Introduction to Python

Kate Hedstrom

January 2011

Outline

- **Why Python?**
- **Examples**
  - Sample functions
  - Adding to a NetCDF file
  - Plotting with PyNGL
- **Interactive or as a script**

Python?

- **Open Source**
- **Matlab can be problematic:**
  - At home
  - At sea
  - Classroom situation
  - Automated web scripts
  - You aren’t rich
- **As a member of the Church of Open Source, I think people should have a free option**

Python Features

- Many scientific modules
- Able to load C/Fortran objects for speed
- Access to NetCDF Files
- Plotting options
- GUI tools
- Convenience of scripting, speed of compiled code

Python Versions

- We use version 2.4 to 2.6
- External packages depend on specific version, plus specific numpy version
- There is a Python 3.0 and 3.1 (even numbers are stable, odd numbers are development)
- "3.2 due for release around the turn of the year"
- Can play, or wait on packages

Example 1

- **Calendar functions in pure Python**

```
#/usr/bin/env python

# This script is used to change a Julian date into
# a dictionary containing the Gregorian equivalent.

from math import *
```

```
def leapyear(year):
    """
    Returns 1 if the provided year is a leap year, 0 if
    the provided year is not a leap year.
    """
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return 1
            else:
                return 0
        else:
            return 1
    else:
        return 0
```

```
def caldate_1900(Julian):
    """
    This is nearly a direct translation of a Matlab script
    to a Python script for changing a Julian date into a
    Gregorian date.
    """
# Bunch of stuff…
    ivd = (1,32,60,91,121,152,182,213,244,274,305,335,366)
    ivdl = (1,32,61,92,122,153,183,214,245,275,306,336,367)

    if (leapyear(yr) == 1):
        yday = ivdl[int(mo-1)]+d-1
    else:
        yday = ivd[int(mo-1)]+d-1
```

```
secs = (JulDay%1)*24*3600
sec = round(secs)
hour = floor(sec/3600)
min = floor((sec%3600)/60)
sec = round(sec%60)

print "Year: "+str(yr)
print "Year Day: "+str(yday)
print "Month: "+str(mo)
print "Day: "+str(d)
print "Hour: "+str(hour)
print "Min: "+str(min)
print "Sec: "+str(sec)
```

```
        cal = {'year':yr,'yearday':yday,'month':mo,'day':d,\
              'hour':hour,'minute':min,'second':sec}

        return cal

if __name__ == "__main__":

        mycal = caldate_1900(36761.5)
        # You can get any of the dictionary values by typing
        # mycal['key'] where key is replaced by the key to
        # the hash table such as 'hour'
        # print mycal['year']
```

Lists and Tuples

- **In this example, idv is a tuple, immutable:**

```
ivd = (1,32,60,91,121,152,182,213,244,274,305,335,366)
```

- **Changeable list would be:**

```
ivd = [1,32,60,91,121,152,182,213,244,274,305,335,366]
```

- **Both are 1-D, can do lists in lists (ugly)**
- **Also have dictionaries**

```
cal = {'year':yr,'yearday':yday,'month':mo,'day':d,\
           'hour':hour,'minute':min,'second':sec}
```

Used Interactively

```
% ipython
Python 2.6.5 (r265:79063, Jul 22 2010, 17:50:24)
Type "copyright", "credits" or "license" for more information.

IPython 0.10.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object'. ?object also works, ??
prints more.

In [1]: from caldate import *

In [2]: caldate_1900(34567.8)
```

PYTHONPATH

- Environment variable that's searched for python scripts
- A place to put stuff like caldate.py that doesn't belong in a system area
- A place to install pyroms if you don't have system permissions

Example 2

- **A little numpy, plus reading and writing NetCDF**

```
#/usr/bin/env python

# A Python Implementation of Adding an Additional
# Variable to a NetCDF File, and adding values
# to that file.

import numpy as np
import netCDF4 as nc
import time
```

```
# Have the netCDF file in local directory
root = nc.Dataset('NEP_grid_5a.nc', 'a') # modes w,a,r

eta_rhos = root.dimensions['eta_rho']
eta_len = len(eta_rhos)
xi_rhos = root.dimensions['xi_rho']
xi_len = len(xi_rhos)
print eta_len, xi_len

spawn_distance = root.createVariable('spawn_dist',\
    'f8', ('eta_rho', 'xi_rho',))
spawn_distance.long_name = "Spawn Distance Variable for
     Given Fish"
spawn_distance.fish_type = "Anchovies"
spawn_distance.created = "Created on: " + time.ctime
     (time.time())
```

```
mask_rho = root.variables['mask_rho']
lat_rho = root.variables['lat_rho']
lon_rho = root.variables['lon_rho']
depth = root.variables['h']

dims = root.dimensions
vars = root.variables
print dims
print vars

shaper = spawn_distance.shape
# This provides numpy objects i,j
for i in np.arange(shaper[0]):
    for j in np.arange(shaper[1]):
         spawn_distance[i,j] = i * j + 10
root.close()
```

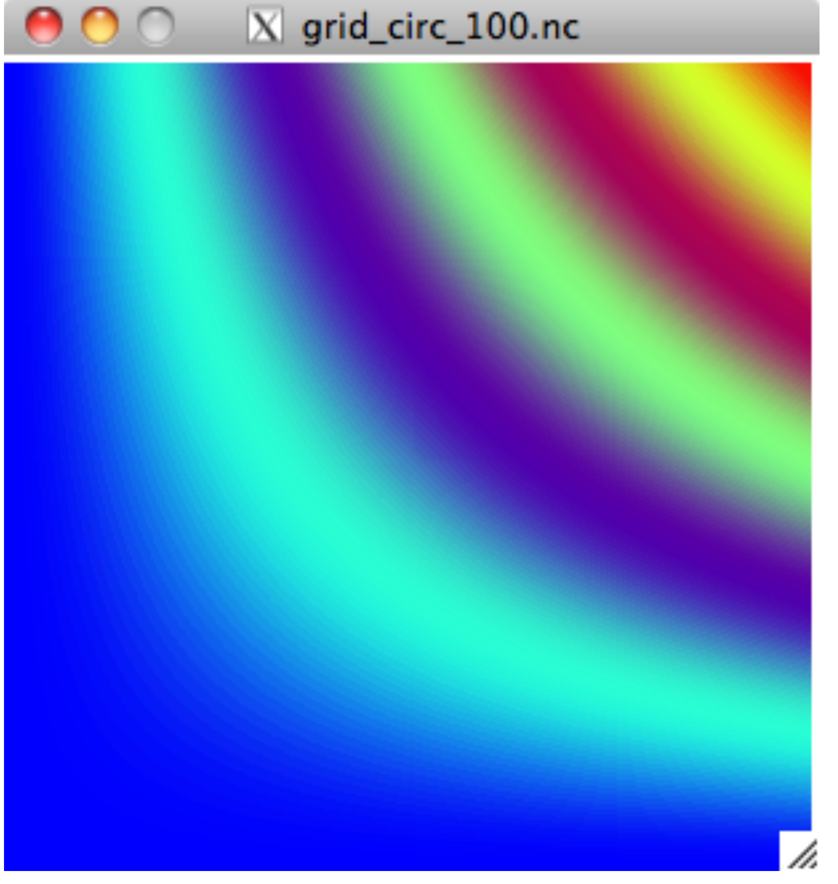
Output

```
102 102
OrderedDict([('xi_psi', <netCDF4.Dimension object at
0x1011cb190>), ('xi_rho', <netCDF4.Dimension object
at 0x1011cb1d0>), ('xi_u', <netCDF4.Dimension object
at 0x1011cb210>), ('xi_v', <netCDF4.Dimension object
at 0x1011cb250>), ('eta_psi', <netCDF4.Dimension
object at 0x1011cb290>), ('eta_rho',
<netCDF4.Dimension object at 0x1011cb2d0>),
('eta_u', <netCDF4.Dimension object at
0x1011cb310>), ('eta_v', <netCDF4.Dimension object
at 0x1011cb350>), ('one', <netCDF4.Dimension object
at 0x1011cb390>), ('two', <netCDF4.Dimension object
at 0x1011cb3d0>), ('bath', <netCDF4.Dimension object
at 0x1011cb410>)])
```

```
OrderedDict([
('xl', <netCDF4.Variable object at 0x1011c1e60>),
('el', <netCDF4.Variable object at 0x1011ca1b8>),
('JPRJ', <netCDF4.Variable object at 0x1011ca230>),
('PLAT', <netCDF4.Variable object at 0x1011ca2a8>),
('PLONG', <netCDF4.Variable object at 0x1011ca320>),
('ROTA', <netCDF4.Variable object at 0x1011ca398>),
('JLTS', <netCDF4.Variable object at 0x1011ca410>),
…
('mask_psi', <netCDF4.Variable object at 0x1011cc668>),
('angle', <netCDF4.Variable object at 0x1011cc6e0>),
('spawn_dist', <netCDF4.Variable object at
0x1011cc758>)])
```

NetCDF File

- **Can check with ncview:**



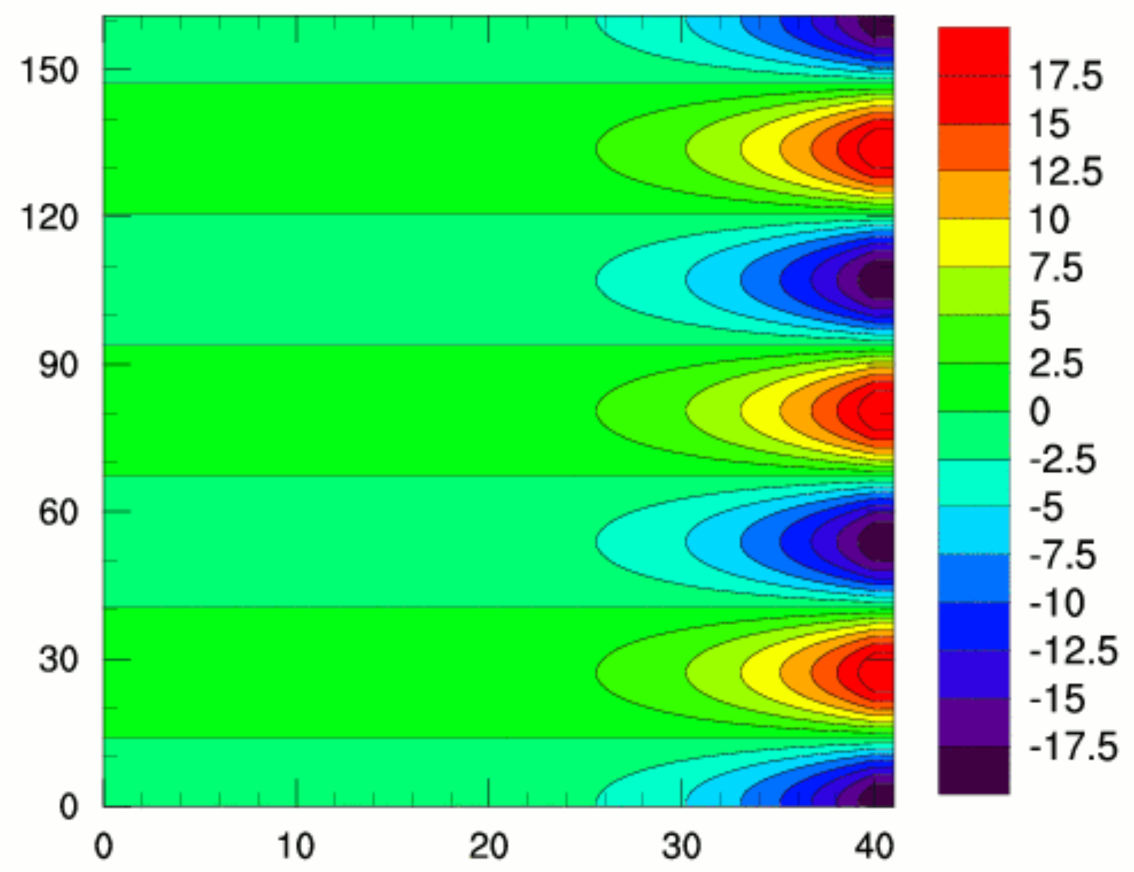
Example 3

- Plotting CIRCLE_POLAR with PyNGL
- Stealing from examples on the PyNGL website

```
#!/usr/bin/env python

import numpy, os
import Ngl
import netCDF4 as nc

root = nc.Dataset('ocean_his.nc', 'r')
zeta = root.variables['zeta']
xr = root.variables['x_rho']
yr = root.variables['y_rho']

zshape = zeta.shape
print zshape                    # gives (41, 162, 42)
```

```
#  Select a colormap and open a workstation.
#
rlist               = Ngl.Resources()
rlist.wkColorMap = "rainbow+gray"
wks_type = "ncgm"       # “ps”, “X11”
wks = Ngl.open_wks(wks_type,"circle",rlist)

resources = Ngl.Resources()

resources.nglSpreadColorStart = 15
resources.nglSpreadColorEnd   = -2

resources.cnFillOn         = True
resources.cnLinesOn        = True
resources.cnLineLabelsOn   = False
```

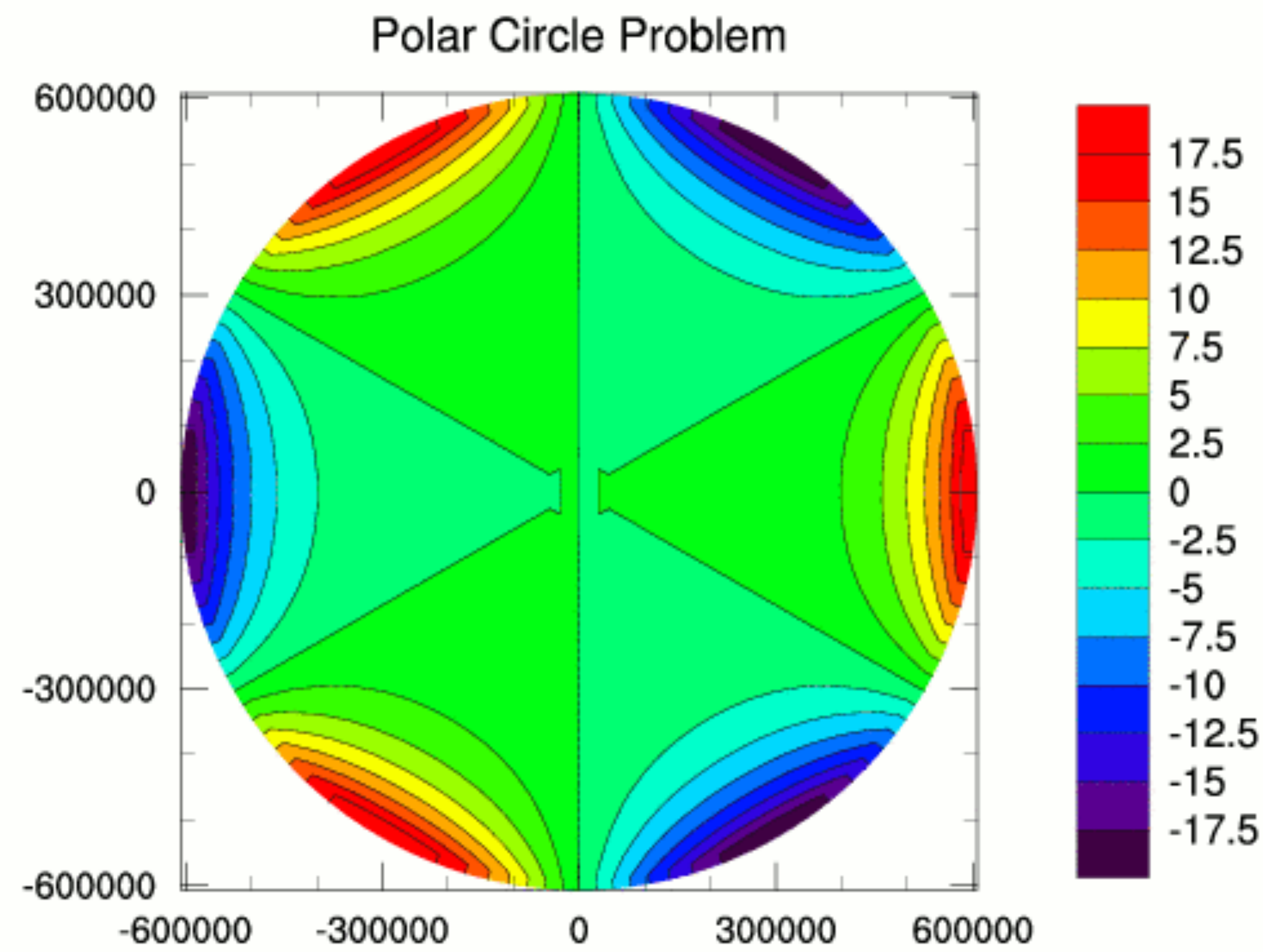
```
for k in numpy.arange(zshape[0]):
  contour = Ngl.contour(wks,zeta[k,:,:],resources)

Ngl.end()
```



Delauney Triangulation

After reading the variables:

```
x = numpy.reshape(xr, -1)
y = numpy.reshape(yr, -1)

# Automatic triangulation for 1-D arrays x,y,z
resources.sfXArray          = x   # X axis data points
resources.sfYArray          = y   # Y axis data points
resources.cnFillMode        = 'AreaFill'
resources.tiMainString      = "Polar Circle Problem"

for k in numpy.arange(zshape[0]):
   z = numpy.reshape(zeta[k,:,:], -1)
   contour = Ngl.contour(wks,z,resources)
Ngl.end()
```


Polar Circle Problem
600000
300000
0
-300000
-600000
-600000
-300000
0
300000
600000
17.5
15
12.5
10
7.5
5
2.5
0
-2.5
-5
-7.5
-10
-12.5
-15
-17.5

Mapping in PyNGL

- The curvilinear x,y coordinate system is not as well supported as the lat,lon coordinate system
- Many, many examples with maps on the website