National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Centre for Environmental Data Analysis
SCIENCE AND TECHNOLOGY FACILITIES COUNCIL
NATURAL ENVIRONMENT RESEARCH COUNCIL

Creating NetCDF files with Python

Ag Stephens & Andy Heaps

We are using netCDF4-python!

There are many options for working with NetCDF files in Python. In this example we have chosen to highlight the use of the **netCDF4-python** module.

The **netCDF4-python** module is useful because:

•It implements the basic “classic” model as well as more advanced features.

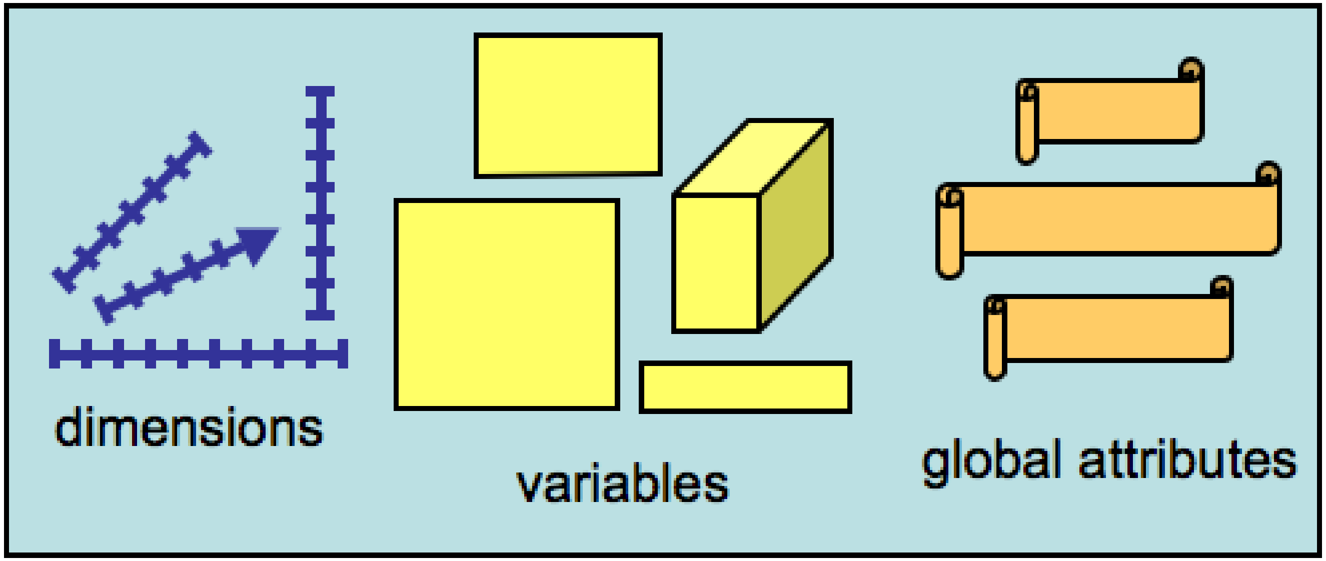
•It provides a simple interface to the NetCDF structure.

•It has been used as the underlying NetCDF I/O layer for many more advanced packages.

Creating a netCDF file

Import Python libraries

Open netCDFfile for writing



Close netCDFfile

Import libraries

```
from netCDF4 import Dataset
import numpy as np
import time as mytime
from numpy.random import uniform
from datetime import datetime, timedelta
from netCDF4 import num2date, date2num
```

We are making a four dimensional dataset based on longitude, latitude, height and time.

We will have a variable called "time" later in the example so we will import the "time" library as "mytime" here to avoid confusion.

Creating/Opening a netCDF file

To create a netCDF file from python, you simply call the **Dataset()** constructor. This is also the method used to open an existing netCDF file.

If the file is open for write access ("w", "r+" or "a"), you may write any type of data including new dimensions, groups, variables and attributes.

This tutorial will focus exclusively on the NetCDF "Classic" data model using: NETCDF4_CLASSIC

Creating a NetCDF file

Open a new NetCDF file ("test.nc") in write ("w") mode:

```
dataset = Dataset('test.nc', 'w', format='NETCDF4_CLASSIC')
```

Creating the dimensions

```
# Create dimensions
longitude = dataset.createDimension('longitude', 144)
latitude = dataset.createDimension('latitude', 73)
level = dataset.createDimension('level', 10)
time = dataset.createDimension('time', None)
```

Variables

NetCDF **variables** behave much like python multi-dimensional arrays in numpy. However, unlike numpy arrays, netCDF4 variables can be appended to along the 'unlimited' dimension (a.k.a. the "record dimension").

To create a netCDF variable, use:

```
  Dataset.createVariable(<var_id>, <type>, <dimensions>)
```

This method has two mandatory arguments: the variable ID (a Python string) and the variable data type. Additionally a tuple of dimensions can be provided.

Creating the variables

```
# Create coordinate variables for 4-dimensions
times = dataset.createVariable('time', np.float64, ('time',))
levels = dataset.createVariable('level', np.int32, ('level',))
latitudes = dataset.createVariable('latitude', np.float32, ('latitude',))
longitudes = dataset.createVariable('longitude', np.float32,
                                    ('longitude',))

# Create the actual 4-d variable
temp = dataset.createVariable('temp', np.float32,
                              ('time','level','latitude','longitude'))
```

Accessing dimensions and variables

All of the dimensions and variables in the Dataset are stored in Python dictionaries:

```
>>> print 'temp variable:', dataset.variables['temp']

>>> for varname in dataset.variables.keys():
...         var = dataset.variables[varname]
...         print varname, var.dtype, var.dimensions, var.shape
```

```
temp variable: <type 'netCDF4.Variable'> float32 temp(time, level, latitude,
longitude) unlimited dimensions: time
current shape = (0, 10, 73, 144)
time float64 (u'time',) (0,)
level int32 (u'level',) (10,)
latitude float32 (u'latitude',) (73,)
longitude float32 (u'longitude',) (144,)
temp float32 (u'time', u'level', u'latitude', u'longitude') (0, 10, 73, 144)
```

Attributes (global)

Global attributes are set by assigning values to Dataset instance variables. Attributes can be strings, numbers or sequences. Returning to our example:

```
# Global Attributes
dataset.description = 'bogus example script'
dataset.history = 'Created ' + mytime.ctime(mytime.time())
dataset.source = 'netCDF4 python module tutorial'
```

Variable attributes

Variable attributes are set by assigning values to Variable instance variables:

```
#Assign variable attributes
longitudes.standard_name='longitude'
longitudes.units = 'degree_east'

latitudes.standard_name='latitude'
latitudes.units = 'degree_north'

levels.standard_name='pressure'
levels.units = 'hPa'

times.standard_name='time'
times.units = 'hours since 0001-01-01 00:00:00'
times.calendar = 'gregorian'

temp.standard_name='air_temperature'
temp.units = 'K'
```

Accessing attributes

Attributes are accessed as attributes of their relevant instances:

```
>>> print dataset.description
bogus example script

>>> print dataset.history
Created Mon Mar 17 01:12:31 2014
```

Writing data

Now that you have a netCDF Variable instance, how do you put data into it? You can just treat it like an array and assign data to a slice.

```
# Writing data
longitudes[:] = np.arange(-180,180,2.5)
latitudes[:] = np.arange(-90,91,2.5)
levels[:] = [1000, 900, 700, 500, 300, 100, 70, 50, 30, 10]
```

Growing data along the unlimited dimension

Unlike NumPy's array objects, netCDF Variable objects that have an unlimited dimension will grow along that dimension if you assign data outside the currently defined range of indices.

```
temp[0:5,:,:,:] = uniform(size=(5,10,73,144))
```

NOTE: uniform is **numpy.random.uniform**(size = X) returns values from a uniform distribution in a numpy array with dimensions expressed in a tuple X.

Defining date/times correctly (1)

Time coordinate values pose a special challenge to netCDF users. Most metadata standards (such as CF and COARDS) specify that time should be measure relative to a fixed date using a certain calendar, with units specified like:

"hours since YY:MM:DD hh-mm-ss"

These units can be awkward to deal with, without a utility to convert the values to and from calendar dates. The functions **num2date()** and **date2num()** are provided with this package to do just that. Here's an example of how they can be used…

Defining date/times correctly (2)

```
# Fill in times
dates = []
for n in range(5):
    dates.append(datetime(2001, 3, 1) + n * timedelta(hours=12))

times[:] = date2num(dates, units = times.units, calendar = times.calendar)
```

num2date() converts numeric values of time in the specified units and calendar to datetime objects, and **date2num()** does the reverse.

Accessing the date/times

```
>>> print 'time values (in units): ', times.units
>>> print times[:]

time values (in units): hours since 0001-01-01 00:00:00.0
[ 17533104. 17533116. 17533128. 17533140. 17533152.]



>>> print 'dates corresponding to time values:'
>>> print num2date(times[:], units=times.units, calendar=times.calendar)

dates corresponding to time values:
 [datetime.datetime(2001, 3, 1, 0, 0)
  datetime.datetime(2001, 3, 1, 12, 0)
  datetime.datetime(2001, 3, 2, 0, 0)
  datetime.datetime(2001, 3, 2, 12, 0)
  datetime.datetime(2001, 3, 3, 0, 0)]
```

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Centre for Environmental Data Analysis
SCIENCE AND TECHNOLOGY FACILITIES COUNCIL
NATURAL ENVIRONMENT RESEARCH COUNCIL

Finally, let’s write the file

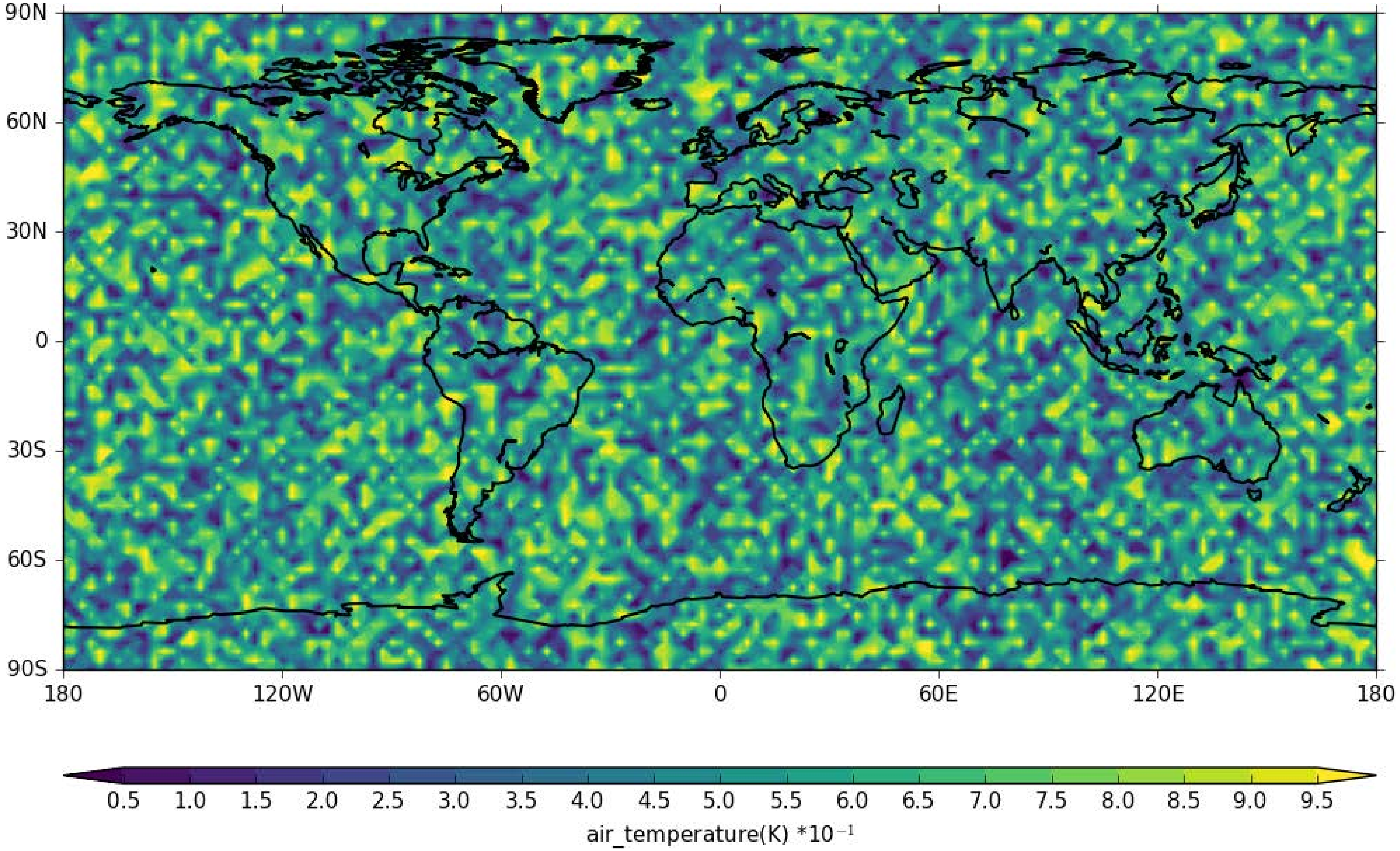
Simply…

```
dataset.close()
```

What does the output look like? $ ncdump -h test.nc

```
netcdf test {
dimensions:
longitude = 144 ;
latitude = 73 ;
level = 10 ;
time = UNLIMITED ; // (5 currently)
variables:
double time(time) ;
                time:standard_name = "time" ;
                time:units = "hours since 0001-01-01 00:00:00" ;
                time:calendar = "gregorian" ;
int level(level) ;
                level:standard_name = "pressure" ;
                level:units = "hPa" ;
float latitude(latitude) ;
                latitude:standard_name = "latitude" ;
                latitude:units = "degree_north" ;
float longitude(longitude) ;
                longitude:standard_name = "longitude" ;
                longitude:units = "degree_east" ;
float temp(time, level, latitude, longitude) ;
                temp:standard_name = "air_temperature" ;
                temp:units = "K" ;

// global attributes:
                :description = "bogus example script" ;
                :history = "Created Wed May  3 16:45:59 2017" ;
                :source = "netCDF4 python module tutorial" ;
}
```



National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Centre for Environmental Data Analysis
SCIENCE AND TECHNOLOGY FACILITIES COUNCIL
NATURAL ENVIRONMENT RESEARCH COUNCIL

Further reading

Python-netCDF4:

http://unidata.github.io/netcdf4-python