SageMath Quick Reference: SFU Math301

Jamie Mulholland

SageMath Version 8.8

For more visit http://wiki.sagemath.org/quickref

Based on work by P. Jipsen, W. Stein, R. Beezer

Sets of Numbers

Some common Sage rings and fields

- `ZZ` integers, ring
- `QQ` rationals, field
- `RR` real numbers
- `CC` complex numbers
- `GF(2)` mod 2, field, specialized implementations
- `GF(p) == FiniteField(p)` p prime, field
- `Integers(6)` integers mod 6

Lists

`range(1,11)` list $[1, 2, \ldots, 10]$

`L = [1,2,"milk","cheese"]` list of objects

`[n for n in [1,2,3] if is_even(n)]` sublists

`[2*n+1 for n in [1,2,3]]` construct new lists

`L.remove(x)` removes first item from list with value x

`L.append(x)` adds item x to end of the list

Caution: first entry of a list is numbered 0

`L[0]` picks off first entry of list L

Sets

`A = Set([1,2,3,4,5])` set of objects

`B = Set([3,4,5,6,7])` set of objects

`A.cardinality()` size of the set

`A.union(B)`, `A.intersection(B)`, `A.difference(B)` are all possible

`Set(n for n in [1,2,3] if is_even(n))` subsets

`Set(2*n+1 for n in [1,2,3])` construct new set

Python/Sage Functions & Commands

defining a function:

```
def function_name( <parameters> ):
        :
    <statement>
        :
    return <expression>
```

lambda function:

```
f = lambda x: x*x
concat = lambda x,y: x+y
```

if statement:

```
if <condition>:
    <statement>
         :
else:
    <statement>
         :
```

while statement:

```
while <condition>:
    <statement>
         :
```

for loop:

```
for  x in <list>:
   <statement>
         :
```

Symmetric Group & Permutations

`S5 = SymmetricGroup(5)`
symmetric group on 5 objects

`A5 = AlternatingGroup(5)`
alternating group on 5 objects

`S5("")` identity permutation in S_5

`a = S5("(2,3)(1,4)")`
permutation with cycle form $(2,3)(1,4)$ in S_5

`b = S5("(2,5,3)")`
permutation with cycle form $(2,5,3)$ in S_5

`a*b`, `a.order()`, `a.inverse()`, `a.sign()`
are all possible

Permutation Groups

`S4 = SymmetricGroup(4)`

`a = S4("(1,2)")`

`b = S4("(1,3)")`

`H = PermutationGroup([a,b])`
subgroup of S_4 generated by a and b

`H.order()` size of subgroup H

`H.center()` center of subgroup H

Vector Constructions

Caution: First entry of a vector is numbered 0

`u = vector(GF(2), [1, 0, 1, 1])` length 4 over $\mathbb{F}_2$

`u = vector(QQ, [1, 3/2, -1])` length 3 over rationals

Matrix Constructions

Caution: Row, column numbering begins at 0

`A = matrix(ZZ, 3, 2, [1,2,3,4,5,6])`
3×2 over the integers

`A = matrix(ZZ, [[1,2],[3,4],[5,6]])`
3×2 over the integers

`B = matrix(QQ, 2, [1,2,3,4,5,6])`
2 rows from a list, so 2×3 over rationals

`Z = matrix(QQ, 2, 2, 0)` 2×2 zero matrix

`D = matrix(QQ, 2, 2, 8)`
diagonal entries all 8, other entries zero

`E = block_matrix([[P,0],[1,R]])`, very flexible input

`II = identity_matrix(5)` 5×5 identity matrix
`I` $= \sqrt{-1}$, do not overwrite with matrix name

`A.block_sum(B)` Diagonal, `A` upper left, `B` lower right

Matrix Multiplication

`u = vector(QQ, [1,2,3])`, `v = vector(QQ, [1,2])`

`A = matrix(QQ, [[1,2,3],[4,5,6]])`

`B = matrix(QQ, [[1,2],[3,4]])`

`v*A`, `A*u`, `B*A`, `B^6`, `B^(-3)` are all possible

Matrix Spaces

`M = MatrixSpace(QQ, 3, 4)` is space of 3×4 matrices

`A = M([1,2,3,4,5,6,7,8,9,10,11,12])`
coerce list to element of `M`, a 3×4 matrix over `QQ`

`M.zero_matrix()`

Matrix Operations

`5*A+2*B` linear combination

`A.inverse()`, `A^(-1)`, `~A`, singular is `ZeroDivisionError`

`A.transpose()`

Echelon Form

A.augment(B) A in first columns, matrix B to the right
A.rref(), A.echelon_form(), A.echelonize()
Note: rref() promotes matrix to fraction field
A = matrix(ZZ,[[4,2,1],[6,3,2]])
A.rref() A.echelon_form()

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A.pivots() indices of columns spanning column space
A.pivot_rows() indices of rows spanning row space

Pieces of Matrices

Caution: row, column numbering begins at 0
A.nrows(), A.ncols() number of rows/columns
A[i,j] entry in row i and column j
A[i] row i as immutable Python tuple:
Caution: OK: A[2,3] = 8, Error: A[2][3] = 8
A.row(i) returns row i as Sage vector
A.column(j) returns column j as Sage vector
A.matrix_from_rows_and_columns([2,4,2],[3,1])
common to the rows and the columns
A.rows() all rows as a list of tuples
A.columns() all columns as a list of tuples
A.submatrix(i,j,nr,nc)
start at entry (i,j), use nr rows, nc cols
A[2:4,1:7], A[0:8:2,3::-1] Python-style list slicing

Scalar Functions on Matrices

A.rank(), A.right_nullity()
A.left_nullity() == A.nullity()
A.determinant() == A.det()

Matrix Properties

.is_zero(); .is_symmetric(); .is_invertible();

Eigenvalues and Eigenvectors

A.eigenvalues() unsorted list, with mutiplicities
A.eigenvectors_right() vectors on right, _left too
Returns, per eigenvalue, a triple: e: eigenvalue;
V: list of eigenspace basis vectors; n: multiplicity

Solutions to Systems

A.solve_right(B) _left too
is solution to A*X = B, where X is a vector **or** matrix
A = matrix(QQ, [[1,2],[3,4]])
b = vector(QQ, [3,4]), then A\b is solution (-2, 5/2)

Constructing Subspaces

span([v1,v2,v3], QQ) span of list of vectors over $\mathbb{Q}$
A.right_kernel() .left_kernel == .kernel too
A.row_space()
A.column_space()
A.eigenspaces_right() vectors on right, _left too
returns pairs: eigenvalues with their right eigenspaces

More Help

"tab-completion" on partial commands
"tab-completion" on <object.> for all relevant methods
dir(<object>) for all relevant methods
type(<object>) for displaying object type
<command>? for summary and examples
<command>?? for complete source code