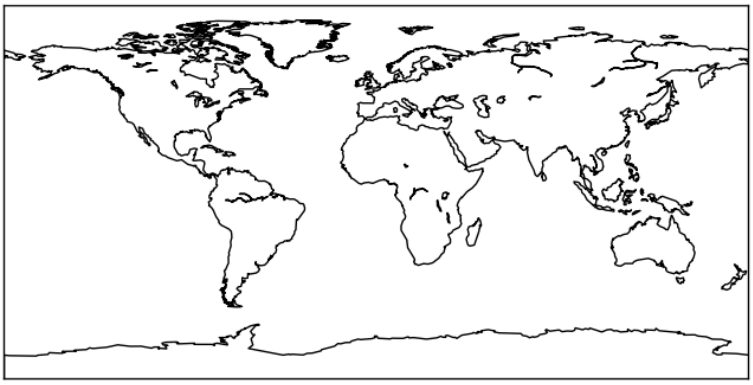
Basemap

Basemap is a great tool for creating maps using python in a simple way. It's a matplotlib extension, so it has got all its features to create data visualizations, and adds the geographical projections and some datasets to be able to plot coast lines, countries, and so on directly from the library.

Drawing the first map

Let's create a the simplest map:

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

map = Basemap()

map.drawcoastlines()

plt.show()
plt.savefig('test.png')
```



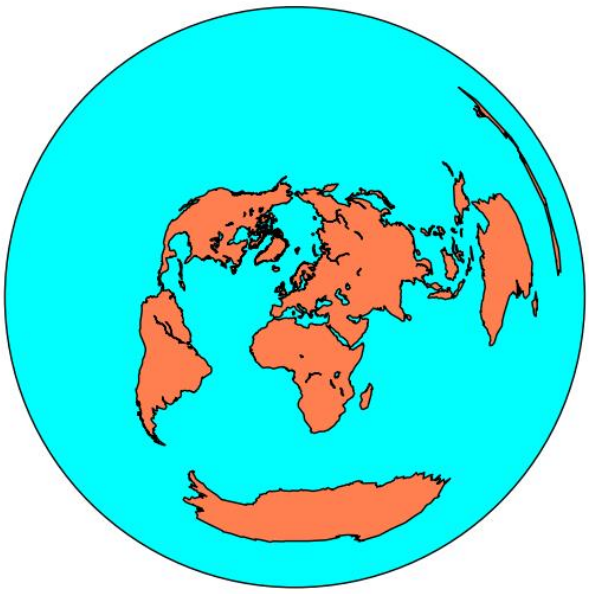
Managing projections

All maps must have a projection. The projection and its features are all assigned when the object *Basemap* is created.

Projection

The projection argument sets the map projection to be used:

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

map = Basemap(projection='aeqd', lon_0 = 10, lat_0 = 50)

map.drawmapboundary(fill_color='aqua')
map.fillcontinents(color='coral',lake_color='aqua')
map.drawcoastlines()

plt.show()
```

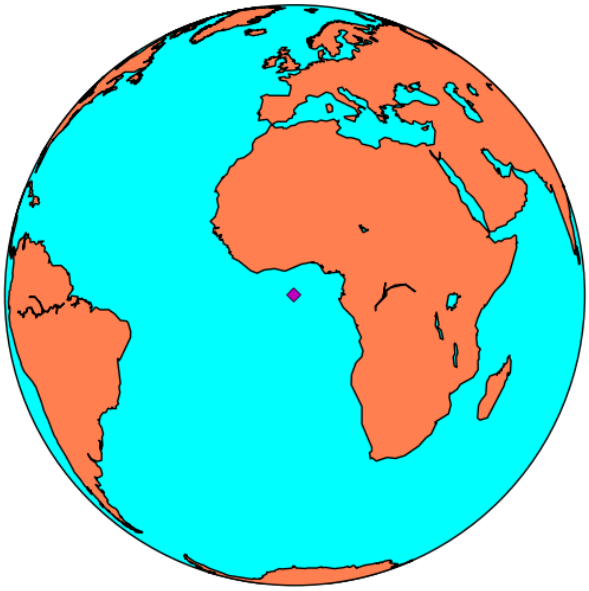


Drawing a point in a map

Drawing a point in a map is usually done using the plot method:

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

map = Basemap(projection='ortho',
              lat_0=0, lon_0=0)

map.drawmapboundary(fill_color='aqua')
map.fillcontinents(color='coral',lake_color='aqua')
map.drawcoastlines()
```

```
x, y = map(0, 0)
map.plot(x, y, marker='D',color='m')

plt.show()
```



Calculating the position of a point on the map

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

map = Basemap(projection='aeqd', lon_0 = 10, lat_0 = 50)

print map(10, 50)
print map(20015077.3712, 20015077.3712, inverse=True)
```

The output will be:

(20015077.3712, 20015077.3712) (10.000000000000002, 50.000000000000014)

When inverse is False, the input is a point in longitude and latitude, and the output is the point in the map coordinates. When inverse is True, the behavior is the opposite.

Working with shapefiles

The way used by Basemap to handle vectorial files is quite different from other libraries

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

```
map = Basemap(llcrnrlon=-0.5,llcrnrlat=39.8,urcrnrlon=4.,urcrnrlat=43.,
             resolution='i', projection='tmerc', lat_0 = 39.5, lon_0 = 1)

map.drawmapboundary(fill_color='aqua')
map.fillcontinents(color='#ddaa66',lake_color='aqua')
map.drawcoastlines()

map.readshapefile('../sample_files/comarques', 'comarques')

plt.show()
```

The first parameter shapefile name must go without the shp extension. The library assumes that all shp, sbf and shx files will exist with this given name

The second parameter is a name to access later to the shapefile information from the Basemap instance

Plotting data

imshow

Plots an image on the map. The image can be a regular rgb image, or a field colored with a cmap.

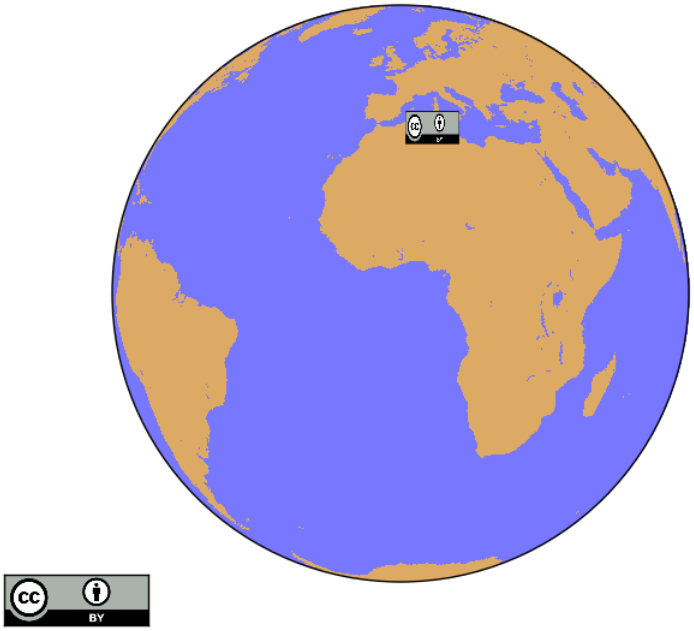
imshow(*args, **kwargs)

As in other cases, the best docs are at the matplotlib documentation.

- The first argument is the image array. If it has three bands, RGB will be assumed, and the image will be plotted. If it has one band, a pseudocolor will be created according to the cmap argument (jet by default). Arrays with two or more than three bands will make the method tp raise an error
- extent sets the position of four corners of the image, in map coordinates. It has to be a sequence with the elements (x0, x1, y0, y1)
- origin changes the initial line position of the image. By default, the first line position is defined by image.origin, and this can be changed using the values ‘upper’ or ‘lower’
- cmap is the desired colormap if the image has one band.
- alpha sets the transparency with values from 0 to 1

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
fig = plt.figure()

map = Basemap(projection='ortho',
              lat_0=0, lon_0=0)
```

```
map.drawlsmask(land_color = "#ddaa66",
               ocean_color="#7777ff",
               resolution = 'l')

x0, y0 = map(1., 31.)
x1, y1 = map(15., 39.)

plt.imshow(plt.imread('../sample_files/by.png'),  extent = (x0, x1, y0, y1))

axicon = fig.add_axes([0.1, 0., 0.15, 0.15])
axicon.imshow(plt.imread('../sample_files/by.png'), origin = 'upper')
axicon.set_xticks([])
axicon.set_yticks([])

plt.show()
```



text

Plots a text on the map

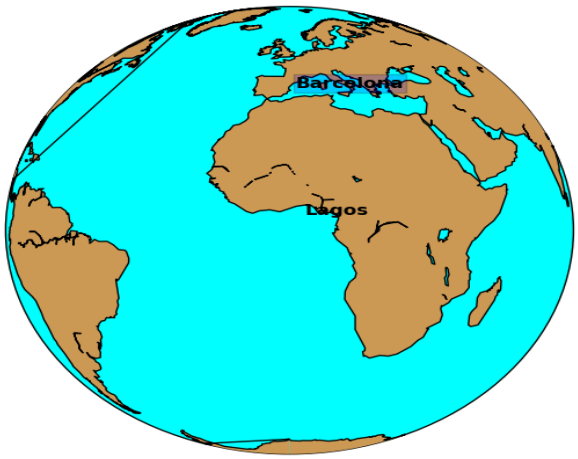
text(x, y, s, fontdict=None, withdash=False, **kwargs)

- The text method does not belong to Basemap, but directly to matplotlib, so it must be called from the plot or axis instance
- x and y are the coordinates in the map projection. Arrays of coordinates are not accepted, so to add multiple labels, call the method multiple times
- s is the text string
- withdash creates a text with a dash when set to true
- fontdict can be used to group the text properties

The text can have many many options such as:

- fontsize the font size
- fontweight font weight, such as bold
- ha horitzontal align, like center, left or right
- va vertical align, like center, top or bottom
- bbox creates a box around the text: bbox=dict(facecolor='red', alpha=0.5)

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

map = Basemap(projection='ortho',
              lat_0=0, lon_0=0)

map.drawmapboundary(fill_color='aqua')
map.fillcontinents(color='#cc9955',lake_color='aqua')
map.drawcoastlines()

lon = 3.4
lat = 3.4
x, y = map(lon, lat)


plt.text(x, y, 'Lagos',fontsize=12,fontweight='bold',
                    ha='left',va='bottom',color='k')
lon = 2.1
lat = 41.
x, y = map(lon, lat)

plt.text(x, y, 'Barcelona',fontsize=12,fontweight='bold',
                    ha='left',va='center',color='k',
                    bbox=dict(facecolor='b', alpha=0.2))
plt.show()
```



Background methods

bluemarble

Plots the bluemarble image on the map.

bluemarble(ax=None, scale=None, **kwargs)

- The scale is useful to downgrade the original image resolution to speed up the process. A value of 0.5 will divide the size of the image by 4
- The image is warped to the final projection, so all projections work properly with this method

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

map = Basemap(llcrnrlon=-10.5,llcrnrlat=33,urcrnrlon=10.,urcrnrlat=46.,
              resolution='i', projection='cass', lat_0 = 39.5, lon_0 = 0.)

map.bluemarble()

map.drawcoastlines()

plt.show()
```

warpimage

Displays an image as a background.

warpimage(image='bluemarble', scale=None, **kwargs)

- By default, displays the NASA Bluemarble image
- The image must be in latlon projection, so the x size must be double than the y size
- The image must cover the whole world, with the longitude starting at -180

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import Image

map = Basemap(projection='ortho',
              lat_0=0, lon_0=0)


tmpdir = '/tmp'

size = [600, 300]
im = Image.open("../sample_files/by.png")

im2 = im.resize(size, Image.ANTIALIAS)
im2.save(tmpdir+'/resized.png', "PNG")

map.warpimage(tmpdir+'/resized.png')

map.drawcoastlines()

plt.show()
```

Basemap utility functions

colorbars

Draws the color legend at one of the edges of the map. The use is almost the same as in matplotlib colorbar

colorbar(mappable=None, location='right', size='5%', pad='2%', fig=None, ax=None, **kwargs)

- mappable is the most important argument. Is the field on the map to be explained with the colorscale. It can be a contourf, a pcolormesh, contour, etc. If the value is None, the last drawn field is represented
- location sets the border of the map where the color scale is drawn. Can be top, right, left or bottom
- size sets the width of the color bar, in % of the parent axis
- pad sets the separation between the axes and the color bar, also in % of the parent axis
- fig is the figure the colorbar is associated with

Most of the matplotlib.colorbar arguments will work too, such as label

makegrid

makegrid method creates an arbitrary grid of equally spaced points in the map projection. Used to get the longitudes and latitudes that would form an an equally spaced grid using the map projection.

makegrid(nx, ny, returnxy=False)

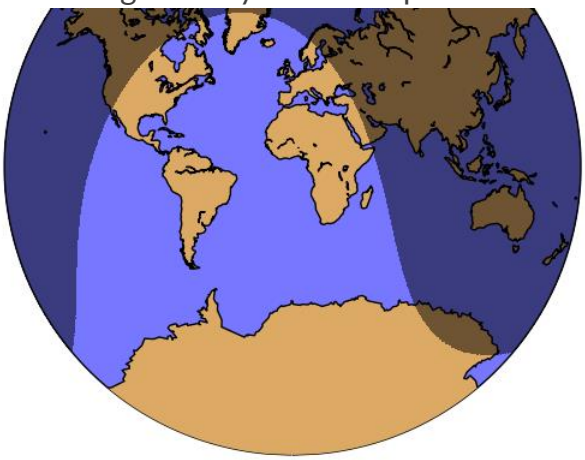
- nx and n define the size of the output grid
- If returnxy is set to True, the positions of the grid in the map projection are also returned
- returns the longitudes and latitudes 2D numpy arrays, and the x and y in the map coordinates arrays if returnxy is set to True

nightshade

Draws the regions of the map which are dark at the specified date

nightshade(date, color='k', delta=0.25, alpha=0.5, ax=None, zorder=2)

- date is a datetime.datetime object
- color is the color of the drawn shadow
- delta is the resolution to which the shadow zone is calculated. By default is 0.25, and small values fail easily
- alpha is the transparency value
- zorder can change the layer vertical position



Multiple maps using subplots

Drawing multiple maps in the same figure is possible using matplotlib's *subplots*. There are several ways to use them, and depending on the complexity of the desired figure, one or other is better:

- Creating the axis using subplot directly with add_subplot
- Creating the subplots with pylab.subplots
- Using subplot2grid
- Creating Inset locators

Basemap in 3D

Creating a basic map

The most important thing to know when starting with 3d matplotlib plots is that the *Axes3D* class has to be used. To add geographical data to the map, the method *add_collection3d* will be used:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```
from mpl_toolkits.basemap import Basemap
map = Basemap()
fig = plt.figure()
ax = Axes3D(fig)
'''
ax.azim = 270
ax.elev = 90
ax.dist = 5
'''
ax.add_collection3d(map.drawcoastlines(linewidth=0.25))
ax.add_collection3d(map.drawcountries(linewidth=0.35))
plt.show()
```

- The ax variable is in this example, an Axes3D instance. All the methods will be used from this instance, so they need to support 3D operations, which doesn't occur in many cases on the basemap methods
- The commented block shows how to rotate the resulting map so the view is better
- To draw lines, just use the add_collection3d method with the output of any of the basemap methods that return an matplotlib.patches.LineCollection object, such as drawcountries