

Python For Data Science Cheat Sheet

Also see NumPy

SciPy - Linear Algebra

Learn More Python for Data Science [Interactively](https://www.datacamp.com) at www.datacamp.com



SciPy

The SciPy library is one of the core packages for scientific computing that provides mathematical algorithms and convenience functions built on the NumPy extension of Python.



Interacting With NumPy

Also see NumPy

```
>>> import numpy as np
>>> a = np.array([1,2,3])
>>> b = np.array([(1+5j,2j,3j), (4j,5j,6j)])
>>> c = np.array([(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)])
```

Index Tricks

```
>>> np.mgrid[0:5,0:5]
>>> np.ogrid[0:2,0:2]
>>> np.r_[3,[0]*5,-1:1:10j]
>>> np.c_[b,c]
```

Create a dense meshgrid
Create an open meshgrid
Stack arrays vertically (row-wise)
Create stacked column-wise arrays

Shape Manipulation

```
>>> np.transpose(b)
>>> b.flatten()
>>> np.hstack((b,c))
>>> np.vstack((a,b))
>>> np.hsplit(c,2)
>>> np.vsplit(d,2)
```

Permute array dimensions
Flatten the array
Stack arrays horizontally (column-wise)
Stack arrays vertically (row-wise)
Split the array horizontally at the 2nd index
Split the array vertically at the 2nd index

Polynomials

```
>>> from numpy import poly1d
>>> p = poly1d([3,4,5])
```

Create a polynomial object

Vectorizing Functions

```
>>> def myfunc(a):
    if a < 0:
        return a*2
    else:
        return a/2
>>> np.vectorize(myfunc)
```

Vectorize functions

Type Handling

```
>>> np.real(b)
>>> np.imag(b)
>>> np.real_if_close(c,tol=1000)
>>> np.cast['f'](np.pi)
```

Return the real part of the array elements
Return the imaginary part of the array elements
Return a real array if complex parts close to 0
Cast object to a data type

Other Useful Functions

```
>>> np.angle(b,deg=True)
>>> g = np.linspace(0,np.pi,num=5)
>>> g[3:] += np.pi
>>> np.unwrap(g)
>>> np.logspace(0,10,3)
>>> np.select([c<4],[c*2])

>>> misc.factorial(a)
>>> misc.comb(10,3,exact=True)
>>> misc.central_diff_weights(3)
>>> misc.derivative(myfunc,1.0)
```

Return the angle of the complex argument
Create an array of evenly spaced values (number of samples)
Unwrap
Create an array of evenly spaced values (log scale)
Return values from a list of arrays depending on conditions
Factorial
Combine N things taken at k time
Weights for Np-point central derivative
Find the n-th derivative of a function at a point

Linear Algebra

You'll use the `linalg` and `sparse` modules. Note that `scipy.linalg` contains and expands on `numpy.linalg`.

```
>>> from scipy import linalg, sparse
```

Creating Matrices

```
>>> A = np.matrix(np.random.random((2,2)))
>>> B = np.asmatrix(b)
>>> C = np.mat(np.random.random((10,5)))
>>> D = np.mat([[3,4], [5,6]])
```

Basic Matrix Routines

Inverse >>> A.I >>> linalg.inv(A)	Inverse Inverse
Transposition >>> A.T >>> A.H	Tranpose matrix Conjugate transposition
Trace >>> np.trace(A)	Trace
Norm >>> linalg.norm(A) >>> linalg.norm(A,1) >>> linalg.norm(A,np.inf)	Frobenius norm L1 norm (max column sum) L inf norm (max row sum)
Rank >>> np.linalg.matrix_rank(C)	Matrix rank
Determinant >>> linalg.det(A)	Determinant
Solving linear problems >>> linalg.solve(A,b) >>> E = np.mat(a).T >>> linalg.lstsq(F,E)	Solver for dense matrices Solver for dense matrices Least-squares solution to linear matrix equation
Generalized inverse >>> linalg.pinv(C) >>> linalg.pinv2(C)	Compute the pseudo-inverse of a matrix (least-squares solver) Compute the pseudo-inverse of a matrix (SVD)

Creating Sparse Matrices

```
>>> F = np.eye(3, k=1)
>>> G = np.mat(np.identity(2))
>>> C[C > 0.5] = 0
>>> H = sparse.csr_matrix(C)
>>> I = sparse.csc_matrix(D)
>>> J = sparse.dok_matrix(A)
>>> E.todense()
>>> sparse.isspmatrix_csc(A)
```

Create a 2X2 identity matrix
Create a 2x2 identity matrix
Compressed Sparse Row matrix
Compressed Sparse Column matrix
Dictionary Of Keys matrix
Sparse matrix to full matrix
Identify sparse matrix

Sparse Matrix Routines

Inverse >>> sparse.linalg.inv(I)	Inverse
Norm >>> sparse.linalg.norm(I)	Norm
Solving linear problems >>> sparse.linalg.spsolve(H,I)	Solver for sparse matrices

Sparse Matrix Functions

```
>>> sparse.linalg.expm(I)
```

Sparse matrix exponential

Asking For Help

```
>>> help(scipy.linalg.diagsvd)
>>> np.info(np.matrix)
```

Matrix Functions

Addition >>> np.add(A,D)	Addition
Subtraction >>> np.subtract(A,D)	Subtraction
Division >>> np.divide(A,D)	Division
Multiplication >>> A @ D >>> np.multiply(D,A) >>> np.dot(A,D) >>> np.vdot(A,D) >>> np.inner(A,D) >>> np.outer(A,D) >>> np.tensordot(A,D) >>> np.kron(A,D)	Multiplication operator (Python 3) Multiplication Dot product Vector dot product Inner product Outer product Tensor dot product Kronecker product
Exponential Functions >>> linalg.expm(A) >>> linalg.expm2(A) >>> linalg.expm3(D)	Matrix exponential Matrix exponential (Taylor Series) Matrix exponential (eigenvalue decomposition)
Logarithm Function >>> linalg.logm(A)	Matrix logarithm
Trigonometric Functions >>> linalg.sinm(D) >>> linalg.cosm(D) >>> linalg.tanm(A)	Matrix sine Matrix cosine Matrix tangent
Hyperbolic Trigonometric Functions >>> linalg.sinhm(D) >>> linalg.coshm(D) >>> linalg.tanhm(A)	Hyperbolic matrix sine Hyperbolic matrix cosine Hyperbolic matrix tangent
Matrix Sign Function >>> np.signm(A)	Matrix sign function
Matrix Square Root >>> linalg.sqrtm(A)	Matrix square root
Arbitrary Functions >>> linalg.funm(A, lambda x: x*x)	Evaluate matrix function

Decompositions

Eigenvalues and Eigenvectors >>> la, v = linalg.eig(A) >>> l1, l2 = la >>> v[:,0] >>> v[:,1] >>> linalg.eigvals(A)	Solve ordinary or generalized eigenvalue problem for square matrix Unpack eigenvalues First eigenvector Second eigenvector Unpack eigenvalues
Singular Value Decomposition >>> U,s,Vh = linalg.svd(B) >>> M,N = B.shape >>> Sig = linalg.diagsvd(s,M,N)	Singular Value Decomposition (SVD) Construct sigma matrix in SVD
LU Decomposition >>> P,L,U = linalg.lu(C)	LU Decomposition

Sparse Matrix Decompositions

```
>>> la, v = sparse.linalg.eigs(F,1)
>>> sparse.linalg.svds(H, 2)
```

Eigenvalues and eigenvectors
SVD

