Python and ecCodes

Xavi Abellan

Xavier.Abellan@ecmwf.int

ECMWF

Python and ecCodes

- Just an appetizer
- Provide you only a small view of the world the Python interface opens to
- Increase your awareness
- You need to explore!

SciPy Software stack

- Python-based ecosystem of open-source software for mathematics, science, and engineering
- It depends on other python packages like:
  - **Numpy**: Base N-dimensional array package
  - **SciPy library** : Fundamental library for scientific computing
  - **Matplotlib**: Comprehensive 2D Plotting
  - **Ipython / Jupyter**: Enhanced Interactive Console, notebooks
  - **Sympy**: Symbolic mathematics
  - **Pandas**: Data structures & analysis

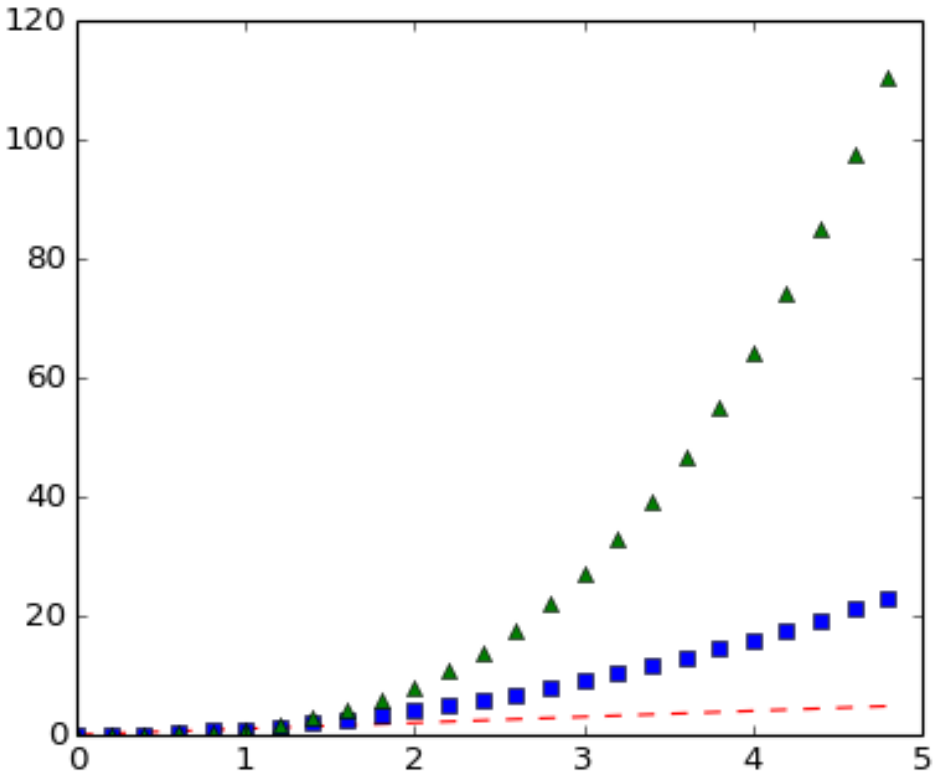
NumPy

- Fundamental Python package for scientific computing
- Provides support for multidimensional arrays
- Good assortment of routines for fast operations on arrays
- Performance comparable to that of C or Fortran
- A growing number of Python-based mathematical and scientific packages are using NumPy
- At its core is the ndarray object, an n-dimensional array of homogenous data

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.size
15
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> a.sum()
105
>>> a.min()
0
>>> a.max()
14
>>> a.mean()
7.0
>>> b*2
array([12, 14, 16])
>>> b-b
array([0, 0, 0])
>>> b*b
array([36, 49, 64])
```

matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



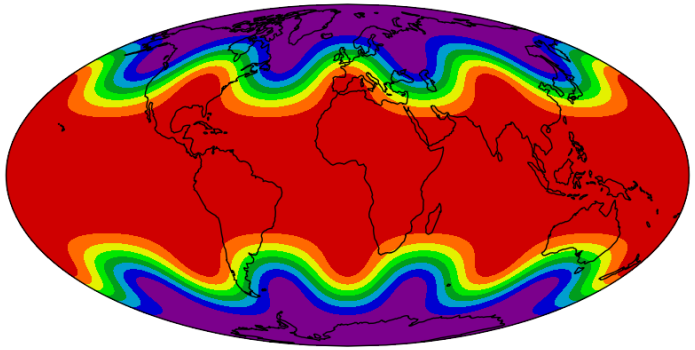
matplotlib

matplotlib – basemap (deprecated)

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np

# make sure the value of resolution is a lowercase L,
#  for 'low', not a numeral 1
map = Basemap(projection='ortho', lat_0=50, lon_0=-100,
              resolution='l', area_thresh=1000.0)

map.drawcoastlines()
map.drawcountries()
map.fillcontinents(color='coral')
map.drawmapboundary()

map.drawmeridians(np.arange(0, 360, 30))
map.drawparallels(np.arange(-90, 90, 30))

plt.show()
```

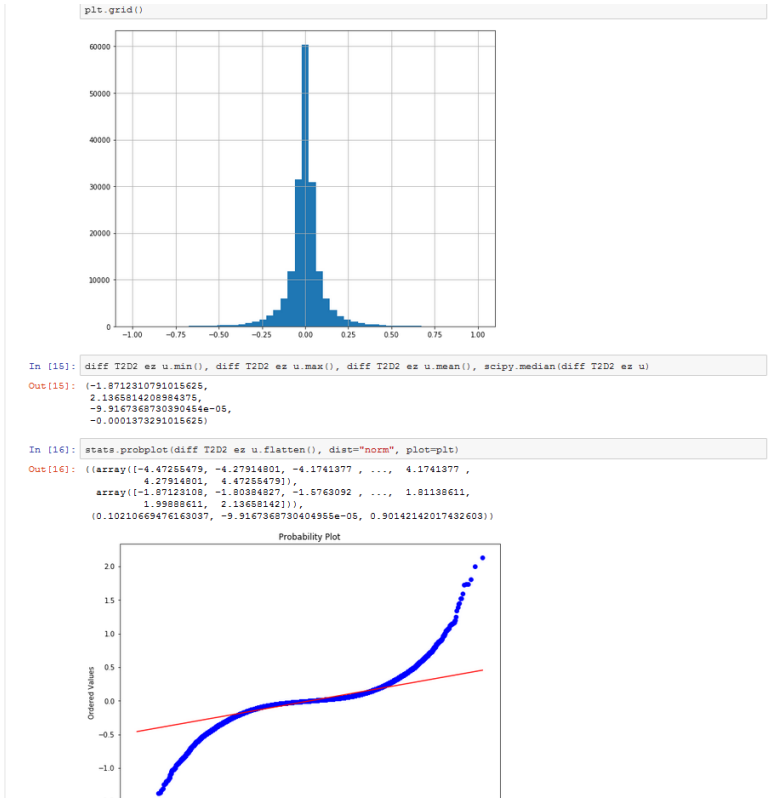
Cartopy

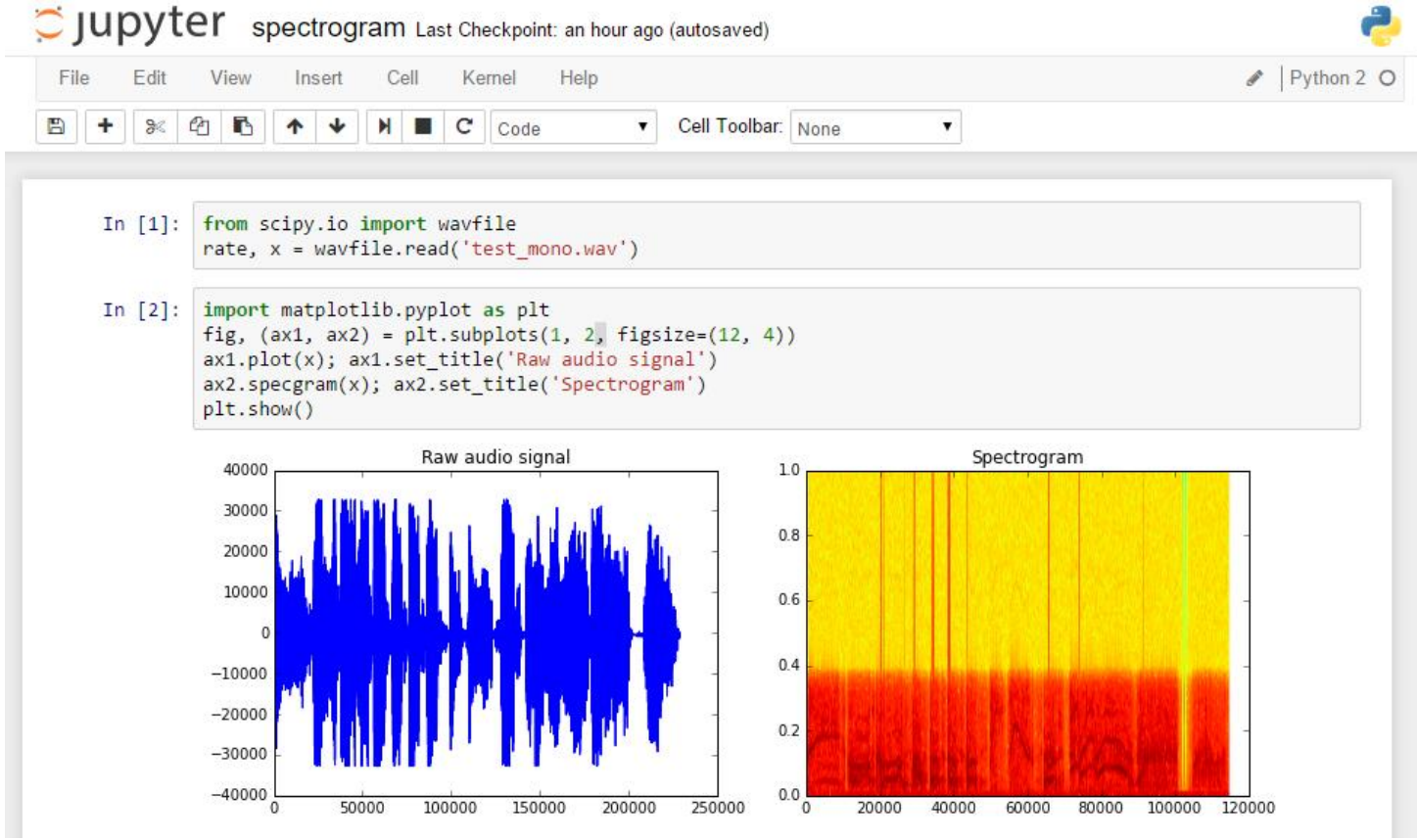
```
import matplotlib.pyplot as plt
import numpy as np
import cartopy.crs as ccrs

fig = plt.figure(figsize=(10, 5))
ax = fig.add_subplot(1, 1, 1, projection=ccrs.Mollweide())

# Fake data
nlats, nlons = (73, 145)
lats = np.linspace(-np.pi / 2, np.pi / 2, nlats)
lons = np.linspace(0, 2 * np.pi, nlons)
lons, lats = np.meshgrid(lons, lats)
wave = 0.75 * (np.sin(2 * lats) ** 8) * np.cos(4 * lons)
mean = 0.5 * np.cos(2 * lats) * ((np.sin(2 * lats)) ** 2 + 2)
lats = np.rad2deg(lats)
lons = np.rad2deg(lons)
data = wave + mean

ax.contourf(lons, lats, data, transform=ccrs.PlateCarree(),
            cmap='nipy_spectral')
ax.coastlines()
ax.set_global()
plt.show()
```



Ipython – Jupyter

- Interactive and enhanced python console
- Server/client web Notebooks





Python at ECMWF

- Currently three interfaces for ECMWF libraries
  - ecCodes
  - Magics++
  - Metview
- ecCharts
- Web plots (ecCodes, magics++)
- Verification (ecCodes, magics++)
- EcFlow (SMS's replacement) - server configuration and client communication
- Copernicus Data Store (Toolbox)
- EFAS (European Flood Alert System) (EcFlow)
- Research

Magics++

- ECMWF's inhouse meteorological plotting software
- Used at ECMWF and in the member states for more than 25 years
- Supports the plotting of contours, wind fields, observations, satellite images, symbols, text, axis and graphs
- Two different ways of plotting
  - Data formats which can be plotted directly: GRIB1, GRIB2, BUFR, ODB, NetCDF and NumPy
  - Data fields can be read with ecCodes, can be modified and then passed to magics++ for plotting
- The produced meteorological plots can be saved in various formats, such as PS, EPS, PDF, GIF, PNG, KML and SVG
- Provides both a procedural and a high-level Python programming interface

Python API – High Level interface

- High-level, more *pythonic* interface

```
with GribFile(filename) as grib:
    # Iterate through each message in the file
    for msg in grib:
        # Access a key from each message
        print(msg[key_name])
        # Report message size in bytes
        msg.size()
        # Report keys in message
        msg.keys()
        # Set scalar value
        msg[scalar_key] = 5
        # Array values are set transparently
        msg[array_key] = [1, 2, 3]
        # Messages can be written to file
        with open(testfile, "w") as test:
            msg.write(test)
        # Messages can be cloned from other messages
        msg2 = GribMessage(clone=msg)
```

https://confluence.ecmwf.int/display/ECC/High-level+Pythonic+Interface+in+ecCodes

Python API – High Level interface

```
f = open("my.grib")

while 1:
   gid = codes_grib_new_from_file(f)
   if gid is None:
      break

   keys = ('dataDate', 'dataTime', 'shortName')
   for key in keys:
      print('%s: %s' % (key, codes_get(gid, key)))


   print(nvalues=%d, avg=%f, min=%f, max=%f' % (
         codes_get_size(gid, 'values'),
         codes_get(gid, 'average'),
         codes_get(gid, 'min'),
         codes_get(gid, 'max')))

   codes_release(gid)

f.close()
```

```
with GribFile("my.grib") as grib:

    for msg in grib:


       keys = ('dataDate', 'dataTime', 'shortName')
       for key in keys:
          print('%s: %s' % (key, msg[key]))


    print(nvalues=%d, avg=%f, min=%f, max=%f' % (
          len(msg),
          msg['average']
          msg['min']
          msg['max']
```

https://confluence.ecmwf.int/display/ECC/High-level+Pythonic+Interface+in+ecCodes

Python API – High Level interface

- High-level, more *pythonic* interface

```
# Write index to file
with GribIndex(filename, keys) as idx:
    idx.write(index_file)

# Read index from file
with GribIndex(file_index=index_file) as idx:
    # Add new file to index
    idx.add(other_filename)
    # Report number of unique values for given key
    idx.size(key)
    # Report unique values indexed by key
    idx.values(key)
    # Request GribMessage matching key, value
    msg = idx.select({key: value})
```

https://confluence.ecmwf.int/display/ECC/High-level+Pythonic+Interface+in+ecCodes

Python API – High Level interface

```
i_keys = ["dataDate", "shortName"]

id = codes_index_new_from_file("my.grib", i_keys)

dates = codes_index_get(id, "dataDate")
names = codes_index_get(id, "shortName")

print dates, names

for date in dates:
    codes_index_select(id, "dataDate", date)
    for name in names:
        codes_index_select(id, "shortName", name)
        gid = codes_new_from_index(id)
        values = codes_get_values(gid)
        if name == "2t":
            values = values - 273.15
        print date, name, values.mean()
        codes_release(gid)

codes_index_release(id)
```

```
i_keys = ["dataDate", "shortName"]

with GribIndex("my.grib", i_keys) as idx:

    dates = idx.values("dataDate")
    names = idx.values("shortName")

    print dates, names

    for date in dates:

        for name in names:
            msg = idx.select({"dataDate": date,
                              "shortName": name})
            values = msg["values"]
            if name == "2t":
                values = values - 273.15
            print date, name, values.mean()
```

https://confluence.ecmwf.int/display/ECC/High-level+Pythonic+Interface+in+ecCodes

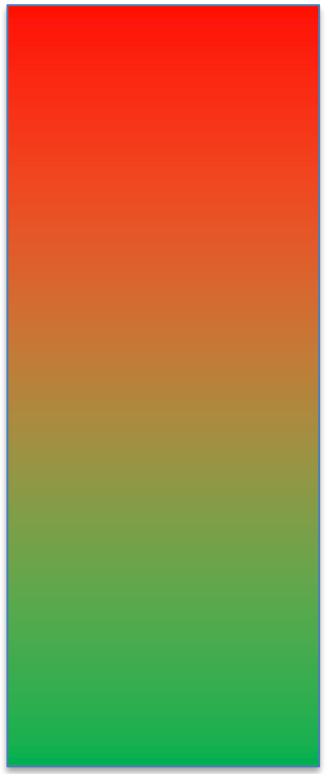
Example scripts

```
$> cd $SCRATCH
$> tar xf ~trx/ecCodes/python-grib-practicals.tar.gz
```

- ecCodes:
  - index.py: example on indexed access
  - reading.py: example on matplotlib usage
  - geo.py: example on iterating over the lat/lon values
- basemap: examples of basemap plotting from grib (deprecated)
- Cartopy: examples of cartopy plotting from grib
- magics: examples of plotting using Magics++
- performance: little example comparing the performance of the tool, the Fortran and the python API
- challenge: Jupyter notebook exercise

THE CHALLENGE

Compute and plot wind speed out of u and v fields.



0 Open the Jupyter Notebook “eccodes_python_challenge.ipynb”

```
$> cd $SCRATCH/python-grib-practicals/challenge
$> jupyter notebook
```

3 Obtain the relevant values for the computation out of the u and v grib fields

5 Print the minimum and maximum values of the wind speed values computed out of the wind components

7 Produce a new file containing **a semantically correct** field for wind speed

9 Produce a plot of the new field (using python)

10 Print the 10 points with maximum wind speeds (with their lat/lon coordinates)

If you finish, see the geek challenge….

THE GEEK CHALLENGE

Create an interactive plot with cartopy and ipywidgets

- Use the file ztuv_ens.grib. It contains ensemble data for different dates
- Inspect it with grib_ls to get a feeling of what is in the file
- Create a new jupyter notebook
- You will need to create widgets to be able to select:
  - The ensemble member to plot
  - The date to plot
- Compute windspeed and update the plot according to the values on the widgets

References

Python specifics

http://www.python.org/

NumPy

http://numpy.scipy.org/

http://www.scipy.org/Numpy_Functions_by_Category

http://docs.scipy.org/numpy/docs/numpy/

http://www.scipy.org/NumPy_for_Matlab_Users

Langtangen, Hans Petter, ”Python scripting for computational science”

References

SciPy

http://www.scipy.org/

Matplotlib / Cartopy / Basemap

http://matplotlib.sourceforge.net/

https://scitools.org.uk/cartopy/docs/latest/

http://matplotlib.org/basemap

ecCodes

https://confluence.ecmwf.int/display/ECC/ecCodes+Home

Magics

https://confluence.ecmwf.int/display/MAGP