

Read and Write Data

Reading NetCDF files with Python

Thanks to all contributors:

Stephen Pascoe, Jeff Whittaker.

So many options!

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.

Opening a netCDF file

To open a netCDF file from python, you simply call the Dataset() constructor as follows:

```
>>> from netCDF4 import Dataset
>>> dataset = Dataset('data.nc')

>>> print(dataset.file_format)
NETCDF4_CLASSIC
```

Working with "classic" NetCDF

The netCDF4 module can read in any netCDF format. This tutorial will focus exclusively on the NetCDF-"classic" data model using: NETCDF4_CLASSIC

The "classic" data model is made up of dimensions, variables and attributes (as discussed earlier).

Interrogating dimensions

You can interrogate dimensions using simple dictionary calls:

```
>>> print(list(dataset.dimensions.keys()))  
['air_pressure', 'latitude_centers', 'time']  
  
>>> print(dataset.dimensions['time'])  
<class 'netCDF4._netCDF4.Dimension': name = 'time',  
size = 12
```

Interrogating variables

You can interrogate variables using simple dictionary calls:

```
>>> print(list(dataset.variables.keys()))  
['tcc', 'time', 'latitude', 'longitude']  
  
>>> tcc_var = dataset.variables['tcc']  
>>> print(tcc_var)  
<class 'netCDF4._netCDF4.Variable'>  
float32 tcc(time, latitude, longitude  
    missing_value: 9.999e+20  
    name: tcc  
    title: Total cloud cover  
unlimited dimensions: time  
current shape = (1, 181, 360)  
filling off
```

Global attributes

Global attributes are available as attributes of the python dataset instance:

```
# Get conventions attribute  
>>> print(dataset.Conventions)  
CF-1.5
```

```
# Or find all NetCDF global attributes  
>>> for attr in dataset.ncattrs():  
...     print(attr, '=', getattr(dataset, attr))  
...  
Conventions = CF-1.5  
history = Written in a hurry on a Tuesday!
```

Variable attributes

Variable attributes are available as attributes of the python variable instance:

```
# Get units attribute
>>> print(tcc_var.units)
0-1
```

```
# Or find all variable attributes
>>> for attr in tcc_var.ncattrs():
...     print(attr, '=', getattr(tcc_var, attr))
...
long_name = Total cloud cover
units = 0-1
```


Accessing the data

Variables contain data, which you can access:

```
# Get a whole variable array
```

```
>>> arr = dataset.variables['tcc'][:]
```

```
>>> arr.shape
```

```
(1, 181, 360)
```

```
>>> type(arr)
```

```
<class 'numpy.ndarray'>
```

This is awesome!

It returns a
NumPy array

```
# Get a sub-slice of the array
```

```
>>> subset = dataset.variables['tcc'][0, 10:15, 5]
```

```
>>> subset.shape
```

```
(5,)
```

For NetCDF variable has missing data - a Masked Array!

Further reading

netCDF4-python website:

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

netCDF4-python code:

<https://github.com/Unidata/netcdf4-python>