# Numpy/Scipy Cheat Sheet

by gonz95alo via cheatography.com/152363/cs/33001/

## Libraries

| | |
|---|---|
| Numpy | `import numpy as np` |
| Linear algebra (numpy) | `import numpy.l inalg as LA` |
| Sparse operations (scipy) | `import scipy.s parse as ss` |
| Sparse normal-izations (sklearn) | `from sklear n.p rep roc essing import normalize as sparse_norm` |

## Array comparisons

| | |
|---|---|
| Locate indices based on condition | `index = np.whe re( array == 4)` |
| Check if all/any satisfy condition | `np.any (array == 1), np.all (array1 == array 2)` |

## Matricial operations

| | |
|---|---|
| Dot product | `np.dot(v1, v2)` or `v1 @ v2` |
| Outer product | `np.out er(v1, v1)` |
| Sum rows (0), columns (1) | `np.sum(A, axis=0)` |
| Eigenvector, eigenvalues | `eigval, eigvec = LA.eig(A)` |

The eigenvector output of the `eig()` function is a matrix containing the usual eigenvectors as its columns.

## Numpy comparisons

| | |
|---|---|
| Check if any element satisfies a condition | `np.any(A == 1)` |
| Find which elements (their indices) satisfy a condition | `np.where(A == 1)` |
| Check if all elements satisfies a condition | `np.all(A == 1)` |
| Closeness up to tolerance | `np.isc los e(1.001, 1)` |

## Numpy utilities

| | |
|---|---|
| Arrange all array elements in a list | `np.rav el(A)` |

## Sparse matrices with scipy

| | |
|---|---|
| Convert numpy matrices to sparse | `A_sparse = ss.csr _ma trix(A)`<br>`A_sparse = ss.csc _ma trix(A)` |
| Change CSC to CSR and viceversa | `A_csr = A_csc.t ocsr()`<br>`A_csc = A_csr.t ocsc()` |
| Row/col. normal-ization | `A_norm = sparse _no rma liz e(A _csc, norm= is=1)` |
| Sparse identity matrix | `sparse_id = ss.ide nti ty(n)` |
| Invert a (CSC) sparse matrix | `sparse_inv = ss.inv (A_ sparse)` |
| Dot product with vector | `A_spar se.d ot(v)` |
| Transpose sparse matrix | `At_sparse = A_spar se.t ra nsp ose()` |

CSC (*Compressed Sparse Column*) are more suitable for column operations while CSR (*Compressed Sparse Row*) are for row operations. For instance, row/col. normalizations.