ESCI 386 – Scientific Programming, Analysis and Visualization with Python

Lesson 14 – Reading NetCDF Files

About NetCDF Files

- NetCDF stands for Network Common Data Form, and is a protocol for machine-independent storage of array-oriented data.

- NetCDF is a very common data storage format for meteorological and oceanographic data and output from numerical weather prediction and climate models.

- There are three variations in the NetCDF format:
  - netCDF classic – This was the original format and is still the default for most applications.
  - netCDF 64-bit – Supports larger variables and file sizes.
  - netCDF4 – This is very similar to the HDF5 data format.

NetCDF Variables

- NetCDF files can contain multiple variables at multiple times and levels.

- NetCDF files contain *data variables* and *dimension variables*.
  - *Dimension variables* hold things such as times, altitude or pressure levels, latitudes, and longitudes for the data.
  - *Data variables* hold the actual data such as temperature, pressures, heights, wind components, etc.

NetCDF and Python

- There are several Python modules for reading and writing NetCDF files. We will discuss two of these. They are:
  - `scipy.io.netcdf`: Reads and writes netCDF classic files
  - `netCDF4`: Reads and writes both the older and newer formats

- The functionality of both libraries is similar.
  - `netCDF4` module can handle the reading and writing of the newer version of NetCDF (Version 4)
  - `scipy.io.netcdf` can only handle Version 3 of the NetCDF format.

NetCDF Variables in Python

- When NetCDF files are accessed using either `scipy.io.netcdf` or `netCDF4` the variables are represented as NumPy arrays.

Opening NetCDF Files

Using scipy.io.netcdf

```
from scipy.io import netcdf
f = netcdf.netcdf_file(filename, ‘r’)
```

Using netCDF4

```
import netCDF4
f = netCDF4.Dataset(filename, ‘r’)
```

Opening File Example

- The file ‘hgt.mon.1981-2010.ltm.nc’ contains monthly-averaged geopotential height data for the globe.

```
import netCDF4
f = netCDF4.Dataset('hgt.mon.1981-2010.ltm.nc','r')
```

Or

```
from scipy.io import netcdf
f = netcdf.netcdf_file('hgt.mon.1981-2010.ltm.nc', 'r')
```

Getting a List of Variables

- `f.variables` is a dictionary containing the variables.

- To get a list of the variables we can iterate over `f.variables`, printing the result.

```
>>> for v in f.variables:
      print(v)


level
lat
lon
time
climatology_bounds
hgt
valid_yr_count
```

In netCDF4 Only!

- You can actually just use `print(f)` and it will print out a bunch of metadata, including the dimensions and variable list.

```
>>> print(f)
<type 'netCDF4.Dataset'>
root group (NETCDF3_CLASSIC file format):
    title: monthly mean geopotential height from the
        NCEP Reanalysis
    history: Created 2011/07/12 by doMonthLTM
    description:  Data from NCEP initialized reanalysis
        (4x/day).  These are interpolated to pressure
        surfaces from model (sigma) surfaces.
    platform: Model
    Conventions: COARDS
    references:
        http://www.esrl.noaa.gov/psd/data/gridded/data.n
        cep.reanalysis.derived.html
    not_missing_threshold_percent: minimum 3% values
        input to have non-missing output value
    dimensions = (u'lon', u'lat', u'level', u'time', u'nbnds')
    variables = (u'level', u'lat', u'lon', u'time',
        u'climatology_bounds', u'hgt', u'valid_yr_count')
    groups = ()
```

Accessing Variables

- To access a variable we use the name of the variable as a dictionary key.

```
>>> f.variables['level'][:]
array([ 1000.,   925.,   850.,   700.,   600.,   500.,   400.,   300.,
        250.,   200.,   150.,   100.,    70.,    50.,    30.,    20.,
         10.], dtype=float32)
>>> f.variables['time'][:]
array([   0.,   31.,   59.,   90.,  120.,  151.,  181.,  212.,  243.,
        273.,  304.,  334.])
```

Accessing Variables (cont.)

- To access a variable we use the name of the variable as a dictionary key.

```
>>> f.variables['lat'][:]
array([ 90. ,  87.5,  85. ,  82.5,  80. ,  77.5,  75. ,  72.5,  70. ,
        67.5,  65. ,  62.5,  60. ,  57.5,  55. ,  52.5,  50. ,  47.5,
        45. ,  42.5,  40. ,  37.5,  35. ,  32.5,  30. ,  27.5,  25. ,
        22.5,  20. ,  17.5,  15. ,  12.5,  10. ,   7.5,   5. ,   2.5,
         0. ,  -2.5,  -5. ,  -7.5, -10. , -12.5, -15. , -17.5, -20. ,
       -22.5, -25. , -27.5, -30. , -32.5, -35. , -37.5, -40. , -42.5,
       -45. , -47.5, -50. , -52.5, -55. , -57.5, -60. , -62.5, -65. ,
       -67.5, -70. , -72.5, -75. , -77.5, -80. , -82.5, -85. , -87.5, -90. ],
      dtype=float32)
```

Accessing Variables (cont.)

- To access a variable we use the name of the variable as a dictionary key.

```
>>> f.variables['lon'][:]
array([   0. ,    2.5,    5. ,    7.5,   10. ,   12.5,   15. ,   17.5,
         20. ,   22.5,   25. ,   27.5,   30. ,   32.5,   35. ,   37.5,
         40. ,   42.5,   45. ,   47.5,   50. ,   52.5,   55. ,   57.5,
         60. ,   62.5,   65. ,   67.5,   70. ,   72.5,   75. ,   77.5,
         80. ,   82.5,   85. ,   87.5,   90. ,   92.5,   95. ,   97.5,
        …
        260. ,  262.5,  265. ,  267.5,  270. ,  272.5,  275. ,  277.5,
        280. ,  282.5,  285. ,  287.5,  290. ,  292.5,  295. ,  297.5,
        300. ,  302.5,  305. ,  307.5,  310. ,  312.5,  315. ,  317.5,
        320. ,  322.5,  325. ,  327.5,  330. ,  332.5,  335. ,  337.5,
        340. ,  342.5,  345. ,  347.5,  350. ,  352.5,  355. ,  357.5], dtype=float32)
```

Finding How Variable is Stored

- To find how a particular variable is stored we use the `dimensions` attribute.

- This example shows that ‘hgt’ is stored as a 4-D array with the indices being ‘time’, ‘level’, ‘lat’, and ‘lon’.

```
>>> f.variables['hgt'].dimensions
('time', 'level', 'lat', 'lon')
```

Example Accessing Height Values

- To access 300 mb Height Values for March at 40N over the United States (125.0W to 72.5W):

March 300mb 40N 235.0E 287.5E

```
>>> f.variables['hgt'][2,7,20,94:116]
array([ 9200.35546875,  9198.53613281,  9194.77246094,  9189.17871094,
        9182.75878906,  9176.89746094,  9172.46679688,  9169.28613281,
        9166.17382812,  9162.09667969,  9156.89648438,  9151.26660156,
        9146.08007812,  9141.55175781,  9137.13476562,  9132.21484375,
        9126.70019531,  9121.01367188,  9116.02148438,  9112.37695312,
        9110.38574219,  9110.02636719], dtype=float32)
```

In-class Exercise

- From the file ‘hgt.mon.1981-2010.ltm.nc’ create the plot shown below.