



# Python for Astronomers

more on numpy & matplotlib

# Numpy arrays and funcs

```
# l8e1.py
import numpy
def sincos(a):
    # automatic conversion to arrays:
    return numpy.sin(a), numpy.cos(a)

x = [0,numpy.pi/4, numpy.pi/2]
s, c = sincos(x)          # tuple unpacking !!!
print s
print c
```

# Numpy arrays and funcs

```
# l8e2.py
import numpy
def addmul(a,b,c):
    return (a+b)*c

a = [1., 4., 9.]
b = [2., 8., 9.]
c = [2., .5, 1/3.]

print addmul(a,b,c)  # fails
```

# Numpy arrays and funcs

```
# best practice to implement functions
# that work with lists and numpy arrays


import numpy
def addmul(a,b,c):
    a = numpy.asarray(a)
    b = numpy.asarray(b)
    c = numpy.asarray(c)
    return (a+b)*c
# BUT don't manipulate the arrays, eg. a+=1!
```

# Numpy arrays and funcs

```
# alternative:


import numpy
def addmul(a,b,c):
    return ( numpy.asarray(a) +      \
             numpy.asarray(b)    ) * \
               numpy.asarray(c)
```

# matplotlib

More useful matplotlib plotting commands

# matplotlib

```
$ ipython -pylab
>>> x = linspace(0, 2*pi, 1000)
>>> plot(x, sin(x))
>>>
>>> fill(x, sin(x))
>>>
>>> # to fill the area between two lines
>>> # only matplotlib 0.98
>>> fill_between(x, sin(x), sin(x)+0.25,
          color='g')
```

# zorder

```
>>> fill_between(x, sin(x), sin(x)+0.1,
                 color='r', zorder=2)
>>>
```

- The zorder attribute determines the order of drawing
- default:
  - patches (zo=1, `hist`), lines (2, `plot`), text (3)
  - then: in order of plotting
- can be set to arbitrary integer numbers, e.g. 10

# errorbars

```
>>> clf()
>>> x = linspace(0, 2*pi, 13)
>>> ye = 0.25*fabs(randn(13))
>>>
>>> errorbar(x, sin(x), xerr=0.1, yerr=ye)
>>>
>>> # “errorbar” plots data and errorbars,
>>> # errors can be scalar or arrays
```

# errorbars

```
>>> errorbar(x, sin(x)+0.2, yerr=ye,
...    elinewidth = 2, ecolor='red', lw=4)
>>>
>>> # Additional attributes (defaults)
>>> # capsize=3
>>> # barsabove=False
>>> # lolims, uplims, xlolims, xuplims = False
>>>
>>> # all other kwargs are passed on to plot
>>> # e.g. supports plot formating like lw=4
```

# contour plots

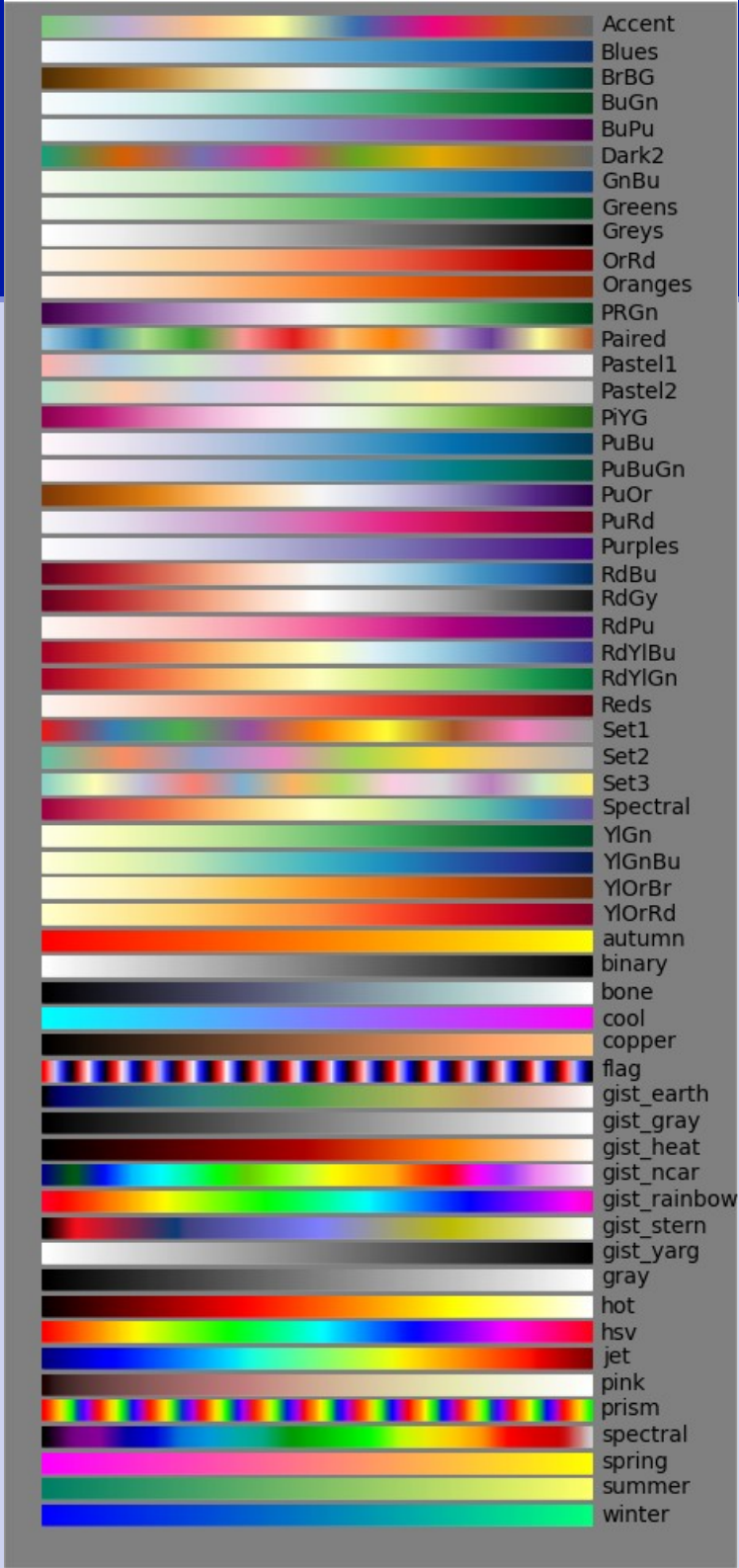
```
>>> x = linspace(-pi, pi, 100)
>>> y = linspace(-pi, pi, 100)
>>>
>>> X,Y = meshgrid(x,y)
>>> Z = cos(X) + 0.5*cos(Y)
>>> contour(X,Y,Z)
```

# contour plots

```
>>> # 10 automatically chosen levels
>>> contour(X,Y,Z, 10)
>>>
>>> # contour lines at specific values
>>> lv = [-sqrt(0.5), 0, sqrt(0.5)]
>>> contour(X,Y,Z, lv)
```

# contour plots

```
>>> # colours of lines
>>> # using a colormap
>>> contour(X,Y,Z, cmap=cm.hot)
>>> # see colormaps.png
>>>
>>> # Or specify colors (note
>>> # the additional s)
>>> contour(X,Y,Z, colors='k')
>>>
```



# contour plots

```
>>> # filled contour plots:
>>> contourf(X,Y,Z, cmap=cm.cold)
>>>                     # no lines are drawn !!!
>>> contour(X,Y,Z, colors='k',
...   linestyles='solid', linewidths=2)
>>>
>>> # linewidths, linestyles & colors can also
>>> # be lists of values !!!
>>> contour(X,Y,Z, colors=['k']*7,
...    linestyles=['solid']*7)
```

# labeling contour plots

```
>>> clf()
>>> CS = contour(X,Y,Z)
>>> clabel(CS, inline=True, inline_spacing=10)
>>>
>>> # additional attributes:
>>> # colors=None (use same color as lines)
>>> # or use string or tuple color definition
>>> #   fmt = '%1.f' as string formatting
```

# axis ticks

```
>>> clf()
>>> x = linspace(0, 8e4 ,12)
>>> plot(x, x**2)
>>>
>>> locs, lbls = xticks()
>>> print locs
>>> print lbls
>>> mylocs = locs[::2]
>>> mylbls = list('abcd')
>>> xticks( locs, mylbls)
```

# axis ticks (the hard way)

```
>>> ax = gca()   # API: get current axes
>>> yfmt = ax.yaxis.get_major_formatter()
>>>    # a formatter object
>>>
>>> yfmt.set_powerlimits((-2,10))
>>> draw()
>>>
>>> yfmt.set_scientific(False)
>>> draw()
```

# axis ticks

- y-/x-axis have
  - major and minor locators:
    Determine **where** the ticks are
  - major and minor formattors
    Control formating of ticks

# Locator and Formatter

```
>>> from matplotlib.ticker import \
...   MultipleLocator
>>>
>>> ax = gca()
>>> loc2 = MultipleLocator(2e9)
>>> ax.yaxis.set_major_locator(loc2)
>>> draw()
>>>
>>> loc2m = MultipleLocator(5e8)
>>> ax.yaxis.set_minor_locator(loc2m)
>>> draw()
```

# Locator and Formatter

```
>>> from matplotlib.ticker import \
...     FormatStrFormatter
>>>
>>> fmt2 = FormatStrFormatter('%.1e')
>>> ax.yaxis.set_major_formatter(fmt2)
>>> draw()
>>>
```

# subplot adjust

```
>>> # adjust plotting of axes
>>> # default values are
>>> subplots_adjust(left = 0.125,
...    right = 0.9,
...    bottom = 0.1,
...    top = 0.9,
...    wspace = 0.2,
...    hspace = 0.2)
```

# Saving figures

```
# l8mpl1.py
from matplotlib import pyplot
import numpy

x = numpy.linspace(0, 2*numpy.pi, 1000)

pyplot.plot(x, numpy.sin(x), 'g--')
pyplot.savefig('sin_function.png')
```

# Customizing matplotlib

- Place a matplotlibrc file here:

  `~/.matplotlib/matplotlibrc`

- Template file

  `/home/mmetz/py2008/matplotlibrc`

  (see also Py2008 homepage)

# Customizing matplotlib

- If your happy with the default config you don't need the matplotlibrc file
- Customize matplotlib
  - well documented template !!!
  - configure backend (GTK, Qt, ...)
  - default figure sizes, line styles, text, ... (e.g. text.usetex)

# Customizing matplotlib

```
>>> # change parameter individually in scripts
>>> print rcParams.keys()
>>>
>>> rcParams['text.usetex'] = True
>>> rcParams['contour.negative_linestyle'] = \
...   'solid'
>>>
>>> rcdefaults()                # reset to default
```

# matplotlib

Have a look at the matplot docs &
especially at the examples !!!