Sage Quick Reference (Basic Math)

Peter Jipsen, version 1.1 (w/modification by nu)
latest version at `wiki.sagemath.org/quickref`

Aim: map standard math notation to Sage commands

Notebook (and commandline)

Evaluate cell: ⟨shift-enter⟩
com⟨tab⟩ tries to complete *command*
command`?`⟨tab⟩ shows documentation
command`??`⟨tab⟩ shows source
`a.`⟨tab⟩ shows all methods for object `a` (more: `dir(a)`)
`search_doc('`*string or regexp*`')` shows links to docs
`search_src('`*string or regexp*`')` shows links to source
`lprint()` toggle LaTeX output mode
`version()` print version of Sage
Insert cell: click on blue line between cells
Delete cell: delete content then backspace

Numerical types

Integers: $\mathbb{Z}$ = `ZZ` e.g. `-2 -1 0 1 10^100`
Rationals: $\mathbb{Q}$ = `QQ` e.g. `1/2 1/1000 314/100 -42`
Decimals: $\mathbb{R} \approx$ `RR` e.g. `.5 0.001 3.14 -42`
Complex: $\mathbb{C} \approx$ `CC` e.g. `1+i 2.5-3*i`

Basic constants and functions

Constants: π = `pi` e = `e` i = `i` ∞ = `oo`
Approximate: `pi.n(digits=18)` = 3.14159265358979324
Functions: `sin cos tan sec csc cot sinh cosh tanh sech csch coth log ln exp`
ab = `a*b` $\frac{a}{b}$ = `a/b` a^b = `a^b` $\sqrt{x}$ = `sqrt(x)`
$\sqrt[n]{x}$ = `x^(1/n)` $|x|$ = `abs(x)` $\log_b(x)$ = `log(x,b)`
Symbolic variables: e.g. `t,u,v,y = var('t u v y')`
Define function: e.g. $f(x) = x^2$
As symbolic function (can integrate, etc): `f(x)=x^2` or
As Python function: `f=lambda x: x^2` or
`def f(x): return x^2`

Operations on expressions

`factor(...)` `expand(...)` `(...).simplify_...`
Symbolic equations: `f(x)==g(x)`
`_` is previous output
`_+a` `_-a` `_*a` `_/a` manipulates equation
Solve $f(x) = g(x)$: `solve(f(x)==g(x),x)`
`solve([f(x,y)==0, g(x,y)==0], x,y)`
`find_root(f(x), a, b)` find $x \in [a,b]$ s.t. $f(x) \approx 0$
$\sum_{i=k}^{n} f(i)$ = `sum([f(i) for i in [k..n]])`
$\prod_{i=k}^{n} f(i)$ = `prod([f(i) for i in [k..n]])`

Calculus

$\lim_{x\to a} f(x)$ = `limit(f(x), x=a)`
$\lim_{x\to a^-} f(x)$ = `limit(f(x), x=a, dir='minus')`
$\lim_{x\to a^+} f(x)$ = `limit(f(x), x=a, dir='plus')`
$\frac{d}{dx}(f(x))$ = `diff(f(x),x)`
$\frac{\partial}{\partial x}(f(x,y))$ = `diff(f(x,y),x)`
`diff` = `differentiate` = `derivative`
$\int f(x)dx$ = `integral(f(x),x)`
`integral` = `integrate`
$\int_a^b f(x)dx$ = `integral(f(x),x,a,b)`
Taylor polynomial, deg n about a: `taylor(f(x),x,a,n)`

2d graphics

`line([(`x_1`,`y_1`),...,(`x_n`,`y_n`)],`*options*`)`
`polygon([(`x_1`,`y_1`),...,(`x_n`,`y_n`)],`*options*`)`
`circle((`x`,`y`),`r`,`*options*`)`
`text("txt",(`x`,`y`),`*options*`)`
options as in `plot.options`,
e.g. `thickness=`*pixel*, `rgbcolor=(`r`,`g`,`b`)`, `hue=`h,
where $0 \le r,b,g,h \le 1$
use option `figsize=[w,h]` to adjust aspect ratio
`plot(f(x),`$x_{\min}$`,`$x_{\max}$`,`*options*`)`
`parametric_plot((f(t),g(t)),`$t_{\min}$`,`$t_{\max}$`,`*options*`)`
`polar_plot(f(t),`$t_{\min}$`,`$t_{\max}$`,`*options*`)`
combine graphs: `circle((1,1),1)+line([(0,0),(2,2)])`
`animate(`*list of graphics objects* `,`*options*`).show(delay=20)`

3d graphics

`line3d([(`x_1`,`y_1`,`z_1`),...,(`x_n`,`y_n`,`z_n`)],`*options*`)`
`sphere((`x`,`y`,`z`),`r`,`*options*`)`
`tetrahedron((`x`,`y`,`z`),`*size*`,`*options*`)`
`cube((`x`,`y`,`z`),`*size*`,`*options*`)`
`octahedron((`x`,`y`,`z`),`*size*`,`*options*`)`
`dodecahedron((`x`,`y`,`z`),`*size*`,`*options*`)`
`icosahedron((`x`,`y`,`z`),`*size*`,`*options*`)`
options e.g. `aspect_ratio=[1,1,1] color='red' opacity`
`plot3d(f(`x`,`y`),[`x_b`,`x_e`],[`y_b`,`y_e`],`*options*`)`
add option `plot_points=[`m`,`n`]` or use `plot3d_adaptive`
`parametric_plot3d((f(`t`),g(`t`),h(`t`)),[`t_b`,`t_e`],`*options*`)`
`parametric_plot3d((f(`u`,`v`),g(`u`,`v`),h(`u`,`v`)),`
`[`u_b`,`u_e`],[`v_b`,`v_e`],`*options*`)`
use `+` to combine graphics objects

Discrete math

$\lfloor x \rfloor$ = `floor(x)` $\lceil x \rceil$ = `ceil(x)`
Remainder of n divided by k = `n%k` $k|n$ iff `n%k==0`
$n!$ = `factorial(n)` $\binom{x}{m}$ = `binomial(x,m)`
ϕ = `golden_ratio` $\phi(n)$ = `euler_phi(n)`
Strings: e.g. `s = 'Hello'` = `"Hello"` = `""+"He"+'llo'`
`s[0]='H'` `s[-1]='o'` `s[1:3]='el'` `s[3:]='lo'`
Lists: e.g. `[1,'Hello',x]` = `[]+[1,'Hello']+[x]`
Tuples: e.g. `(1,'Hello',x)` (immutable)
Sets: e.g. $\{1,2,1,a\}$ = `Set([1,2,1,'a'])` $(= \{1,2,a\})$
List comprehension $\approx$ set builder notation, e.g.
$\{f(x) : x \in X, x > 0\}$ = `Set([f(x) for x in X if x>0])`

Linear algebra

$\begin{pmatrix}1\\2\end{pmatrix}$ = `vector([1,2])`
$\begin{pmatrix}1&2\\3&4\end{pmatrix}$ = `matrix([[1,2],[3,4]])`
$\begin{vmatrix}1&2\\3&4\end{vmatrix}$ = `det(matrix([[1,2],[3,4]]))`
Av = `A*v` A^{-1} = `A^-1` A^t = `A.transpose()`
methods: `nrows()` `ncols()` `nullity()` `rank()` `trace()`...

Sage modules and packages

`from` *module_name* `import *` (many preloaded)
e.g. `calculus coding combinat crypto functions games geometry graphs groups logic matrix numerical plot probability rings sets stats`
`sage.`*module_name*`.all.`⟨tab⟩ shows exported commands
Std packages: Maxima GP/PARI GAP Singular R Shell...
Opt packages: Biopython Fricas(Axiom) Gnuplot Kash...
`%`*package_name* then use package command syntax
`time` *command* to show timing information