Maple 9: A Quick Reference

Prepared by:
Douglas Meade
Department of Mathematics
University of South Carolina

March 2004, updated for Maple 9
(Earlier editions for Maple 8, Maple 7, Maple 6, Maple V, Release 5, and Maple V, Release 4)

Symbols and Abbreviations

Symbol	Description	Example
`:=`	assignment	`f := x^2/y^3;`
`;`	terminate command; display result	`int( x^2, x );`
`:`	terminate command; hide result	`int( x^2, x ):`
`..`	specify a range or interval	`plot( t*exp(-2*t), t=0..3 );`
`{ }`	set delimiter (a set is an unordered list)	`{ y, x, y };`
`[ ]`	list delimiter (lists are ordered)	`[ y, x, y ];`
`%`	refers to previous result (percent) *Note*: Was `"` until Maple V, Release 5	`Int( exp(x^2), x=0..1 ):` `% = evalf( % );`
`" "` (see `?strings`)	string delimiter (double quote) *Note*: Changed in Maple V, Release 5 (see `%`)	`plot( sin(10*x) + 3*sin(x), x=0..2*Pi,` `title="An interesting plot" );`
`` ` ` ``(see `?names`)	name delimiter (back quote)	`` `A name` := `This is a name.`; ``
`||` (see also `?cat`)	concatenate string or name *Note*: Was `.` prior to Maple 6	`a||3;` `a||(1..3);`
`´ ´`(see `?uneval`)	delayed evaluation (single quote)	`x := ´x´;`
`->` (see `?->` and `?proc`)	mapping (procedure) definition	`f := (x,y) -> x^2*sin(x-y);` `f(Pi/2,0);`
`@`	composition operator	`(cos@arcsin)(x);`
`@@`	repeated composition operator	`(D@@2)(ln);`

Mathematical Operations, Functions, and Constants

Symbol	Description	Example
`+`, `-`, `*`, `/`, `^`	add, subtract, multiply, divide, power	`3*x^(-4) + x/Pi;`
`sin`, `cos`, `tan`, `cot`, `sec`, `csc`	trigonometric functions	`sin( theta-Pi/5 ) - sec( theta^2 );`
`arcsin`, `arccos`, `arctan`, `arccot`, `arcsec`, `arccsc`	inverse trigonometric functions	`arctan( 2*x );`
`exp`	exponential function	`exp( 2*x );`
`ln`	natural logarithm	`ln( x*y/2 );`
`log10`	common logarithm (base 10)	`log10( 1000 );`
`abs`	absolute value	`abs( (-3)^5 );`
`sqrt`	square root	`sqrt( 24 );`
`!`	factorial	`k!;`
`=`, `<>`, `<`, `<=`, `>`, `>=`	equations and inequalities *Note*: `E` no longer exists; use `exp(1)`	`diff( y(x), x ) + x*y(x) = F(x);` `exp(Pi) > Pi^exp(1);`
`Pi`, `I`	π, i (mathematical constants) *Note*: Maple is case-sensitive	`exp( Pi*I );`
`infinity`	infinity (∞)	`int( x^(-2), x=1..infinity );`

NOTES:

- The document is also available on the World Wide Web in either PDF (http://www.math.sc.edu/ meade/maple/maple-ref.pdf) or PostScript (http://www.math.sc.edu/ meade/maple/maple-ref.ps).
- Please send comments, corrections, and suggestions for improvements to meade@math.sc.edu.

Commands

Command	Description	Example
`restart`	clear all Maple definitions	`restart:`
`with`	load a Maple package	`with( DEtools ); with( plots ):`
`help` (also `?`)	display Maple on-line help	`?DEplot`
`limit`	calculate a limit	`limit( sin(a*x)/x, x=0 );`
`diff`	compute the derivative of an expression	`diff( a*x*exp(b*x^2)*cos(c*y), x )`
`int`	definite or indefinite integration	`int( sqrt(x), x=0..Pi );`
`Limit`	inert (unevaluated) form of `limit`	`Limit( sin(a*x)/x, x=0 );`
`Diff`	inert (unevaluated) form of `diff`	`Diff( a*x*exp(b*x^2)*cos(c*y), x );`
`Int`	inert (unevaluated) form of `int`	`Int( sqrt(x), x=0..Pi );`
`value`	evaluate an inert expression (typically used with `Limit`, `Diff`, or `Int`)	`G := Int( exp(-x^2), x );` `value( G );`
`plot`	create a 2-dimensional plot	`plot( u^3, u=0..1, title="cubic" );`
`plot3d`	create a 3-dimensional plot	`plot3d(sin(x)*cos(y),x=0..4*Pi,y=0..Pi);`

`display`	combine multiple plot structures into a single plot or modify optional settings in a plot (in `plots` package)	`F:=plot( exp(x), x=0..3, style=line );`
		`G:=plot( 1/x, x=0..3, style=point );`
		`plots[display]([F,G], title="2 curves");`

`solve`	solve equations or inequalities	`solve( x^4 - 5*x^2 + 6*x = 2, { x } );`
`fsolve`	solve using floating-point arithmetic	`fsolve( t/10 + t*exp(-2*t) = 1, t );`
`dsolve`	solve ordinary differential equations; see `?dsolve` for a list of available options	`dsolve( diff(y(x),x)-y(x)=1, y(x) );`

`odeplot`	create 2D and 3D plots from solutions obtained by `dsolve` (with `type=numeric`); see `?odeplot` for more options (in `plots` package)	`S:=diff(x(t),t)=-y(t),diff(y(t),t)=x(t):`
		`IC:=x(0)=1,y(0)=1:`
		`P:=dsolve({S,IC}, {x(t),y(t)}, numeric):`
		`odeplot(P, [[t,x(t)],[t,y(t)]], 0..Pi);`
		`odeplot(P, [x(t),y(t)], 0..Pi);`

`DEplot`	create plot associated with an ODE or system of ODEs; see `?DEplot` for more information (in `DEtools` package)	`ODE := diff( y(x),x ) = 2*x*y(x);`
		`DEplot( ODE, [y(x)], x=-2..2,`
		`  y=-1..1, arrows=SMALL );`

`D`	differential operator (often used when specifying derivative initial conditions for `dsolve`)	`ODE :=  diff(y(x),x$2) +y(x) = 1;`
		`IC := y(0)=1, D(y)(0)=1;`
		`dsolve( { ODE, IC }, y(x) );`

`simplify`	apply simplification rules to an expression	`simplify( exp( a+ln(b*exp(c)) ) );`
`factor`	factor a polynomial	`factor( (x^3-y^3)/(x^4-y^4) );`
`convert`	convert an expression to a different form	`convert( x^3/(x^2-1), parfrac, x );`
`collect`	collect coefficients of like powers	`collect( (x+1)^3*(x+2)^2, x );`
`rhs`	right-hand side of an equation	`rhs( y = a*x^2 + b );`
`lhs`	left-hand side of an equation	`lhs( y = a*x^2 + b );`
`numer`	extract the numerator of an expression	`numer( (x+1)^3/(x+2)^2 );`
`denom`	extract the denominator of an expression	`denom( (x+1)^3/(x+2)^2 );`
`subs`	substitute values into an expression	`subs( x=r^(1/3), 3*x*ln(x^3) );`
`eval`	evaluate an expression with specific values	`eval( 3*x*ln(x^3), x=r^(1/3));`
`evalf`	evaluate using floating-point arithmetic	`evalf( exp( Pi^2 ) );`
`evalc`	evaluate a complex-valued expression (returns a value in the form `a+I*b`)	`evalc( exp( alpha+I*omega ) );`
`evalb`	evaluate a Boolean expression (returns `true` or `false` or `FAIL`)	`evalb( evalf( exp(Pi) > Pi^exp(1) ) );`
`assign`	perform assignments (often used after `solve` or `dsolve`)	`S:=solve( {x+y=1, 2*x+y=3}, {x,y} );` `assign( S ); x; y;`
`seq`	create a sequence	`seq( [0,i], i=-3..3 );`

`for ... from ... to ... by ... in ... while ... do ... end do`	repetition statement; see `?do` for syntax (Note: `od` is an acceptable substitute for `end do`)	`tot := 0;`
		`for i from 11 by 2 while i < 100 do`
		`  tot := tot + i^2`
		`end do;`

`if ... then ... elif ... else ... end if`	conditional statement; see `?if` for syntax (Note: `fi` is an acceptable substitute for `end if`)	`if type(x,name) then 'f'(x)` `  else x+1 end if;`
`assume`	inform Maple of additional properties of objects	`assume( t>0 );`
`about`	check assumptions on Maple objects	`about( t );`