Maple 11 Cheat Sheet

Syntax

`;` Ends a command with a semicolon. *e.g.* `5+6; plot(x);`

`:` Suppresses the display of output by ending a command with a colon. Useful for lengthy outputs or loading packages. *e.g.* `with(plots): 5000!:`

`:=` Assigns an expression to a variable. *e.g.* `a:=3; b:=a+x;` assigns 3 to a and $3 + x$ to b. `x:='x';` unassigns the variable x.

`=` Defines mathematical equations. *e.g.* `y = x^2 + 3*x + 4;` produces the equation $y = x^2 + 3x + 4$.

`%` Refers to the last result. n of the % symbols refers to the n^{th} previous result. *e.g.* `%%%` gives the third previous presult.

`f:=(x,y,...)->...` Defines a function. *e.g.* `f := (x,y) -> x^2+y^2;` defines the function $f(x, y) = x^2 + y^2$. `f(0,1)` evaluates $f(0, 1) = 0^2 + 1^2 = 1$. `plot3d(f(x,y), x=0..1, y=0..1);` plots the function.

`L:=[x1, x2, ..., xn]` Defines a list (ordered sequence) L of expressions x_1, x_2, ..., x_n. Refer to the n^{th} list item by $L[n]$. To extract the contents of a list, use the empty selection operator []. *e.g.* `A:=[1,2,3]; A[3];` returns 3. `A[]` returns 1,2,3.

`S: :={x1, x2, ..., xn}` Defines a set S of expressions x_1, x_2, ..., x_n. Use the empty selection operator[] to extract the contents of a set. *e.g.* `S:={5,3,3,2,1}; S[];` returns 1,3,4,5.

`?`*`topic`* Displays help on topic.

All identifiers (variables and functions) are **case sensitive**. *e.g.* `X` is different from `x`. `Pi` and `pi` are different!

In general, a function whose name begins with a capital letter is an inert form of the function who has the same name but begins with lower case. Inert functions are unevaluated and may be manipulated and printed in a prettyprinted format. *e.g.* `Int(x,x);` returns $\int xdx$ and is the inert form of `int(x,x);`, which evaluates to $x^2/2$.

Usages

Right-click an expression to display a context-sensitive menu of applicable options.

!!! Click the $\boxed{\text{!!!}}$ icon to execute the entire worksheet. Useful when you have changed expressions that affect subsequent commands.

Keyboard Shortcuts

`Enter`	Evaluate
`Ctrl + =`	Evaluate and display inline (Document Mode)
`Ctrl + Space`	Complete symbol/command
`F5`	Toggle Math/Text entry (Document Mode) Toggle 2-D/1-D Math entry (Worksheet Mode)
`Ctrl + F1`	Maple help

Defined Constants

`Pi`	$\pi \approx 3.14159265...$
`I`	complex number $I = \sqrt{-1}$
`infinity`	∞
`gamma`	Euler's constant $\gamma \approx 0.5772156649...$
`Catalan`	Catalan's constant $\approx 0.915965594...$
`exp(1)`	$e \approx 2.718281828$

Commands

General

`with(`*`package`*`);` Loads the specified Maple package.

`unassign(`*`var`*`);` Deletes a value stored in the given variable. *e.g.* `a:=1; unassign('a');` unassigns the identifier a so that it does not contain the value of 1 anymore.

`restart;` Clears internal memory. The settings of all identifiers are resetted.

Common Mathematical Operations

`x + y - z;`	addition and subtraction
`x * y;`	multiplication
`x / y;`	division
`x^y;`	power x^y
`sqrt(x);`	square root $\sqrt{x}$
`exp(x);`	exponential e^x
`ln(x);`	natural log $ln(x)$
`log[b](x);`	logarithm $log_b(x)$
`surd(x,n);`	real n^{th} root $\sqrt[n]{x}$
`sin(x); cos(x); tan(x);`	trigonometric functions

`arcsin(x); arccos(x);`
`arctan(x);`
	inverse trig functions

Numerical Manipulation

`eval(`*`expression`*`);` Evaluates the given expression. *e.g.* `a:=b^2; b:=c+1; c:=2; eval(a);` returns 9.

`eval(`*`expression`*`, x=`*`value`*`);` Evaluates expression at the given point $x = value$. *e.g.* `eval(x^2+5*x, x=1);` evaluates the polynomial $x^2 + 5x$ at $x = 1$ and returns 6.

`eval(expression, {x=value1, y=value2,...});` Evaluates expression at the given points $x = value1$, $y = value2$,...

`subs(x=value,expression);` Substitutes the given value into expression. *e.g.* `subs(x=2,x^2+2*x+1);` gives 9. `subs(x=0, sin(x)/cos(x));` returns $sin(0)/cos(0)$.

`evalf(expression);` Numerically evaluates expression and returns its decimal approximation. *e.g.* `evalf(Pi);` returns 3.141592654.

`value(expression);` Evaluates the given inert expression. *e.g.* `F:=Sum(i,i=1..5); value(F);` evaluates the inert sum $\sum_{i=1}^{5} i$ and returns 15.

`assume(x, domain);` Restricts variable x to *domain*. Examples of *domain* are `positive`, `negative`, `posint`, `integer`, `real`, and `complex`. *e.g.* `assume(x, 'integer');` forces x to be an integer.

`assume(relation);` Enforces the given relational property. *e.g.* `assume(x > 0);` restricts x to be positive.

`additionally(x, domain); additionally(relation);` Places further restrictions on the given variable. Usages are similar to that for `assume`. *e.g.* `assume(x, real); additionally(x > 0);` forces x to be real as well as positive.

Algebra

`simplify(expression);` Applies simplification rules to the given expression. *e.g.* `simplify(cos(Pi*cos(x)^2+Pi*sin(x)^2));` returns -1.

`collect(expression, variable);` Combines like terms in expression with respect to the given variable. *e.g.* `collect(a^2*x+b*x+5, x);` returns $5 + (a^2 + b)x$.

`normal(expression);` Simplifies and normalizes the given rational expression so that the result is of factored normal form, where the numerator and denomator are relatively prime polynomials with integer coefficients. *e.g.* `normal(1/x+x/(x+1));` returns $\frac{x+1+x^2}{x(x+1)}$.

`factor(expression);` Factors the given expression of a multivariate polynomial. Does NOT factor integers or integer coefficients in a polynomial. *e.g.* `factor(4*x^2+12*x+8)` returns $4(x + 1)(x + 2)$.

`ifactor(expression);` Factors an integer or rational number into a product of primes. *e.g.* `ifactor(24/19);` returns $\frac{(2)^3(3)}{(19)}$. `ifactor(2^10-1);` returns (3)(11)(31).

`expand(expression);` Distributes the given expression. *e.g.* `expand((x+3)*(x+5));` returns $x^2 + 8x + 15$.

`solve(equations, variables);` Solves for the unknown variables in the given equations or inequalities.

e.g. `solve(x^2-25=0, x);` solves the equation $x^2 - 25 = 0$ and returns 5,-5.

e.g. `solve({x+y+z = 6, x-y+2*z = 5, 2*x+2*y+z = 9}, [x, y, z])` solves the system of three equations and returns the solution [[x = 1, y = 2, z = 3]].

e.g. `solve(abs(x+5) > 3, x);` solves the inequality $|x+5| > 3$ and returns $RealRange(Open(-2), infinity)$, $RealRange(-infinity, Open(-8))$.

`fsolve(equations, variable, [complex]);` Numerically solves for the unknown *variable* in *equations*. Use the `complex` option to find a complex solution. *e.g.* `fsolve(x^2+x+1 = 0, x, complex);` returns $-.5000000000 - .8660254038I$, $-.5000000000 + .8660254038I$.

Calculus

`diff(f, x1, ..., xj);` Differentiates f with respect to variables $x_1, \ldots, x_j$. *e.g.* `diff(sin(x), x);` takes the first derivative of $sin(x)$. `diff(f(x,y),x,y);` computes $\frac{\partial^2}{\partial y \partial x} f(x, y)$.

`diff(f, x$n);` Computes the n^{th} derivative of f. *e.g.* `diff(x^4, x$2);` computes the second derivative of x^4 and returns $12x^2$.

`int(f, x);` Computes an indefinite integral of f with respect to the variable x. *e.g.* `int(cos(x), x);` computes $\int cos(x)dx$ and returns $sin(x)$.

`int(f, x=a..b);` Computes the definite integral of f with respect to the variable x on the interval from a to b. *e.g.* `int(x^2, x=0..2);` computes $\int_0^2 x^2 dx$ and returns 8/3.

`limit(f, x=a, [dir]);` Computes the limit of f as x approaches a. a can be any algebraic expression or `infinity`. Direction *dir* is optional and is real bidirectional by default (except for ∞ and $-\infty$). Possible values of direction are `left`, `right`, `real`, and `complex`. *e.g.* `limit(1/exp(x), x=infinity);` computes $\lim_{x \to \infty} \frac{1}{e^x}$ and returns 0.

`sum(f, k=m..n);` Returns the summation $\sum_{k=m}^{n} f(k)$. *e.g.* `sum(x^2, x=1..n);` computes $\sum_{x=1}^{n} x^2$.

Plots

`plot(`*f*`, `*`x=xmin..xmax, options`*`);` Creates a **two-dimensional** plot of the real function $f(x)$ over the horizontal range from *xmin* to *xmax*. Options are specified in the form `option=value` (see box below).

- *f* is a function with an independent variable. *e.g.* `plot(x^2, x=-5..5);`.
- *f* is represented parametrically: `[x(t),y(t),t=t0..t1]`. *e.g.* `plot([cos(t),sin(t),t=-2*Pi..2*Pi]);`
- *f* is a list of functions to be graphed on the same plot: `[f1, f2, ..., fn]`. *e.g.* `plot([1,x,x^2], x=-2..2);` puts the functions $y = 1$, $y = x$, and $y = x^2$ on the same plot.

`implicitplot(`*`eqn, x=xmin..xmax, y=ymin..ymax, options`*`);` In the `plots` package. i.e. Must be preceded by `with(plots);` Creates the two-dimensional plot of an implicitly defined curve *eqn* on the specified intervals: $[xmin, xmax]$ and $[ymin, ymax]$. Options are specified in the form `option=value` (see box below). *e.g.* `implicitplot(x^2+y^2=1, x=-1..1, y=-1..1);`.

`inequal(`*`ineqs, x=xmin..xmax, y=xmin..xmax, options`*`);` In the `plots` package. Plots regions defined by inequalities *ineqs* in the specified x and y intervals. Options are in the form `optionsfeasible / optionsopen / optionsclosed / optionsexcluded` = (`optionsList`), where *optionsList* is of the format (`option=value, option2=value2, ...`). *e.g.* `inequal(x+y>0, x-y<=1, x=-3..3, y=-3..3, optionsexcluded=(color=blue,thickness=2));`

Options for Plot, Implicitplot, and Inequal

Type of axes	`axes=boxed/frame/none/normal`
Color of curves	`color=blue/black/green/red/etc.`
Determine input discontinuities	`discont=true/false`
Draw gridlines	`gridlines=true/false`
Label Axes	`labels=[x,y]`
Scaling	`scaling=constrained/unconstrained`
Line thickness	`thickness=number`
Title	`title="plot title"`
Min/max y values	`y=ymin..ymax`
View window	`view=[xmin..xmax,ymin..ymax]`

`plot3d(`*`f, x=a..b, y=c..d, options`*`);` Creates a **three-dimensional** plot of the real function $f(x, y)$ over the horizontal range $[a, b]$ and vertical range $[c, d]$. Options are specified in the form `option=value` (see box below).

- *f* is a function with two independent variables. *e.g.* `plot(sin(x+y), x=-1..1, y=-1..1);`.
- *f* is represented parametrically: `[f1(x,y), f2(x,y), f3(x,y)]`. *e.g.* `plot3d([x*sin(x)*cos(y), x*cos(x)*cos(y), x*sin(y)], x=0..2*Pi, y=0..Pi);`
- *f* is a list of functions to be graphed on the same plot: `[f1(x,y), f2(x,y), ..., fn(x,y)]`. If there are three functions, use the `plotlist` option to avoid a parametric plot. *e.g.* `plot3d([sin(x*y),cos(x*y),x+y], x=-1..1, y=-1..1, plotlist);` puts the functions $z = sin(xy)$, $z = cos(xy)$, and $z = x + y$ on the same plot.

`implicitplot3d(`*`eqn, x=a..b, y=c..d, z=i..j, options`*`);` In the `plots` package. Creates the three-dimensional plot of an implicitly defined surface *eqn* on the specified intervals: $x = [a, b]$, $y = [c, d]$ and $z = [i, j]$. Options are specified in the form `option=value` (see box below). *e.g.* `implicitplot3d(x^2+y^2+z^2=1, x=-1..1, y=-1..1, z=-1..1);`.

Options for Plot3d and Implicitplot3d

Type of axes	`axes=boxed/frame/none/normal`
Color of curves	`color=blue/black/green/red/etc.`
Contours	`contours=number`
Coordinate System	`coords=cartesian/cylindrical/spherical/etc.`
Grid Dimensions	`grid=[m,n]`
Label Axes	`labels=[x,y,z]`
Scaling	`scaling=constrained/unconstrained`
Line thickness	`thickness=number`
Title	`title="plot title"`
View window	`view=[xmin..xmax,ymin..ymax, zmin..zmax]`

`animate(`*`plotcommand, plotargs, t=a..b, options`*`);` In the `plots` package. Creates a 2-D or 3-D animation on paramter t, ranging from a to b. *plotcommand* is a Maple command that generates a 2-D or 3-D plot (*e.g.* `plot, plot3d, implicitplot`). *plotargs* is a list of arguments to the plot command. Possible options are those used in the `plot` command or the following:

Number of frames	`frames=n`
Display a trace of n frames	`trace=n`

e.g. `animate(plot, [A*sin(x), x=0..10], A=0..2, frames=50, trace=5);`

`display(L, options);` In the `plots` package. Combines the list L of plot structures into a single plot or animation. *options* are those used for `plot` or `plot3d`.

e.g.
```
with(plots):
p1:=plot3d(sin(x*y), x=-Pi..Pi, y=-Pi..Pi):
p2:=plot3d([x+y, sin(x)], x=-Pi..Pi, y=-Pi..Pi):
display([p1,p2], axes=boxed, title="test plot");
```