# Python Data Analysis Reference Card

*Fabricio Ferrari* www.ferrari.pro.br

A,B: 2D array x,y: 1D vector M,N integers

## Numpy Arrays

| | |
|---|---|
| **Creation** | |
| zeros((M,N)) | Returns array filled zeros, M lines, N columns |
| ones((M,N)) | Returns array filled with ones, MxN |
| empty((M,N)) | Returns array not filled (random values), MxN |
| zeros_like(A) | Return an array of zeros with shape and type of input. |
| ones_like(A) | Return an array of ones with shape and type of input. |
| empty_like(A) | Return an empty array with shape and type of input. |
| random.random((M,N)) | Returns array filled with random numbers [0..1] |
| identity(3,float) | Identity 3x3 array of floats |
| array([(1.5,2,3),(4,5,6)]) | specify values, 2x3 array of floats |
| mgrid[1:3,2:5] | rectangular mesh grid with x values [1,2] and y values [2,3,4] |
| fromfunction(f, (3,3)) | Returns 3x3 array with function f(i,j) evaluated for all combinations of indices |
| arange(1., 10., 0.5) | Array with range and step of values |
| linspace(0,2,9) | 9 numbers from 0 to 2 |
| **Methods** | |
| A.sum() | Sum array (may specify axis) |
| A.min() | Minimum value |
| A.max() | Maximum Value |
| A.mean() | Average value |
| A.std() | Standard Deviation |
| A.var() | Variance |
| A.trace() | Array trace |
| A.size() | Number of elements |
| A.shape() | Shape |
| A.ptp() | Peak-to-peak (maximum - minimum) |
| A.ravel() | 1-d version of A |
| A.transpose(), A.T | transpose indices of array |
| A.resize(M,N) | Replicate or truncate array to new shape *in place* |
| A.reshape(M,N) | Returns array with new shape |
| A.clip(Amin,Amax) | Clip values of array a at values Amin, Amax |
| A.compress(condition, axis=None) | Selects elements from array A based on condition |
| A.conjugate() | Return an array with all complex-valued elements conjugated. |
| A.copy | Return a copy of the array. |
| A.cumprod(axis=0) | Cumulative product along specied axis. |
| A.cumsum(axis=0) | Return the cumulative sum of the elements along the given axis. |
| A.diagonal(offset=0, axis1=0, axis2=1) | Returns diagonal of 2D matrix with optional offsets. |
| A.fill(value) | fills in with a specified value |

## Numpy Arrays (cont.)

| **Operations** | *Arithmetic operarions are elementwise* |
|---|---|
| `C = A-B` | Elementwise subtraction ($C_{i,j} = A_{i,j} - B_{i,j}$) |
| `A**2` | Returns array with each `A` element squared |
| `dot(A,B)` | Matrix product |
| `mat(A) * mat(B)` | Matrix product |
| `matrixmultiply(A, B)` | Matrix product |
| `inner(A, B)` | Inner product |
| `outer(A, B)` | Outer product |
| `product(A, axis=0)` | Net product of elements along specified axis. |
| `concatenate(arrays, axis=0)` | Concatenate arrays contained in sequence of arrays |
| `vstack(A,B)` | Vertically stacks two arrays |
| `hstack(A,B)` | Horizontally stacks two arrays |
| `vsplit(A,2)` | Vertically splits A in 2 |
| `hsplit(A,2)` | Horizontally Splits A in 2 |
| **Indexing** | |
| `x[2]` | 3rd elements (zero based indexing) |
| `x[-2]` | Counting from the end |
| `x[2:5]` | subarray, `[x[2], x[3], x[4]]` |
| `x[:5]` | from the beginning to the 4th element |
| `x[2:]` | from the 3rd to the end |
| `x[:]` | the whole array |
| `x[2:9:3]` | every 3 elements, `[x[2], x[5], x[8]]` |
| `x[numpy.where(x>7)]` | returns elements in `x` that satisfy criteria |
| `indices(shape, type=None)` | Generate array with values corresponding to position of selected index of the array |
| `A[j][i]` | Indexing convention (`j` row, `i` column) |
| `A[3][2]` | 3rd element in the 4th row |
| `A[1]` | 2nd row |

**Numpy Arrays Caution:** The cumulative class of operations that either sum or multiply elements of an array (sum, product, cumsum, cumproduct) use the input array type as the type to perform the cumulative sum or product. If you are not using the largest type for the array (i.e., int64, Float64, etc.) you are asking for trouble. One is likely to run into integer overflows or Floating point precision problems if you don't specifcally ask for a higher precision result, especially for large array. One should get into the habit of specifying the keyword argument dtype to a highest precision type when using these functions, e.g. sum(arr, dtype=Float64) .

## Numpy Arrays (cont.)

| | |
|---|---|
| **Questions** | |
| `all(A, axis=None )` | are all elements of array nonzero? [also method],identical to alltrue() |
| `allclose(A, B, rtol=1.e-5, atol=1.e-8)` | True if all elements within specied amount (between two arrays) |
| `alltrue(A, axis=0)` | Are all elements nonzero along specied axis true. |
| `any(A , axis=None)` | Are any elements of an array nonzero [also method], identical to sometrue() |
| `sometrue(A, axis=0)` | Are any elements along specified axis true |
| `where(a)` | Find `true` locations in array `a` |
| **Ordering** | |
| `argmax(A, axis=-1), argmin(a,axis=-1 )` | Return array with min/max locations for selected axis |
| `argsort(A, axis=-1)` | Returns indices of results of sort on an array |
| `searchsorted(bin, A)` | Return indices of mapping values of an array a into a monotonic array bin |
| `sort(A, axis=-1)` | Sort array elements along selected axis |
| `choose(selector, population, clipmode=CLIP)` | Fills specied array by selecting corresponding values from a set of arrays using integer selection array (population is a tuple of arrays) |
| **File Ops** | |
| `fromfile(file, type, shape=None)` | Use binary data in file to form new array of specified type. |
| `fromstring(datastring, type, shape=None)` | Use binary data in datastring to form new array of specified shape and type |
| **Numeric Types** | |
| `A.astype(type)` | Copy of the array, cast to a specified type. |
| `int8, uint8, int16, uint16, int32, uint32, int64, uint64, float32,float64, complex64, complex128` | Possible numeric types `dtype=` |

## PyFits

| | |
|---|---|
| `import pyfits` | load FITS module |
| `pyfits.info('pix.fits')` | show info about file |
| `img = pyfits.getdata('pix.fits')` | read image data from file |
| `hdr = pyfits.getheader('pix.fits')` | read header from file |
| `hdr['date']` | header keyword value |
| `hdr['date'] = '4th of July'` | modify value |
| `hdr.update('flatfile','flat17.fits')` | add new keyword 'flatfile' |
| `pyfits.writeto('newfile.fits',data,hdr)` | create new fits file |
| `pyfits.append('existingfile.fits',data, hdr)` | append to file |
| `pyfits.update('existingfile.fits',data, hdr, ext=3)` | update file |

## Matplotlib/pylab

| Plot functions | |
|---|---|
| `acorr(x)` | plot autocorrelation function |
| `bar(x,y)` | bar charts |
| `barh(x,y)` | horizontal bar charts |
| `broken_barh(x,y)` | a set of horizontal bars with gaps |
| `boxplot(x)` | box and whisker plots |
| `cohere(x,y)` | plot of coherence |
| `contour(A)` | contour plot |
| `contourf(A)` | filled contours |
| `csd(x,y)` | plot of cross spectral density |
| `errorbar(x,y,yerr,xerr)` | errorbar plot |
| `hist(x)` | histogram plot |
| `imshow(A)` | display image within axes boundaries (resamples image) |
| `loglog(x,y)` | log log plot |
| `matshow(A)` | display a matrix in a new figure preserving aspect |
| `pcolor(A)` | make a pseudocolor plot |
| `pcolormesh(A)` | make a pseudocolor plot using a quadrilateral mesh |
| `pie(x)` | pie chart |
| `plot(x,y)` | basic x, y plots |
| `plot_date(x,y)` | plot using x or y argument as date values and label axis accordingly |
| `polar(x)` | polar plot |
| `psd(x)` | power spectral density (FFT) |
| `quiver(x,y)` | vector field plot |
| `scatter(x,y)` | scatter plot |
| `semilogx(x,y)` | log x, linear y, x y plot |
| `semilogy(x,y)` | linear x, log y, x y plot |
| `specgram` | spectrogram plot (FFT) |
| `stem(x,y)` | stem plot (similar to histogram) |
| `spy(A)` | plot sparsity pattern using markers |
| `xcorr(x,y)` | plot the autocorrelation function of x and y |

**Examples**

```
from matplotlib.font_manager import fontManager, FontProperties
font= FontProperties(size='x-small')
pylab.legend(loc='lower left', prop=font)

params = {'backend': 'ps',
          'axes.labelsize': 10,
          'text.fontsize': 10,
          'legend.fontsize': 10,
          'xtick.labelsize': 8,
          'ytick.labelsize': 8,
          'text.usetex': False,
          'figure.figsize':  [3.4, 2.1]}
pylab.rcParams.update(params)
```

Based on:

⋆ *Using Python for Interactive Data Analysis* by Perry Greenfield and Robert Jedrzejewski, Space Telescope Science Institute, May 10, 2007

⋆ *Tentative NumPy Tutorial*, `www.scipy.org`, March 01, 2009.

## Matplotlib/Pylab

| | |
|---|---|
| **Decorations** | annotate something in figure |
| `annotate` | |
| `arrow` | add arrow to plot |
| `axhline` | plot horizontal line across axes |
| `axvline` | plot vertical line across axes |
| `axhspan` | plot horizontal bar across axes |
| `axvspan` | plot vertical bar across axes |
| `clabel` | label contour lines |
| `clim` | adjust color limits of current image |
| `fill` | make filled polygons |
| `grid` | set whether grids are visible |
| `legend` | add legend to current axes |
| `rgrids` | customize the radial grids and labels for polar plots |
| `table` | add table to axes |
| `text` | add text to axes |
| `thetagrids` | for polar plots |
| `title` | add title to axes |
| `xlabel` | add x axes label |
| `ylabel` | add y axes label |
| `xlim` | set/get x axes limits |
| `ylim` | set/get y axes limits |
| `xticks` | set/get x ticks |
| `yticks` | set/get y ticks |
| **Figure functions** | |
| `colorbar` | add colorbar to current Figure |
| `figimage` | display unresampled image in Figure |
| `figlegend` | display legend for Figure |
| `figtext` | add text to Figure |

## Matplotlib/Pylab

| Objects | |
|---|---|
| `axes` | create axes object on current figure |
| `box` | set axis frame state on or off |
| `cla` | clear current axes |
| `clf` | clear current figure |
| `close` | close a figure window |
| `delaxes` | delete axes object from the current figure |
| `draw` | force a redraw of the current figure |
| `figure` | create or change active figure |
| `gca` | get the current axes object |
| `gcf` | get the current figure |
| `gci` | get the current image |
| `getp` | get a handle graphics property |
| `hold` | set the hold state (overdraw or clear?) |
| `ioff` | set interactive mode off |
| `ion` | set interactive mode on |
| `isinteractive` | test for interactive mode |
| `ishold` | test for hold mode |
| `plotting` | list plotting commands |
| `rc` | control the default parameters |
| `savefig` | save the current figure |
| `setp` | set a handle graphics property |
| `show` | show the current figures (for non-interactive mode) |
| `subplot` | create an axes within a grid of axes |
| `subplot_tool` | launch the subplot configuration tool |
| **Color Maps** | |
| `autumn` | set the default colormap to autumn |
| `bone` | set the default colormap to bone |
| `cool` | set the default colormap to cool |
| `copper` | set the default colormap to copper |
| `flag` | set the default colormap to flag |
| `gray` | set the default colormap to gray |
| `hot` | set the default colormap to hot |
| `hsv` | set the default colormap to hsv |
| `jet` | set the default colormap to jet |
| `pink` | set the default colormap to pink |
| `prism` | set the default colormap to prism |
| `spring` | set the default colormap to spring |
| `summer` | set the default colormap to summer |
| `winter` | set the default colormap to winter |
| `spectral` | set the default colormap to spectral |