OPeNDAP subsetting with python

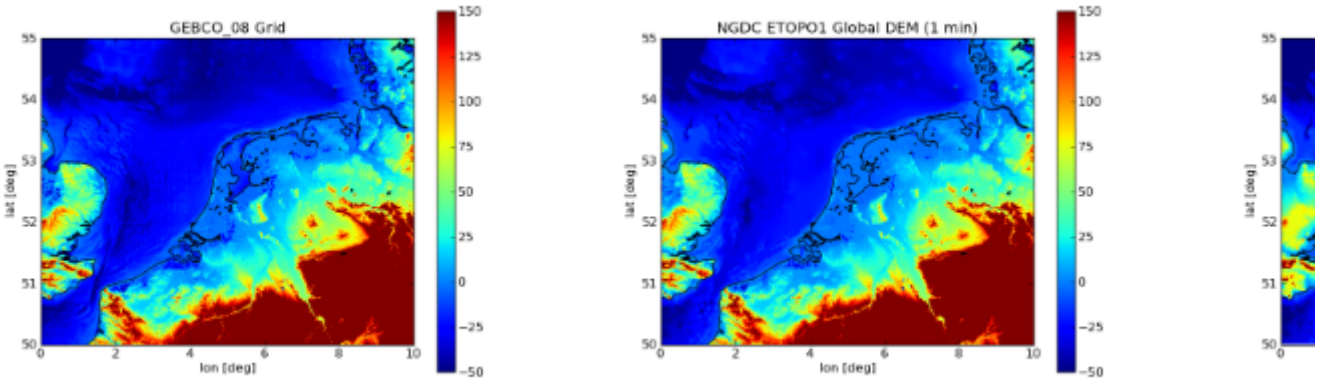
In this example we will show how to benefit from OPeNDAP subsetting. We will request subsets from worldwide digital elevevation maps (DEMs) that are too big to request as a whole: resulting in approx. 500 MB (over 10,000 x 20,000 points) or 2 GB (over 20,000 x 40,000) for respectively a 1 minute or 30 second grids.

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

#OPENDAP_SUBSETTING_WITH_PYTHON_TUTORIAL how to benefit from OPeNDAP subsetting in python
#
# This tutorial is also available for Matlab
#
#See also: OPeNDAP_access_with_Matlab_tutorial.py

# $Id:  $
# $Date:  $
# $Author:  $
# $Revision:  $
# $HeadURL: https://repos.deltares.nl/repos/OpenEarthTools/trunk/python/io/opendap
/OPeNDAP_subsetting_with_python_tutorial.py $
# $Keywords: $

# This document is also posted on a wiki: http://public.deltares.nl/display/OET/OPeNDAP+access+with+python

# Read data from an opendap server
import urllib
import numpy as np
import netCDF4
import pydap
import matplotlib
import matplotlib.pyplot as plt
import pylab

# from opendap import opendap # OpenEarthTools module, see above that makes pypdap quack like netCDF4

# Define data on an opendap server
# for converting non-open access GECBO grids to same netCDF structure: see https://repos.deltares.nl/repos
/OpenEarthRawData/trunk/gebco/
gridurls =  ['http://geoport.whoi.edu/thredds/dodsC/bathy/etopo1_bed_g2',
             'http://geoport.whoi.edu/thredds/dodsC/bathy/etopo2_v2c.nc',
             'http://geoport.whoi.edu/thredds/dodsC/bathy/srtm30plus_v1.nc',
             'http://geoport.whoi.edu/thredds/dodsC/bathy/srtm30plus_v6',
             'http://geoport.whoi.edu/thredds/dodsC/bathy/smith_sandwell_v9.1.nc',
             'http://geoport.whoi.edu/thredds/dodsC/bathy/smith_sandwell_v11',
             r'F:\checkouts\OpenEarthRawData\gebco\raw\gebco_30sec.nc',
             r'F:\checkouts\OpenEarthRawData\gebco\raw\gebco_1min.nc']

lineurl  = 'http://opendap.deltares.nl/thredds/dodsC/opendap/noaa/gshhs/gshhs_i.nc';

# Get line data: 1D vectors are small, so we can get all data
linedata = netCDF4.Dataset(lineurl) # opendap(url_line) # when netCDF4 was not compiled with OPeNDAP

lines = dict(
   lon=linedata.variables['lon'][:],
   lat=linedata.variables['lat'][:]
   )
linedata.close()

# Define bounding box
BB = dict(
   lon=[ 0, 10],
   lat=[50, 55]
   )

for i,ncfile in enumerate(gridurls):

   print ncfile

   # Get grid data
```

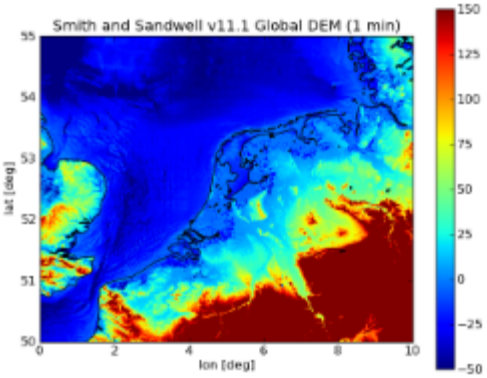
```
grid   = netCDF4.Dataset(ncfile) # opendap(ncfile) # when netCDF4 was not compiled with OPeNDAP

# Get all data from lat and lon, but just a pointer to the topography
# Because getting it all takes a bit too long
lat = grid.variables['lat'][:]
lon = grid.variables['lon'][:]

# nonzero returns a tuple of idx per dimension
# we're unpacking the tuple here so we can lookup max and min
(latidx,) = np.logical_and(lat >= BB['lat'][0], lat < BB['lat'][1]).nonzero()
(lonidx,) = np.logical_and(lon >= BB['lon'][0], lon < BB['lon'][1]).nonzero()

#
assert lat.shape[0] == grid.variables['topo'].shape[0], 'We assume first dim is lat here'
assert lon.shape[0] == grid.variables['topo'].shape[1], 'We assume 2nd dim is lon here'

# get rid of the non used lat/lon now
lat = lat[latidx]
lon = lon[lonidx]
# Get the topography data with the new indices
topo = grid.variables['topo'][latidx.min():latidx.max(),
                              lonidx.min():lonidx.max()]
title = grid.title # remember so we can close the file before plotting
grid.close()

Lon,Lat = np.meshgrid(lon, lat)
# make a pcolormesh (the fastest way to plot simple grids)
mesh = plt.pcolormesh(Lon,Lat,topo)
plt.plot(lines['lon'],lines['lat'],'k')
# some customizations (called on the axes and figure)
mesh.axes.set_title(title)
mesh.axes.set_aspect(1/np.cos(np.pi*np.mean(BB['lat'])/180.)) # set aspect to match meters
plt.xlim(BB['lon'][0],BB['lon'][1])
plt.ylim(BB['lat'][0],BB['lat'][1])
mesh.axes.set_xlabel('lon [deg]')
mesh.axes.set_ylabel('lat [deg]')
plt.clim(-50,150)
mesh.figure.colorbar(mesh) # use the mesh to make a colorbar
# Save the figure
mesh.figure.savefig('%s.png' % title)
mesh.figure.clf()
```

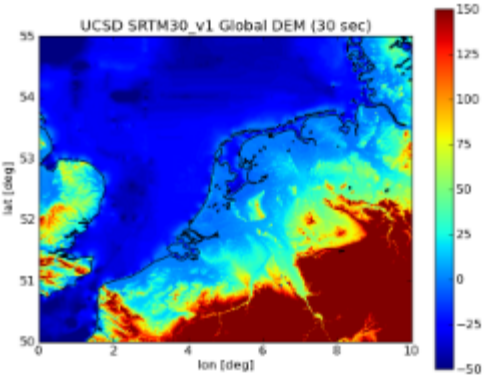
Download the code of this python example (repos,manual download)

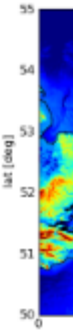
See also: OPeNDAP access with **python**, OPeNDAP subsetting with **Matlab**,OPeNDAP subsetting with **R**, PostgreSQL access with **python**



non-open GEBCO data for comparison [The GEBCO_08 Grid version 20091120, http://www.gebco.net]


Smith and Sandwell v11.1 Global DEM (1 min)
lat [deg]
lon [deg]
55
54
53
52
51
50
0
2
4
6
8
10
150
125
100
75
50
25
0
−25
−50


UCSD SRTM30_v1 Global DEM (30 sec)
lat [deg]
lon [deg]
55
54
53
52
51
50
0
2
4
6
8
10
150
125
100
75
50
25
0
−25
−50


lat [deg]
55
54
53
52
51
50
0