...]`	(concatenate lists together)
`Union[`t_1`, `t_2`, ...]`	(union of lists)	`Intersection[`t_1`, `t_2`, ...]`	(common to all lists)
`Sort[`t`]`	(sort elements in standard order)	`Reverse[`t`]`	(reverse the order of elements)
`RotateLeft[`t`, `n`]`	(rotate n places to the left)	`RotateRight[`t`, `n`]`	(rotate n places to the right)

Input/Output

`<<`$file$	(read expressions from $file$, return last $expr$)	`Save["`$file$`", `x`]`	(save the definition of x to $file$)
$expr$`>>`$file$	(write $expr$ to a $file$)	`Display["!psfix>`$file$`",`$graph$`]`	(save as PS file)
$expr$`>>>`$file$	(append $expr$ to a $file$)	`ReadList["`$file$`",`$type$`]`	(read objects of a given $type$)
`!!`$file$	(display the content of a $file$)	`PSPrint[`$graph$`]`	(print a hardcopy of graphics)
`<<Calculus‘VectorAnalysis‘`	(load a package)	`!`$command$	(issue a UNIX command)

Patterns

`_`	(any $expr$)	`x_`	(any $expr$ named x)	`x:`$pattern$	(match pattern)
`x_`h	(pattern with head h)	$pattern$ `/;` $condition$	(conditional)	$pattern$`?`$test$	(if $test$ is `True`)
`x__`	(sequence of $expr$)	`x___`	(zero or more $expr$)	`x_:`v	($expr$ with default)

Expression in Different Formats

`FullForm[`e`]`	(full form)	`CForm[`e`]`	(C codes)	`FortranForm[`e`]`	(fortran)	`StandardForm[`e`]`	(math)
`InputForm[`e`]`	(input)	`OutputForm[`e`]`	(out)	`MatrixForm[`e`]`	(matrix)	`TeXForm[`e`]`	(TEX)

Input Editing and Help

`%`	(last result)	`?`$Name$	(help on $Name$)	`*`	(represent 0 or more characters)
`%`n	(result on `Out[`n`]`)	`??`$Name$	(more help on $Name$)	`@`	(1 or more lower-case letters)
`Shift-Enter`	(evaluate $expr$)	`Ctrl-C`	(interrupt execution)	`!`	(shell escape)