Sage quick reference sheet

Revision: May 15, 2011

By Anders Damsgaard Christensen, anders.damsgaard@geo.au.dk.
This document is a summary of basic Sage commands, presented in the tutorial and the Sage quick reference, http://wiki.sagemath.org/quickref.

Interactive Sage session control

`<cmd>?`	Display help on command
`<cmd>??`	Display source code on command
`help(<class>)`	Display help on class
`_`	Previous output
`__`	Next previous output
`_oh`	List of all output
`%hist`	Listing of all input
`%macro <name> #-#`	Save input lines in range # to # as macro
`!<cmd>`	Execute external shell command
`logstart <filename>`	Log input (and optionally output)
`load "<filename>"`	Load and execute log
`save(<obj>,<fname>)`	Save object. Save as ASCII text with `open(...,'w')` and `write(str(<obj>))`
`load('<filename>')`	Load contents from file
`save_session(<name>)`	Save entire session
`load_session(<name>)`	Load session (`*.sobj`)

Arithmetic

`a*b` $= ab$, `a/b` $= \frac{a}{b}$, `a**b` $= a^b$, `a%b`: remainder
`a**(1/b)` $= \sqrt[b]{a}$, `abs(a)` $= |a|$, `log(a,b)` $= \log_b(a)$
`sum(f(i) for i in (k..n))` $\sum_{i=k}^{n} f(i)$
`prod(f(i) for i in (k..n))` $\prod_{i=k}^{n} f(i)$
`[...,...]`: list, `(...,...)`: tuple (immutable)
`1, 2, ...`: Integer numbers
`1.0, 1., .2, 1.2`: Floating-point numbers

Matrix algebra

`v = vector([1,2])`	Create row vector (use `.column` to turn)
`matrix([[1,2],[3,4]])`	Create matrix

A^{-1} = `A^-1`, A^t = `A.transpose()`, $|A|$ = `det(A)`
Solve $Ax = v$: `A\v` or `A.solve_right(v)`
Solve $xA = v$: `A.solve_left(v)`

Numbers

`ZZ`: integers, `QQ`: rationals, `RR`: reals, `CC`: complex, `CDF` or `RDF`: double precision (**c**omplex & **r**eal)
Primes: `rime_range(n,m)`, `is_prime`, `next_prime`,
Factor: `factor(n)`

Constants and functions

`pi` $= \pi$, `e` $= e$, `i` $= i$, `oo` $= \infty$,
`golden_ratio` $= \phi$, `euler_gamma` $= \gamma$
`simplify(...)`: Simplify symbolic/numeric expression
Loop: `for i in range(n): <function(i)>`

Symbolic expressions

Relations: `==`, `!=`, `>`, `<`, `<=`, `>=`
Boolean: `or`, `and`, `not`, `in`, `not in`, `is`, `is not`

`x,a = var('x a')`	Define symbolic variables
`f(x) = x**2`	Define symbolic expression, e.g. $f(x) = x^2$

Calculus

`limit(f(x),x=a)`	$\lim_{x\to a} f(x)$
`diff(f(x),x)`	$\frac{\partial}{\partial x}(f(x))$
`diff(f(x,y),x)`	$\frac{\partial}{\partial x}(f(x,y))$
`integral(f(x),x)`	$\int f(x)dx$
`integral(f(x),x,a,b)`	$\int_a^b f(x)dx$

`numerical_integral(f(x),x,a,b)` $\approx \int_a^b f(x)dx$

Linear algebra

`K^n`	Vector space, e.g. `QQ^3`
`span(<vectors>,field)`	Vector space

Output, plots and graphs

`latex(<expr>)`: Display LaTeX-math syntax for expression
`print("<str>")`: Print string to console
`text("<txt>",(`x`,`y`),options)`
`plot(f(`x`),(`x`, `$x_{\min}$`, `$x_{\max}$`),options)`
`plot3d(f(`x,y`),(`x`,`x_b`,`x_e`),(`y`,`y_b`,`y_e`),options)`
`animate(<list>, options).show(delay=20)`
Options: as in `<plot>.options`,
Combine plots with the + operator.
3D viewers: `jmol`, `tachyon`, `java3d`, `canvas3d`

Python functions and modules

`import <module>`	Import Python module, 3rd party module or custom `*.py` file
`help(<module>)`	Show module documentation
`def func(...):`	Define function, followed by indented code block. Return: `return ...`. Put `@interact` on the line before the function for interactive behavior

Profiling and debugging

`%time <cmd>`	Time duration of execution of command (for blocks of code, use `cputime()`
`%prun <cmd>`	Profiling of command execution
`%pdb`	Turn on interactive debugger

Python reference manual

http://docs.python.org/reference/index.html

Sage reference manual

http://sagemath.org/doc/reference/

Sage tutorial

http://sagemath.org/doc/tutorial/