Sage Quick Reference for 2MMC10

(based on work by Peter Jipsen and Wiliam Stein)

Python

Python and sage use indentation for expressing syntax. Interactive versions need an extra `return` to run.

conditional statement: `if expr: statements .. elif expr: statements .. else: statements ..`

value identity `if a==1:.. if a!=1:.. if a>= 1:..`

while loop `while expr: statements`

escape while loop `while True: ... if cond: break`

for loop `for target in iter: statements`

end loop/jump to next `break` / `continue`

print `print ("hello world")`

Sage 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

Sage 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: $\mathbf{Z}$ = `ZZ` e.g. `-2 -1 0 1 10^100`

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

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

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

Mod n: $\mathbf{Z}/n\mathbf{Z}$ = `Zmod` e.g. `Mod(2,3) Zmod(3)(2)`

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

`x` is assumed to be a polynomial variable, all other variables need to be declared `y=var("y")` or as follows:

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)`

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

Number field: `R.<x>=QQ[];K.<a>=NumberField(x^3+x+1)`

Integers

n divided by m has *remainder* `n % m`

`gcd(n,m)`, `gcd(`*list*`)`

extended gcd $g = sa + tb = \gcd(a,b)$: `g,s,t=xgcd(a,b)`

`lcm(n,m)`, `lcm(`*list*`)`

binomial coefficient $\binom{m}{n}$ = `binomial(m,n)`

digits in a given base: `n.digits(`*base*`)`

number of digits: `n.ndigits(`*base*`)`

(*base* is optional and defaults to 10)

divides $n \mid m$: `n.divides(m)` if $nk = m$ some k

divisors – all d with $d \mid n$: `n.divisors()`

factorial – $n!$ = `n.factorial()`

Prime Numbers and Number theory

primality testing: `is_prime(n)`, `is_pseudoprime(n)`

prime power testing: `is_prime_power(n)`

$\pi(x) = \#\{p : p \leq x \text{ is prime}\}$ = `prime_pi(x)`

set of prime numbers: `Primes()`

$\{p : m \leq p < n \text{ and } p \text{ prime}\}$ =`prime_range(m,n)`

first n primes: `primes_first_n(n)`

next and previous primes: `next_prime(n)`, `previous_prime(n)`, `next_probable_prime(n)`

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

Continued fractions: `continued_fraction(x)`

Discrete math

$\lfloor x \rfloor$ = `floor(x)` $\lceil x \rceil$ = `ceil(x)`

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)

Length of list or tuple `len(l)`, `len(t)`

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

Adjoin elements of t to s `s.update(t)`

Intersect t and s `s.intersection_update(t)`

Remove elements in t from s `s.difference_update(t)`

List comprehension ≈ set builder notation, e.g.

$\{f(x) : x \in X, x > 0\}$ = `Set([f(x) for x in X if x>0])`

Modular Arithmetic and Congruences

a modulo n as element of $\mathbf{Z}/n\mathbf{Z}$: `Mod(a, n)`

Remainder of n divided by k = `n%k` $k|n$ iff `n%k==0`

Euler's $\phi(n)$ function: `euler_phi(n)`

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

Quadratic residues: `quadratic_residues(n)`

Quadratic non-residues: `quadratic_residues(n)`

ring $\mathbf{Z}/n\mathbf{Z}$ = `Zmod(n) = IntegerModRing(n)`

primitive root modulo n = `primitive_root(n)`

inverse of $n \pmod m$: `n.inverse_mod(m)`

power $a^n \pmod m$: `power_mod(a, n, m)`

Chinese remainder theorem: `x = crt(a,b,m,n)`

finds x with $x \equiv a \pmod m$ and $x \equiv b \pmod n$

discrete log: `log(Mod(6,7), Mod(3,7))`

order of $a \pmod n$ =`Mod(a,n).multiplicative_order()`

square root of $a \pmod n$ =`Mod(a,n).sqrt()`

Matrix algebra

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

$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ = `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()`

Groups

Order of group G `G.cardinality()`

Generators of G `G.gens()`

Elliptic curves

`E = EllipticCurve(K, [a1,a2,a3,a4,a6])`

`E = EllipticCurve(K, [c4,c6])`

The field parameter `K` is optional if $a_i \in K$

Point $P = (s,t)$: `P = E(s,t)`

Scalar multiplication: `5*P`

Point at infinity `P = E(0,1,0)` or `P = E(0)`