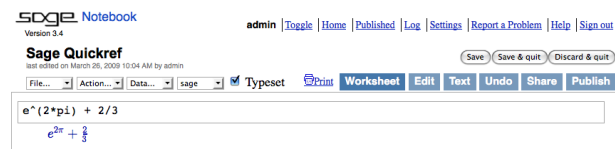


Sage Quick Reference

William Stein (based on P. Jipsen's quickref)
GNU Free Document License, extend for your own use

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*?<tab>` 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 all source code

`_` is previous output

Numbers

Integers: $\mathbf{Z} = \mathbb{Z}$ e.g. -2 -1 0 1 10^{100}

Rationals: $\mathbf{Q} = \mathbb{Q}$ e.g. $1/2$ $1/1000$ $314/100$ $-2/1$

Reals: $\mathbf{R} \approx \mathbb{R}$ e.g. .5 0.001 3.14 $1.23e10000$

Complex: $\mathbf{C} \approx \mathbb{C}$ e.g. $\mathbb{C}(1,1)$ $\mathbb{C}(2.5,-3)$

Double precision: RDF and CDF e.g. $\text{CDF}(2.1,3)$

Mod n : $\mathbf{Z}/n\mathbf{Z} = \mathbb{Z}_{\text{mod}}$ e.g. $\text{Mod}(2,3)$ $\mathbb{Z}_{\text{mod}}(3)(2)$

Finite fields: $\mathbf{F}_q = \mathbb{GF}$ e.g. $\mathbb{GF}(3)(2)$ $\mathbb{GF}(9, 'a') \cdot 0$

Polynomials: $R[x,y]$ e.g. $\mathbb{R}\langle x,y \rangle = \mathbb{Q}\mathbb{Q}[]$ $x+2*y^3$

Series: $R[[t]]$ e.g. $\mathbb{R}\langle t \rangle = \mathbb{Q}\mathbb{Q}[[[]]]$ $1/2+2*t+0(t^2)$

p -adic numbers: $\mathbf{Z}_p = \mathbb{Z}_p$, $\mathbf{Q}_p = \mathbb{Q}_p$ e.g. $2+3*5+0(5^2)$

Algebraic closure: $\overline{\mathbf{Q}} = \mathbb{Q}\overline{\mathbb{Q}}$ e.g. $\mathbb{Q}\overline{\mathbb{Q}}(2^{1/5})$

Interval arithmetic: RIF e.g. `sage: RIF((1,1.00001))`

Number field: $\mathbb{R}\langle x \rangle = \mathbb{Q}\mathbb{Q}[]$; $\mathbb{K}\langle a \rangle = \text{NumberField}(x^3+x+1)$

Arithmetic

$ab = \mathbf{a*b}$ $\frac{a}{b} = \mathbf{a/b}$ $a^b = \mathbf{a^b}$ $\sqrt{x} = \mathbf{sqrt(x)}$

$\sqrt[n]{x} = \mathbf{x^(1/n)}$ $|x| = \mathbf{abs(x)}$ $\log_b(x) = \mathbf{log(x,b)}$

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

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

Constants and functions

Constants: $\pi = \mathbf{pi}$ $e = \mathbf{e}$ $i = \mathbf{i}$ $\infty = \mathbf{oo}$

$\phi = \mathbf{golden_ratio}$ $\gamma = \mathbf{euler_gamma}$

Approximate: $\mathbf{pi.n(digits=18)} = 3.14159265358979324$

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=[1..10], s=(1..10), c=Color('red')):
    print n^s
```

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

$\lim_{x \rightarrow a} f(x) = \mathbf{limit(f(x), x=a)}$

$\frac{d}{dx}(f(x)) = \mathbf{diff(f(x), x)}$

$\frac{\partial}{\partial x}(f(x,y)) = \mathbf{diff(f(x,y), x)}$

`diff = differentiate = derivative`

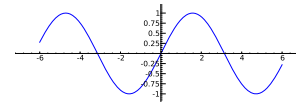
$\int f(x)dx = \mathbf{integral(f(x), x)}$

`integral = integrate`

$\int_a^b f(x)dx = \mathbf{integral(f(x), x, a, b)}$

Taylor polynomial, deg n about a : `taylor(f(x), x, a, n)`

2D graphics



`line([(x1,y1), ..., (xn,yn)], options)`

`polygon([(x1,y1), ..., (xn,yn)], 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 $0 \leq r, b, g, h \leq 1$

`show(graphic, options)`

use `figsize=[w,h]` to adjust size

use `aspect_ratio=number` to adjust aspect ratio

`plot(f(x), (x, xmin, xmax), options)`

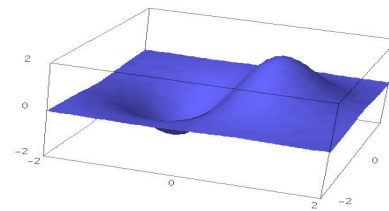
`parametric_plot((f(t), g(t)), (t, t_min, t_max), options)`

`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([(x1,y1,z1), ..., (xn,yn,zn)], 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,xb,xe), (y,yb,ye), options)`

`parametric_plot3d((f,g,h), (t,tb,te), options)`

`parametric_plot3d((f(u,v), g(u,v), h(u,v)), (u,ub,ue), (v,vb,ve), options)`

options: `aspect_ratio=[1,1,1]`, `color='red'`
`opacity=0.5`, `figsize=6`, `viewer='tachyon'`

Discrete math

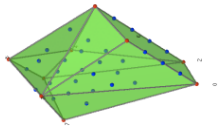
$\lfloor x \rfloor = \text{floor}(x)$ $\lceil x \rceil = \text{ceil}(x)$
Remainder of n divided by $k = n\%k$ $k|n$ iff $n\%k==0$
 $n! = \text{factorial}(n)$ $\binom{x}{m} = \text{binomial}(x,m)$
 $\phi(n) = \text{euler_phi}(n)$
Strings: e.g. `s = 'Hello' = "Hello" = ""+"He"+"llo"`
`s[0]='H'` `s[-1]='o'` `s[1:3]='el'` `s[3:]='llo'`
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])`

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

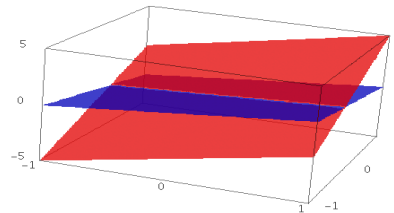


Integer sequences: `sloane_find(list)`, `sloane.<tab>`
Partitions: `Partitions(n)`, `number_of_partitions(n)`
Combinations: `Combinations(list)`
Tableau: `Tableau([[1,2,3],[4,5]])`
Words: `W=Words('abc')`; `W('aabca')`
Root systems: `RootSystem(['A',3])`
Posets: `Poset([[1,2],[4],[3],[4],[]])`
Polytopes: `LatticePolytope(3-row integer matrix)`

Matrix algebra

$\begin{pmatrix} 1 \\ 2 \end{pmatrix} = \text{vector}([1,2])$
 $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \text{matrix}(QQ, [[1,2],[3,4]], \text{sparse=False})$
 $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \text{matrix}(QQ, 2, 3, [1,2,3, 4,5,6])$
 $\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} = \text{det}(\text{matrix}(QQ, [[1,2],[3,4]]))$
 $Av = A*v$ $A^{-1} = A^{-1}$ $A^t = A.\text{transpose}()$
Solve $Ax = v$: `A\v` or `A.solve_right(v)`
Solve $xA = v$: `A.solve_left(v)`
Reduced row echelon form: `A.echelon_form()`
Hessenberg form: `A.hessenberg_form()`
Characteristic polynomial: `A.charpoly()`
Eigenvalues: `A.eigenvalues()`
Eigenvectors: `A.eigenvectors_right()` (and left)
Gram-Schmidt: `A.gram_schmidt()`
Visualize: `plot(A)`
Methods: `A.nrows()` `A.ncols()` `A.rank()` ...
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 as subspace: `A.kernel()`
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)`

Number theory

Primes: `prime_range(n,m)`, `is_prime`, `next_prime`
Factor: `factor(n)`, `qsieve(n)`, `ecm.factor(n)`
Kronecker symbol: $\left(\frac{a}{b}\right) = \text{kronecker_symbol}(a,b)$
Bernoulli numbers: `bernoulli(n)`, `bernoulli_mod_p(p)`
Elliptic curves: `EllipticCurve([a1,a2,a3,a4,a6])`
Continued fractions: `continued_fraction(x)`
Dirichlet characters: `DirichletGroup(N)`
Modular forms: `ModularForms(level, weight)`
Modular symbols: `ModularSymbols(level, weight, sign)`
Brandt modules: `BrandtModule(level, weight)`
Modular abelian varieties: `JO(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)
