

Python/Healpy Tutorial

<http://github.com/zonca/healpytut/>

Preparation

Carver:

```
module load python
module load cmbdev
```

Your laptop

install Entough Python Distribution for Win/Mac http://www.enthought.com/products/epd_free.php

Components

- Ipython: Interactive shell
- Numpy: Array math
- Matplotlib: Plotting
- Scipy: Advanced scientific tools [FFT, spline, signal processing]
- Healpy: Healpix for python

Overview

- Basic types
- Arrays
- Plotting
- Code organization: modules, packages
- Healpy

Setup environment

```
.ipython/ipythonrc pdb 1, autocall 2
get help on function by calling: healpy.nside2npix?
```

Basic types

Lists

```
52
53 test_list = []
54 test_list.append(9)
55 print(test_list)
```

[9]

```
56
57 test_list.append("quite a long string")
58 test_list.append([1, 3, 4])
59 test_list.append(10)
60 print(test_list)
```

[9, 'quite a long string', [1, 3, 4], 10]

```
60
61 #Replace
62 test_list[2] = 1
63 print(test_list)

[9, 'quite a long string', 1, 10]
```

```
64
65 #Slicing
66 print(test_list[:2])

[9, 'quite a long string']
```

```
67 print(test_list[-1:])

[10]
first python WARNING Last element is excluded!!!
71 print(test_list[1:2])

['quite a long string']
this is C
```

```
76
77 for i in range(len(test_list)):
78     print(test_list[i])

9
quite a long string
1
10
this is Python
```

```
81
82 for element in test_list:
83     print(element)

9
quite a long string
1
10
```

Tuple

Like lists but not mutable, used for string interpolation, return of functions

```
87 test_tuple = (3, 4)
88 print(test_tuple[0])

3
```

```
89
90 #test_tuple[0] = 2
```

Dictionary

```
95
96 test_dict = {}
97
98 test_dict["LFI28M"] = 127.
99 test_dict["LFI28S"] = 12.
100
101 print(test_dict)

{'LFI28S': 12.0, 'LFI28M': 127.0}
```

```
102
103 print(test_dict["LFI28M"])

127.0

107
108 for k,v in test_dict.iteritems(): #Dictionary is **NOT ORDERED**
109
110     print("Channel %s has value %.2f" % (k,v)) #C-style string formatting

Channel LFI28S has value 12.00
Channel LFI28M has value 127.00
```

Strings

```
109
110 # type of quotes does not matter
111 test_string = "a quite long string"
112 test_string = 'a quite long string'
113
114 # multiline strings
115 test_string = """
116 This is a multiline
117 string,
118 keeps formatting"""
119
120 print(test_string)

This is a multiline
string,
keeps formatting

121
122 #strings interpolation
123
124 print("either using " + str(1.0) + " concatenation or interpolation for int %04d
    , float %.2f, exp %.1e" % (3, 1/3., 2.3))

either using 1.0 concatenation or interpolation for int 0003, float 0.33, exp
2.3e+00
```

Functions

```
133
134 def sum_diff(a, b, take_abs=False):
135     if take_abs:
136         return abs(a+b), abs(a-b)
137     else:
138         return a+b, a-b
139
140
141 a=2; b=3
142 absum, abdiff = sum_diff(a, b)
143 ab_sumdiff = sum_diff(a, b)
144
145 print(absum)

5

141 print(ab_sumdiff)
```

```
(5, -1)
```

Integer division

second python **WARNING**

```
146 # 1/2 = 0 because they are integers
147 # 1./2 = .5 because 1. is float
148 # to avoid do at beginning of software
149 # from __future__ import division
```

Arrays

```
156
157 import numpy as np
158 a = np.array([1, 4, 5])
159 print(a.dtype)
```

```
int32
```

```
159 a[0] = .9
160 print(a)
```

```
[0 4 5]
```

Warning type is integer

```
164
165 a = np.array([1, 4, 5], dtype=np.double)
166 a[0] = .9
167 print(a)
```

```
[ 0.9  4.  5. ]
```

```
167
168 #same slicing as lists
169 a = np.arange(20)
170 print(a[10:18:2]) #2 is the step
```

```
[10 12 14 16]
```

```
171
172 #2D same as IDL, shape is always a **tuple**
173 a = np.zeros((3, 4))
174 a[1, 3] = 2
175 print(a)
```

```
[[ 0.  0.  0.  0.]
 [ 0.  0.  0.  2.]
 [ 0.  0.  0.  0.]]
```

```
176
177 #array itself is an object, so it has methods associated
178 print(a.mean())
```

```
0.166666666666667
```

```
179 print(a.std())
```

```
0.552770798393
```

```
180 print(a.flatten())
```

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  2.  0.  0.  0.  0.]
```

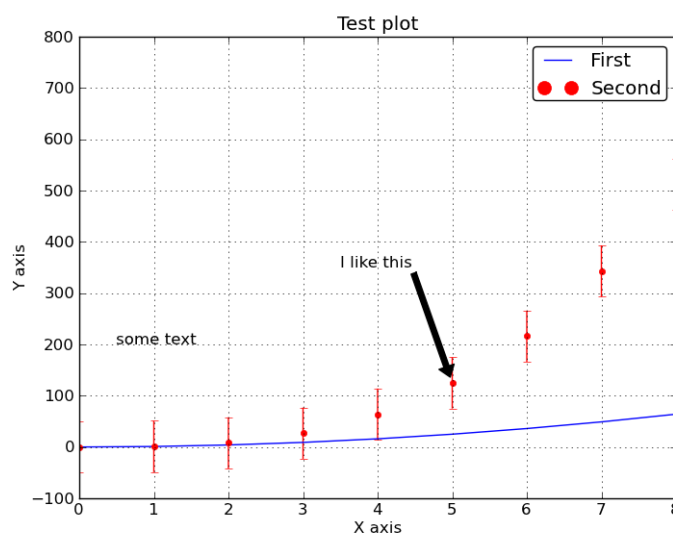
Plotting

Interactively with ipython -pylab

```

188
189 from pylab import *
190 plot(arange(10), arange(10)**2, label='First')
191 errorbar(arange(10), arange(10)**3, 50., None, 'r.', markersize=10, label='
    Second')
192 annotate('I like this', xy=(5, 125), xytext=(3.5, 350),
193         arrowprops=dict(facecolor='black', shrink=0.05),
194         )
195 text(0.5, 200, 'some text')
196 grid()
197 legend(loc=0)
198 xlabel('X axis'); ylabel('Y axis')
199 xlim([0, 8])
200 title('Test plot')
201 savefig('plot.png')
202 show()

```



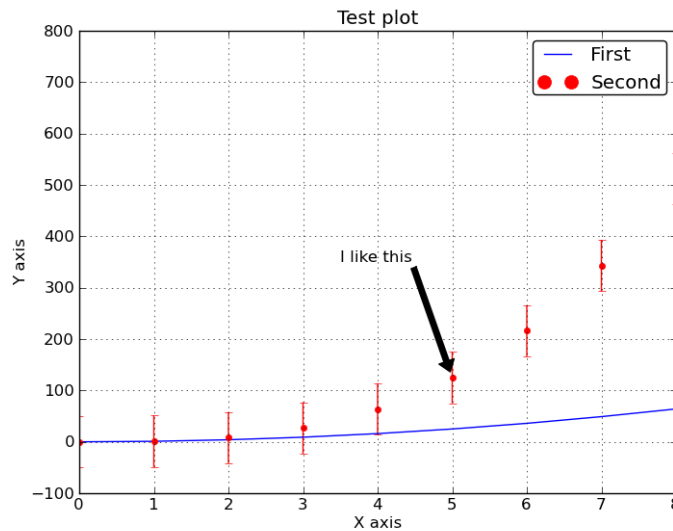
In software

USE NAMESPACES

```

208
209 import matplotlib.pyplot as plt
210 import numpy as np
211 plt.figure()
212 plt.plot(np.arange(10), np.arange(10)**2, label='First')
213 plt.errorbar(np.arange(10), np.arange(10)**3, 50., None, 'r.', markersize=10,
214             label='Second')
215 plt.annotate('I like this', xy=(5, 125), xytext=(3.5, 350),
216             arrowprops=dict(facecolor='black', shrink=0.05),
217             )
218 plt.grid()
219 plt.legend(loc=0)
220 plt.xlabel('X axis'); plt.ylabel('Y axis')
221 plt.xlim([0, 8])
222 plt.title('Test plot')
223 plt.savefig('plot.png')
224 show()

```



no namespaces?

```
225 title = "My title"
226 title("Other title")
```

Error:

Traceback (most recent call last):

```
File "/home/zonca/p/software/pyreport/pyreport/main.py", line 180, in executeblock
    exec block_text in self.namespace
```

```
File "<string>", line 3, in <module>
```

TypeError: 'str' object is not callable

Code organization

Modules

```
233
234 # Modules are just .py files containing functions, simplest library
235 # usually they can be imported in other scripts or executed
236
237 if __name__ == '__main__':
238     print('Executing this just if directly called as python this_script.py')
239
240 #example
241 import healpy
242 print(healpy.pixelfunc)
```

```
<module 'healpy.pixelfunc' from '/usr/local/lib/python2.7/dist-packages/healpy
/pixelfunc.pyc'>
```

```
243 print(healpy.pixelfunc.nside2npix)
```

```
<function nside2npix at 0xb11f4fc>
```

Packages

collection of modules in a folder with an `__init__.py` file which defines what is imported on the main level

```
250
251 # for example:
252
253 import healpy
```

```

254 print (healpy)

    <module 'healpy' from '/usr/local/lib/python2.7/dist-packages/healpy/__init__.pyc'>
255 print (healpy.nside2npix)

```

```

    <function nside2npix at 0xb11f4fc>

```

Best practice Start with a single module and then split into several modules importing in `__init__.py` the most important functions and classes, *NOT* internal functions.

Healpy

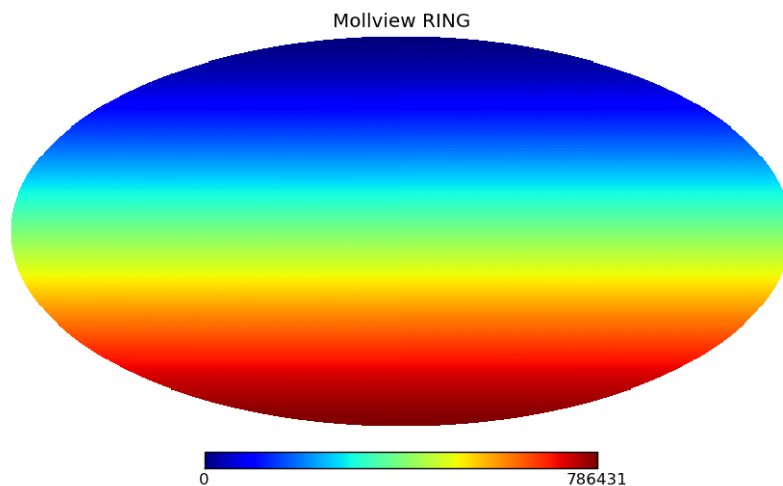
in background calling C++ Healpix for most transforms

healpy by default works in RING

```

266
267 import healpy
268
269 m = np.arange(healpy.nside2npix(256))
270 healpy.mollview(m, min=0, max=m.max(), title='Mollview RING', nest=False)
271 show()

```

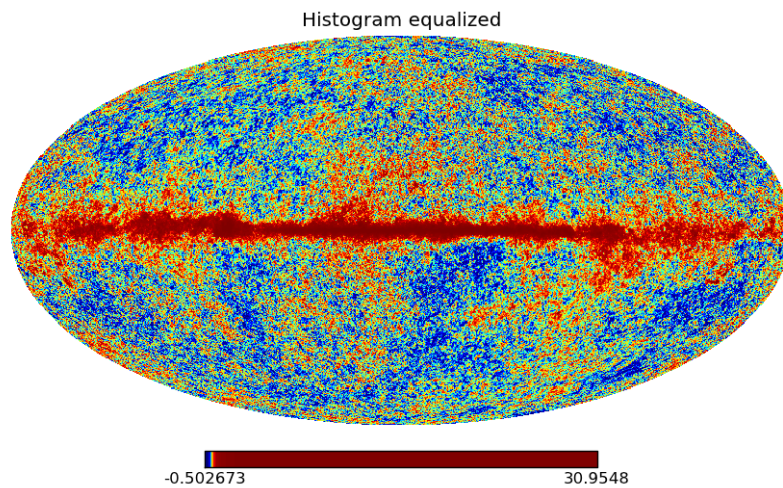


http://lambda.gsfc.nasa.gov/data/map/dr4/skymaps/7yr/raw/wmap_band_imap_r9_7yr_W_v4.fits

```

274
275 filename = 'wmap_band_imap_r9_7yr_W_v4.fits'
276 #filename = '/global/scratch/sd/planck/user/zonca/healpytut/
    wmap_band_imap_r9_7yr_W_v4.fits'
277 m = healpy.read_map(filename) #by default converts to RING!!
278
279 healpy.mollview(m, title='Histogram equalized', nest=False, norm='hist')
280 show()

```



```

280 m = healpy.read_map(filename, nest=True) #keeps nested
281
282 healpy.mollview(m, coord=['G','E'], title='Linear scale', unit='mK', nest=True,
283               min=-1,max=1, xsize=2000) #xsize increases resolution
284 healpy.graticule()

```

0.0 180.0 -180.0 180.0

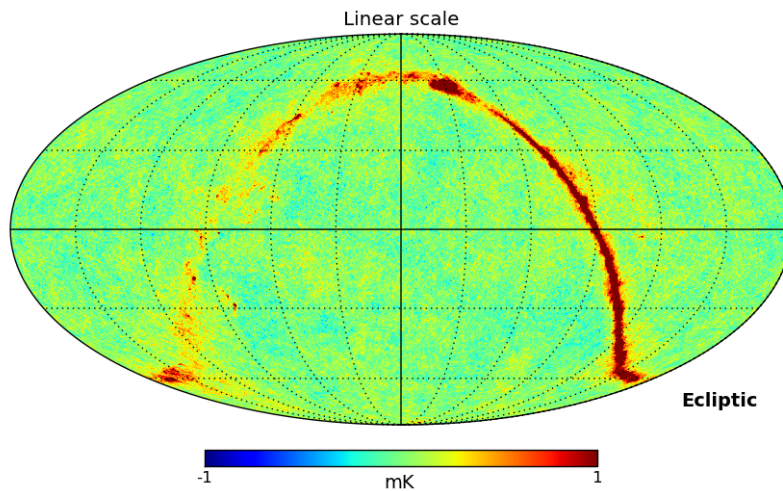
The interval between parallels is 30 deg -0.00'.

The interval between meridians is 30 deg -0.00'.

```

282 show()

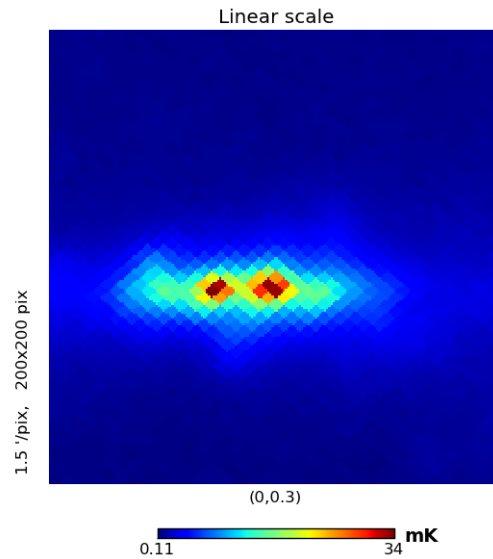
```



```

284
285 healpy.gnomview(m, rot=[0,0.3], title='Linear scale', unit='mK', format='%.2g',
286               nest=True)
286 show()

```

```

288
289 print (healpy.fit_dipole (m, gal_cut=20)) # degrees
      (0.01858625016436204, array([-0.00071935,  0.00231184,  0.00520954]))

```

Smoothing

```

292
293 m_smoothed = healpy.smoothing (m, fwhm=60, arcmin=True)
294 healpy.mollview (m_smoothed, min=-1, max=1, title='Map smoothed 1 deg')

```

Rotator

```

298
299 rot = healpy.Rotator (coord=['G', 'E'])
300 theta_gal, phi_gal = np.pi/2., 0.
301 theta_ecl, phi_ecl = rot (theta_gal, phi_gal)
302 print (theta_ecl, phi_ecl)
      (1.6674228671489519, -1.625964003063237)

```

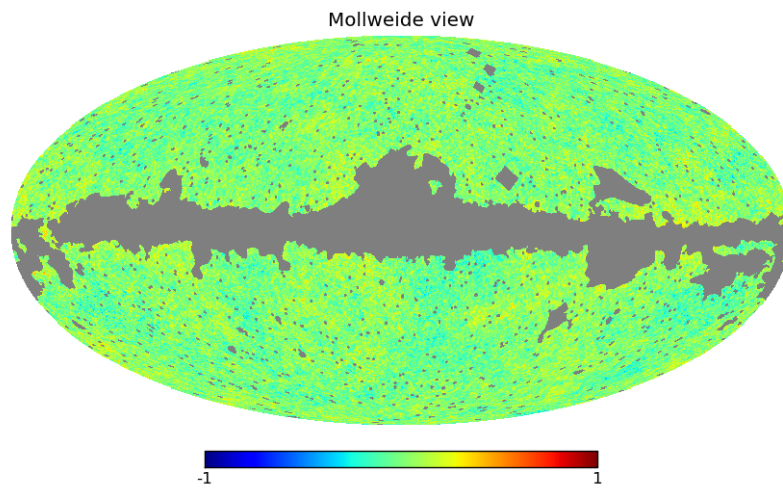
Masking

http://lambda.gsfc.nasa.gov/data/map/dr4/ancillary/masks/wmap_temperature_analysis_mask_r9_7yr_v4.fits

```

307
308 mask_filename = 'wmap_band_imap_r9_7yr_W_v4.fits'
309 #mask_filename = '/global/scratch/sd/planck/user/zonca/healpytut/
      wmap_band_imap_r9_7yr_W_v4.fits'
310 mask = healpy.read_map (mask_filename).astype (np.bool)
311
312 m = healpy.read_map (filename)
313
314 #method 1: multiply arrays
315 m_masked = m.copy ()
316 m_masked[np.logical_not (mask)] = healpy.UNSEEN
317 healpy.mollview (m_masked, min=-1, max=1)
318 show ()

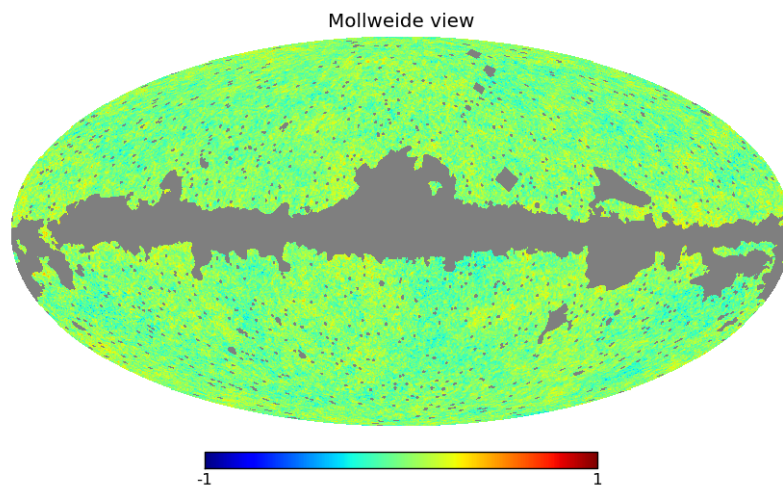
```



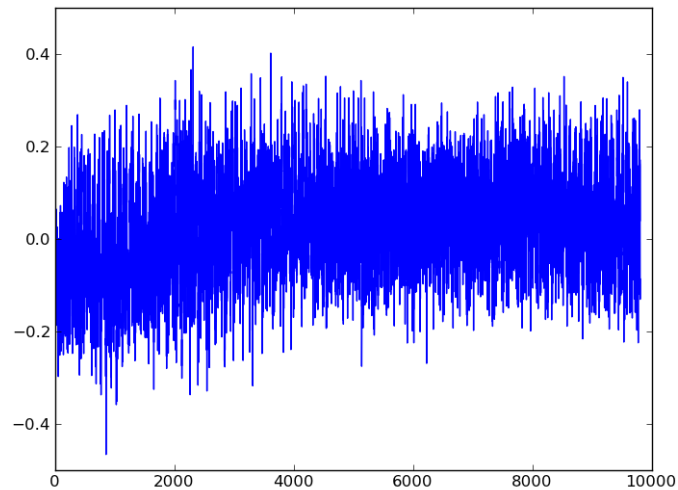
```
318
319 #method 2: numpy masked arrays
320 m_masked = healpy.ma(m)
321 print(m_masked)

[-0.12779275 -0.08507241  0.08297058 ...,  0.0255827  0.09494673
  0.03039758]
```

```
322 m_masked.mask = np.logical_not(mask)
323 healpy.mollview(m_masked.filled(), min=-1, max=1)
324 show()
```



```
325 figure()
326 plot(m_masked[:10000].compressed())
327 show()
```

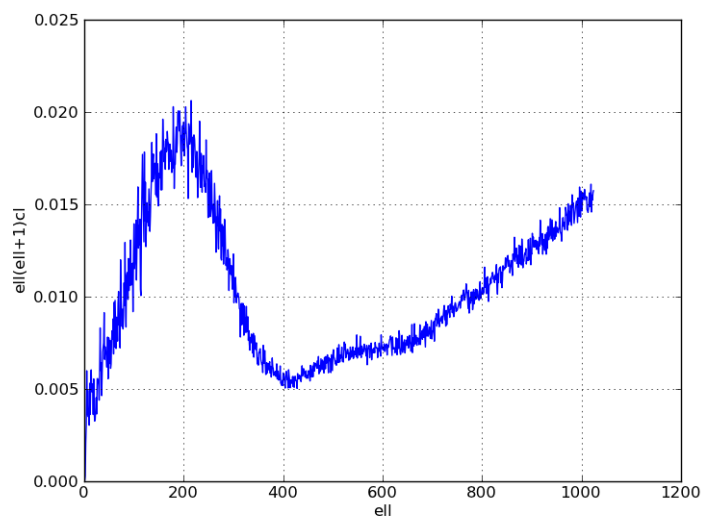


```
329  
330 healpy.write_map('wmap_masked.fits', m_masked.filled(), coord='G')
```

Overwriting existing file 'wmap_masked.fits'.

Spectra

```
334  
335 cl = healpy.anafast(m_masked.filled(), lmax=1024)  
336 ell = np.arange(len(cl))  
337 plt.figure()  
338 plt.plot(ell, ell * (ell+1) * cl)  
339 plt.xlabel('ell'); plt.ylabel('ell(ell+1)cl'); plt.grid()  
340 show()
```



```
341  
342 healpy.write_cl('cl.fits', cl)
```

Overwriting existing file 'cl.fits'.

```
344  
345 from glob import glob #bash like file pattern matching  
346  
347 print(glob('*.fits'))
```

```
[ 'wmap_masked.fits', 'wmap_band_imap_r9_7yr_W_v4.fits', 'cl.fits', '  
  wmap_temperature_analysis_mask_r9_7yr_v4.fits' ]
```