

# Mathematica commands summary (cheat sheet)

Hugo Touchette

National Institute for Theoretical Physics (NITheP), Stellenbosch, South Africa

Last updated: May 18, 2016

See last page for *Mathematica* 8 updates

Abbreviations	
approx	Approximation
arg	Argument
char	Character
cnt	Counter
cond	Condition
dist	Distribution
elem	Element
eqn	Equation
eval	Evaluation
expr	Expression
fct	Function
img	Image
patt	Pattern
rand	Random
str	String
val	Value
var	Variable

## Equalities, assignments [DC]

=	Assign value	
:=	Delayed eval	Also used for repeated eval
==	Equality	
=.	Clear value	Same as Clear
===	Boolean test	Same as SameQ
f[x_]:=f[x]=...	Remember computed vals	

## Rules and replacements [DC, Tut]

patt->expr	Transformation rule	Can be assigned to names
patt:->expr	Delayed rule	Same as :>
expr/.rule	Replacement	Same as ReplaceAll
	$1+2x/.x->3$	Output: 7
expr/.{rule1,rule2}	Multiple replacements	
expr//.rule	Replace until no changes	Same as ReplaceRepeated

## Applications, functions and maps [DC, Tut]

f@expr	Prefix form f[expr]	
expr//f	Postfix form f[expr]	
x <sup>f</sup> y	Infix form f[x,y]	
f@expr	Replaces the head of expr by f	Same as Apply[f,expr]
	Plus@{1,2}	Output: 1 + 2 = 3
	Similar to: f[x]/.f->g	Output: g[x]
f/@list	Map f on list	Same as Map[f,list]
f/@expr	Map f on parts of expr	
	Map[f,a+b]	Output: f[a]+f[b]
Function[x,body]	Function with arg x	Same as pure functions
	Function[x,x^2][n]	Output: n <sup>2</sup>
	Map[Function[x,x^2],a+b]	Output: a <sup>2</sup> + b <sup>2</sup>
Function[{vars},...]	Many args	
body&	Pure (anonymous) function	
	(#^2&)[5]	Output: 25
	Map[#^2&,a+b]	Output: a <sup>2</sup> + b <sup>2</sup>
	(1+g[#])&/@{1,2}	Output: {1+g[1],1+g[2]}
#n	nth arg in pure function	
##	Seq of all args	

## Formatting [DC]

FullForm	Display full head form
MatrixForm	Display in matrix form
Column	Display in column form
TableForm	Display in tabular form
TreeForm	Display in tree structure

## Shortcuts [DC]

<b>Ctrl</b> -@	$\sqrt{\quad}$
<b>Ctrl</b> -/	Fraction
<b>Ctrl</b> -^	Superscript
<b>Ctrl</b> -_	Subscript
<b>Ctrl</b> -Enter	New row
<b>Ctrl</b> -,	New column

## Patterns [DC, Tut]

-	Blank: any expr	
--	Double blank: any seq of expr	
---	Triple blank: zero or more args	
x_	Any expr named x	
x_h	Expr named x with head h	
x	Exact matching x	
	$x+2/.x->2$	Output: 4
x:patt	Named expr matching patt	
	$f[a^b]/.f[x:_^_]->p[x]$	Output: p[a <sup>b</sup> ]
x_:v	Optional arg with default val v	
x_h:v	Typed arg with optional val v	
f[n_]	f with any named arg n	
f[n_,n_]	f with identical args	
x^n_	x to any named power n	
x_^n_	Any expr to any power	
a.+b_	Sum of two named exprs	
{a1_,a2_}	List of two named exprs	
patt;/cond	Pattern with cond	Read: patt such that cond
	$(x./;NumberQ[x]&&Im[x]==0)$	Any real number
rule;/cond	Rule with cond	
patt?test	Pattern matching boolean test	
	$f[x_?NumberQ]:=x+2$	
(patt1 patt2 ...)	patt1 or patt2 or ...	
	$\{1,x,x^2,y^3\}/.(x x^n)->q$	Output: {1,q,q,y <sup>3</sup> }
expr/.rule	Replacement in expr with rule	
	$x+2/.x->2$	Output: 4
	$f[a]+f[b]/.f[x_]->x^2$	Output: a <sup>2</sup> +b <sup>2</sup>
	Position[f[a],g[b],f[b],f[x_]]	Output: {{1},{2}}
	$\{1,x,x^2,x^3\}/.x^n->r[n]$	Output: {1,x,r[2],r[3]}
Except[c]	Anything except c	
Except[c,patt]	Pattern patt except c	
	Cases[{1,0,2},Except[0]]	Output: {1,2}
	Cases[{1,x,0},Except[0,_Integer]]	Output: {1}

## Type specification

Integer	N	<b>Esc</b> -dsN- <b>Esc</b>
Real	R	<b>Esc</b> -dsR- <b>Esc</b>
Complex	C	<b>Esc</b> -dsC- <b>Esc</b>
Rational	Z	<b>Esc</b> -dsZ- <b>Esc</b>
List	List type or head	
Symbol	Any symbol	
String	String type of head	
_Head	Type Head	
		MatchQ[x <sup>2</sup> ,_Power]

## Files [DC]

Import["file"]	Import file or url
Export["file",expr]	Export expr in file
ImageResolution->pts	Image resolution in pts
ReadList["file"]	Returns list from file
<<File'	Load content of file

Lists [DC, Tut]			Boolean functions [DC]		Programming [DC, Tut1, Tut2]	
{i,imax}	Simple counter from 1 to imax		SameQ[x,y]	Same? (x===y)	Do[expr,cnt]	Do loop
{i,imin,imax,di}	Counter with step di		OrderedQ[list]	list ordered?	While[test,body]	While loop
Table[expr,cnt]	Generate list	Multiple counters possible	IntegerQ[x]	x integer?	If[cond,t,f]	If statement
list[[i]]	i <sup>th</sup> elem (list or expr)	Same as Part[list,i] or list[[i]]	MemberQ[list,patt]	Elms match patt?	Return[expr]	Output expr and stop
list[[i,j]]	(i,j) elem	Same as list[[i]][[j]]	MatchQ[expr,patt]	expr matches patt?	Module[{x,...},expr]	Proc with local vars (vars localization)
list[[i;;j]]	Range i to j	Same as Part[list,i;;j]	ValueQ[expr]	expr has value?	Block[{x,...},expr]	Eval with global vars
Drop[list,n]	First n elems dropped	Last if -n	AtomQ[expr]	expr is atomic?	With[{x=x0,...},expr]	Local constants
Drop[list,{n}]	n <sup>th</sup> elem dropped		FreeQ[expr,patt]	expr free of patt?	Trace[expr]	Trace eval of expr
Extract[expr,list]	Extract in expr at pos list		EvenQ[n]	n even integer?	Compile[{vars},expr]	Compiled expr
Array[f,n]	{f(1),f(2),...,f(n)}		OddQ[n]	n odd integer?	i++	Incrementation
	<b>Array[1+#^2&amp;,2]</b>	<b>Output: {2,5}</b>	PrimeQ[n]	n prime integer?	i--	Decrementation
Range[n]	{1,2,3,...,n}	Same as Array[#&,n]	NumberQ[expr]	expr a number?	i+=di	Increment by di
Range[nmin,nmax,dn]	Full syntax		NumericQ[expr]	expr numerical?	i-=di	Decrement by di
Position[expr,patt]	List matching positions		PolynomialQ[expr,var]	expr a polynomial?	i*=c	Multiply by c and override val
Count[list,patt]	Number of matches		VectorQ[expr]	expr 1D array (list)?	i/=c	Divide by c and override val
Min[list], Max[list]	Min, max of list	Same as: Min[i,j,...]	MatrixQ[expr]	expr 2D array (list)?		
Append[expr,elem]	expr with elem added at end		DigitQ[str]	All digits in str?		
AppendTo[expr,elem]	Append and update expr	Same as expr=Append[expr,elem]	LetterQ[str]	All letters in str?		
Prepend[expr,elem]	Inset at beginning	See also: PrependTo				
Insert[list,elem,n]	Insert elem in list at pos n		<b>Strings [DC, Tut]</b>			
Delete[expr,elem,n]	Delete elem in list at pos n		"chars"	String		
ReplacePart[expr,i->new]	Replaces expr[[i]] by new		ToString[expr]	Transforms expr to string		
ReplaceList[expr,rule]	Replace according to rule		StringJoin[str1,str2,...]	Join strings	Same as <>	
Flatten[list]	Flattens nested lists		StringLength[str]	Number of chars in str		
Flatten[list,n]	Flattens to level n		StringTake[str,n]	n first chars of str	Last if -n	
Take[list,n]	Take first n elem		StringDrop[str,n]	First n chars dropped	Last if -n	
	<b>Take[#,1]&amp;/@{{1,2},{3,1}}</b>	<b>Output: {{1},{3}}</b>	StringInsert[str1,str2,n]	Insert str2 at pos n		
Cases[list,patt]	Elms of list matching patt		StringPosition[str,substr]	Start and end of substr in str		
	<b>Cases[{2,x^2},x^n]</b>	<b>Output: x^2</b>	StringCount[str,substr]	Number of occurrences of substr		
Cases[list,patt,n]	First n matching patt		StringReplace[str,rule]	Replace with rule		
Cases[list,rule]	Results for which rule applies		StringMatchQ[str,patt]	patt appears in str?		
Select[list,boolean]	Boolean selection in list		StringFreeQ[str,patt]	str free of patt?		
Position[expr,patt]	Pos where patt appears		StringSplit[str]	Split where whitespace		
Join[l1,l2,...]	Join lists		StringSplit[str,patt]	Split according to patt		
Union[l1,l2,...]	Set union	Same as ∪	Sort	Alphabetical (canonical) sort		
Riffle[{e1,e2,...},x]	{e1,x,e2,x,...}		DateString["elem"]	String version of date		
Intersection[l1,l2,...]	Set intersection	Same as ∩	Characters[str]	List of chars		
Subsets[list]	All subsets of list		ToUpperCase[str]	Capitalize		
DeleteDuplicates	Delete duplicates from list		ToLowerCase[str]	Lower case		
Thread[f[{a,b,c}]]	{f[a],f[b],f[c]}		CharacterRange["char1","char2"]	List from char1 to char2		
Sort[list]	Sorts list		patt..	patt followed by symbols		
Reverse[list]	list in reverse order		Whitespace	White space		
PadLeft[list,n,x]	Add x n times left of list	Pad with 0 if no x	<b>Dynamic objects [DC, Tut2]</b>			
PadRight[list,n,x]	Add x n times right of list		Dynamic[expr]	Dynamically updated expr		
Partition[list,n]	Partition list in n sublists		DynamicModule[{vars},...]	Module with dynamical vars		
Split[list]	Splits list in identical elems		Manipulate[expr,{u,...}]	Interactive widget		
Tuples[list,n]	n-tuples of list					
Permutations[list]	Permutations					
Outer[f,l1,l2]	Outer (cartesian) product					

## Useful functions

Sum[f, {i, imin, imax, di}]	Sum
Product[f, {i, imin, imax, di}]	Product
Mod[m, n]	Remainder of $m/n$
Solve[f[x]==0, x]	Solves polynomial eqn
Expand[p]	Expand polynomial p
ComplexExpand[expr]	Expand expression (takes care of complex conjugates)
Factor[p]	Factorize p
Length[expr]	Number of elems in expr
Numerator[expr]	Numerator of expr
Denominator[expr]	Denominator of expr
Options[cmd]	Display options for cmd
SetOptions[cmd]	Set options for cmd
Evaluate[expr]	Explicit eval
Piecewise[{{val, test}, ...}]	Piecewise definition
InverseFunction[f][x]	Inverse fct
Nest[f, x, n]	$n$ -fold fct composition
NestList[f, x, n]	List of compositions
FixedPoint[f, x]	Iterate until fixed point
FixedPointList[f, x]	Iteration list
NestWhile[f, x, test]	While iteration
NestWhileList	While iteration list
LinearModelFit	Linear model
NonlinearModelFit	Nonlinear model
GraphPlot[m]	Graph (network) for adjacency $m$
GraphData["name"]	Graph with name
FinancialData["tag", "date"]	Financial data from date
CountryData["tag", "prop"]	Country data
WeatherData["tag", "prop"]	Weather data

## Probability and statistics [DC]

RandomReal[{a, b}]	Uniform rand real in $[a, b]$	From $[0, 1]$ if no arg
RandomReal[dist]	Rand real from dist	
RandomInteger[{i, j}]	Uniform integer in $[i, j]$	Uniform 0 or 1 if no arg
RandomInteger[dist]	Rand integer from dist	
RandomSample[list, n]	$n$ samples from list	
RandomSeed[]	Reset seed	
PDF[dist, x]	Density fct of dist	
CDF[dist, x]	Cumulant density fct of dist	
Histogram[list, w]	Histogram (bin width $w$ )	Option: "Probability" "ProbabilityDensity"
BinCounts[list, w]	Counting dist of list (bin width $w$ )	
Mean[list]	Also Mean[dist]	
Variance[list]		
StandardDeviation[list]		
ChiSquareDistribution[ $\nu$ ]		
ExponentialDistribution[ $\lambda$ ]		
NormalDistribution[ $\mu, \sigma$ ]		
BinomialDistribution[n, p]		
BernoulliDistribution[p]		

## Linear algebra [DC]

a.b	Dot or matrix product
Cross[a, b]	Cross product
Norm[a]	Euclidean norm
IdentityMatrix[n]	$n \times n$ identity matrix
Diagonal[m]	Diagonal of $m$
Diagonal[m, k]	$k$ th elem in diagonal of $m$
Dimensions[m]	Dimensions of $m$
Inverse[m]	Inverse square matrix $m$
Det[m]	Determinant square matrix
Tr[m]	Trace matrix or tensor $m$
Transpose[l]	Transpose first two levels
Eigenvalues[m]	Eigenvals square matrix
Eigenvectors	Eigenvecs square matrix

## Parallel computation [DC, Tut]

Parallelize[expr]	Direct parallelization
ParallelEvaluate[expr]	Eval on kernels
DistributeDefinitions	Put def on kernels
ParallelNeeds	Package on kernels
ParallelMap	Parallel Map
ParallelSum	Parallel Sum
ParallelProduct	Parallel Product
ParallelTable	Parallel Table
ParallelDo	Parallel Do
ParallelSubmit	Submit to kernel
	Collect with WaitAll
WaitAll	Results from kernels

## Numerical routines [DC, Tut]

N[expr]	Numerical eval of expr	
SetPrecision[expr, n]	Precision to $n$ digits	\$Pre=SetPrecision[#, n]& Set session precision to $n$ digits
NSolve[eqn, var]	Approx polynomial eqn	
FindRoot[eqn, {x, x0}]	Approx eqn in var x with seed $x_0$	
FindRoot[eqn, {x, x0, a, b}]	Approx eqn in $[a, b]$	
NDSolve[eqn, y, {x, a, b}]	Solve differential eqn	Initial cond specified in eqn
NIntegrate[f, {x, a, b}]	Numerical integration. Methods: "SymbolicProcessing" -> 0 "GlobalAdaptive" "LocalAdaptive" "Oscillatory" "PrincipalValue" "MonteCarlo" "AdaptiveMonteCarlo"	Fully numerical eval Default ID integrals; automatic detection Cauchy principal value Random sampling
FindMinimum[f, x, x0]	Find local min with seed $x_0$	
NMinimize[f, x]	Find global min. Methods: "DifferentialEvolution" "NelderMead" "RandomSearch" "SimulatedAnnealing"	Option: "SearchPoints" Option: "SearchPoints" Option: "SearchPoints"
NumericalCalculus package:		
NLimit[expr, x->x0]	Numerical limit	
ND[expr, x, x0]	Numerical D at $x_0$	
NSeries[f, {x, x0, n}]	Numerical series	

## Other

\$Pre	Pre-processing var
\$Post	Post-processing var

## Some new/updated in Mathematica 8

Free form input	Type = then input
WolframAlpha["query"]	Query to WolframAlpha server
RandomVariate[dist]	Merges RandomReal and RandomInteger
Probability[pred, x $\approx$ dist]	Probability of predicate with distribution dist
NProbability	
Expectation[expr, x $\approx$ dist]	Expected value with distribution dist
NExpectation	
EstimatedDistribution[data, dist]	Fit dist to data
EmpiricalDistribution[data]	Distribution for data
HistogramDistributon	Similar to Histogram
Distributed[x, dist]	x $\approx$ dist. Same as <code>Esc-dist-Esc</code>
Conditioned[expr, cond]	expr conditioned on cond. Same as <code>Esc-cond-Esc</code>
Cumulant[dist, r]	r <sup>th</sup> cumulant of dist
MomentGeneratingFunction[dist, t]	Moment function in var t
StableDistribution[type, $\alpha, \beta, \mu, \sigma$ ]	Stable (Lévy) distribution
TransformedDistribution[trans, dist]	
Graph	
GraphPlot	
GraphStyle	
AdjacencyGraph	
IncidenceGraph	
GridGraph	
RandomGraph	
ButterflyGraph	
HighlightGraph	
AdjacencyMatrix	
VertexDegree	
ClosenessCentrality	
BetweennessCentrality	
PageRankCentrality	
GraphQ	
ConnectedGraphQ	
SmoothHistogram	
SmoothHistogram3D	
DensityHistogram	
SmoothDensityHistogram	
Texture	Use image for plot rendering
TextRecognize	Text recognition from image
Speak	Speak content
SpokenString	
ImageCapture[]	Open camera utility
CurrentImage[]	Returns current image
EdgeDetect[img]	Returns image with edges

## Notes