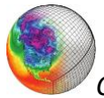
Community Data Management System



CDMS Cheat Sheet

Files (f)

f = cdms.open('myfile')	Open file "myfile"

Querying

f.showglobal()	Print file's global attributes
attdic=f.attributes	Return file attribute's dictionary
attlist=f. attributes.keys()	Return file attribute's list
nm=f.name	Returns attribute value (i.e. 'name')
f.key='value'	Set an attribute to a new value
dims=f.listdimension()	Returns file's dimensions list
dims=f.listdimension('myvar')	Returns file's dimensions list

Variable

f.showvariable()	Display all variables in the file
vardic=f.variables	Returns variable dictionary
varlist=f.variables.keys()	Returns variable list

File

dims=f.listdimension('myvar')	Returns dimensions list for 'myvar'
f.showdimension('myvar'	Same as above
f.showattribute('myvar')	Prints the attributes of 'myvar'
f.showall("myvar')	Prints information of 'myvar'

Transient Variable (TV)

Retrieving

```
tv=f('myvar')   # gets 'myvar' from cchns file f
```

tv now is a Transient Variable containing the variable 'myvar' from the file' 'myfile'
Let's say 'myvar' is 3D: time/latitude/longitude, if we wish to retrieve the longitude rom -180 to 180: =f('myvar' ,longitude=(-180 ,180))

Note: *By default the retrieval edges are closed/open, i.e the second value is not included in the retrieval procedure, if we wish to retrieve 180 then we would pass: s=f('myvar' ,longitude=(-180,180 ,'cc')) i.e.: closed/closed*

Querying

attdic=s.attributes	Returns a dictionary
attlist=s.listattribute()	Return a list of the variable attribute
dmnames=s.getAxisld()	Returns al isl of the dimensions names
sh=s. shape	Returns a tuple with the length of each dim.
s.info()	Display description of the slab: attr. & dim.

Dimension=Axis

Retrieving

ax=s.getAxis(0)	Returns axis dimension at index (0)
itim=s.getAxisIndex('time')	Returns axis dimension index (0, 1, 2, …)
axlist=s.getAxisList()	Returns axis list
tim=s.getTime ()	Returns axis time
lev=s.getLevel()	Returns axis level
lat=s.getLatitude()	Returns axis latitude
lon=s.getLongitude()	Returns axis longitude

Querying

id=ax.id	Returns axis name
val=ax[:]	Returns axis values' list
att=ax.attributes	Returns attribute's dictionary
bounds=ax.getBounds()	Returns boundary list of each coordinate
ax.isTime()	Returns 1 if the axis is **time**, 0 otherwise
ax.isLevel()	Returns 1 if the axis is **level**, 0 otherwise
ax.isLatitude()	Returns 1 if the axis is **latitude**, 0 otherwise
ax.isLongitude()	Returns 1 if the axis is **longitude**, 0 if not
ax.isCircular()	Returns 1 if the axis is **circular**
ax.modulo	Returns the value of the modulo (for circular axis)

Altering

ax[:]=newvalues	Change axis values (*Ax*):
ax.id='my new name'	Changes axis name
ax.units='my new units'	Changes axis units
ax.myspecialattribute='my special attribute value'	Set axis attribute
ax.designateTime()	Sets axis designation time
ax.designateLevel()	Sets axis designation levels
ax.designateLatitude()	Sets axis designation latitude
ax.designateLongitude()	Sets axis designation longitude
ax.designateCircular(value)	Sets axis designation as circular: modulo "value"

Writing data to a file

Example:
Write a Transient Variable (TV) having the dimensions: time, latitude, longitude to a file. The data shape is 12,64,128 and the grid is gaussian T42 (from -180, to 180 and south to north), and represent the 12 months of year 2000.

1-Preparing the dimensions/axis

time=cdms2.createAxis(range(12))	Create the "raw" axis
time.id='time'	Set the name
time. units='months since 2000'	
time.designateTime()	Specify the axis as "time" (independently from the name)
lons=MV2.arange(128, typecode=MV2.float)*2.8125-180.	
lon=cdms2.createAxis(lons)	
lon.id='longitude'	
lon. units='degrees_east'	
lon. designateLongitude()	
lat=cdms2.createGaussianAxis(64)	Creates a gaussian axis with 64 latitudes
lat.units='degrees_ north'	

2- Setting the axes into the TV (tv)

```
tv.setAxis(0,time)
tv.setAxis(1,lat)
tv.setAxis(2,lon)
```

3- Writing to the file

f=cdms.open('myfile.nc','w')	To create a new dataset
f=cdms.open('myfile.nc','r+')	To open an existing file

```
f.write(tv)
# or if we're not sure that we set the axes on the tv: f.write(tv,axe=(tim,lat,lon))
f.close()
```

4- Unlimited dimension

```
By default, time is set as unlimited and can be extended:
f=cdms2.open('myfile.nc','r+')
for i in range(len(time)):
   t=tv[i]
   f.write(t)
f.close()
Writes all the time one after the other.
```

***5*- Full form of the write function**

fv = write(var, attributes=None, axes=None, extbounds=None, id=None, extend=None, fill_ value=None, index=None)

- var: is a variable or array.
- attributes: is a dictionary of attributes associated to the variable.
- axes: is the list of file axes including the domain of the variable.
- extbounds: is the extended dimension bounds. Default: var.getAxis(0).getBounds(0)
- id: is the variable name in the file. Default is var.id.
- extend: causes the first dimension to be 'extensible': iteratively writeable. The default is None, in which case the first dimension is extensible if it is time.
- fill_value: is the missing value flag. Index is the extended dimension index to write to. The default index is determined by lookup relative to the existing extended dimension.