

Basic Maple Commands

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

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

Command	Description
5. Character Strings	
<pre>str := "This is a string"; length(str); substring(str, m..n); [seq(substring(str, k..k), k = 1.. length(str)] searchtext(st, str) cat(s1, s2, ...) convert(expr, string); type(str, string)</pre>	<p>defining a character string</p> <p>the number of characters in a string</p> <p>extract a substring from string <i>str</i> starting with the <i>m</i>th and ending with the <i>n</i>th character</p> <p>give the list of characters in a string</p> <p>find the place where <i>st</i> occurs in string <i>str</i></p> <p>join the strings <i>s1</i>, <i>s2</i>, ... together</p> <p>convert an expression to a string (textual form)</p> <p>check whether <i>str</i> is a string (<i>true</i> or <i>false</i>)</p>
6. Boolean expressions	
<pre>b := true; b := false; =, <>, <, <=, >, >= and, or, not evalb(bool) type(b, boolean)</pre>	<p>assigning true/false to the variable <i>b</i></p> <p>relation operators (equal, not equal, less than, etc.); can be used to form boolean expressions</p> <p>logical operators (\rightarrow boolean expressions)</p> <p>evaluate the boolean expression <i>bool</i> (gives <i>true</i> or <i>false</i>)</p> <p>check whether <i>b</i> is a boolean expression (<i>true</i> or <i>false</i>)</p>
7. Looping control	
<pre>for i to m do; expr; od; for i from n to m by s do; expr; od; while test do; expr; od; for i from n to m by s while test do; expr; od; return(expr)</pre>	<p>evaluate <i>expr</i> (= sequence of commands) repeatedly with <i>i</i> varying from 1 to <i>m</i> in steps of 1</p> <p>evaluate <i>expr</i> repeatedly with <i>i</i> varying from <i>n</i> to <i>m</i> in steps of <i>s</i></p> <p>evaluate <i>expr</i> until <i>test</i> becomes false</p> <p>evaluate <i>expr</i> repeatedly with <i>i</i> varying from <i>n</i> to <i>m</i> in steps of <i>s</i> as long as <i>test</i> is true</p> <p>(explicit) return from a subroutine, assigning the value <i>expr</i> to the subroutine</p>
8. Conditionals	
<pre>if test then statmt fi; if test then statmt₁ else statmt₂ fi;</pre>	<p>execute the statement (sequence) <i>statmt</i> only if <i>test</i> is true</p> <p>execute the statement (sequence) <i>statmt₁</i> if <i>test</i> is true, otherwise execute <i>statmt₂</i></p>

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