# Python & Pylab Cheat Sheet

## Running

| | |
|---|---|
| `python3` | standard python shell. |
| `ipython3` | improved interactive shell. |
| `ipython3 --pylab` | ipython including pylab |
| `python3` *file.py* | run *file.py* |
| `python3 -i` *file.py* | run *file.py*, stay in interactive mode |

To quit use `exit()` or [ctrl]+[d]

## Getting Help

| | |
|---|---|
| `help()` | interactive Help |
| `help(`*object*`)` | help for *object* |
| *object*`?` | ipython: help for *object* |
| *object*`??` | ipython: extended help for *object* |
| `%magic` | ipython: help on magic commands |

## Import Syntax, e.g. for $\pi$

| | |
|---|---|
| `import numpy` | use: `numpy.pi` |
| `import numpy as np` | use: `np.pi` |
| `from numpy import pi` | use: `pi` |
| `from numpy import *` | use: `pi` (use sparingly) |

## Types

| | | | | |
|---|---|---|---|---|
| `i = 1` | Integer | | | |
| `f = 1.` | Float | | | |
| `c = 1+2j` | Complex | with this: | | |
| `True/False` | Boolean | | `c.real` | 1.0 |
| `'abc'` | String | | `c.imag` | 2.0 |
| `"abc"` | String | | `c.conjugate()` | 1-2j |

## Operators

| mathematics | | comparison | |
|---|---|---|---|
| `+` | addition | `=` | assign |
| `-` | subtraction | `==` | equal |
| `*` | multiplication | `!=` | unequal |
| `i/i` | float division | `<` | less |
| `i//i` | int division | `<=` | less-equal |
| `**` | power | `>=` | greater-equal |
| `%` | modulo | `>` | greater |

## Basic Syntax

| | |
|---|---|
| `input('`*foo*`')` | read string from command-line |
| `class` *Foo*`(Object):` ... | class definition |
| `def` *bar*`(args):` ... | function/method definition |
| `if` *c*`:` ... `elif` *c*`:` ... `else:` | branching |
| `try:` ... `except` *Error*`:` ... | exception handling |
| `while` *cond*`:` ... | while loop |
| `for` *item* `in` *list*`:` ... | for loop |
| `[`*item* `for` *item* `in` *list*`]` | for loop, list notation |

## Useful tools

| | |
|---|---|
| `pylint` *file.py* | static code checker |
| `pydoc3` *file* | parse docstring to man-page |
| `python3 -m doctest` *f.py* | run examples in docstring |
| `python3 -m pdb` *file.py* | run in debugger |

# NumPy & Friends

The following import statement is assumed:
`from pylab import *`

## General Math

`f`: float, `c`: complex:

| | |
|---|---|
| `abs(c)` | absolute value of f or c |
| `sign(c)` | get sign of f or c |
| `fix(f)` | round towards 0 |
| `floor(f)` | round towards − inf |
| `ceil(f)` | round towards + inf |
| `round(f, p)` | round f to p places |
| `angle(c)` | angle of complex number |
| `sin(c)` | sinus of argument |
| `arcsin(c)` | arcsin of argument |
| `cos, tan,...` | analogous |

## Defining Lists, Arrays, Matrices

`l`: list, `a`: array:

| | |
|---|---|
| `[[1,2],[3,4,5]]` | basic list |
| `array([[1,2],[3,4]])` | array from "rectangular" list |
| `matrix([[1,2],[3,4]])` | matrix from 2d-list |
| `range(min, max, step)` | integers in [min, max) |
| `list(range(...))` | list from range() |
| `arange(min, max, step)` | integer array in [min, max) |
| `frange(min, max, step)` | float array in [min, max] |
| `linspace(min, max, num)` | num samples in [min, max] |
| `meshgrid(x,y)` | create coord-matrices |
| `zeros, ones, eye` | generate special arrays |

## Element Access

| | |
|---|---|
| `l[row][col]` | list: basic access |
| `l[min:max]` | list: range access [min,max) |
| `a[row,col]` or `a[row][col]` | array: basic access |
| `a[min:max,min:max]` | array: range access [min,max) |
| `a[`*list*`]` | array: select indices in *list* |
| `a[np.where(`*cond*`)]` | array: select where *cond* true |

## List/Array Properties

| | |
|---|---|
| `len(l)` | size of first dim |
| `a.size` | total number of entries |
| `a.ndim` | number of dimensions |
| `a.shape` | size along dimensions |
| `ravel(l)` or `a.ravel()` | convert to 1-dim |
| `a.flat` | iterate all entries |

## Matrix Operations

`a`: array, `M`: matrix:

| | |
|---|---|
| `a*a` | element-wise product |
| `dot(a,a)` or `M*M` | dot product |
| `cross(a,a)` | cross product |
| `inv(a)` or `M.I` | inverted matrix |
| `transpose(a)` or `M.T` | transposed matrix |
| `det(a)` | calculate determinate |

## Statistics

| | |
|---|---|
| `sum(l,d)` or `a.sum(d)` | sum elements along d |
| `mean(l,d)` or `a.mean(d)` | mean along d |
| `std(l,d)` or `a.std(d)` | standard deviation along d |
| `min(l,d)` or `a.min(d)` | minima along d |
| `max(l,d)` or `a.max(d)` | maxima along d |

## Misc functions

| | |
|---|---|
| `loadtxt(`*file*`)` | read values from *file* |
| `polyval(coeff,xvals)` | evaluate polynomial at xvals |
| `roots(coeff)` | find roots of polynomial |

# Plotting

## Plot Types

| | |
|---|---|
| `plot(xvals, yvals, 'g+')` | mark 3 points with green + |
| `errorbar()` | like `plot` with error bars |
| `semilogx(), semilogx()` | like `plot`, semi-log axis |
| `loglog()` | double logarithmic plot |
| `polar(phi_vals, rvals)` | plot in polar coordinates |
| `hist(vals, n_bins)` | create histogram from values |
| `bar(low_edge, vals, width)` | create bar-plot |
| `contour(xvals,yvals,zvals)` | create contour-plot |

## Pylab Plotting Equivalences

| | |
|---|---|
| `figure()` | `fig = figure()`<br>`ax = axes()` |
| `subplot(2,1,1)` | `ax = fig.add_subplot(2,1,1)` |
| `plot()` | `ax.plot()` |
| `errorbar()` | `ax.errorbar()` |
| `semilogx, ...` | analogous |
| `polar()` | `axes(polar=True)` and `ax.plot()` |
| `axis()` | `ax.set_xlim(), ax.set_ylim()` |
| `grid()` | `ax.grid()` |
| `title()` | `ax.set_title()` |
| `xlabel()` | `ax.set_xlabel()` |
| `legend()` | `ax.legend()` |
| `colorbar()` | `fig.colorbar(plot)` |

## Plotting 3D

`from mpl_toolkits.mplot3d import Axes3D`

`ax = fig.add_subplot(...,projection='3d')`

| | |
|---|---|
| or `ax = Axes3D(fig)` | create 3d-axes object |
| `ax.plot(xvals, yvals, zvals)` | normal plot in 3d |
| `ax.plot_wireframe` | wire mesh |
| `ax.plot_surface` | colored surface |