Basic Maple Commands

Command	Description
1. General Commands and Conventions	
$f(a)$	evaluating a function f at a; e.g. $sin(Pi)$
;	command end/result displayed
:	" " /result not displayed
% (previously: ")	output of previous line
cursor on *name*, click on *help*	help for *name*
$settime := time(); expression;$ $time() - settime;$	to get elapsed time for computing an expression
$a := expression;$	assignment
$a\hat{\ }n;$	n-th power of a
$sqrt(a);$	the (exact) square root of a
$evalf(expression, n);$	numerical value of *expression* to n-digit accuracy
$evalb(a = b);$	logical comparison (gives *true* or *false*)
$a[n];$	n-th element of list a
$plot(expression, x = a..b);$	2-dim plot of *expression* for x between a and b
$plot3d(expr, x = a..b, y = c..d);$	3-dim plot of *expr* for x between a and b and y between c and d
$f := x-> expr$	definition of a one-variable function $f(x)$
$f := [x, y, \ldots]-> expr$	definition of multi-variable function $f(x, y, \ldots)$
$a := proc(x, y)\, local\, z, w; ...; end;$	definition of subroutine a
2. Elementary Number Theory	
$iquo(a, b);$ or $floor(a/b);$	integral part of the quotient a/b
$irem(a, b);$ or $modp(a, b);$	remainder of division of a by b
$frac(x);$	the fractional part of x
$igcd(a, b);$	the gcd of a and b
$igcdex(a, b, 'x', 'y');$	the extended gcd
$x; y;$	to extract the values of the above extended gcd
$ithprime(n);$	the n-th prime number
$isprime(n);$	test whether or not n is prime (gives *true* or *false*)
$ifactor(n);$	factor n into its prime factors
$a\&\hat{\ }n \bmod m;$ or $Power(a, n) \bmod m;$	compute a^n mod m efficiently

Command	Description
3. Sets and Lists: Basic Structure	
$s := \{1,2,3,4,5\};$	defines a set s: an unordered sequence of elements
$a := [1,2,3,4,5];$	defines a list a: an ordered sequence of elements
$s := \{seq(f, i = 1..5)\};$	create the set s consisting of the elements $f(1),\ldots,$ $f(5)$; here f is an expression (depending on i)
$a := [seq(f, i = 1..5)];$	create the list a consisting of the elements $f(1)$, $\ldots$, $f(5)$; here f is an expression (depending on i)
$nops(a);$	the number of elements in list a
$a[i]$	the ith element of the list a
$[a[i..j]]$ or $[op(i..j, a)]$	the list consisting of elements i through j (inclusive)
$select(k{-}{>}\ k < m\, or\, k > n, a);$	list a with elements m through n dropped
$member(e, a);$	test whether e occurs in list a ($true$ or $false$)
$member(e, a,' p'); p;$	the position(s) at which e occurs in a
$type(s, set);$	check whether s is a set (has type "set"); gives $true$ or $false$
$type(a, list);$	check whether s is a list (has type "list"); gives $true$ or $false$
4. Operations on Sets and Lists	
$s := convert(a, set);$	convert a list to a set
$a := convert(s, list);$	convert a set to a list
$s\ \ union\ \ t$; or $`union`(s, t, \ldots)$	combine sets s, t, $\ldots$, removing repeated elements
$s\ \ intersect\ \ t;$	intersection of sets s and t
$s\ \ minus\ \ t$	the set of elements which are in s but not in t
$[op(a), op(b), \ldots]$	concatenate (join) the lists $a, b, \ldots$
$a := [e, op(a)];$	add element e at the beginning of list a
$a := [op(a), e];$	add element e at the end of list a
$a := subsop(i = e, a);$	replace the ith element of the list a by e
$a := subsop(i = NULL, a);$	delete ith element from list a
$[a[1..n-1], e, a[n..nops(a)]];$	insert e at position n in list a
$sort(a);$	sort the elements of list a (into a standard order)
$[select(bool, a)];$	list consisting of the elements of a for which the boolean-valued function *bool* is true
$map(f, a);$	apply the function f to each element of the list a

Command	Description
5. Character Strings	
$str := "This\,is\,a\,string"$;	defining a character string
$length(str)$;	the number of characters in a string
$substring(str, m..n)$;	extract a substring from string str starting with the mth and ending with the nth character
$[seq(substring(str, k..k), k = 1..$ $length(str)]$	give the list of characters in a string
$searchtext(st, str)$	find the place where st occurs in string str
$s1.s2.\ldots$ or $cat(s1, s2, \ldots)$	join the strings $s1$, $s2$, … together
$convert(expr, string)$;	convert an expression to a string (textual form)
$type(str, string)$	check whether str is a string ($true$ or $false$)
6. Boolean expressions	
$b := true; b := false$;	assigning true/false to the variable b
=, <>, <, <=, >, >=	relation operators (equal, not equal, less than, etc.); can be used to form boolean expressions
and, *or*, *not*	logical operators ($\rightarrow$ boolean expressions)
$evalb(bool)$	evaluate the boolean expression $bool$ (gives $true$ or $false$)
$type(b, boolean)$	check whether b is a boolean expression ($true$ or $false$)
7. Looping control	
$for\ i\ to\ m\ do; expr; od$;	evaluate $expr$ repeatedly with i varying from 1 to m in steps of 1
$for\ i\ from\ n\ to\ m\ by\ s\ do; expr; od$;	evaluate $expr$ repeatedly with i varying from n to m in steps of s
$while\ test\ do; expr; od$;	evaluate $expr$ until $test$ becomes false
$for\ i\ from\ n\ to\ m\ by\ s\ while\ test$ $do; expr; od$;	evaluate $expr$ repeatedly with i varying from n to m in steps of s as long as $test$ is true
$RETURN(expr)$	(explicit) return from a subroutine, assigning the value $expr$ to the subroutine
8. Conditionals	
$if\ test\ then\ statmt\ fi$;	execute the statement (sequence) $statmt$ only if $test$ is true
$if\ test\ then\ statmt_1\ else\ statmt_2\ fi$;	execute the statement (sequence) $statmt_1$ if $test$ is true, otherwise execute $statmt_2$

Command	Description
9. Complex Numbers	
$z := x + y * I;$	defining a complex number
$abs(expr);$	the absolute value of *expr*
$argument(expr)$	the argument of *expr*
$Re(expr); Im(expr);$	the real and imaginary part of *expr*
$conjugate(expr);$	the complex conjugate of *expr*
$evalc(expr)$	evaluating an expression (as a complex number)
$convert(expr, polar)$	convert *expr* to its polar form
$type(expr, complex)$	check that *expr* has type “complex”
10. Polynomials	
$f := x\hat{}n + a_1 * x\hat{}(n-1) + \dots;$	defining a polynomial $f = f(x)$ (assuming that x has no value)
$type(f, polynom(integer, x))$	check that f is an integer polynomial in x
$degree(f, x)$	degree of f in x
$coeff(f, x, n)$	extract the coefficient of x^n in f
$coeffs(f, x)$	list of coefficients of $f(x)$
$lcoeff(f, x)$	the leading (highest) coefficient of $f(x)$
$tcoeff(f, x)$	the constant (trailing) coefficient of $f(x)$
$collect(f, x)$	collect all coefficients of f which have the same powers in x
$expand(expr)$	distribute products over sums
$sort(f)$	sort into decreasing order
$subs(x = a, f)$	evaluate $f(x)$ at $x = a$
$Eval(f, x = a)\ mod\ p;$	evaluate $f(x)$ (mod p) at $x = a$
$f\ mod\ n;$	reduce the coefficients of f modulo n
$quo(f, g, x);\ rem(f, g, x);$	the quotient and remainder of division of f by g (viewed as polynomials in x)
$gcd(f, g, x)$	the greatest common divisor of $f(x)$ and $g(x)$
$gcd(f, g, x,' s',' t')$	the extended Euclidean algorithm of $f(x)$ and $g(x)$; i.e. s, t satisfy $f * s + g * t = g := gcd(f, g)$
$factor(f)$	factor f into its irreducible factors
$Factor(f)\ mod\ p$	factor f modulo p
$roots(f)$	find the rational roots of f
$interp(x, y, t)$	The Lagrange Interpolation polynomial