Adding data to the map

Now we know how to make and customize basic maps. Great. But we want to use these maps to convey information or results, ie to display data.

So how do we add data to our maps?

credit

As always, lots of this lesson is based on Ryan Abernathy's course: https://rabernat.github.io/research_computing_2018/maps-with-cartopy.html. Parts too are from the person who wrote the Cartopy package, Phil Elson , tutorial here: https://github.com/SciTools/cartopy-tutorial/tree/42cb77062a08063a53e7a511a9681bdb15e70fe7.

In [1]:
```
# import statements

import cartopy.crs as ccrs
import cartopy

import matplotlib.pyplot as plt

%matplotlib inline
```

Cartopy plays well with matplotlib

Because our map *is* a matplotlib axis, we can use all the familiar maptplotlib plotting tools and commands to make plots. By default, the map extent will be adjusted to match the data. We can override this with the `.set_global` or `.set_extent` commands.

In [2]:
```
# create some test data
new_york_lon = -74.0060
new_york_lat = 40.7128
honolulu_lon = -157.8583
honolulu_lat = 21.3069

lons = [new_york_lon, honolulu_lon]
lats = [new_york_lat, honolulu_lat]
```

Key point:

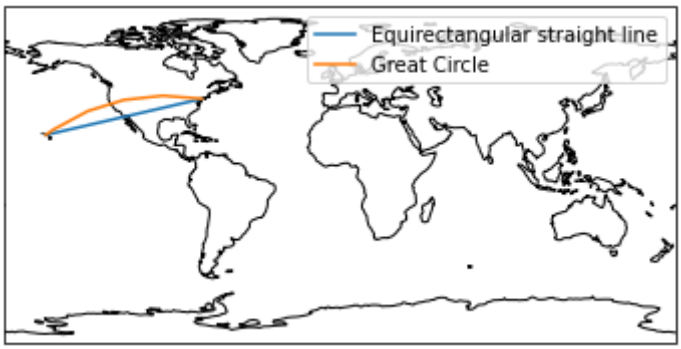
The data *ALSO* have to be transformed to the projection space. This is done via the `transform=` keyword in the plotting method. The argument is another `cartopy.crs` object. If you don't specify a transform, Cartopy assume that the data is using the same projection as the underlying GeoAxis.

> ### From the Cartopy Documentation:
>
> The core concept is that the projection of your axes is independent of the coordinate system your data is defined in. The `projection` argument is used when creating plots and determines the projection of the resulting plot (i.e. what the plot looks like). The `transform` argument to plotting functions tells Cartopy what coordinate system your data are defined in.

```
ax = plt.axes(___=ccrs.PlateCarree())
ax.plot(lons, lats, label='Equirectangular straight line')
ax.plot(lons, lats, label='Great Circle', ___=ccrs.Geodetic())
ax.coastlines()
ax.legend()
ax.set_global()
```

In [4]:

```
ax = plt.axes(projection=ccrs.PlateCarree())
ax.plot(lons, lats, label='Equirectangular straight line')
ax.plot(lons, lats, label='Great Circle', transform=ccrs.Geodetic())
ax.coastlines()
ax.legend()
ax.set_global()
```



deconstruct the plot above

First, we just used a normal matplotlib `.plot()` command to plot our lat an lon data. This is actually just the normal command, its nothing special from Cartopy. In the first case we didn't add the `transform=` argument, so the map assumed that the data are already in the map projection (what is the map projection here?).

In the second case we told the plotting function that the original data we want to plot (`lon` , `lat`) are in `ccrs.Geodetic()` projection, so it then transformed them into the defined map projection.

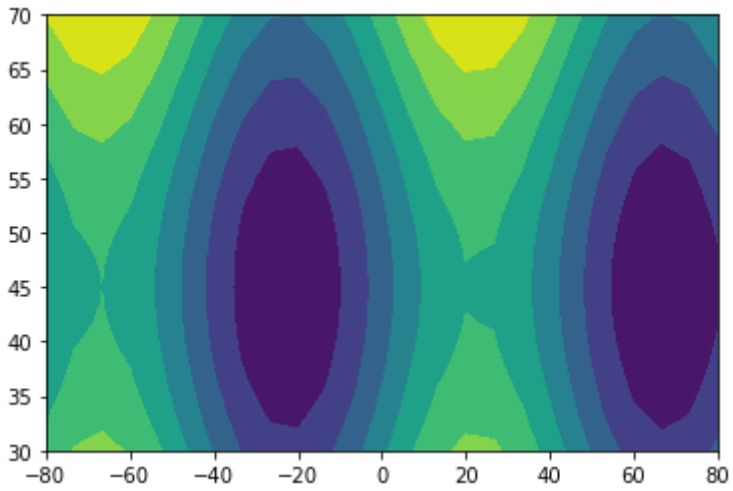
Plotting 2D (Raster) Data

The same principles apply to 2D data. Below we create some fake example data defined in regular lat / lon coordinates.

When we plot this there is no projection, just good old standard matplotlib.

In [9]:
```
import numpy as np

lon = np.linspace(-80, 80, 25)
lat = np.linspace(30, 70, 25)
lon2d, lat2d = np.meshgrid(lon, lat)

data = np.cos(np.deg2rad(lat2d) * 4) + np.sin(np.deg2rad(lon2d) * 4)

plt.contourf(lon2d, lat2d, data)
```

Out[9]:
```
<matplotlib.contour.QuadContourSet at 0x7fc788dc6d90>
```

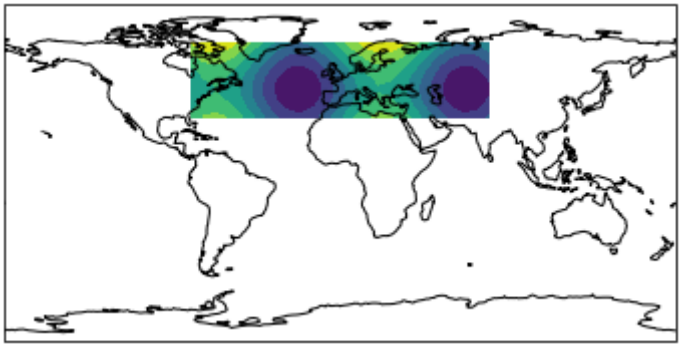


Now we create a `PlateCarree` projection and plot the data on it without any `transform` keyword. This happens to work because `PlateCarree` is the simplest projection of lat / lon data.

```
ax = plt.axes(___=ccrs.PlateCarree())
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data)
```

In [10]:
```
ax = plt.axes(projection=ccrs.PlateCarree())
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data)
```

Out[10]:
```
<cartopy.mpl.contour.GeoContourSet at 0x7fc798388d60>
```



However, if we try the same thing with a different projection, we get the wrong result.

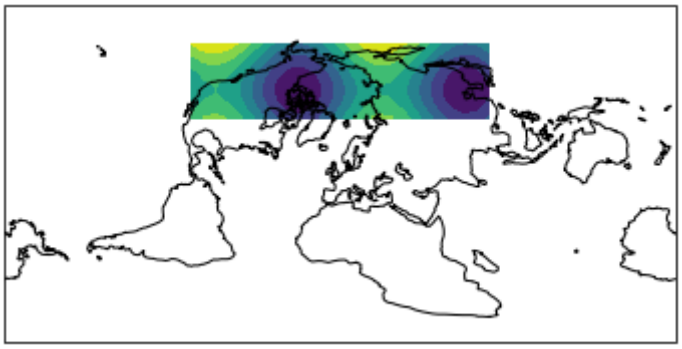
```
projct = ccrs.RotatedPole(pole_longitude=-177.5, pole_latitude=37.5)
ax = plt.axes(projection=projct)
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data)
```

In [11]:

```
projct = ccrs.RotatedPole(pole_longitude=-177.5, pole_latitude=37.5)
ax = plt.axes(projection=projct)
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data)
```

Out[11]:

```
<cartopy.mpl.contour.GeoContourSet at 0x7fc7584d33a0>
```



To make this work, we need to make sure to give the `transform=` argument in `contourf`. In particular we need to tell contourf that the data is in the standard `PlateCarree` projection. This is almost always going to be the original data projection (as far as I know, unless you know for sure otherwise). It's generally a good bet to pass the `transform=ccrs.PlateCarree()` in whatever matplotlib plotting function you are using in a map.

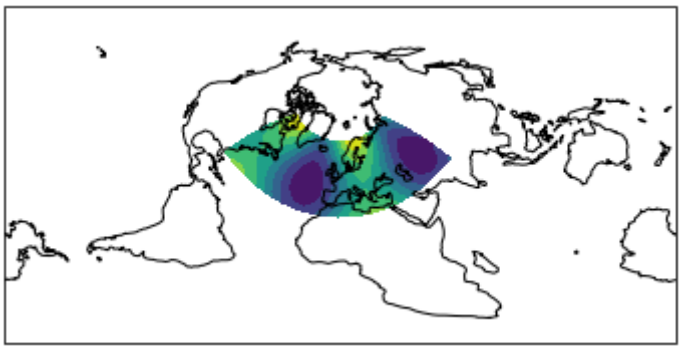
```
projection = ccrs.RotatedPole(pole_longitude=-177.5, pole_latitude=37.5)
ax = plt.axes(projection=projection)
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data, ___=ccrs.PlateCarree())
```

In [12]:

```
projection = ccrs.RotatedPole(pole_longitude=-177.5, pole_latitude=37.5)

ax = plt.axes(projection=projection)
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data, transform=ccrs.PlateCarree())
```

Out[12]:

```
<cartopy.mpl.contour.GeoContourSet at 0x7fc788cd7070>
```



Showing images

We can plot a satellite image easily on a map if we know its lat/lon extent. There are lots more ways to plot satellite data we can dig into if you want

make a figure with the Satellite data

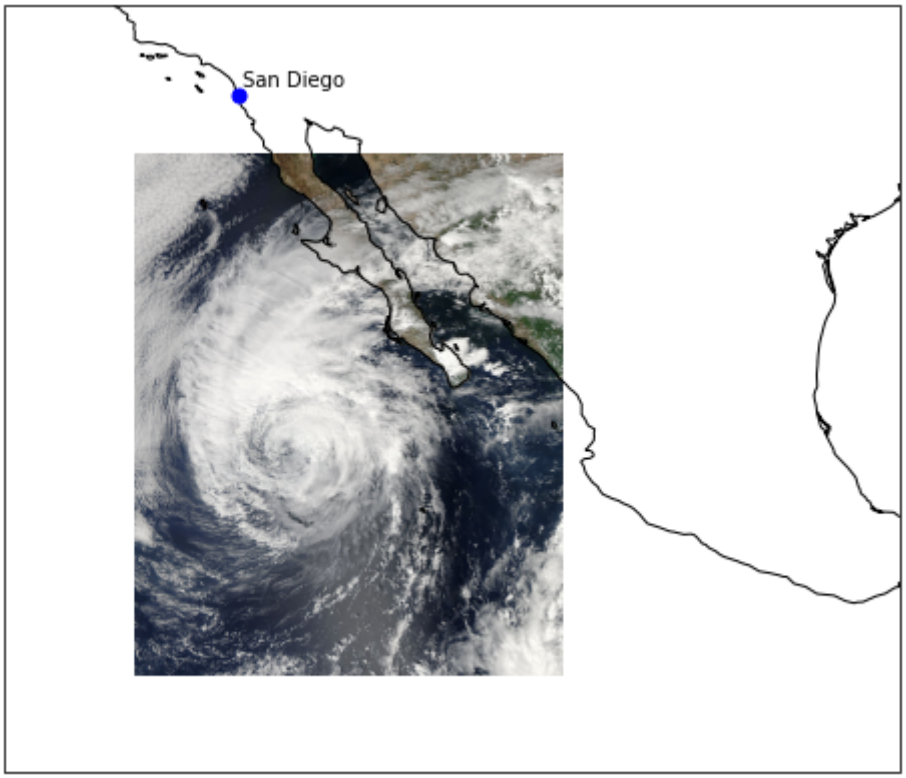
We are going to use a matplotlib function called `imshow()` . What do we need to add to it?

```
fig = plt.figure(figsize=(8, 10))

# load image & load coordinates of the corners
fname = '../data/Miriam.A2012270.2050.2km.jpg'
img_extent = (-120.67660000000001, -106.32104523100001,
13.2301484511245, 30.766899999999502)
img = plt.imread(fname)

# define projection for axes
_____________________

# add the image. Because this image was a tif, the "origin" of the image
is in the
# upper left corner
ax.imshow(____, origin='upper', extent=_____, transform=_______)
ax.coastlines(resolution='50m', color='black', linewidth=1)

# mark a known place to help us geo-locate ourselves
ax.plot(-117.1625, 32.715, 'bo', markersize=7)
ax.text(-117, 33, 'San Diego')

# set map extent to larger area
___________________
```

In [27]:

```
fig = plt.figure(figsize=(8, 10))

# load image & load coordinates of the corners
fname = '../data/Miriam.A2012270.2050.2km.jpg'
img_extent = (-120.67660000000001, -106.32104523100001, 13.2301484511245, 30.766
img = plt.imread(fname)
```

```
# define projection for axes
ax = plt.axes(projection=ccrs.PlateCarree())

# add the image. Because this image was a tif, the "origin" of the image is in t
# upper left corner
ax.imshow(img, origin='upper', extent=img_extent, transform=ccrs.PlateCarree())
ax.coastlines(resolution='50m', color='black', linewidth=1)

# mark a known place to help us geo-locate ourselves
ax.plot(-117.1625, 32.715, 'bo', markersize=7)
ax.text(-117, 33, 'San Diego')

# set map extent to larger area
ax.set_extent([-125, -95,10,35])
```



In [ ]: