Lectures 4

Python Basics

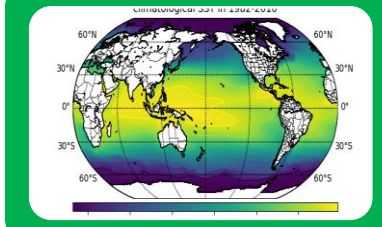
- Background
- Installation & setup
- Python Language

Python Advanced

- I/O & Exceptions Handling
- Modules & Packages
- Object-Oriented Programming in Python

Python for Scientific Computation

- Array computation with Numpy
- Common scientific computation with Scipy
- Draw common 2D figures with Matplotlib



Python for Oceanography

- Read/Write netCDF files
- Draw data on maps with basemap

Specific packages for geography

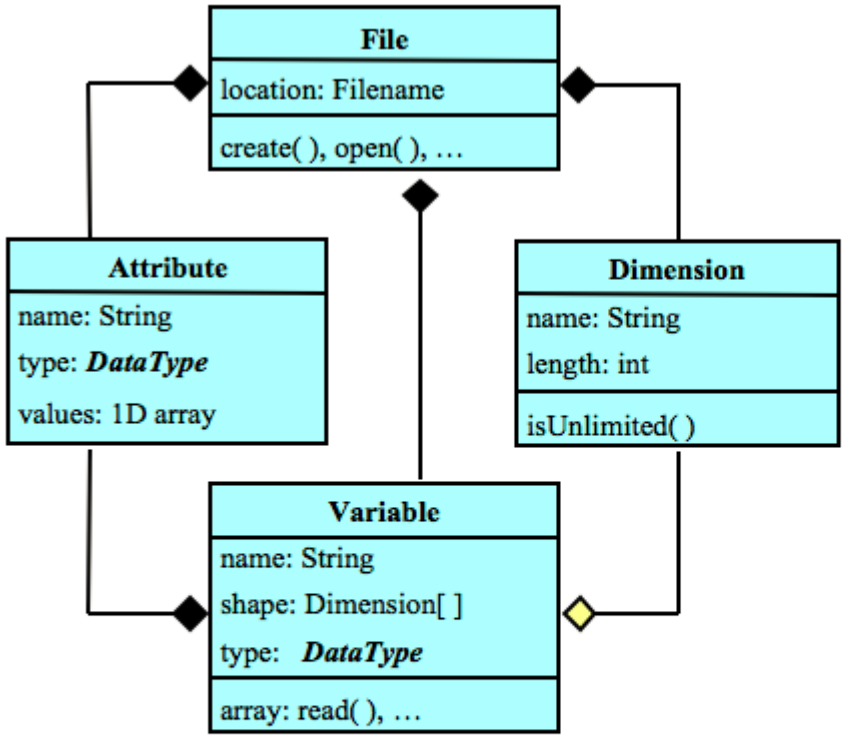
- netCDF4
- basemap
- cartopy

netCDF4 format introduction

- A format that is widely used in geoscience education and research community, developed by Unidata.
  - Array-oriented
  - Self-describing
  - Machine-independent

- netCDF Data Model: the logical structure of data
  - Classic Data Model
    - Dimensions, variables and attributes
  - Enhanced Data Model (Common Data Model)
    - Completely backward-compatible to the classic data model
    - Also supports groups, multiple unlimited dimensions and user-defined types
    - The new features are only used with netCDF4/HDF5
  - For maximum interoperability with existing code, new data should be created with the The Classic Model.

netCDF Classic Data Model



A file has named variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One dimension may be of unlimited length.

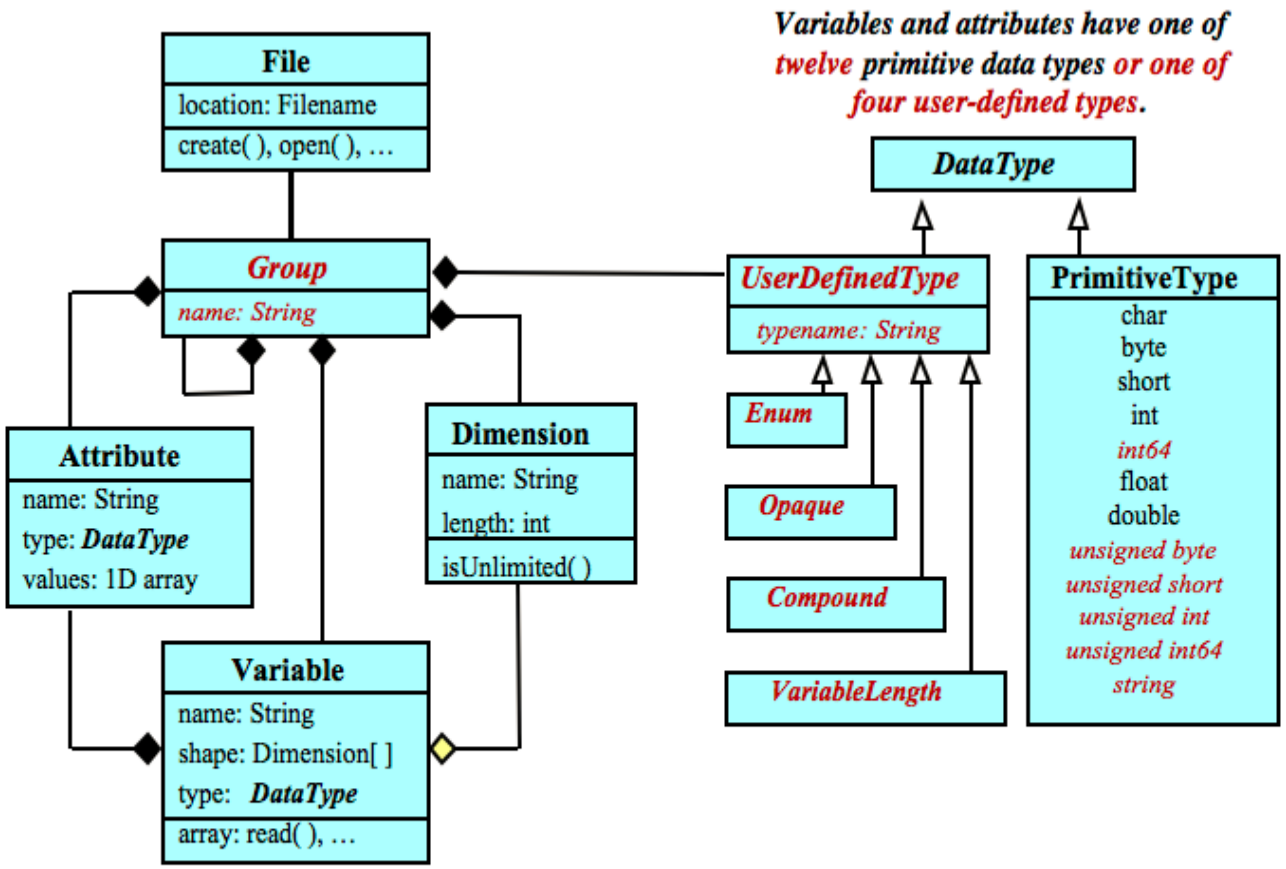
source: [NetCDF Documentation]

Variables and attributes have one of six primitive data types.

DataType
char
byte
short
int
float
double

```
In [366]: import netCDF4 as nc

In [367]: dataset = nc.Dataset("OISST/OISST_1982-2010.nc", "r")

In [368]: dataset
Out[368]:
<class 'netCDF4._netCDF4.Dataset'>
root group (NETCDF3_CLASSIC data model, file format NETCDF3):
    dimensions(sizes): X(360), T(348), Y(181)
    variables(dimensions): float32 X(X), float32 I(T), float32 Y(Y), float32 sst(T,Y,X)
    groups:

In [369]:
```

netCDF Enhanced Data Model



A file has a top-level unnamed group. Each group may contain one or more named subgroups, user-defined types, variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One or more dimensions may be of unlimited length.

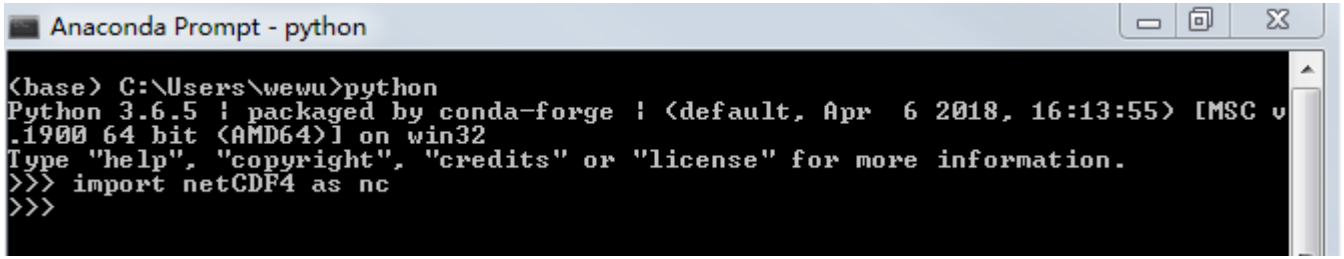
source: NetCDF Documentation

```
In [371]: dataset_nc4 = nc.Dataset("OISST/OISST_avg_1982_2019.nc", "r")

In [372]: dataset_nc4
Out[372]:
<class 'netCDF4._netCDF4.Dataset'>
root group (NETCDF4 data model, file format HDF5):
    dimensions(sizes):
    variables(dimensions):
    groups: processed_data

In [373]:
```

Install netCDF4 package with conda

- Installation

- Verification



Reading netCDF files with netCDF4

- Import the package
- Open a netCDF file
- Query global attributes
- Query dimensions
- Query variables
- Close the file

```
In [384]: import netCDF4 as nc

In [385]: root_group = nc.Dataset("OISST/OISST_1982-2010.nc", "r")

In [386]: root_group.ncattrs()
Out[386]: []

In [387]: root_group.dimensions
Out[387]:
OrderedDict([('X',
              <class 'netCDF4._netCDF4.Dimension'>: name = 'X', size = 360),
             ('T',
              <class 'netCDF4._netCDF4.Dimension'>: name = 'T', size = 348),
             ('Y',
              <class 'netCDF4._netCDF4.Dimension'>: name = 'Y', size = 181)])

In [388]: root_group.variables
Out[388]:
OrderedDict([('X', <class 'netCDF4._netCDF4.Variable'>
              float32 X(X)
                  standard_name: longitude
                  pointwidth: 1.0
                  gridtype: 1
                  units: degree_east
              unlimited dimensions:
              current shape = (360,)
              filling on, default _FillValue of 9.969209968386869e+36 used),
             ('T', <class 'netCDF4._netCDF4.Variable'>
              float32 T(T)
                  standard_name: time
                  pointwidth: 1.0
                  calendar: 360
                  gridtype: 0
                  units: months since 1960-01-01
```

Writing netCDF files with netCDF4

- Import the package
- Open a netCDF file
- Create global attributes
- Create dimensions
- Create variables
- Close the file

```
# create a netCDF file with classic data model
nc_file = 'OISST\sample_nc3_file.nc'
root_group = nc.Dataset(nc_file, "w", format='NETCDF3_CLASSIC')

# create global attributes
root_group.setncattr(name="description", value="this is just a netCDF sample file.")
root_group.version = "1.0"
root_group.setncattr("created time", str(date.today()))

# create dimensions for variables
lat = root_group.createDimension('latitude', size=181)
lon = root_group.createDimension('longitude', size=360)
time = root_group.createDimension('time', size=None)

# create variables
latitudes = root_group.createVariable('latitude', 'f4', ('latitude'))
longitudes = root_group.createVariable('longitude', 'f4', ('longitude'))
times = root_group.createVariable('time', 'i4', ('time'))
temperatures = root_group.createVariable('temperature', 'f8', ('time', 'latitude', 'longitude'))
# create variable attributes
temperatures.units = "Celsius Degree"

# write data into the created variables
latitudes[:] = np.linspace(-90, 90, 181)
longitudes[:] = np.linspace(0, 359, 360)
times[:] = np.array([1])
temperatures[0] = np.random.random((latitudes.size, longitudes.size)) * 25

# close the nc file
root_group.close()
```

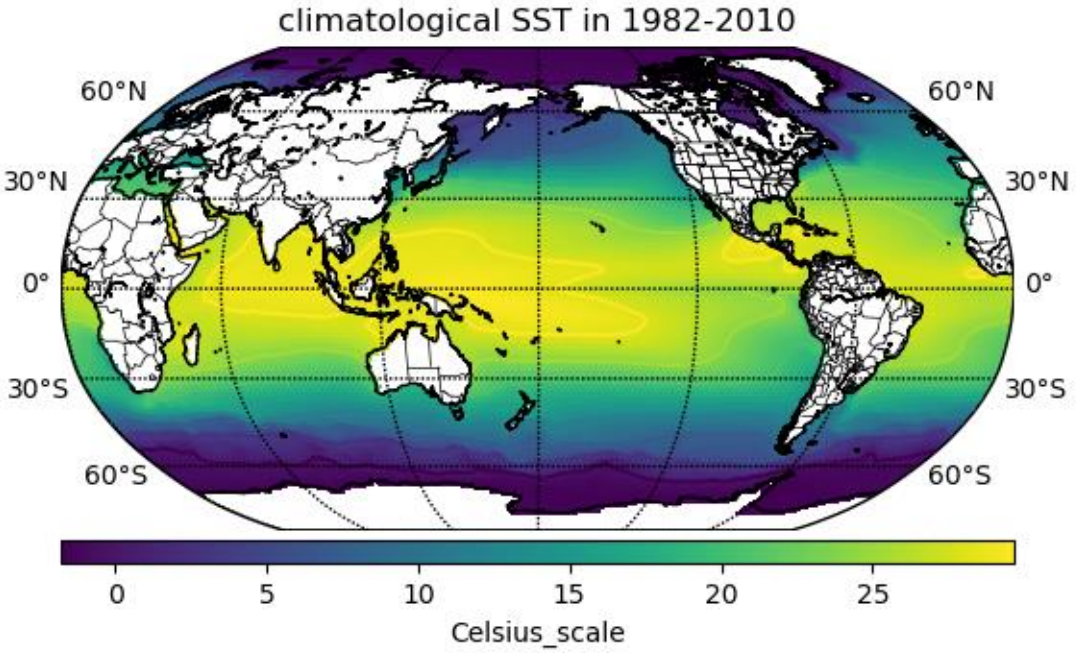
netCDF4: References & Exercises

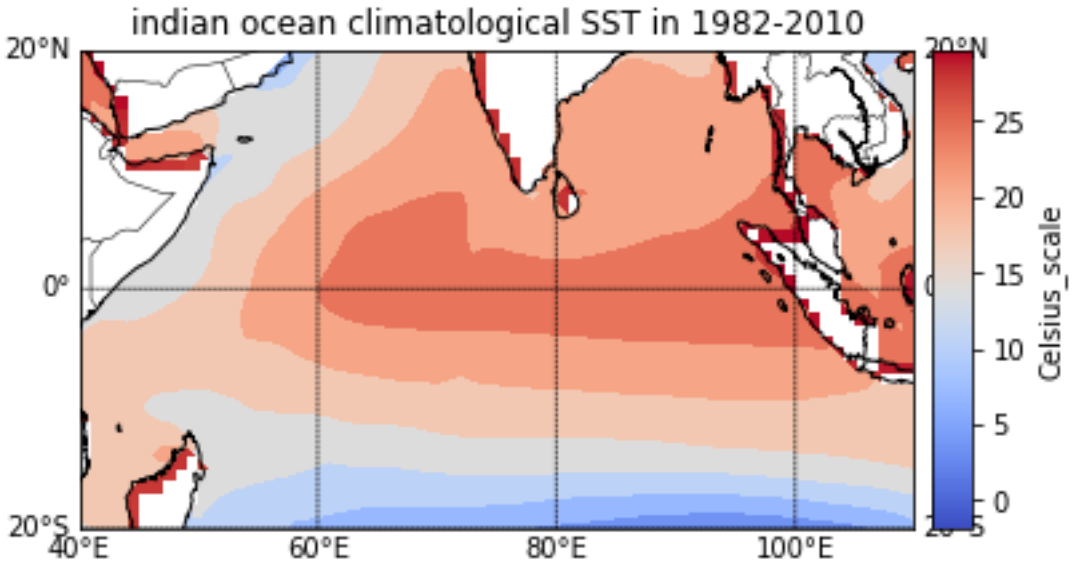
- Ref:
  - http://unidata.github.io/netcdf4-python/
  - http://joehamman.com/2013/10/12/plotting-netCDF-data-with-Python/
  - https://github.com/Unidata/netcdf4-python
- Exercise
  - Write a Python version ncdump (–h) to show the metadata of any netCDF4 file

```
localhost:ERSST may$ ncdump -h sst.mnmean.nc
netcdf sst.mnmean {
dimensions:
        lon = 180 ;
        lat = 89 ;
        nbnds = 2 ;
        time = UNLIMITED ; // (1964 currently)
variables:
        float lat(lat) ;
                lat:units = "degrees_north" ;
                lat:long_name = "Latitude" ;
                lat:actual_range = 88.f, -88.f ;
                lat:standard_name = "latitude" ;
                lat:axis = "Y" ;
                lat:coordinate_defines = "center" ;
        float lon(lon) ;
                lon:units = "degrees_east" ;
```

data visualization with basemap

- basemap: a matplotlib extension to plot data on maps
  - for earth scientists, particularly oceanographers and meteorologists
  - to retire in 2020 and be replaced by Cartopy

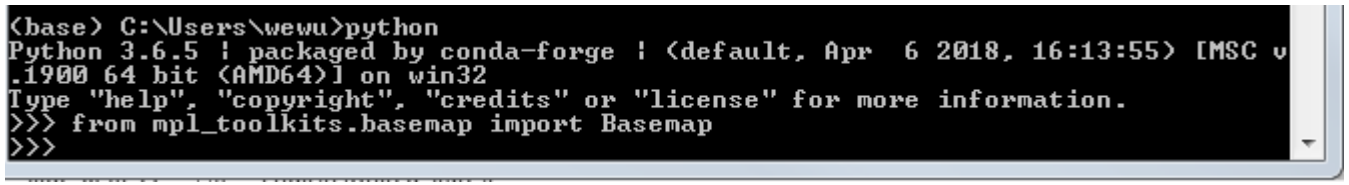




Install basemap package with conda

- Installation

- Verification



Draw figures with basemap

- Create a Basemap instance
  - Choose a projection
  - Setup the drawing region
- Plot data on the map
- Draw coast lines, continents, etc.
- Draw some markers on the map
- Show the figure

```
# draw another figure for Indian Ocean region
lower_left_corner_lon = 40
lower_left_corner_lat = -20
upper_right_corner_lon = 110
upper_right_corner_lat = 20

# Create basemap instance
base_map = Basemap(projection='cyl', resolution='l',
                   llcrnrlon=lower_left_corner_lon, llcrnrlat=lower_left_corner_lat,
                   urcrnrlon=upper_right_corner_lon, urcrnrlat=upper_right_corner_lat)

# Plot data on the map
cmp = plt.cm.coolwarm   # choose a beautiful color map
cor = base_map.pcolormesh(lons, lats, sst_avg, latlon=True, cmap=cmp)
ctr = base_map.contourf(lons, lats, sst_avg, latlon=True, cmap=cmp)

# Add colorbar
cbar = base_map.colorbar(cor, location='right')
cbar.set_label(oisst.variables['sst'].units)

# Add grid lines
base_map.drawparallels(np.arange(lower_left_corner_lat, upper_right_corner_lon, 20.),
                       labels = [1, 1, 0, 0], fontsize=10)
base_map.drawmeridians(np.arange(lower_left_corner_lon, upper_right_corner_lon, 20.),
                       labels = [0, 0, 0, 1], fontsize=10)

# Add coastlines, states, and country boundaries
base_map.drawcoastlines()
base_map.drawstates()
base_map.drawcountries()

# Add title
plt.title('indian ocean climatological SST in 1982-2010')
plt.show()
```

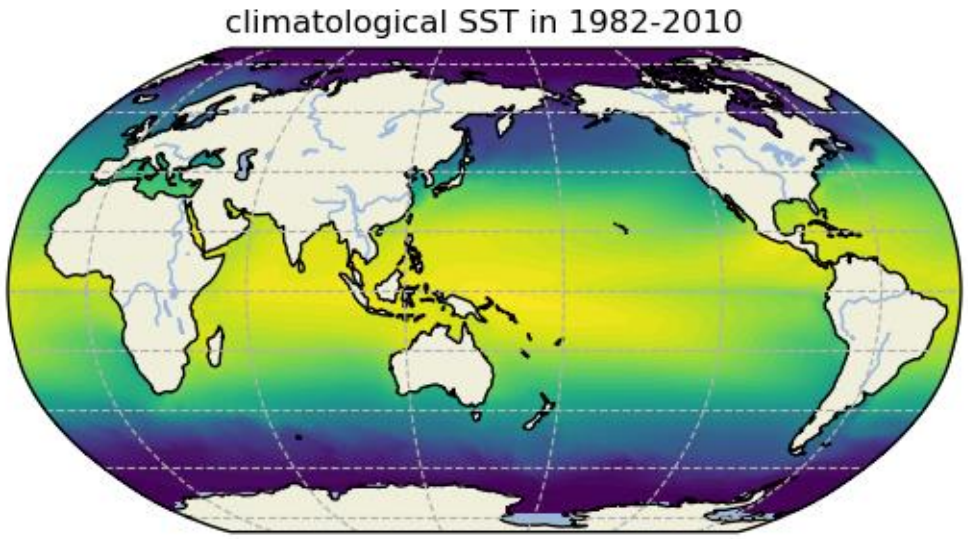
basemap: References & Exercises

- Ref:
  - http://basemaptutorial.readthedocs.io/en/latest/index.html
  - https://matplotlib.org/basemap/index.html
  - https://matplotlib.org/basemap/users/examples.html
  - https://matplotlib.org/cmocean/

- Exercise
  - Use basemap to draw some oceanographic figures

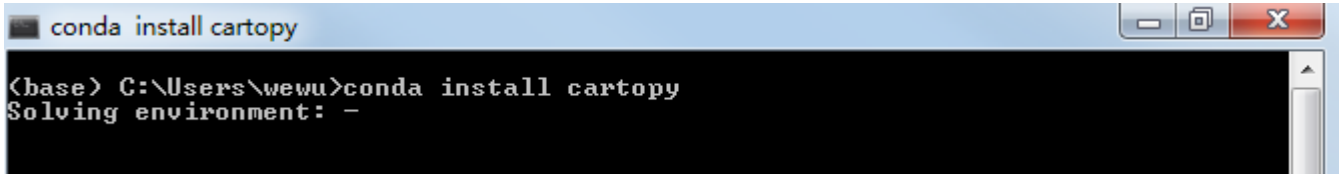
data visualization with cartopy

- cartopy:
  - a Python package designed for geospatial data processing in order to produce maps and other geospatial data analyses
  - basemap's successor and exceeder

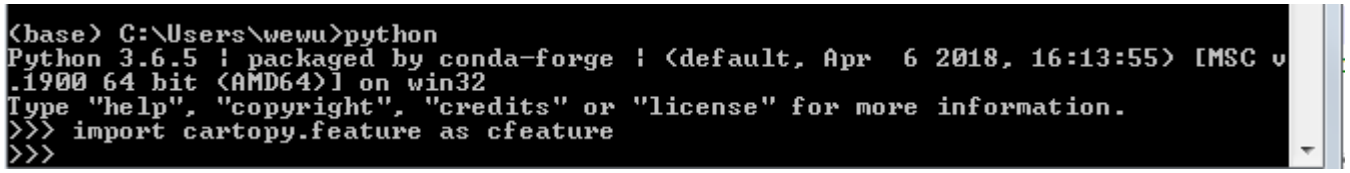


Install cartopy package with conda

- Installation



- Verification



Reference

- https://scitools.org.uk/cartopy/docs/latest/index.html
- https://scitools.org.uk/cartopy/docs/latest/cartopy_outline.html
- https://github.com/SciTools/cartopy

Machine learning frameworks for data science

- TensorFlow
- PyTorch

Lecture 8:
Hardware and Software

source: Fei-Fei Li's CNN for Visual Recognition

Reference

- https://github.com/josephmisiti/awesome-machine-learning
- Feifei Li's course lecture: Deep Learning Hardware and Software
- http://tensorfly.cn/tfdoc/get_started/introduction.html