

1 Introduction

The following is a collection of synonyms for various operations in the computer algebra systems Axiom, Derive, GAP, Gmp, DoCon, Macsyma, Magnus, Maxima, Maple, Mathematica, MuPAD, Octave, Pari, Reduce, Scilab, Sumit and Yacas. This collection does not attempt to be comprehensive, but hopefully it will be useful in giving an indication of how to translate between the syntaxes used by the different systems in many common situations. Note that a blank entry means either (a) that there may be an exact translation of a particular operation for the indicated system, but we don't know what it is or (b) there is no exact translation but it may still be possible to work around this lack with a related functionality.

While commercial systems are not provided on this CD the intent of the Rosetta effort is to make it possible for experienced Computer Algebra users to experiment with other systems. Thus the commands for commercial systems are included to allow users of those systems to translate.

Some of these systems are special purpose and do not support a lot of the functionality of the more general purpose systems. Where they do support an interpreter the commands are provided.

Originally written by Michael Wester. Modified for Rosetta by Timothy Daly, Alexander Hulpke (GAP).

2 System availability

System	License	Status (May 2002)	Web Location
Axiom	BSD	available	http://www.algor.org
Axiom	open source	pending	http://home.earthlink.net/~jgg964/axiom.html
Derive	commercial	available	http://www.mathware.com
DoCon	open source	available	http://www.haskell.org/docon
GAP	GPL	Rosetta	http://www.gap-system.org/ gap
Gmp	GPL	Rosetta	http://www.swox.com/gmp
Macsyma	commercial	dead	unavailable
Magnus	GPL	Rosetta	http://sourceforge.net/projects/magnus
Maxima	GPL	Rosetta	http://www.ma.utexas.edu/maxima.html
Maple	commercial	available	http://www.maplesoft.com
Mathematica	commercial	available	http://www.wolfram.com
MuPAD	commercial	available	http://www.mupad.de
Octave	GPL	Rosetta	http://www.octave.org
Pari	GPL	Rosetta	http://www.parigp-home.de
Reduce	commercial	available	http://www.zib.de/Symbolik/reduce
Scilab	Scilab	available	http://www-rocq.inria.fr/scilab
Sumit		available	http://www-sop.inria.fr/cafe/soft-f.html
Yacas	GPL	available	http://yacas.sourceforge.net

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

System	Type	Interpreted or Compiled
Axiom	General Purpose	both
Derive	General Purpose	
DoCon	General Purpose	Interpreted in Haskell
GAP	Group Theory	
Gmp	arb. prec. arithmetic	
Macsyma	General Purpose	
Magnus	Infinite Group Theory	
Maxima	General Purpose	
Maple	General Purpose	
Mathematica	General Purpose	
MuPAD	General Purpose	
Octave	Numerical Computing	
Pari	Number Theory	
Reduce	General Purpose	
Scilab	General Purpose	
Sumit	Functional Equations	
Yacas	General Purpose	

3 Programming and Miscellaneous

	Unix/Microsoft user initialization file	
Axiom	~/axiom.input	
GAP	~/gaprc	GAP.RC
Gmp		
DoCon		
Derive		derive.ini
Macsyma	~/macsyma-init.macsyma	mac-init.mac
Magnus		
Maxima	~/macsyma-init.macsyma	mac-init.mac
Maple	~/mapleinit	maplev5.ini
Mathematica	~/init.m	init.m
MuPAD	~/mupadinit	\mupad\bin\userinit.mu
Octave		
Pari		
Reduce	~/reducerc	reduce.rc
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Describe <i>keyword</i>	Find keywords containing <i>pattern</i>			
Axiom)what operations pattern			
Derive					
DoCon					
GAP	?keyword	??keyword			
Gmp					
Macsyma	describe("keyword")\$	apropos("pattern");			
Magnus					
Maxima	describe("keyword")\$	apropos("pattern");			
Maple	?keyword	?pattern ¹			
Mathematica	?keyword	?*pattern*			
MuPAD	?keyword	?*pattern*			
Octave	help -i keyword				
Pari					
Reduce					
Scilab					
Sumit					
Yacas					
	Comment	Line continuation	Prev. expr.	Case sensitive	Variables assumed
Axiom	-- comment	input _<CR>input	%	Yes	real
Derive	"comment"	input ~<CR>input		No	real
DoCon					
GAP	# comment	input\<CR>input	last	Yes	no assumption
Gmp					
Macsyma	/* comment */	input<CR>input;	%	No	real
Magnus					
Maxima	/* comment */	input<CR>input;	%	No	real
Maple	# comment	input<CR>input;	%	Yes	complex
Mathematica	(* comment *)	input<CR>input	%	Yes	complex
MuPAD	# comment #	input<CR>input;	%	Yes	complex
Octave	##			Yes	
Pari					
Reduce	% comment	input<CR>input;	ws	No	complex
Scilab					
Sumit					
Yacas					

¹Only if the pattern is not a keyword and then the matches are simplistic.

	Load a file	Time a command	Quit
Axiom)read "file")quiet)set messages time on)quit
Derive	[Transfer Load Derive]		[Quit]
DoCon			
GAP	Read("file");	time; (also see Runtime());	quit;
Gmp			
Macsyma	load("file")\$	showtime: all\$	quit();
Magnus			
Maxima	load("file")\$	showtime: all\$	quit();
Maple	read("file"):	readlib(showtime): on;	quit
Mathematica	@<< file	Timing[command]	Quit[]
MuPAD	read("file"):	time(command);	quit
Octave	load file	tic(); cmd ; toc()	quit <i>or</i> exit
Pari			
Reduce	in "file"\$	on time;	quit;
Scilab			quit
Sumit			
Yacas			

	Display output	Suppress output	Substitution: $f(x, y) \rightarrow f(z, w)$
Axiom	input	input;	subst(f(x, y), [x = z, y = w])
Derive	input	var:= input	[Manage Substitute]
DoCon			
GAP	input;	input;;	Value(f, [x,y], [z,w]); ²
Gmp			
Macsyma	input;	input\$	subst([x = z, y = w], f(x, y));
Magnus			
Maxima	input;	input\$	subst([x = z, y = w], f(x, y));
Maple	input;	input:	subs({x = z, y = w}, f(x, y));
Mathematica	input	input;	f[x, y] /. {x -> z, y -> w}
MuPAD	input;	input:	subs(f(x, y), [x = z, y = w]);
Octave	input	input;	
Pari			
Reduce	input;	input\$	sub({x = z, y = w}, f(x, y));
Scilab			
Sumit			
Yacas			

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Set	List	Matrix
Axiom	<code>set [1, 2]</code>	<code>[1, 2]</code>	<code>matrix(@[[1, 2],[3, 4]])</code>
Derive	<code>{1, 2}</code>	<code>[1, 2]</code>	<code>@[[1,2], [3,4]]</code>
DoCon			
GAP	<code>Set([1,2])</code>	<code>[1, 2]</code>	<code>@[[1,2], [3,4]]³</code>
Gmp			
Macsyma	<code>[1, 2]</code>	<code>[1, 2]</code>	<code>matrix([1, 2], [3, 4])</code>
Magnus			
Maxima	<code>[1, 2]</code>	<code>[1, 2]</code>	<code>matrix([1, 2], [3, 4])</code>
Maple	<code>{1, 2}</code>	<code>[1, 2]</code>	<code>matrix(@[[1, 2], [3, 4]])</code>
Mathematica	<code>{1, 2}</code>	<code>{1, 2}</code>	<code>{{1, 2}, {3, 4}}</code>
MuPAD	<code>{1, 2}</code>	<code>[1, 2]</code>	<code>export(Dom): export(linalg): matrix:= ExpressionField(normal): matrix(@[[1, 2], [3, 4]])</code>
Octave			
Pari			
Reduce	<code>{1, 2}</code>	<code>{1, 2}</code>	<code>mat((1, 2), (3, 4))</code>
Scilab		<code>list(1,2)</code>	<code>A=[1,2;3,4]</code>
Sumit			
Yacas			

	Equation	List element	Matrix element	Length of a list
Axiom	<code>x = 0</code>	<code>1 . 2</code>	<code>m(2, 3)</code>	<code>#1</code>
Derive	<code>x = 0</code>	<code>1 SUB 2</code>	<code>m SUB 2 SUB 3</code>	<code>DIMENSION(1)</code>
DoCon				
GAP	<code>x=0</code>	<code>1[2]</code>	<code>m[2][3]</code>	<code>Length(1)</code>
Gmp				
Macsyma	<code>x = 0</code>	<code>1[2]</code>	<code>m[2, 3]</code>	<code>length(1)</code>
Magnus				
Maxima	<code>x = 0</code>	<code>1[2]</code>	<code>m[2, 3]</code>	<code>length(1)</code>
Maple	<code>x = 0</code>	<code>1[2]</code>	<code>m[2, 3]</code>	<code>nops(1)</code>
Mathematica	<code>x == 0</code>	<code>1@[[2]]</code>	<code>m@[[2, 3]]</code>	<code>Length[1]</code>
MuPAD	<code>x = 0</code>	<code>1[2]</code>	<code>m[2, 3]</code>	<code>nops(1)</code>
Octave				
Pari				
Reduce	<code>x = 0</code>	<code>part(1, 2)</code>	<code>m(2, 3)</code>	<code>length(1)</code>
Scilab		<code>1(2)</code>		
Sumit				
Yacas				

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Prepend/append an element to a list		Append two lists
Axiom	<code>cons(e, l)</code>	<code>concat(l, e)</code>	<code>append(l1, l2)</code>
Derive	<code>APPEND([e], l)</code>	<code>APPEND(l, [e])</code>	<code>APPEND(l1, l2)</code>
DoCon			
GAP	<code>Concatenation([e], l)</code>	<code>Add(l, e)</code>	<code>Append(l1, l2)</code>
Gmp			
Macsyma	<code>cons(e, l)</code>	<code>endcons(e, l)</code>	<code>append(l1, l2)</code>
Magnus			
Maxima	<code>cons(e, l)</code>	<code>endcons(e, l)</code>	<code>append(l1, l2)</code>
Maple	<code>[e, op(l)]</code>	<code>[op(l), e]</code>	<code>[op(l1), op(l2)]</code>
Mathematica	<code>Prepend[l, e]</code>	<code>Append[l, e]</code>	<code>Join[l1, l2]</code>
MuPAD	<code>[e, op(l)]</code>	<code>append(l, e)</code>	<code>l1 . l2</code>
Octave			
Pari			
Reduce	<code>e . l</code>	<code>append(l, e)</code>	<code>append(l1, l2)</code>
Scilab			
Sumit			
Yacas			

	Matrix column dimension	Convert a list into a column vector
Axiom	<code>ncols(m)</code>	<code>transpose(matrix([l]))</code>
Derive	<code>DIMENSION(m SUB 1)</code>	<code>[l]^`</code>
DoCon		
GAP	<code>Length(mat[l])</code>	objects are identical
Gmp		
Macsyma	<code>mat_ncols(m)</code>	<code>transpose(matrix(l))</code>
Magnus		
Maxima	<code>mat_ncols(m)</code>	<code>transpose(matrix(l))</code>
Maple	<code>linalg[coldim](m)</code>	<code>linalg[transpose](matrix([l]))</code>
Mathematica	<code>Dimensions[m][[2]]</code>	<code>Transpose[{l}]</code>
MuPAD	<code>linalg::ncols(m)</code>	<code>transpose(matrix([l]))</code> ⁴
Octave		
Pari		
Reduce	<code>load_package(linalg)\$ column_dim(m)</code>	<code>matrix v(length(l), 1)\$ for i:=1:length(l) do v(i, 1):= part(l, i)</code>
Scilab		
Sumit		
Yacas		

⁴See the definition of `matrix` above.

	Convert a column vector into a list
Axiom	<code>[v(i, 1) for i in 1..nrows(v)]</code>
Derive	<code>v` SUB 1</code>
DoCon	
GAP	objects are identical
Gmp	
Macsyma	<code>part(transpose(v), 1)</code>
Magnus	
Maxima	<code>part(transpose(v), 1)</code>
Maple	<code>op(convert(linalg[transpose](v), listlist))</code>
Mathematica	<code>Flatten[v]</code>
MuPAD	<code>[op(v)]</code>
Octave	
Pari	
Reduce	<code>load_package(linalg)\$</code> <code>for i:=1:row_dim(v) collect(v(i, 1))</code>
Scilab	
Sumit	
Yacas	

	True	False	And	Or	Not	Equal	Not equal
Axiom	true	false	and	or	not	=	~=
Derive	TRUE	FALSE	AND	OR	NOT	=	/=
DoCon							
GAP	true	false ⁵	and	or	not	=	<>
Gmp							
Macsyma	true	false	and	or	not	=	#
Magnus							
Maxima	true	false	and	or	not	=	#
Maple	true	false	and	or	not	=	<>
Mathematica	True	False	&&		!	==	!=
MuPAD	true	false	and	or	not	=	<>
Octave							
Pari							
Reduce	t	nil	and	or	not	=	neq
Scilab	%t	%f					
Sumit							
Yacas							

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	If+then+else statements	Strings (concatenated)
Axiom	if _ then _ else if _ then _ else _	concat(["x", "y"])
Derive	IF(., ., IF(., ., .))	"xy"
DoCon		
GAP	if _ then _ elif _ then _ else _ fi	Concatenation("x","y")
Gmp		
Macsyma	if _ then _ else if _ then _ else _	concat("x", "y")
Magnus		
Maxima	if _ then _ else if _ then _ else _	concat("x", "y")
Maple	if _ then _ elif _ then _ else _ fi	"x" . "y"
Mathematica	If[., ., If[., ., .]]	"x" <> "y"
MuPAD	if _ then _ elif _ then _ else _ end_if	"x" . "y"
Octave		
Pari		
Reduce	if _ then _ else if _ then _ else _	"xy" or mkid(x, y)
Scilab		
Sumit		
Yacas		
	Simple loop and Block	Generate the list $[1, 2, \dots, n]$
Axiom	for i in 1..n repeat (x; y)	[f(i) for i in 1..n]
Derive	VECTOR([x, y], i, 1, n)	VECTOR(f(i), i, 1, n)
DoCon		
GAP	for i in [1..n] do _ od;	[1..n] or [1,2..n]
Gmp		
Macsyma	for i:1 thru n do (x, y);	makelist(f(i), i, 1, n);
Magnus		
Maxima	for i:1 thru n do (x, y);	makelist(f(i), i, 1, n);
Maple	for i from 1 to n do x; y od;	[f(i) \$ i = 1..n];
Mathematica	Do[x; y, {i, 1, n}]	Table[f[i], {i, 1, n}]
MuPAD	for i from 1 to n do x; y end_for;	[f(i) \$ i = 1..n];
Octave		
Pari		
Reduce	for i:=1:n do @<<x; y>>;	for i:=1:n collect f(i);
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Complex loop iterating on a list		
Axiom	for x in [2, 3, 5] while x**2 < 10 repeat output(x)		
Derive			
DoCon			
GAP	for x in [2, 3, 5] do while x^2<10 do Print(x);od;od;		
Gmp			
Macsyma	for x in [2, 3, 5] while x^2 < 10 do print(x)\$		
Magnus			
Maxima	for x in [2, 3, 5] while x^2 < 10 do print(x)\$		
Maple	for x in [2, 3, 5] while x^2 < 10 do print(x) od:		
Mathematica	For[l = {2, 3, 5}, l != {} && l@[[1]]^2 < 10, l = Rest[l], Print[l@[[1]]]]		
MuPAD	for x in [2, 3, 5] do if x^2 < 10 then print(x) end_if end_for:		
Octave			
Pari			
Reduce	for each x in {2, 3, 5} do if x^2 < 10 then write(x)\$		
Scilab			
Sumit			
Yacas			
	Assignment	Function definition	Clear vars and funs
Axiom	y:= f(x)	f(x, y) == x*y)clear properties y f
Derive	y:= f(x)	f(x, y):= x*y	y:= f:=
DoCon			
GAP	y:= f(x);	f:=function(x, y) return x*y; end;	There are no symbolic variables
Gmp			
Macsyma	y: f(x);	f(x, y):= x*y;	remvalue(y)\$ remfunction(f)\$
Magnus			
Maxima	y: f(x);	f(x, y):= x*y;	remvalue(y)\$ remfunction(f)\$
Maple	y:= f(x);	f:= proc(x, y) x*y end;	y:= 'y': f:= 'f':
Mathematica	y = f[x]	f[x_, y_]:= x*y	Clear[y, f]
MuPAD	y:= f(x);	f:= proc(x, y) begin x*y end_proc;	y:= NIL: f:= NIL:
Octave			
Pari			
Reduce	y:= f(x);	procedure f(x, y); x*y;	clear y, f;
Scilab			
Sumit			
Yacas			

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Function definition with a local variable	
Axiom	<code>f(x) == (local n; n:= 2; n*x)</code>	
Derive		
DoCon		
GAP	<code>f:=function(x) local n; n:=2;return n*x; end;</code>	
Gmp		
Macsyma	<code>f(x):= block([n], n: 2, n*x);</code>	
Magnus		
Maxima	<code>f(x):= block([n], n: 2, n*x);</code>	
Maple	<code>f:= proc(x) local n; n:= 2; n*x end;</code>	
Mathematica	<code>f[x_]:= Module[{n}, n = 2; n*x]</code>	
MuPAD	<code>f:= proc(x) local n; begin n:= 2; n*x end_proc;</code>	
Octave		
Pari		
Reduce	<code>procedure f(x); begin scalar n; n:= 2; return(n*x) end;</code>	
Scilab		
Sumit		
Yacas		
	Return unevaluated symbol	Define a function from an expression
Axiom	<code>e:= x*y; 'e</code>	<code>function(e, f, x, y)</code>
Derive	<code>e:= x*y 'e</code>	<code>f(x, y):= e</code>
DoCon		
GAP	No unevaluated symbols ⁶	
Gmp		
Macsyma	<code>e: x*y\$ 'e;</code>	<code>define(f(x, y), e);</code>
Magnus		
Maxima	<code>e: x*y\$ 'e;</code>	<code>define(f(x, y), e);</code>
Maple	<code>e:= x*y: 'e';</code>	<code>f:= unapply(e, x, y);</code>
Mathematica	<code>e = x*y; HoldForm[e]</code>	<code>f[x_, y_] = e</code>
MuPAD	<code>e:= x*y: hold(e);</code>	<code>f:= hold(func)(e, x, y);</code>
Octave		
Pari		
Reduce	<code>e:= x*y\$</code>	<code>for all x, y let f(x, y):= e;</code>
Scilab		
Sumit		
Yacas		

⁶Variables can be assigned to generators of a suitable free object, for example `x:=X(Rationals,"x");` or `f:=FreeGroup(2);x:=f.1;`

	Fun. of an indefinite number of args	Apply “+” to sum a list
Axiom		<code>reduce(+, [1, 2])</code>
Derive	<code>LST 1:= 1</code>	
DoCon		
GAP	<code>lst:=function(args) _ end;</code>	<code>Sum([1,2])</code>
Gmp		
Macsyma	<code>lst([1]):= 1;</code>	<code>apply("+", [1, 2])</code>
Magnus		
Maxima	<code>lst([1]):= 1;</code>	<code>apply("+", [1, 2])</code>
Maple	<code>lst:=proc() [args[1..nargs]] end;</code>	<code>convert([1, 2], `+`)</code>
Mathematica	<code>lst[l_]:= {1}</code>	<code>Apply[Plus, {1, 2}]</code>
MuPAD	<code>lst:= proc(l) begin [args()]</code> <code>end_proc;</code>	<code>_plus(op([1, 2]))</code>
Octave		
Pari		
Reduce		<code>xapply(+, {1, 2})</code> ⁶
Scilab		
Sumit		
Yacas		
	Apply a fun. to a list of its args	Map an anonymous function onto a list
Axiom	<code>reduce(f, l)</code>	<code>map(x +-> x + y, [1, 2])</code>
Derive		<code>x:= [1, 2]</code> <code>VECTOR(x SUB i + y, i, 1, DIMENSION(x))</code>
DoCon		
GAP	<code>List(l,f)</code>	<code>List([1,2],x->x+y)</code>
Gmp		
Macsyma	<code>apply(f, l)</code>	<code>map(lambda([x], x + y), [1, 2])</code>
Magnus		
Maxima	<code>apply(f, l)</code>	<code>map(lambda([x], x + y), [1, 2])</code>
Maple	<code>f(op(l))</code>	<code>map(x -> x + y, [1, 2])</code>
Mathematica	<code>Apply[f, l]</code>	<code>Map[# + y &, {1, 2}]</code>
MuPAD	<code>f(op(l))</code>	<code>map([1, 2], func(x + y, x))</code>
Octave		
Pari		
Reduce	<code>xapply(f, l)</code> ⁶	<code>for each x in {1, 2} collect x + y</code>
Scilab		
Sumit		
Yacas		

⁶`procedure xapply(f, lst); lisp(f . cdr(lst))$`

	Pattern matching: $f(3y) + f(zy) \rightarrow 3f(y) + f(zy)$
Axiom	<pre>f:= operator('f); (rule f((n integer?(n)) * x) == n*f(x))(_ f(3*y) + f(z*y))</pre>
Derive	
DoCon	
GAP	
Gmp	
Macsyma	<pre>matchdeclare(n, integerp, x, true)\$ defrule(fnx, f(n*x), n*f(x))\$ apply1(f(3*y) + f(z*y), fnx);</pre>
Magnus	
Maxima	<pre>matchdeclare(n, integerp, x, true)\$ defrule(fnx, f(n*x), n*f(x))\$ apply1(f(3*y) + f(z*y), fnx);</pre>
Maple	<pre>map(proc(q) local m; if match(q = f(n*y), y, 'm') and type(rhs(op(m)), integer) then subs(m, n * f(y)) else q fi end, f(3*y) + f(z*y));</pre>
Mathematica	<pre>f[3*y] + f[z*y] /. f[n_Integer * x_] -> n*f[x]</pre>
MuPAD	<pre>d:= domain("match"): d::FREEVARIABLE:= TRUE: n:= new(d, "n", func(testtype(m, DOM_INT), m)): x:= new(d, "x", TRUE): map(f(3*y) + f(z*y), proc(q) local m; begin m:= match(q, f(n*x)); if m = FAIL then q else subs(hold("n" * f("x")), m) end_if end_proc);</pre>
Octave	
Pari	
Reduce	<pre>operator f; f(3*y) + f(z*y) where {f(~n * ~x) => n*f(x) when fixp(n)};</pre>
Scilab	
Sumit	
Yacas	

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Define a new infix operator and then use it		
Axiom	<p>One can overload existing infix operators for ones own purposes</p> <pre> infix("~")\$ "~"(x, y) := sqrt(x^2 + y^2)\$ 3 ~ 4; infix("~")\$ "~"(x, y) := sqrt(x^2 + y^2)\$ 3 ~ 4; `&~~ := (x, y) -> sqrt(x^2 + y^2): 3 &~ 4; x_ \[Tilde] y_ := Sqrt[x^2 + y^2]; 3 \[Tilde] 4 tilde:= proc(x, y) begin sqrt(x^2 + y^2) end_proc: 3 &tilde 4; infix \$ procedure (x, y); sqrt(x^2 + y^2)\$ 3 4; </pre>		
Derive			
DoCon			
GAP			
Gmp			
Macsyma			
Magnus			
Maxima			
Maple			
Mathematica			
MuPAD			
Octave			
Pari			
Reduce			
Scilab			
Sumit			
Yacas			
	Main expression operator	1 st operand	List of expression operands
Axiom ⁷		kernel(s)(e) . 1	kernel(s)(e)
Derive			<i>various</i> ⁸
DoCon			
GAP			There are no formal unevaluated expressions
Gmp			
Macsyma	part(e, 0)	part(e, 1)	args(e)
Magnus			
Maxima	part(e, 0)	part(e, 1)	args(e)
Maple	op(0, e)	op(1, e)	[op(e)]
Mathematica	Head[e]	e@[[1]]	ReplacePart[e, List, 0]
MuPAD	op(e, 0)	op(e, 1)	[op(e)]
Octave			
Pari			
Reduce	part(e, 0)	part(e, 1)	for i:=1:arglength(e) collect part(e, i)
Scilab			
Sumit			
Yacas			

⁷The following commands work only on expressions that consist of a single level (e.g., $x + y + z$ but not $a/b + c/d$).

⁸TERMS, FACTORS, NUMERATOR, LHS, etc.

	Print text and results	
Axiom	output(concat(["sin(", string(0), ") = ", string(sin(0))])));	
Derive	"sin(0)" = sin(0)	
DoCon		
GAP	Print("There is no sin, but factors(10)= ",Factors(10), "\n")	
Gmp		
Macsyma	print("sin(", 0, ") =", sin(0))\$	
Magnus		
Maxima	print("sin(", 0, ") =", sin(0))\$	
Maple	printf("sin(%a) = %a\n", 0, sin(0)):	
Mathematica	Print[StringForm["sin(``) = ``", 0, Sin[0]]];	
MuPAD	print(Unquoted, "sin(".0.)" = sin(0)):	
Octave		
Pari		
Reduce	write("sin(", 0, ") = ", sin(0))\$	
Scilab		
Sumit		
Yacas		
	Generate FORTRAN	Generate T _E X/L ^A T _E X
Axiom	outputAsFortran(e)	outputAsTex(e)
Derive	[Transfer Save Fortran]	
DoCon		
GAP		Print(LaTeX(e));
Gmp		
Macsyma	fortran(e)\$ or gentran(eval(e))\$	tex(e);
Magnus		
Maxima	fortran(e)\$ or gentran(eval(e))\$	tex(e);
Maple	fortran([e]);	latex(e);
Mathematica	FortranForm[e]	TexForm[e]
MuPAD	generate::fortran(e);	generate::TeX(e);
Octave		
Pari		
Reduce	on fort; e; off fort; or	load_package(tri)\$
Scilab	load_package(gentran)\$ gentran e;	on TeX; e; off TeX;
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Import two space separated columns of integers from file
Axiom	
Derive	[Transfer Load daTa] (from file.dat)
DoCon	
GAP	
Gmp	
Macsyma	xy: read_num_data_to_matrix("file", nrow, 2)\$
Magnus	
Maxima	xy: read_num_data_to_matrix("file", nrow, 2)\$
Maple	xy:= readdata("file", integer, 2):
Mathematica	xy = ReadList["file", Number, RecordLists -> True]
MuPAD	
Octave	
Pari	
Reduce	
Scilab	
Sumit	
Yacas	

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Export two space separated columns of integers to file ⁷
Axiom	<pre>)set output algebra "file" (creates file.spout) for i in 1..n repeat output(concat([string(xy(i, 1)), " ", string(xy(i, 2))])))set output algebra console xy [Transfer Print Expressions File] (creates file.prt)</pre>
Derive	
DoCon	
GAP	<pre>PrintTo("file");for i in [1..n] do AppendTo("file",xy[i][1]," ",xy[i][2],"\n");od;</pre>
Gmp	
Macsyma	<pre>writefile("file")\$ for i:1 thru n do print(xy[i, 1], xy[i, 2])\$ closefile()</pre>
Magnus	
Maxima	<pre>writefile("file")\$ for i:1 thru n do print(xy[i, 1], xy[i, 2])\$ closefile()</pre>
Maple	<pre>writedata("file", xy); outfile = OpenWrite["file"]; Do[WriteString[outfile, xy@[[i, 1]], " ", xy@[[i, 2]], "\n"], {i, 1, n}] Close[outfile];</pre>
Mathematica	
MuPAD	<pre>fprint(Unquoted, Text, "file", ("\\n", xy[i, 1], xy[i, 2]) \$ i = 1..n):</pre>
Octave	
Pari	
Reduce	<pre>out "file"; for i:=1:n do write(xy(i, 1), " ", xy(i, 2)); shut "file";</pre>
Scilab	
Sumit	
Sumit	
Yacas	

4 Mathematics and Graphics

Since GAP aims at discrete mathematics, it does not provide much of the calculus functionality listed in the following section.

⁷Some editing of file will be necessary for all systems but Maple and Mathematica.

	e	π	i	$+\infty$	$\sqrt{2}$	$2^{1/3}$
Axiom	%e	%pi	%i	%plusInfinity	sqrt(2)	2**(1/3)
Derive	#e	pi	#i	inf	SQRT(2)	2^(1/3)
DoCon						
GAP			E(4)	infinity	ER(2) ⁸	
Gmp						
Macsyma	%e	%pi	%i	inf	sqrt(2)	2^(1/3)
Magnus						
Maxima	%e	%pi	%i	inf	sqrt(2)	2^(1/3)
Maple	exp(1)	Pi	I	infinity	sqrt(2)	2^(1/3)
Mathematica	E	Pi	I	Infinity	Sqrt[2]	2^(1/3)
MuPAD	E	PI	I	infinity	sqrt(2)	2^(1/3)
Octave						
Pari						
Reduce	e	pi	i	infinity	sqrt(2)	2^(1/3)
Scilab						
Sumit						
Yacas						

	Euler's constant	Natural log	Arctangent	$n!$
Axiom		log(x)	atan(x)	factorial(n)
Derive	euler_gamma	LOG(x)	ATAN(x)	n!
DoCon				
GAP		LogInt(x,base)		Factorial(n)
Gmp				
Macsyma	%gamma	log(x)	atan(x)	n!
Magnus				
Maxima	%gamma	log(x)	atan(x)	n!
Maple	gamma	log(x)	arctan(x)	n!
Mathematica	EulerGamma	Log[x]	ArcTan[x]	n!
MuPAD	EULER	ln(x)	atan(x)	n!
Octave				
Pari				
Reduce	Euler_Gamma	log(x)	atan(x)	factorial(n)
Scilab				
Sumit				
Yacas				

⁸ER represents special cyclotomic numbers and is not a root function.

	Legendre polynomial	Chebyshev poly. of the 1 st kind
Axiom	<code>legendreP(n, x)</code>	<code>chebyshevT(n, x)</code>
Derive	<code>LEGENDRE_P(n, x)</code>	<code>CHEBYCHEV_T(n, x)</code>
DoCon		
GAP		
Gmp		
Macsyma	<code>legendre_p(n, x)</code>	<code>chebyshev_t(n, x)</code>
Magnus		
Maxima	<code>legendre_p(n, x)</code>	<code>chebyshev_t(n, x)</code>
Maple	<code>orthopoly[P](n, x)</code>	<code>orthopoly[T](n, x)</code>
Mathematica	<code>LegendreP[n, x]</code>	<code>ChebyshevT[n, x]</code>
MuPAD	<code>orthpoly::legendre(n, x)</code>	<code>orthpoly::chebyshev1(n, x)</code>
Octave		
Pari		
Reduce	<code>LegendreP(n, x)</code>	<code>ChebyshevT(n, x)</code>
Scilab		
Sumit		
Yacas		

	Fibonacci number	Elliptic integral of the 1 st kind
Axiom	<code>fibonacci(n)</code>	
Derive	<code>FIBONACCI(n)</code>	<code>ELLIPTIC_E(phi, k^2)</code>
DoCon		
GAP	<code>Fibonacci(n)</code>	
Gmp		
Macsyma	<code>fib(n)</code>	<code>elliptic_e(phi, k^2)</code>
Magnus		
Maxima	<code>fib(n)</code>	<code>elliptic_e(phi, k^2)</code>
Maple	<code>combinat[fibonacci](n)</code>	<code>EllipticE(sin(phi), k)</code>
Mathematica	<code>Fibonacci[n]</code>	<code>EllipticE[phi, k^2]</code>
MuPAD	<code>numlib::fibonacci(n)</code>	
Octave		
Pari		
Reduce		<code>EllipticE(phi, k^2)</code>
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	$\Gamma(x)$	$\psi(x)$	Cosine integral	Bessel fun. (1 st)
Axiom	Gamma(x)	psi(x)	real(Ei(%i*x))	besselJ(n, x)
Derive	GAMMA(x)	PSI(x)	CI(x)	BESSEL_J(n, x)
DoCon				
GAP				
Gmp				
Macsyma	gamma(x)	psi[0](x)	cos_int(x)	bessel_j[n](x)
Magnus				
Maxima	gamma(x)	psi[0](x)	cos_int(x)	bessel_j[n](x)
Maple	GAMMA(x)	Psi(x)	Ci(x)	BesselJ(n, x)
Mathematica	Gamma[x]	PolyGamma[x]	CosIntegral[x]	BesselJ[n, x]
MuPAD	gamma(x)	psi(x)		besselJ(n, x)
Octave				
Pari				
Reduce	Gamma(x)	Psi(x)	Ci(x)	BesselJ(n, x)
Scilab				
Sumit				
Yacas				
	Hypergeometric fun. ${}_2F_1(a, b; c; x)$		Dirac delta	Unit step fun.
Axiom				
Derive	GAUSS(a, b, c, x)			STEP(x)
DoCon				
GAP				
Gmp				
Macsyma	hgfred([a, b], [c], x)		delta(x)	unit_step(x)
Magnus				
Maxima	hgfred([a, b], [c], x)		delta(x)	unit_step(x)
Maple	hypergeom([a, b], [c], x)		Dirac(x)	Heaviside(x)
Mathematica	HypergeometricPFQ[{a,b},{c},x]		@<< Calculus`DiracDelta`	
MuPAD			dirac(x)	heaviside(x)
Octave				
Pari				
Reduce	hypergeometric({a, b}, {c}, x)			
Scilab				
Sumit				
Yacas				

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Define $ x $ via a piecewise function
Axiom	
Derive	<code>a(x):= -x*CHI(-inf, x, 0) + x*CHI(0, x, inf)</code>
DoCon	
GAP	
Gmp	
Macsyma	<code>a(x):= -x*unit_step(-x) + x*unit_step(x)\$</code>
Magnus	
Maxima	<code>a(x):= -x*unit_step(-x) + x*unit_step(x)\$</code>
Maple	<code>a:= x -> piecewise(x < 0, -x, x):</code>
Mathematica	<code>@<< Calculus`DiracDelta` a[x_]:= -x*UnitStep[-x] + x*UnitStep[x]</code>
MuPAD	<code>a:= proc(x) begin -x*heaviside(-x) + x*heaviside(x) end_proc:</code>
Octave	
Pari	
Reduce	
Scilab	
Sumit	
Yacas	

	Assume x is real	Remove that assumption
Axiom		
Derive	<code>x :epsilon Real</code>	<code>x:=</code>
DoCon		
GAP		
Gmp		
Macsyma	<code>declare(x, real)\$</code>	<code>remove(x, real)\$</code>
Magnus		
Maxima	<code>declare(x, real)\$</code>	<code>remove(x, real)\$</code>
Maple	<code>assume(x, real);</code>	<code>x:= 'x':</code>
Mathematica	<code>x/: Im[x] = 0;</code>	<code>Clear[x]</code>
MuPAD	<code>assume(x, Type::RealNum):</code>	<code>unassume(x, Type::RealNum):</code>
Octave		
Pari		
Reduce		
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Assume $0 < x \leq 1$	Remove that assumption
Axiom		
Derive	<code>x :epsilon (0, 1]</code>	<code>x:=</code>
DoCon		
GAP		
Gmp		
Macsyma	<code>assume(x > 0, x <= 1)\$</code>	<code>forget(x > 0, x <= 1)\$</code>
Magnus		
Maxima	<code>assume(x > 0, x <= 1)\$</code>	<code>forget(x > 0, x <= 1)\$</code>
Maple	<code>assume(x > 0);</code> <code>additionally(x <= 1);</code>	<code>x:= 'x':</code>
Mathematica	<code>Assumptions -> 0 < x <= 1</code> ⁸	
MuPAD	<code>assume(x > 0): assume(x <= 1):</code>	<code>unassume(x):</code>
Octave		
Pari		
Reduce		
Scilab		
Sumit		
Yacas		
	Basic simplification of an expression e	
Axiom	<code>simplify(e) or normalize(e) or complexNormalize(e)</code>	
Derive	<code>e</code>	
DoCon		
GAP	<code>e</code>	
Gmp		
Macsyma	<code>ratsimp(e) or radcan(e)</code>	
Magnus		
Maxima	<code>ratsimp(e) or radcan(e)</code>	
Maple	<code>simplify(e)</code>	
Mathematica	<code>Simplify[e] or FullSimplify[e]</code>	
MuPAD	<code>simplify(e) or normal(e)</code>	
Octave		
Pari		
Reduce	<code>e</code>	
Scilab		
Sumit		
Yacas		

⁸This is an option for `Integrate`.

	Use an unknown function	Numerically evaluate an expr.
Axiom	<code>f := operator('f);</code> <code>f(x)</code>	<code>exp(1) :: Complex Float</code>
Derive	<code>f(x) :=</code> <code>f(x)</code>	<code>Precision := Approximate</code> <code>APPROX(EXP(1))</code> <code>Precision := Exact</code>
DoCon		
GAP		<code>EvalF(123/456)</code>
Gmp		
Macsyma	<code>f(x)</code>	<code>sfloat(exp(1));</code>
Magnus		
Maxima	<code>f(x)</code>	<code>sfloat(exp(1));</code>
Maple	<code>f(x)</code>	<code>evalf(exp(1));</code>
Mathematica	<code>f[x]</code>	<code>N[Exp[1]]</code>
MuPAD	<code>f(x)</code>	<code>float(exp(1));</code>
Octave		
Pari		
Reduce	<code>operator f;</code> <code>f(x)</code>	<code>on rounded; exp(1);</code> <code>off rounded;</code>
Scilab		
Sumit		
Yacas		
	$n \bmod m$	Solve $e \equiv 0 \bmod m$ for x
Axiom	<code>rem(n, m)</code>	<code>solve(e = 0 :: PrimeField(m), x)</code>
Derive	<code>MOD(n, m)</code>	<code>SOLVE.MOD(e = 0, x, m)</code>
DoCon		
GAP	<code>n mod m</code>	solve using finite fields
Gmp		
Macsyma	<code>mod(n, m)</code>	<code>modulus: m\$ solve(e = 0, x)</code>
Magnus		
Maxima	<code>mod(n, m)</code>	<code>modulus: m\$ solve(e = 0, x)</code>
Maple	<code>n mod m</code>	<code>msolve(e = 0, m)</code>
Mathematica	<code>Mod[n, m]</code>	<code>Solve[{e == 0, Modulus == m}, x]</code>
MuPAD	<code>n mod m</code>	<code>solve(poly(e = 0, [x], IntMod(m)), x)</code>
Octave		
Pari		
Reduce	<code>on modular;</code> <code>setmod m\$ n</code>	<code>load_package(modsr)\$ on modular;</code> <code>setmod m\$ m.solve(e = 0, x)</code>
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Put over common denominator	Expand into separate fractions
Axiom	$a/b + c/d$	$(a*d + b*c)/(b*d) :: -$ MPOLY([a], FRAC POLY INT)
Derive	FACTOR(a/b + c/d, Trivial)	EXPAND((a*d + b*c)/(b*d))
DoCon		
GAP	$a/b+c/d$	
Gmp		
Macsyma	xthru(a/b + c/d)	expand((a*d + b*c)/(b*d))
Magnus		
Maxima	xthru(a/b + c/d)	expand((a*d + b*c)/(b*d))
Maple	normal(a/b + c/d)	expand((a*d + b*c)/(b*d))
Mathematica	Together[a/b + c/d]	Apart[(a*d + b*c)/(b*d)]
MuPAD	normal(a/b + c/d)	expand((a*d + b*c)/(b*d))
Octave		
Pari		
Reduce	$a/b + c/d$	on div; (a*d + b*c)/(b*d)
Scilab		
Sumit		
Yacas		
	Manipulate the root of a polynomial	
Axiom	a:= rootOf(x**2 - 2); a**2	
Derive		
DoCon		
GAP	x:=X(Rationals,"x"); a:=RootOfDefiningPolynomial(AlgebraicExtension(Rationals,x^2-2)); a^2	
Gmp		
Macsyma	algebraic:true\$ tellrat(a^2 - 2)\$ rat(a^2);	
Magnus		
Maxima	algebraic:true\$ tellrat(a^2 - 2)\$ rat(a^2);	
Maple	a:= RootOf(x^2 - 2): simplify(a^2);	
Mathematica	a = Root[#^2 - 2 &, 2] a^2	
MuPAD		
Octave		
Pari		
Reduce	load_package(arnum)\$ defpoly(a^2 - 2); a^2;	
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Noncommutative multiplication	Solve a pair of equations
Axiom		<code>solve([eqn1, eqn2], [x, y])</code>
Derive	<code>x :epsilon Nonscalar</code> <code>y :epsilon Nonscalar</code> <code>x . y</code>	<code>SOLVE([eqn1, eqn2], [x, y])</code>
DoCon		
GAP	<code>*</code>	
Gmp		
Macsyma	<code>x . y</code>	<code>solve([eqn1, eqn2], [x, y])</code>
Magnus		
Maxima	<code>x . y</code>	<code>solve([eqn1, eqn2], [x, y])</code>
Maple	<code>x &* y</code>	<code>solve({eqn1, eqn2}, {x, y})</code>
Mathematica	<code>x ** y</code>	<code>Solve[{eqn1, eqn2}, {x, y}]</code>
MuPAD		<code>solve({eqn1, eqn2}, {x, y})</code>
Octave		
Pari		
Reduce	<code>operator x, y;</code> <code>noncom x, y;</code> <code>x() * y()</code>	<code>solve({eqn1, eqn2}, {x, y})</code>
Scilab		
Sumit		
Yacas		
	Decrease/increase angles in trigonometric functions	
Axiom	<code>simplify(normalize(sin(2*x)))</code>	
Derive	<code>Trigonometry:= Expand</code> <code>sin(2*x)</code>	<code>Trigonometry:= Collect</code> <code>2*sin(x)*cos(x)</code>
DoCon		
GAP		
Gmp		
Macsyma	<code>trigexpand(sin(2*x))</code>	<code>trigreduce(2*sin(x)*cos(x))</code>
Magnus		
Maxima	<code>trigexpand(sin(2*x))</code>	<code>trigreduce(2*sin(x)*cos(x))</code>
Maple	<code>expand(sin(2*x))</code>	<code>combine(2*sin(x)*cos(x))</code>
Mathematica	<code>TrigExpand[Sin[2*x]]</code>	<code>TrigReduce[2*Sin[x]*Cos[x]]</code>
MuPAD	<code>expand(sin(2*x))</code>	<code>combine(2*sin(x)*cos(x), sincos)</code>
Octave		
Pari		
Reduce	<code>load_package(assist)\$</code> <code>trigexpand(sin(2*x))</code>	<code>trigreduce(2*sin(x)*cos(x))</code>
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Gröbner basis
Axiom	groebner([p1, p2, ...])
Derive	
DoCon	
GAP	
Gmp	
Macsyma	grobner([p1, p2, ...])
Magnus	
Maxima	grobner([p1, p2, ...])
Maple	Groebner[gbasis]([p1, p2, ...], plex(x1, x2, ...))
Mathematica	GroebnerBasis[{p1, p2, ...}, {x1, x2, ...}]
MuPAD	groebner::gbasis([p1, p2, ...])
Octave	
Pari	
Reduce	load_package(groebner)\$ groebner({p1, p2, ...})
Scilab	
Sumit	
Yacas	
	Factorization of e over $i = \sqrt{-1}$
Axiom	factor(e, [rootOf(i**2 + 1)])
Derive	FACTOR(e, Complex)
DoCon	
GAP	Factors(GaussianIntegers,e)
Gmp	
Macsyma	gfactor(e); or factor(e, i^2 + 1);
Magnus	
Maxima	gfactor(e); or factor(e, i^2 + 1);
Maple	factor(e, I);
Mathematica	Factor[e, Extension -> I]
MuPAD	QI:= Dom::AlgebraicExtension(Dom::Rational, i^2 + 1); QI::name:= "QI": Factor(poly(e, QI));
Octave	
Pari	
Reduce	on complex, factor; e; off complex, factor;
Scilab	
Sumit	
Yacas	

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Real part	Convert a complex expr. to rectangular form
Axiom	<code>real(f(z))</code>	<code>complexForm(f(z))</code>
Derive	<code>RE(f(z))</code>	<code>f(z)</code>
DoCon		
GAP	<code>(f(z)+GaloisCyc(f(z),-1))/2</code>	
Gmp		
Macsyma	<code>realpart(f(z))</code>	<code>rectform(f(z))</code>
Magnus		
Maxima	<code>realpart(f(z))</code>	<code>rectform(f(z))</code>
Maple	<code>Re(f(z))</code>	<code>evalc(f(z))</code>
Mathematica	<code>Re[f[z]]</code>	<code>ComplexExpand[f[z]]</code>
MuPAD	<code>Re(f(z))</code>	<code>rectform(f(z))</code>
Octave		
Pari		
Reduce	<code>repart(f(z))</code>	<code>repart(f(z)) + i*impart(f(z))</code>
Scilab		
Sumit		
Yacas		

	Matrix addition	Matrix multiplication	Matrix transpose
Axiom	<code>A + B</code>	<code>A * B</code>	<code>transpose(A)</code>
Derive	<code>A + B</code>	<code>A . B</code>	<code>A`</code>
DoCon			
GAP	<code>A + B</code>	<code>A * B</code>	<code>TransposedMat(A)</code>
Gmp			
Macsyma	<code>A + B</code>	<code>A . B</code>	<code>transpose(A)</code>
Magnus			
Maxima	<code>A + B</code>	<code>A . B</code>	<code>transpose(A)</code>
Maple	<code>evalm(A + B)</code>	<code>evalm(A &* B)</code>	<code>linalg[transpose](A)</code>
Mathematica	<code>A + B</code>	<code>A . B</code>	<code>Transpose[A]</code>
MuPAD	<code>A + B</code>	<code>A * B</code>	<code>transpose(A)</code>
Octave			
Pari			
Reduce	<code>A + B</code>	<code>A * B</code>	<code>tp(A)</code>
Scilab			
Sumit			
Yacas			

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Solve the matrix equation $Ax = b$	
Axiom	<code>solve(A, transpose(b)) . 1 . particular :: Matrix ---</code>	
Derive		
DoCon		
GAP	<code>SolutionMat(TransposedMat(A),b)</code>	
Gmp		
Macsyma	<code>xx: genvector('x, mat_nrows(b))\$ x: part(matlinsolve(A . xx = b, xx), 1, 2)</code>	
Magnus		
Maxima	<code>xx: genvector('x, mat_nrows(b))\$ x: part(matlinsolve(A . xx = b, xx), 1, 2)</code>	
Maple	<code>x:= linalg[linsolve](A, b)</code>	
Mathematica	<code>x = LinearSolve[A, b]</code>	
MuPAD		
Octave		
Pari		
Reduce		
Scilab		
Sumit		
Yacas		
	Sum: $\sum_{i=1}^n f(i)$	Product: $\prod_{i=1}^n f(i)$
Axiom	<code>sum(f(i), i = 1..n)</code>	<code>product(f(i), i = 1..n)</code>
Derive	<code>SUM(f(i), i, 1, n)</code>	<code>PRODUCT(f(i), i, 1, n)</code>
DoCon		
GAP	<code>Sum([1..n],f)</code>	<code>Product([1..n],f)</code>
Gmp		
Macsyma	<code>closedform(sum(f(i), i, 1, n))</code>	<code>closedform(product(f(i), i, 1, n))</code>
Magnus		
Maxima	<code>closedform(sum(f(i), i, 1, n))</code>	<code>closedform(product(f(i), i, 1, n))</code>
Maple	<code>sum(f(i), i = 1..n)</code>	<code>product(f(i), i = 1..n)</code>
Mathematica	<code>Sum[f[i], {i, 1, n}]</code>	<code>Product[f[i], {i, 1, n}]</code>
MuPAD	<code>sum(f(i), i = 1..n)</code>	<code>product(f(i), i = 1..n)</code>
Octave		
Pari		
Reduce	<code>sum(f(i), i, 1, n)</code>	<code>prod(f(i), i, 1, n)</code>
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Limit: $\lim_{x \rightarrow 0^-} f(x)$	Taylor/Laurent/etc. series
Axiom	<code>limit(f(x), x = 0, "left")</code>	<code>series(f(x), x = 0, 3)</code>
Derive	<code>LIM(f(x), x, 0, -1)</code>	<code>TAYLOR(f(x), x, 0, 3)</code>
DoCon		
GAP		
Gmp		
Macsyma	<code>limit(f(x), x, 0, minus)</code>	<code>taylor(f(x), x, 0, 3)</code>
Magnus		
Maxima	<code>limit(f(x), x, 0, minus)</code>	<code>taylor(f(x), x, 0, 3)</code>
Maple	<code>limit(f(x), x = 0, left)</code>	<code>series(f(x), x = 0, 4)</code>
Mathematica	<code>Limit[f[x], x->0, Direction->1]</code>	<code>Series[f[x], {x, 0, 3}]</code>
MuPAD	<code>limit(f(x), x = 0, Left)</code>	<code>series(f(x), x = 0, 4)</code>
Octave		
Pari		
Reduce	<code>limit!-(f(x), x, 0)</code>	<code>taylor(f(x), x, 0, 3)</code>
Scilab		
Sumit		
Yacas		

	Differentiate: $\frac{d^3 f(x,y)}{dx dy^2}$	Integrate: $\int_0^1 f(x) dx$
Axiom	<code>D(f(x, y), [x, y], [1, 2])</code>	<code>integrate(f(x), x = 0..1)</code>
Derive	<code>DIF(DIF(f(x, y), x), y, 2)</code>	<code>INT(f(x), x, 0, 1)</code>
DoCon		
GAP		
Gmp		
Macsyma	<code>diff(f(x, y), x, 1, y, 2)</code>	<code>integrate(f(x), x, 0, 1)</code>
Magnus		
Maxima	<code>diff(f(x, y), x, 1, y, 2)</code>	<code>integrate(f(x), x, 0, 1)</code>
Maple	<code>diff(f(x, y), x, y\$2)</code>	<code>int(f(x), x = 0..1)</code>
Mathematica	<code>D[f[x, y], x, {y, 2}]</code>	<code>Integrate[f[x], {x, 0, 1}]</code>
MuPAD	<code>diff(f(x, y), x, y\$2)</code>	<code>int(f(x), x = 0..1)</code>
Octave		
Pari		
Reduce	<code>df(f(x, y), x, y, 2)</code>	<code>int(f(x), x, 0, 1)</code>
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Laplace transform	Inverse Laplace transform
Axiom	<code>laplace(e, t, s)</code>	<code>inverseLaplace(e, s, t)</code>
Derive	<code>LAPLACE(e, t, s)</code>	
DoCon		
GAP		
Gmp		
Macsyma	<code>laplace(e, t, s)</code>	<code>ilt(e, s, t)</code>
Magnus		
Maxima	<code>laplace(e, t, s)</code>	<code>ilt(e, s, t)</code>
Maple	<code>inttrans[laplace](e,t,s)</code>	<code>inttrans[invlaplace](e,s,t)</code>
Mathematica	<code>@<< Calculus`LaplaceTransform` LaplaceTransform[e, t, s]</code>	<code>InverseLaplaceTransform[e,s,t]</code>
MuPAD	<code>transform::laplace(e,t,s)</code>	<code>transform::ilaplace(e, s, t)</code>
Octave		
Pari		
Reduce	<code>load_package(laplace)\$ laplace(e, t, s)</code>	<code>load_package(defint)\$ invlap(e, t, s)</code>
Scilab		
Sumit		
Yacas		
	Solve an ODE (with the initial condition $y'(0) = 1$)	
Axiom	<code>solve(eqn, y, x)</code>	
Derive	<code>APPLY_IC(RHS(ODE(eqn, x, y, y-)), [x, 0], [y, 1])</code>	
DoCon		
GAP		
Gmp		
Macsyma	<code>ode_ibc(ode(eqn, y(x), x), x = 0, diff(y(x), x) = 1)</code>	
Magnus		
Maxima	<code>ode_ibc(ode(eqn, y(x), x), x = 0, diff(y(x), x) = 1)</code>	
Maple	<code>dsolve({eqn, D(y)(0) = 1}, y(x))</code>	
Mathematica	<code>DSolve[{eqn, y'[0] == 1}, y[x], x]</code>	
MuPAD	<code>solve(ode({eqn, D(y)(0) = 1}, y(x)))</code>	
Octave		
Pari		
Reduce	<code>odesolve(eqn, y(x), x)</code>	
Scilab		
Sumit		
Yacas		

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Define the differential operator $L = D_x + I$ and apply it to $\sin x$
Axiom	DD : LODO(Expression Integer, e +-> D(e, x)) := D(); L:= DD + 1; L(sin(x))
Derive	
DoCon	
GAP	
Gmp	
Macsyma	load(opalg)\$ L: (diffop(x) - 1)\$ L(sin(x));
Magnus	
Maxima	load(opalg)\$ L: (diffop(x) - 1)\$ L(sin(x));
Maple	id:= x -> x: L:= (D + id): L(sin)(x);
Mathematica	L = D[#, x]& + Identity; Through[L[Sin[x]]]
MuPAD	L:= (D + id): L(sin)(x);
Octave	
Pari	
Reduce	
Scilab	
Sumit	
Yacas	
	2D plot of two separate curves overlayed
Axiom	draw(x, x = 0..1); draw(acsch(x), x = 0..1); [Plot Overlay]
Derive	
DoCon	
GAP	
Gmp	
Macsyma	plot(x, x, 0, 1)\$ plot(acsch(x), x, 0, 1)\$
Magnus	
Maxima	plot(x, x, 0, 1)\$ plot(acsch(x), x, 0, 1)\$
Maple	plot({x, arccsch(x)}, x = 0..1):
Mathematica	Plot[{x, ArcCsch[x]}, {x, 0, 1}];
MuPAD	plotfunc(x, acsch(x), x = 0..1):
Octave	
Pari	
Reduce	load_package(gnuplot)\$ plot(y = x, x = (0 .. 1))\$ plot(y = acsch(x), x = (0 .. 1))\$
Scilab	
Sumit	
Yacas	

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.

	Simple 3D plotting
Axiom	<code>draw(abs(x*y), x = 0..1, y = 0..1);</code>
Derive	<code>[Plot Overlay]</code>
DoCon	
GAP	
Gmp	
Macsyma	<code>plot3d(abs(x*y), x, 0, 1, y, 0, 1)\$</code>
Magnus	
Maxima	<code>plot3d(abs(x*y), x, 0, 1, y, 0, 1)\$</code>
Maple	<code>plot3d(abs(x*y), x = 0..1, y = 0..1):</code>
Mathematica	<code>Plot3D[Abs[x*y], {x, 0, 1}, {y, 0, 1}];</code>
MuPAD	<code>plotfunc(abs(x*y), x = 0..1, y = 0..1):</code>
Octave	
Pari	
Reduce	<code>load_package(gnuplot)\$</code> <code>plot(z = abs(x*y), x = (0 .. 1), y = (0 .. 1))\$</code>
Scilab	
Sumit	
Yacas	

Based on material originally published in *Computer Algebra Systems: A Practical Guide* edited by Michael J. Wester, John Wiley & Sons, Chichester, United Kingdom, ISBN 0-471-98353-5, xvi+436 pages, 1999.