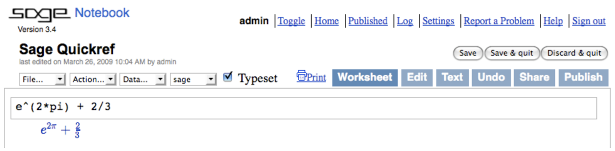
Sage Quick Reference

William Stein (based on work of P. Jipsen) (mod. by nu)

Notebook



Evaluate cell: ⟨shift-enter⟩
Evaluate cell creating new cell: ⟨alt-enter⟩
Split cell: ⟨control-;⟩
Join cells: ⟨control-backspace⟩
Insert math cell: click blue line between cells
Insert text/HTML cell: shift-click blue line between cells
Delete cell: delete content then backspace

Command line

com⟨tab⟩ complete *command*
`*bar*?` list command names containing "*bar*"
command`?`⟨tab⟩ shows documentation
command`??`⟨tab⟩ shows source code
`a.`⟨tab⟩ shows methods for object `a` (more: `dir(a)`)
`a._`⟨tab⟩ shows hidden methods for object `a`
`search_doc("`*string or regexp*`")` fulltext search of docs
`search_src("`*string or regexp*`")` search source code
`_` is previous output

Numbers

Integers: $\mathbb{Z}$ = `ZZ` e.g. `-2 -1 0 1 10^100`
Rationals: $\mathbb{Q}$ = `QQ` e.g. `1/2 1/1000 314/100 -2/1`
Reals: $\mathbb{R} \approx$ `RR` e.g. `.5 0.001 3.14 1.23e10000`
Complex: $\mathbb{C} \approx$ `CC` e.g. `CC(1,1) CC(2.5,-3)`
Double precision: `RDF` and `CDF` e.g. `CDF(2.1,3)`
Mod n: $\mathbb{Z}/n\mathbb{Z}$ = `Zmod` e.g. `Mod(2,3) Zmod(3)(2)`
Finite fields: $\mathbb{F}_q$ = `GF` e.g. `GF(3)(2) GF(9,"a").0`
Polynomials: $R[x,y]$ e.g. `S.<x,y>=QQ[] x+2*y^3`
Series: $R[[t]]$ e.g. `S.<t>=QQ[[]] 1/2+2*t+O(t^2)`
p-adic numbers: $\mathbb{Z}_p \approx$`Zp`, $\mathbf{Q}_p \approx$`Qp` e.g. `2+3*5+O(5^2)`
Algebraic closure: $\overline{\mathbb{Q}}$ = `QQbar` e.g. `QQbar(2^(1/5))`
Interval arithmetic: `RIF` e.g. `RIF((1,1.00001))`
Number field: `R.<x>=QQ[]; K.<a>=NumberField(x^3+x+1)`

Arithmetic

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)`

Sums: $\sum_{i=k}^{n} f(i)$ = `sum(f(i) for i in (k..n))`

Products: $\prod_{i=k}^{n} f(i)$ = `prod(f(i) for i in (k..n))`

Constants and functions

Constants: π = `pi` e = `e` i = `i` ∞ = `oo`
ϕ = `golden_ratio` γ = `euler_gamma`
Approximate: `pi.n(digits=18)` = 3.14159265358979324
Functions: `sin cos tan sec csc cot sinh cosh tanh sech csch coth log ln exp` ...
Python function: `def f(x): return x^2`

Interactive functions

Put `@interact` before function (vars determine controls)

```
@interact
def f(n=[0..4], s=(1..5), c=Color("red")):
    var("x")
    show(plot(sin(n+x^s),-pi,pi,color=c))
```

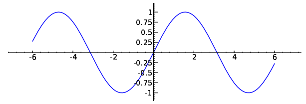
Symbolic expressions

Define new symbolic variables: `var("t u v y z")`
Symbolic function: e.g. $f(x) = x^2$ `f(x)=x^2`
Relations: `f==g f<=g f>=g f<g f>g`
Solve $f = g$: `solve(f(x)==g(x), x)`
`solve([f(x,y)==0, g(x,y)==0], x,y)`
`factor(...)` `expand(...)` `(...).simplify_...`
`find_root(f(x), a, b)` find $x \in [a,b]$ s.t. $f(x) \approx 0$

Calculus

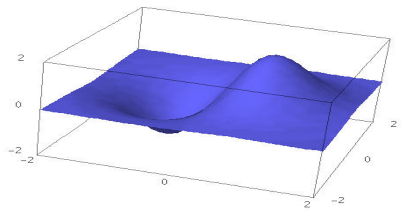
$\lim_{x \to a} f(x)$ = `limit(f(x), x=a)`
$\frac{d}{dx}(f(x))$ = `diff(f(x),x)`
$\frac{\partial}{\partial x}(f(x,y))$ = `diff(f(x,y),x)`
`diff` = `differentiate` = `derivative`
$\int f(x)dx$ = `integral(f(x),x)`
$\int_a^b f(x)dx$ = `integral(f(x),x,a,b)`
$\int_a^b f(x)dx \approx$ `numerical_integral(f(x),a,b)`
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 $0 \le r, b, g, h \le 1$
`show(`*graphic*`, `*options*`)`
use `figsize=[w,h]` to adjust size
use `aspect_ratio=`*number* to adjust aspect ratio
`plot(f(`x`),(`$x, x_{\min}, x_{\max}$`),`*options*`)`
`parametric_plot((f(`t`),g(`t`)),(`$t, t_{\min}, t_{\max}$`),`*options*`)`
`polar_plot(f(`t`),(`$t, t_{\min}, t_{\max}$`),`*options*`)`
combine: `circle((1,1),1)+line([(0,0),(2,2)])`
`animate(`*list of graphics, 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*`)`
`text3d("txt", (`x`,`y`,`z`), `*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*`)`
`plot3d(f(`x`,`y`),(`x, x_b, x_e`), (`y, y_b, y_e`),`*options*`)`
`parametric_plot3d((f,g,h),(`t, t_b, t_e`),`*options*`)`
`parametric_plot3d((f(`u`,`v`),g(`u`,`v`),h(`u`,`v`)),`
`(`u, u_b, u_e`),(`v, v_b, v_e`),`*options*`)`
options: `aspect_ratio=[1,1,1]`, `color="red"`,
`opacity=0.5`, `figsize=6`, `viewer="tachyon"`

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)`

$\phi(n)$ = `euler_phi(n)`

Strings: e.g. `s = "Hello" = "He"+'llo'`
  `s[0]="H"` `s[-1]="o"` `s[1:3]="el"` `s[3:]="lo"`

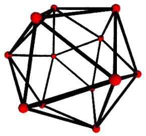
Lists: e.g. `[1,"Hello",x] = []+[1,"Hello"]+[x]`

Tuples: e.g. `(1,"Hello",x)` (immutable)

Sets: e.g. $\{1,2,1,a\}$ = `Set([1,2,1,"a"])` $(=\{1,2,a\})$

List comprehension ≈ set builder notation, e.g.
  $\{f(x)|x \in X, x > 0\}$ =`Set([f(x) for x in X if x>0])`

Graph theory



Graph: `G = Graph({0:[1,2,3], 2:[4]})`

Directed Graph: `DiGraph(`*dictionary*`)`

Graph families: `graphs.`⟨tab⟩

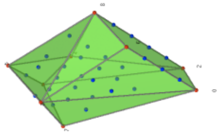
Invariants: `G.chromatic_polynomial()`, `G.is_planar()`

Paths: `G.shortest_path()`

Visualize: `G.plot()`, `G.plot3d()`

Automorphisms: `G.automorphism_group()`,
  `G1.is_isomorphic(G2)`, `G1.is_subgraph(G2)`

Combinatorics



Integer sequences: `sloane_find(`*list*`)`, `sloane.`⟨tab⟩

Partitions: `P=Partitions(`n`)` `P.count()`

Combinations: `C=Combinations(`*list*`)` `C.list()`

Cartesian product: `CartesianProduct(P,C)`

Tableau: `Tableau([[1,2,3],[4,5]])`

Words: `W=Words("abc"); W("aabca")`

Posets: `Poset([[1,2],[4],[3],[4],[]])`

Root systems: `RootSystem(["A",3])`

Crystals: `CrystalOfTableaux(["A",3], shape=[3,2])`

Lattice Polytopes: `A=random_matrix(ZZ,3,6,x=7)`
`L=LatticePolytope(A)` `L.npoints()` `L.plot3d()`

Matrix algebra

$\begin{pmatrix}1\\2\end{pmatrix}$ = `vector([1,2])`

$\begin{pmatrix}1 & 2\\3 & 4\end{pmatrix}$ = `matrix(QQ,[[1,2],[3,4]], sparse=False)`

$\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\end{pmatrix}$ = `matrix(QQ,2,3,[1,2,3, 4,5,6])`

$\begin{vmatrix}1 & 2\\3 & 4\end{vmatrix}$ = `det(matrix(QQ,[[1,2],[3,4]]))`

Av = `A*v` A^{-1} = `A^-1` A^t = `A.transpose()`

Solve $Ax = v$: `A\v` or `A.solve_right(v)`

Solve $xA = v$: `A.solve_left(v)`

Reduced row echelon form: `A.echelon_form()`

Rank and nullity: `A.rank()` `A.nullity()`

Hessenberg form: `A.hessenberg_form()`

Characteristic polynomial: `A.charpoly()`

Eigenvalues: `A.eigenvalues()`

Eigenvectors: `A.eigenvectors_right()` (also left)

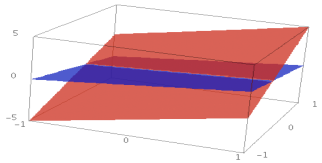
Gram-Schmidt: `A.gram_schmidt()`

Visualize: `A.plot()`

LLL reduction: `matrix(ZZ,...).LLL()`

Hermite form: `matrix(ZZ,...).hermite_form()`

Linear algebra



Vector space K^n = `K^n` e.g. `QQ^3 RR^2 CC^4`

Subspace: `span(vectors,` *field*`)`
  E.g., `span([[1,2,3], [2,3,5]], QQ)`

Kernel: `A.right_kernel()` (also left)

Sum and intersection: `V + W` and `V.intersection(W)`

Basis: `V.basis()`

Basis matrix: `V.basis_matrix()`

Restrict matrix to subspace: `A.restrict(V)`

Vector in terms of basis: `V.coordinates(`*vector*`)`

Numerical mathematics

Packages: `import numpy, scipy, cvxopt`

Minimization: `var("x y z")`
  `minimize(x^2+x*y^3+(1-z)^2-1, [1,1,1])`

Number theory

Primes: `prime_range(n,m)`, `is_prime`, `next_prime`

Factor: `factor(n)`, `qsieve(n)`, `ecm.factor(n)`

Kronecker symbol: $\left(\frac{a}{b}\right)$ = `kronecker_symbol(`a,b`)`

Continued fractions: `continued_fraction(x)`

Bernoulli numbers: `bernoulli(n)`, `bernoulli_mod_p(p)`

Elliptic curves: `EllipticCurve([`a_1, a_2, a_3, a_4, a_6`])`

Dirichlet characters: `DirichletGroup(`N`)`

Modular forms: `ModularForms(`*level*`,` *weight*`)`

Modular symbols: `ModularSymbols(`*level*`,`*weight*`,`*sign*`)`

Brandt modules: `BrandtModule(`*level*`,` *weight*`)`

Modular abelian varieties: `J0(`N`)`, `J1(`N`)`

Group theory

`G = PermutationGroup([[(1,2,3),(4,5)],[(3,4)]])`

`SymmetricGroup(`n`)`, `AlternatingGroup(`n`)`

Abelian groups: `AbelianGroup([3,15])`

Matrix groups: `GL`, `SL`, `Sp`, `SU`, `GU`, `SO`, `GO`

Functions: `G.sylow_subgroup(p)`, `G.character_table()`,
  `G.normal_subgroups()`, `G.cayley_graph()`

Noncommutative rings

Quaternions: `Q.<i,j,k> = QuaternionAlgebra(a,b)`

Free algebra: `R.<a,b,c> = FreeAlgebra(QQ, 3)`

Python modules

`import` *module_name*

`module_name.`⟨tab⟩ and `help(module_name)`

Profiling and debugging

`time` *command*: show timing information

`timeit("command")`: accurately time command

`t = cputime(); cputime(t)`: elapsed CPU time

`t = walltime(); walltime(t)`: elapsed wall time

`%pdb`: turn on interactive debugger (command line only)

`%prun command`: profile command (command line only)