Plotting review, 2-D plots, and mapping

ESS 116 | Fall 2024

Prof. Henri Drake, Prof. Jane Baldwin, and Prof. Michael Pritchard

(Modified from Ethan Campbell and Katy Christensen's materials for UW's Ocean 215)

What we'll cover in this lesson

1. Review of plotting concepts

2. 2-D plotting

3. Mapping with Cartopy

What we'll cover in this lesson

1. **Review of plotting concepts**

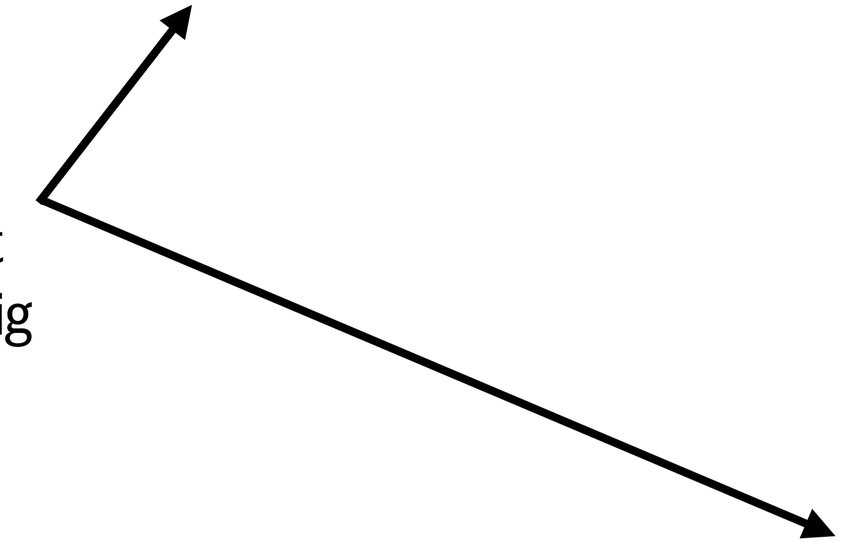
2. 2-D plotting

3. Mapping with Cartopy

Creating a figure

```
fig = plt.figure( )
```

You can add the `figsize` argument here to customize how big your figure is



```
fig, ax = plt.subplots(nrows=   , ncols=   )
```

Creating a figure

```
fig = plt.figure( )
```

When creating a figure with a single axes, you can create a variable for your axes object with this code:

```
ax=plt.axes()
```

You can add the `figsize` argument here to customize how big your figure is

```
fig, ax = plt.subplots(nrows=   , ncols=   )
```

To call the axis that is currently being plotted on, use the code: `plt.gca()`

Line plot

```
plt.plot(x, y, c='r', ls='-', marker='o', ms=5, lw=2)
```

These are all optional formatting arguments

Replace this with `ax` if you have an axes object

Line plot

```
plt.plot(x, y, c='r', ls='-', marker='o', ms=5, lw=2)
```

Replace this with `ax` if you have an axes object

`color`
input: string

`linestyle`
input: string

`marker`
input: string

`markersize`
input: float

`linewidth`
input: float

These are all optional formatting arguments

Line plot

```
plt.plot(x, y, c='r', ls='-', marker='o', ms=5, lw=2)
```

Replace this with `ax` if you have an axes object

`color`
input: string

`linestyle`
input: string

`marker`
input: string

`markersize`
input: float

`linewidth`
input: float

These are all optional formatting arguments

Line plot

```
plt.plot(x, y, 'ro-', ms=5, lw=2)
```

Color, linestyle, and marker can be combined in a single string if the color is a single letter shortcut

Optional formatting arguments

Markers

character	description
`'.'`	point marker
`','`	pixel marker
`'o'`	circle marker
`'v'`	triangle_down marker
`'^'`	triangle_up marker
`'<'`	triangle_left marker
`'>'`	triangle_right marker
`'1'`	tri_down marker
`'2'`	tri_up marker
`'3'`	tri_left marker
`'4'`	tri_right marker
`'s'`	square marker
`'p'`	pentagon marker
`'*'`	star marker
`'h'`	hexagon1 marker
`'H'`	hexagon2 marker
`'+'`	plus marker
`'x'`	x marker
`'D'`	diamond marker
`'d'`	thin_diamond marker
`'|'`	vline marker
`'_'`	hline marker

Line Styles

character	description
`'-'`	solid line style
`'--'`	dashed line style
`'-.'`	dash-dot line style
`':'`	dotted line style

Colors

The supported color abbreviations are the single letter codes

character	color
`'b'`	blue
`'g'`	green
`'r'`	red
`'c'`	cyan
`'m'`	magenta
`'y'`	yellow
`'k'`	black
`'w'`	white

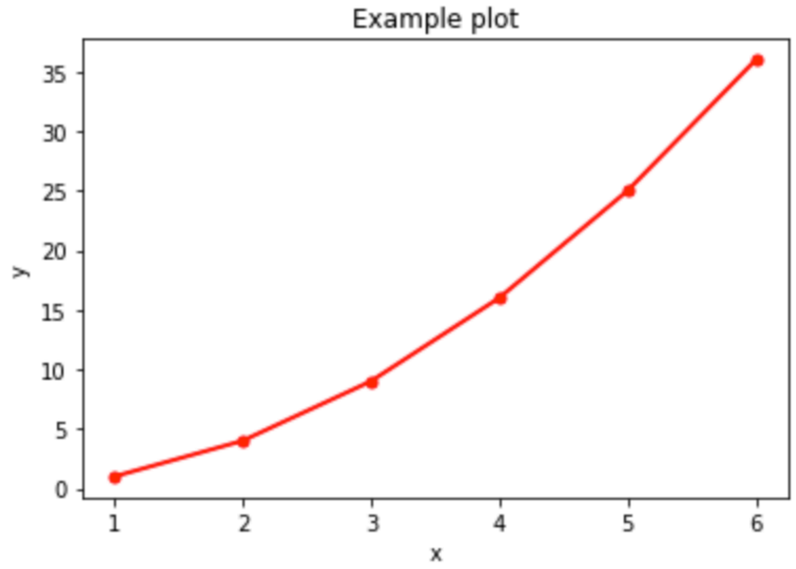
matplotlib
Version 3.3.2

https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.plot.html

Customizing line plots

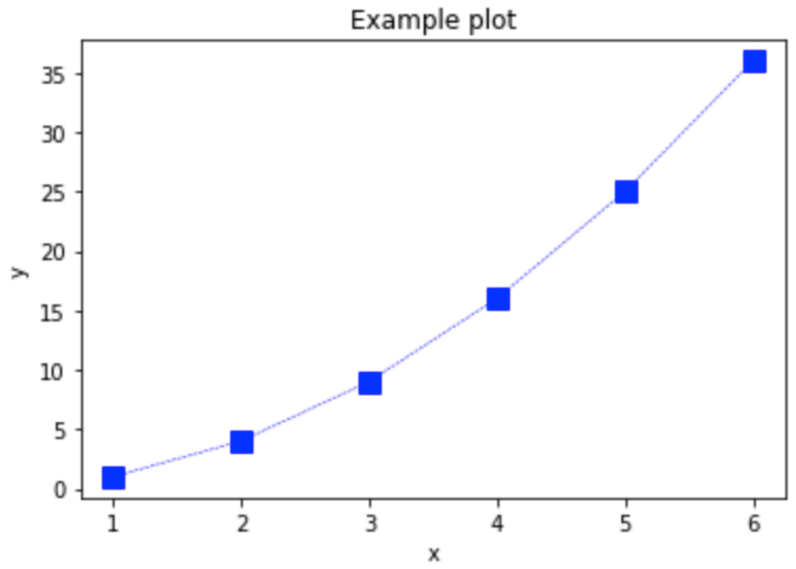
```
1 x = np.array([1,2,3,4,5,6])
2 y = x**2
3
4 fig = plt.figure()
5 plt.plot(x, y, c='r' ,ls='-' ,marker='o',ms=5 , lw=2)
6 plt.xlabel('x')
7 plt.ylabel('y')
8 plt.title('Example plot')
```

```
Text(0.5, 1.0, 'Example plot')
```




```
1 x = np.array([1,2,3,4,5,6])
2 y = x**2
3
4 fig = plt.figure()
5 plt.plot(x, y, c='b' ,ls='--' ,marker='s',ms=10 , lw=0.5)
6 plt.xlabel('x')
7 plt.ylabel('y')
8 plt.title('Example plot')
```

```
Text(0.5, 1.0, 'Example plot')
```



Scatter plot

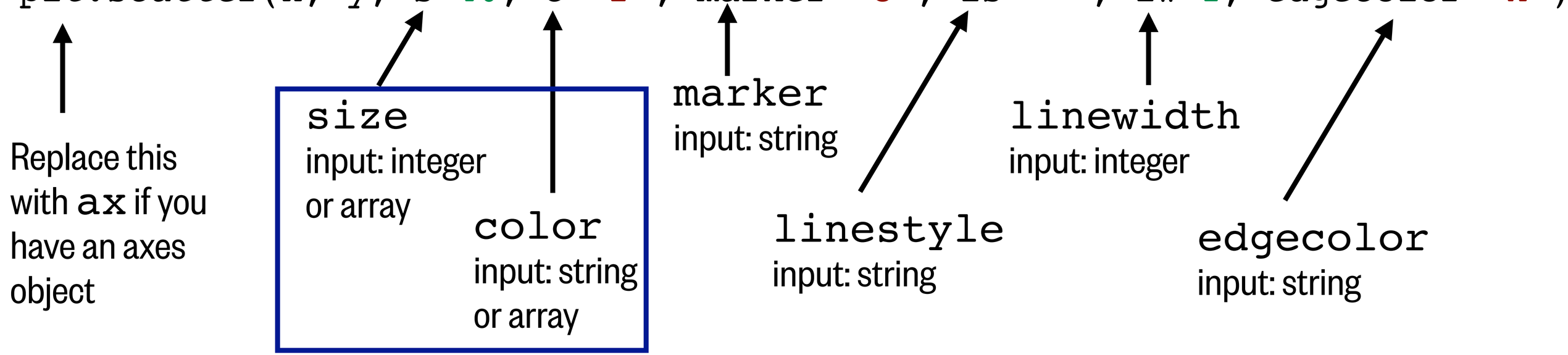
```
plt.scatter(x, y, s=40, c='r', marker='o', ls='-', lw=1, edgecolor='k')
```

These are all optional formatting arguments

Replace this with `ax` if you have an axes object

Scatter plot

```
plt.scatter(x, y, s=40, c='r', marker='o', ls='-', lw=1, edgecolor='k')
```



Scatter plot

```
plt.scatter(x, y, s=40, c='r', marker='o', ls='-', lw=1, edgecolor='k')
```

Replace this with `ax` if you have an axes object

`size`
input: integer or array

`color`
input: string or array

`marker`
input: string

`linestyle`
input: string

`linewidth`
input: integer

`edgecolor`
input: string

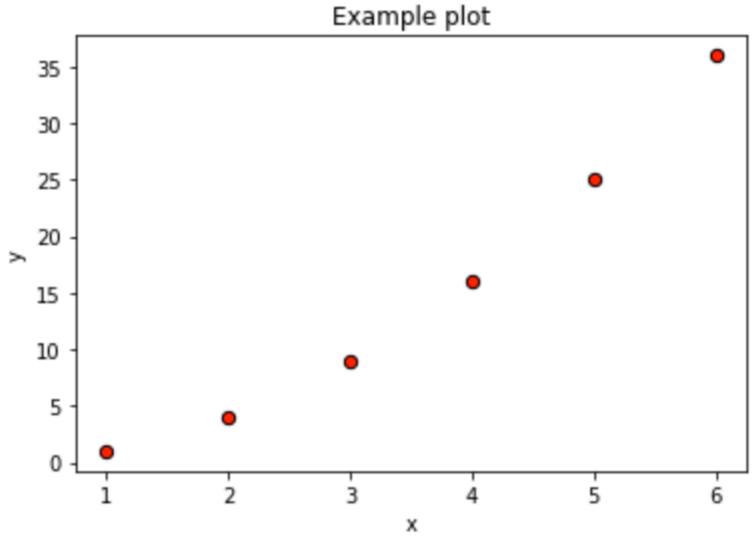
These are all optional formatting arguments

Line arguments in a scatter plot are different from line arguments in a line plot…

Scatter plot

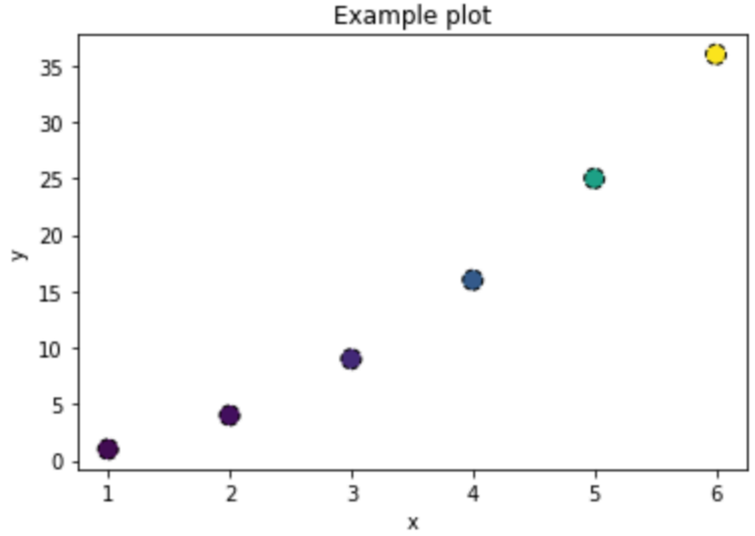
```
1 x = np.array([1,2,3,4,5,6])
2 y = x**2
3 z = x**3
4
5 fig = plt.figure()
6 plt.scatter(x,y, s=40, c='r', marker='o', ls='-', lw=1, edgecolor='k')
7 plt.xlabel('x')
8 plt.ylabel('y')
9 plt.title('Example plot')
```

```
Text(0.5, 1.0, 'Example plot')
```




```
1 x = np.array([1,2,3,4,5,6])
2 y = x**2
3 z = x**3
4
5 fig = plt.figure()
6 plt.scatter(x,y, s=100, c=z, marker='o', ls='--', lw=1, edgecolor='k')
7 plt.xlabel('x')
8 plt.ylabel('y')
9 plt.title('Example plot')
```

```
Text(0.5, 1.0, 'Example plot')
```

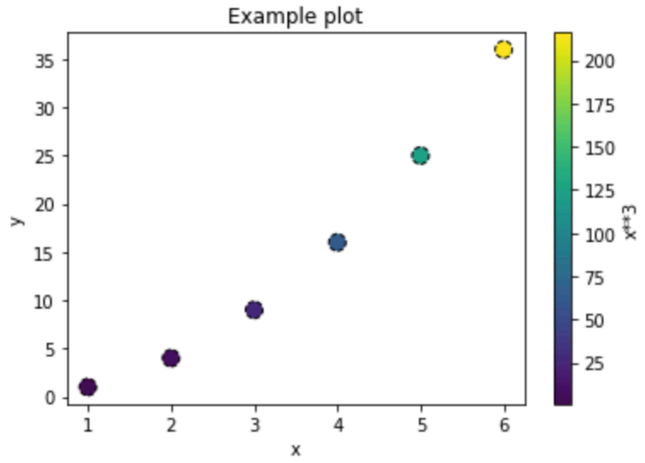


Considering color

Because our color can change using scatter plot, we have to show what the colors mean using a colorbar.

Notice that I created a variable name (**`ax`**) for my axes object and a variable name (**`scat`**) for my plot.

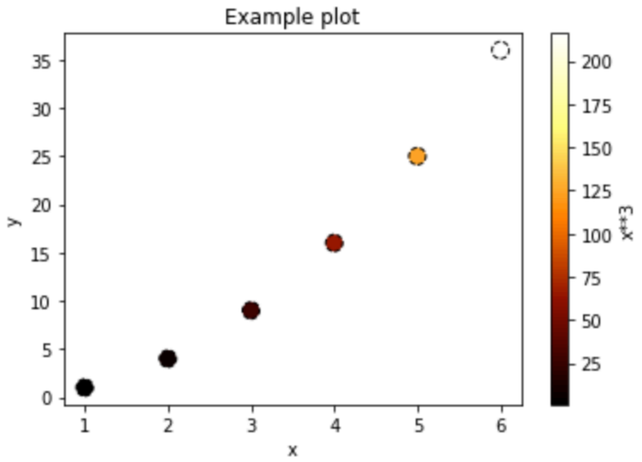
```
x = np.array([1,2,3,4,5,6])
y = x**2
z = x**3

fig = plt.figure()
ax = plt.gca()
scat = plt.scatter(x,y, s=100, c=z, marker='o', ls='--', lw=1, edgecolor='k')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Example plot')

c = plt.colorbar(scat, ax=ax)
c.set_label('x**3')
```



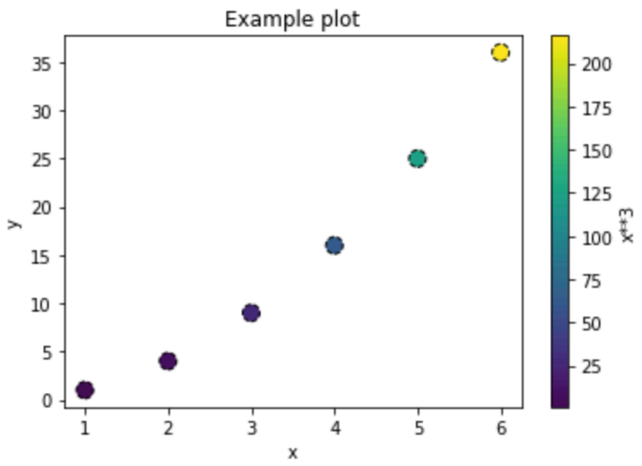
`ax` and **`scat`** are used as argument for my colorbar. This ensures that my colorbar is linked to my plot.

Colormaps

```
x = np.array([1,2,3,4,5,6])
y = x**2
z = x**3

fig = plt.figure()
ax = plt.gca()
scat = plt.scatter(x,y, s=100, c=z, marker='o', ls='--', lw=1, edgecolor='k', cmap='afmhot')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Example plot')

c = plt.colorbar(scat, ax=ax)
c.set_label('x**3')
```

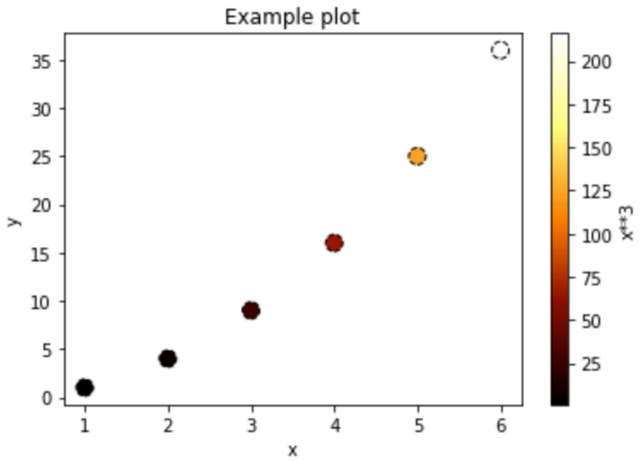



```
x = np.array([1,2,3,4,5,6])
y = x**2
z = x**3

fig = plt.figure()
ax = plt.gca()
scat = plt.scatter(x,y, s=100, c=z, marker='o', ls='--', lw=1, edgecolor='k')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Example plot')

c = plt.colorbar(scat, ax=ax)
c.set_label('x**3')
```



Colormaps

```
x = np.array([1,2,3,4,5,6])
y = x**2
z = x**3

fig = plt.figure()
ax = plt.gca()
scat = plt.scatter(x,y, s=100, c=z, marker='o', ls='--', lw=1, edgecolor='k', cmap='afmhot')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Example plot')

c = plt.colorbar(scat, ax=ax)
c.set_label('x**3')
```



matplotlib
Version 3.3.2

https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html

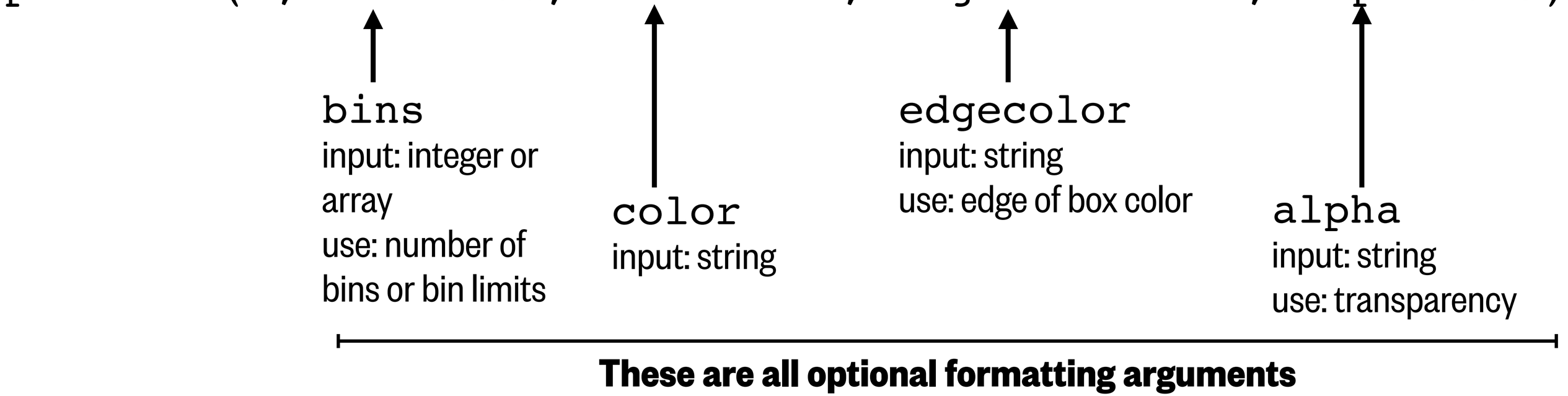
Histograms

```
plt.hist(x, bins=100, color='b', edgecolor='k', alpha=0.5)
```

These are all optional formatting arguments

Histograms

```
plt.hist(x, bins=100, color='b', edgecolor='k', alpha=0.5)
```

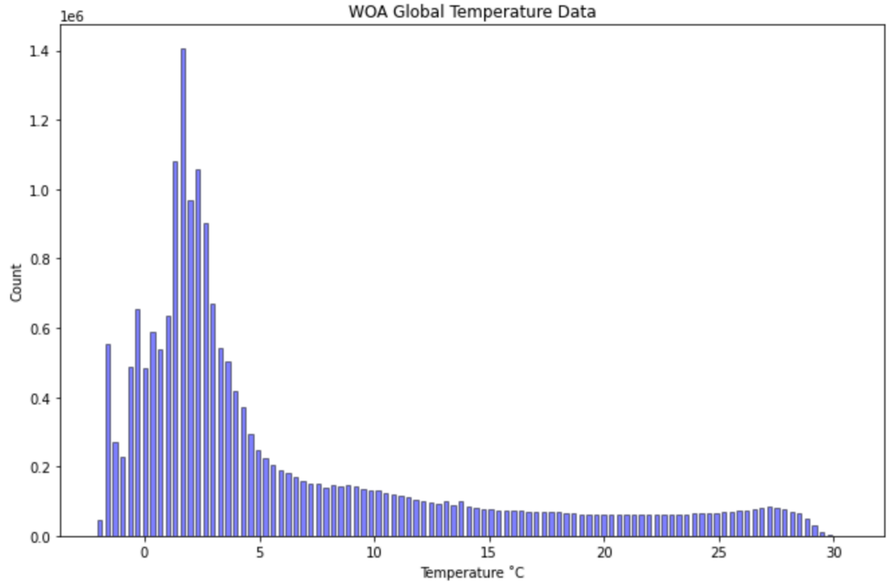


Histograms - example

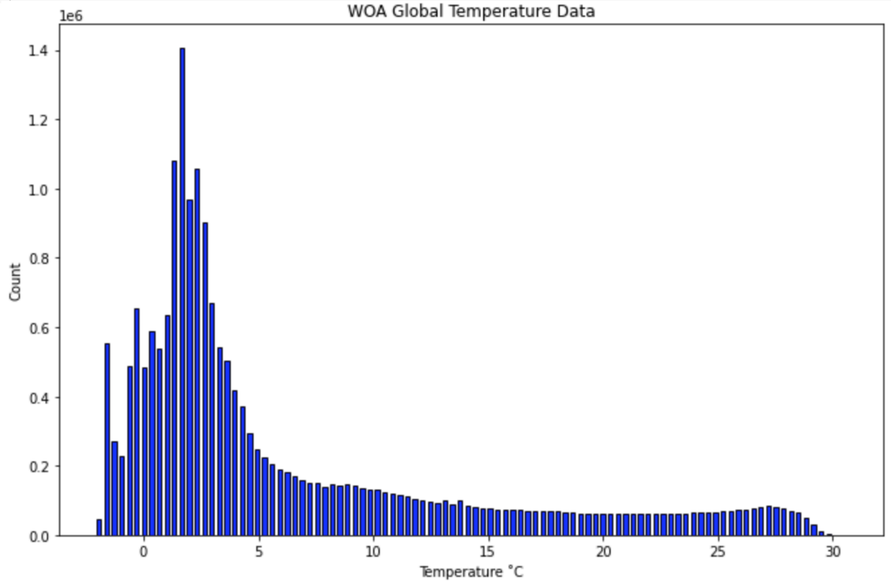
From class #11 activity: WOA temperature dataset

```
filepath = '/content/drive/My Drive/Data_folder/woa18_temp.nc'
temp = xr.open_dataset(filepath)
t_data = temp['t_an'].values.flatten()

```

```
fig = plt.figure(figsize=(11,7))
h = plt.hist(t_data, bins=100, rwidth=0.6, color='b',edgecolor='k', alpha=0.5)
plt.xlabel('Temperature °C')
plt.ylabel('Count')
plt.title('WOA Global Temperature Data')
```




```
fig = plt.figure(figsize=(11,7))
h = plt.hist(t_data, bins=100, rwidth=0.6, color='b',edgecolor='k', alpha=0.9)
plt.xlabel('Temperature °C')
plt.ylabel('Count')
plt.title('WOA Global Temperature Data')
```

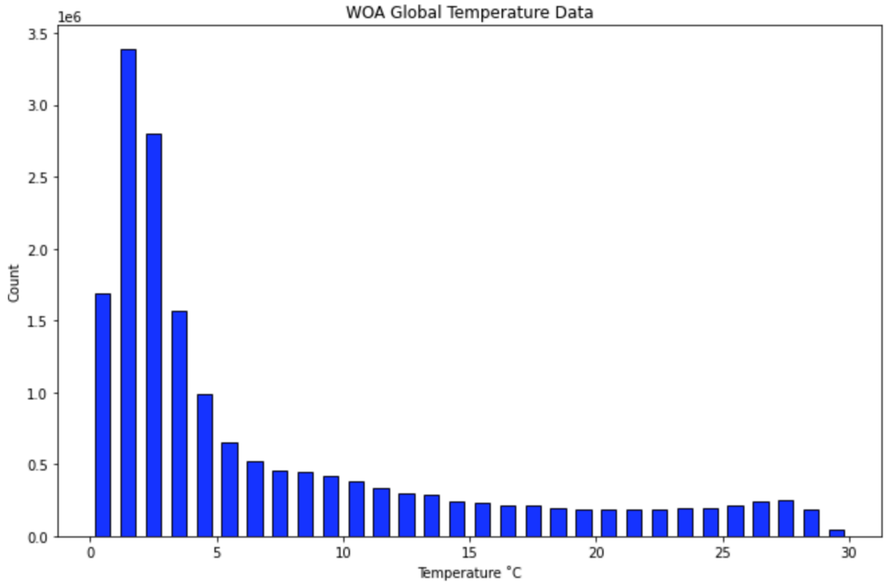


Histograms - example

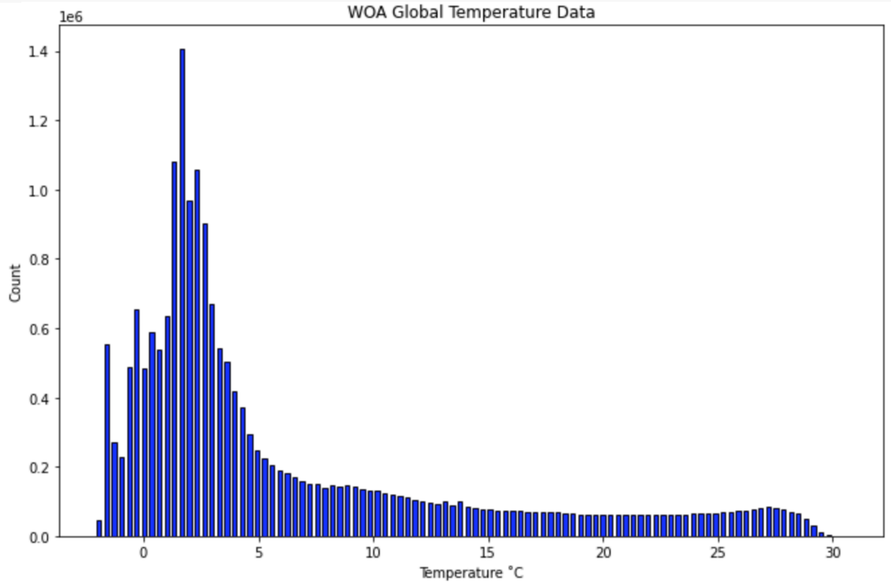
From class #11 activity: WOA dataset

```
filepath = '/content/drive/My Drive/Data_folder/woa18_temp.nc'
temp = xr.open_dataset(filepath)
t_data = temp['t_an'].values.flatten()

```

```
fig = plt.figure(figsize=(11,7))
h = plt.hist(t_data, bins=range(0,31), rwidth=0.6, color='b',edgecolor='k', alpha=0.9)
plt.xlabel('Temperature ˚C')
plt.ylabel('Count')
plt.title('WOA Global Temperature Data')
```




```
fig = plt.figure(figsize=(11,7))
h = plt.hist(t_data, bins=100, rwidth=0.6, color='b',edgecolor='k', alpha=0.9)
plt.xlabel('Temperature ˚C')
plt.ylabel('Count')
plt.title('WOA Global Temperature Data')
```



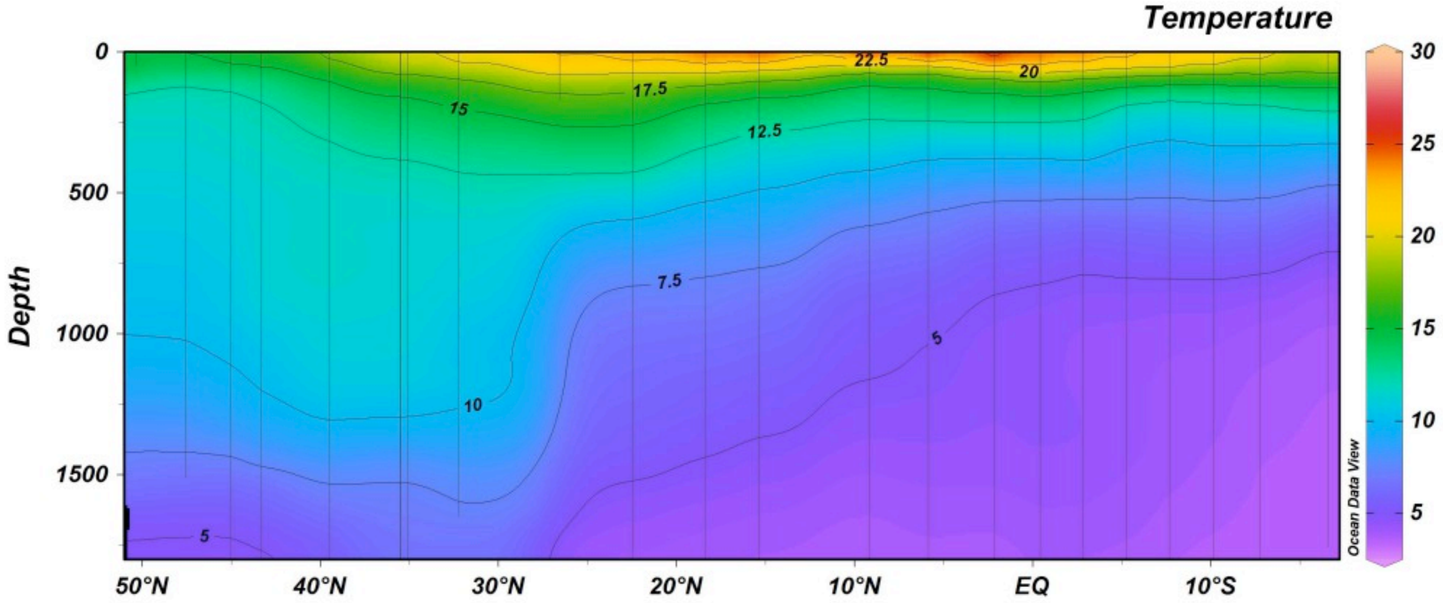
What we'll cover in this lesson

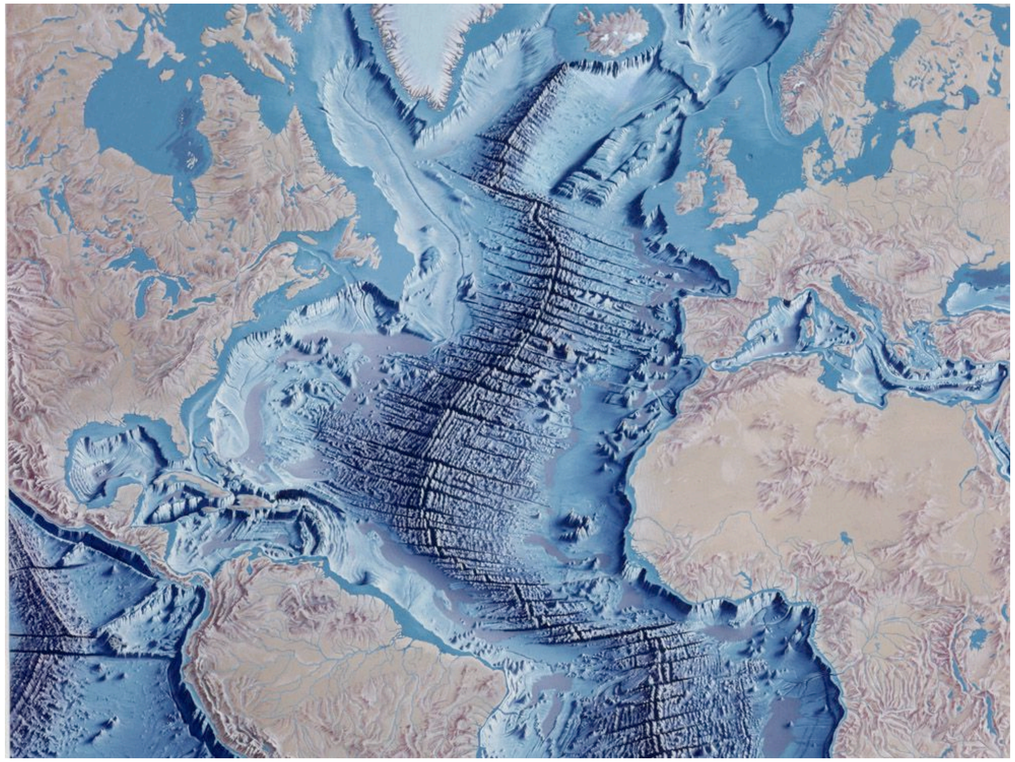
1. Review of plotting concepts

2. **2-D plotting**

3. Mapping with Cartopy

What are 2-dimensional plots?





2-dimensional plots show data in an x-y space

- Maps
- Transects
- Time vs distance evolution (Hoffmuller)

Contour plots

```
plt.contour(x, y, z, levels= , colors= , linewidths= , linestyles= )
```

input: integer or array

or

`cmap=`

These are all optional formatting arguments

Notice that colors, linewidths, and linestyles are plural! (not like with line/scatter plots)

Contour plots are used for showing data (z) on a 2-dimensional plot with corresponding x and y axes.
The length of x corresponds to the number of columns in z
The length of y corresponds to the number of rows in z

Contour plots - example

WOA oxygen dataset

```
1
2 filepath = '/content/drive/My Drive/Data_folder/woa18_oxy.nc'
3 oxy = xr.open_dataset(filepath,decode_times=False)
4
5 display(oxy)
6
7 o_data = oxy['o_an'].mean(dim='lon').isel(time=0).values
8 lat = oxy['lat'].values
9 depth = oxy['depth'].values
10
```

xarray.Dataset

▶ Dimensions: (**depth**: 102, **lat**: 180, **lon**: 360, **time**: 1)

▼ Coordinates:

time	(time)	float32	8214.0
depth	(depth)	float32	0.0 5.0 10.0 ... 5400.0 5500.0
lat	(lat)	float32	-89.5 -88.5 -87.5 ... 88.5 89.5
lon	(lon)	float32	-179.5 -178.5 ... 178.5 179.5

▼ Data variables:

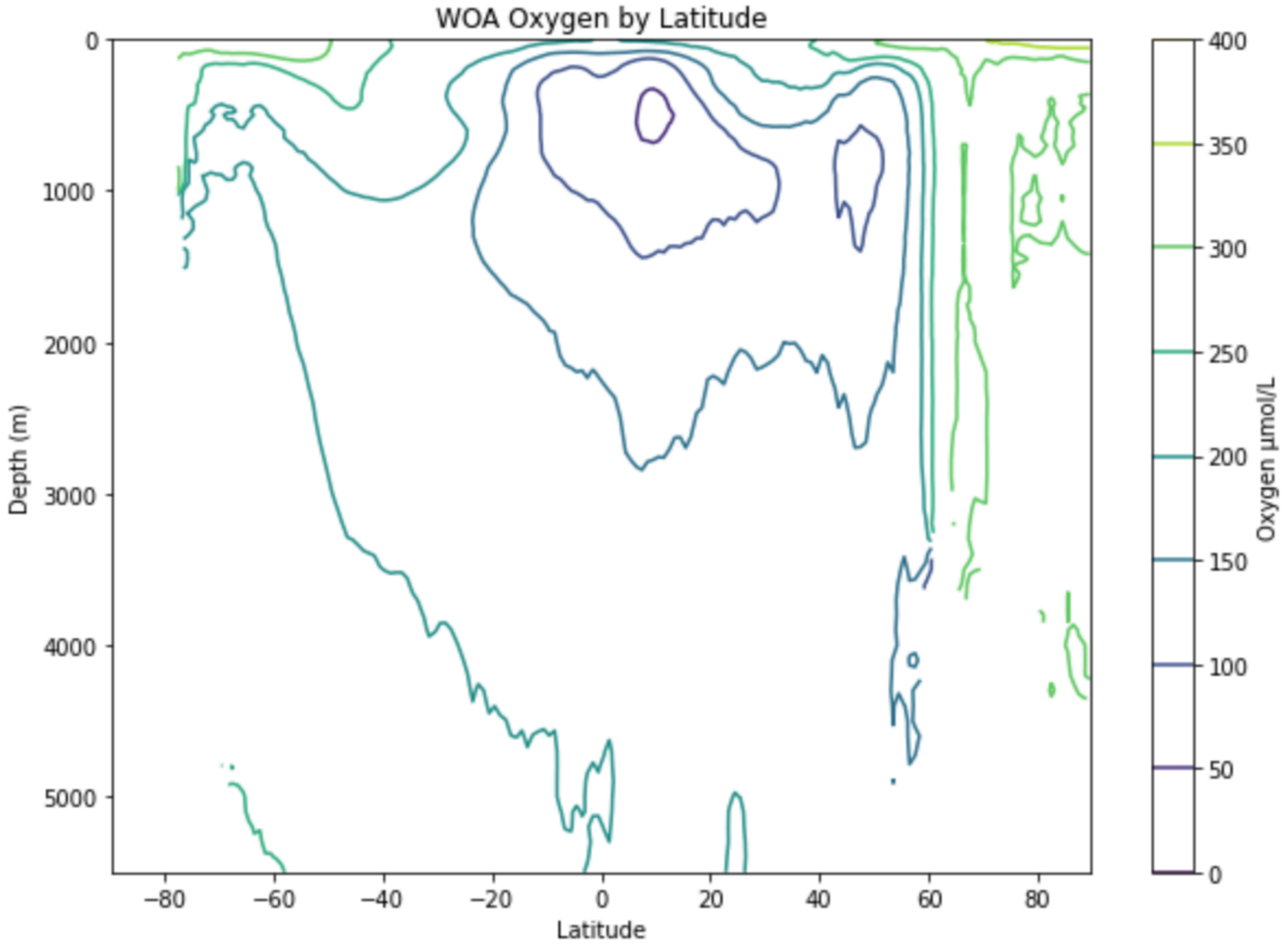
o_an	(time, depth, lat, lon)	float32	...

▶ Attributes: (50)

Contour plots - example

WOA oxygen dataset

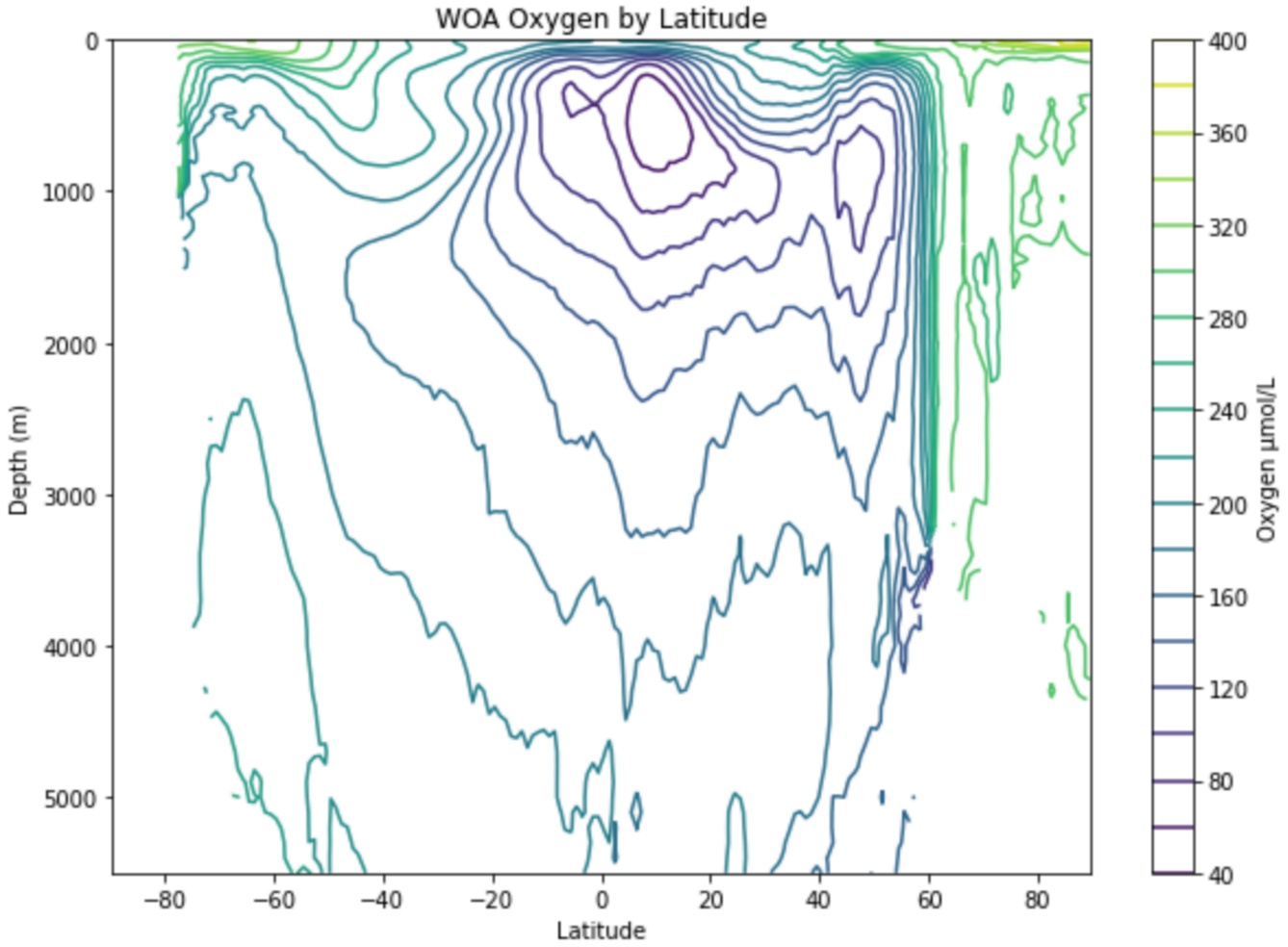
```
fig = plt.figure(figsize=(10,7))
ax = plt.gca()

cntr = plt.contour(lat, depth, o_data)
c = plt.colorbar(cntr, ax=ax)
c.set_label('Oxygen µmol/L')

ax.invert_yaxis()
plt.xlabel('Latitude')
plt.ylabel('Depth (m)')
plt.title('WOA Oxygen by Latitude')
```



Contour plots - example

WOA oxygen dataset

```
fig = plt.figure(figsize=(10,7))
ax = plt.gca()

cntr = plt.contour(lat, depth, o_data, levels=20)
c = plt.colorbar(cntr, ax=ax)
c.set_label('Oxygen µmol/L')

ax.invert_yaxis()
plt.xlabel('Latitude')
plt.ylabel('Depth (m)')
plt.title('WOA Oxygen by Latitude')
```

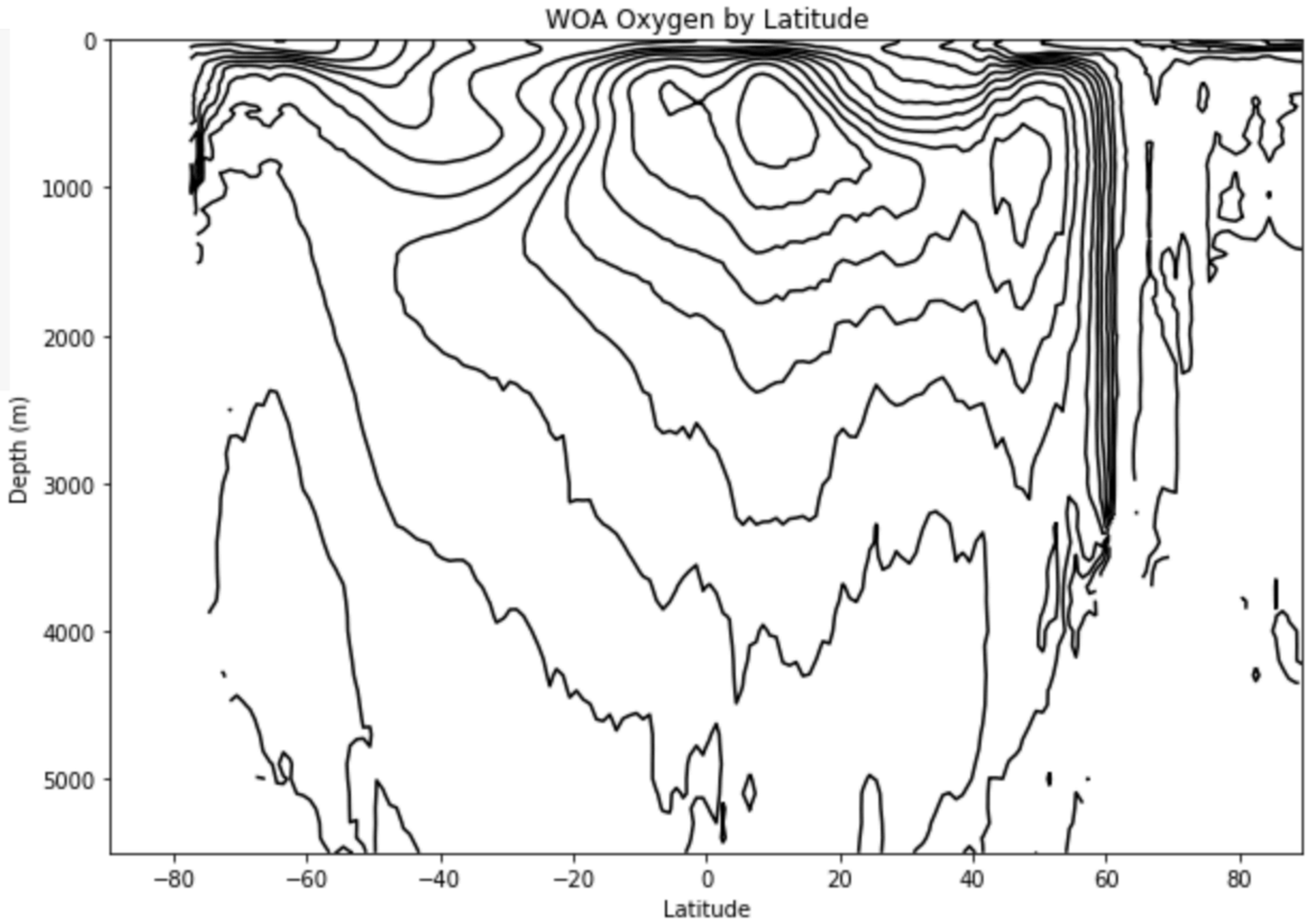


Contour plots - example

WOA oxygen dataset

```
fig = plt.figure(figsize=(10,7))
ax = plt.gca()

cntr = plt.contour(lat, depth, o_data, levels=20, colors='k')

ax.invert_yaxis()
plt.xlabel('Latitude')
plt.ylabel('Depth (m)')
plt.title('WOA Oxygen by Latitude')
```

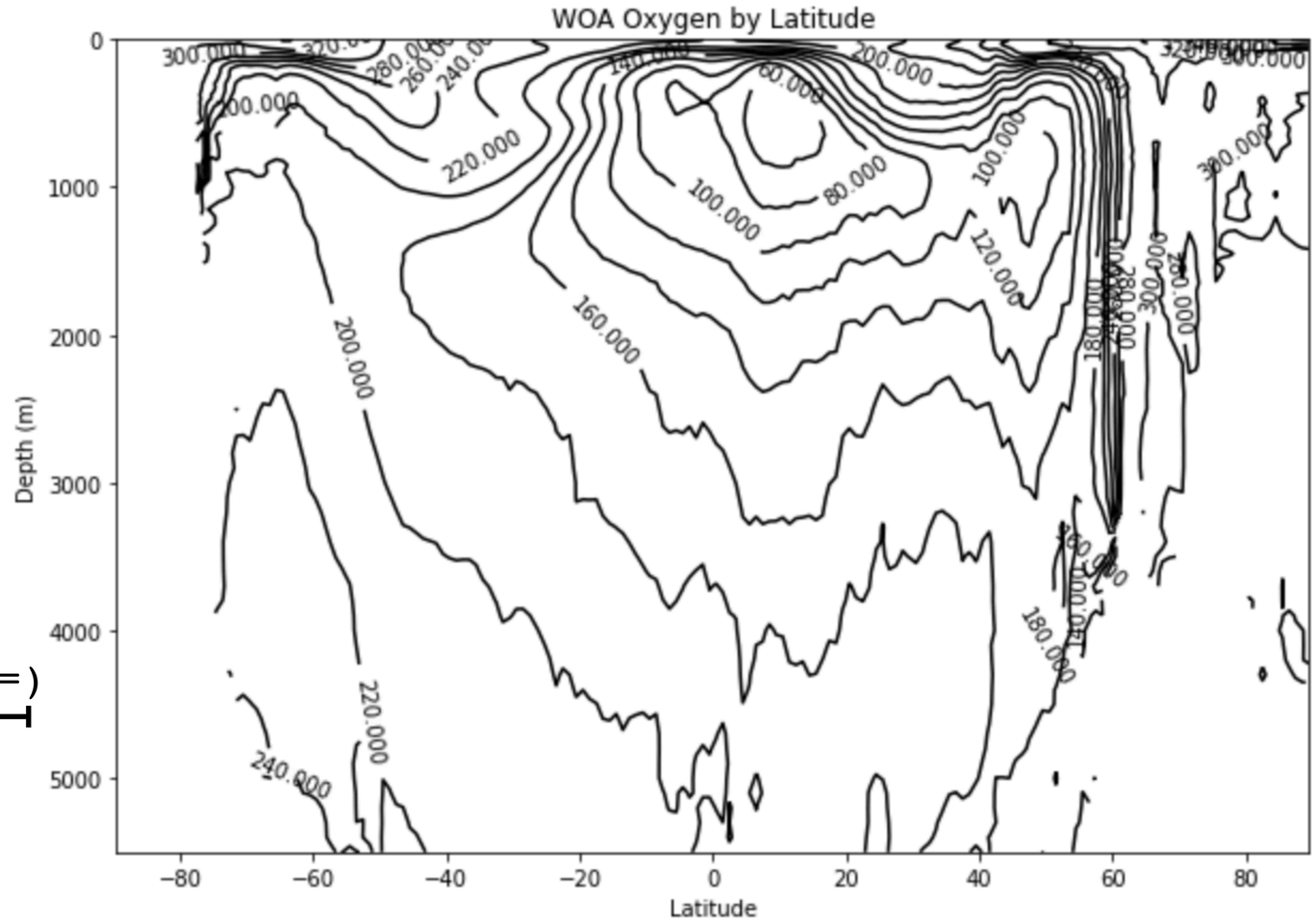
Now we don't know what the contours mean!



Contour plots - example

WOA oxygen dataset

```
2 fig = plt.figure(figsize=(10,7))
3 ax = plt.gca()
4
5 cntr = plt.contour(lat, depth, o_data, levels=20, colors='k')
6
7 ax.invert_yaxis()
8 plt.xlabel('Latitude')
9 plt.ylabel('Depth (m)')
10 plt.title('WOA Oxygen by Latitude')
11
12 plt.clabel(cntr)
```



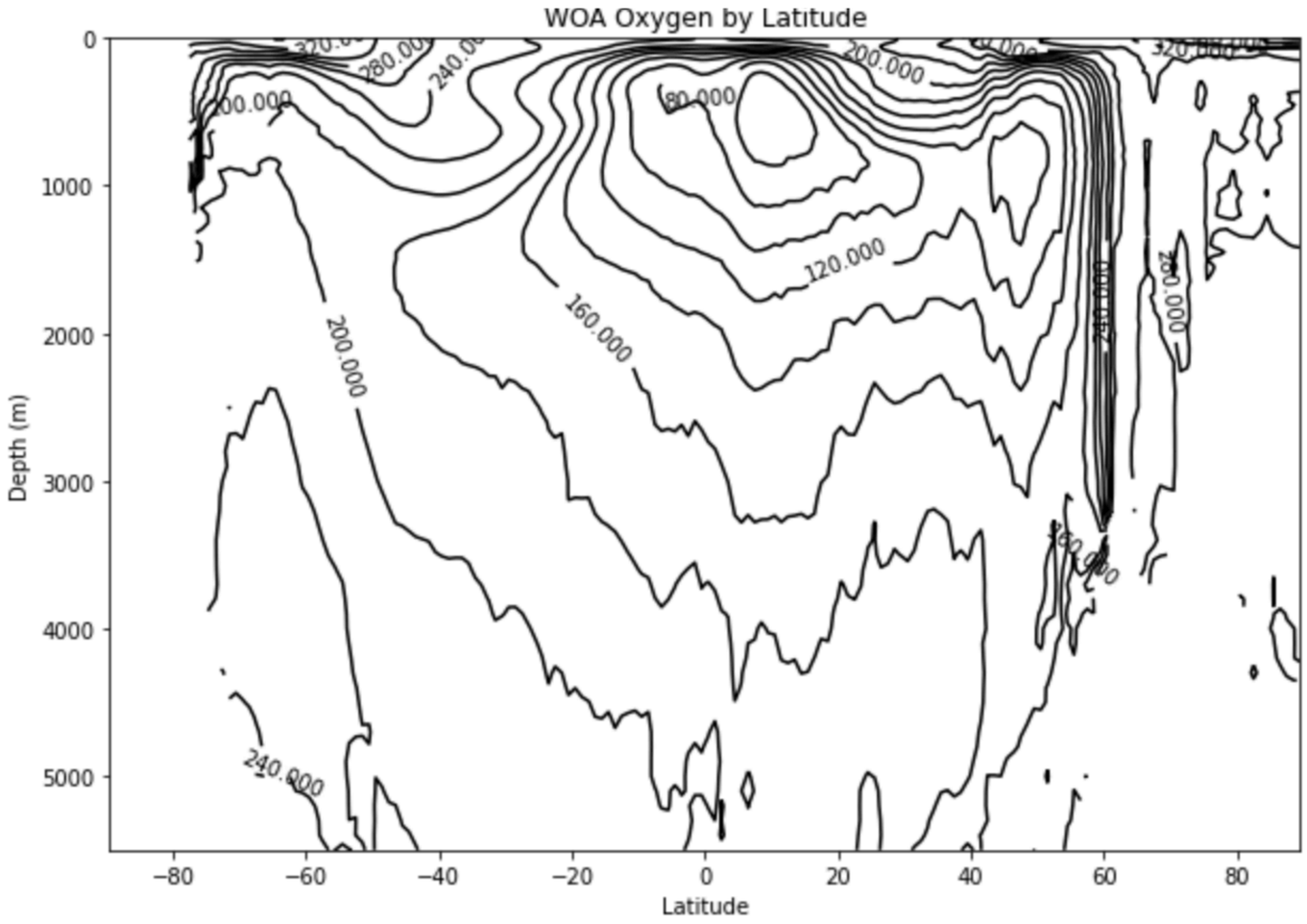
`plt.clabel(`plot variable`, levels=, fontsize=, colors=)`

Optional

Contour plots - example

WOA oxygen dataset

```
fig = plt.figure(figsize=(10,7))
ax = plt.gca()

cntr = plt.contour(lat, depth, o_data, levels=20, colors='k')

ax.invert_yaxis()
plt.xlabel('Latitude')
plt.ylabel('Depth (m)')
plt.title('WOA Oxygen by Latitude')

plt.clabel(cntr, levels=cntr.levels[::2])
```



Contourf plots

```
plt.contour(x, y, z, levels= , colors= , linewidths= , linestyles= )
```

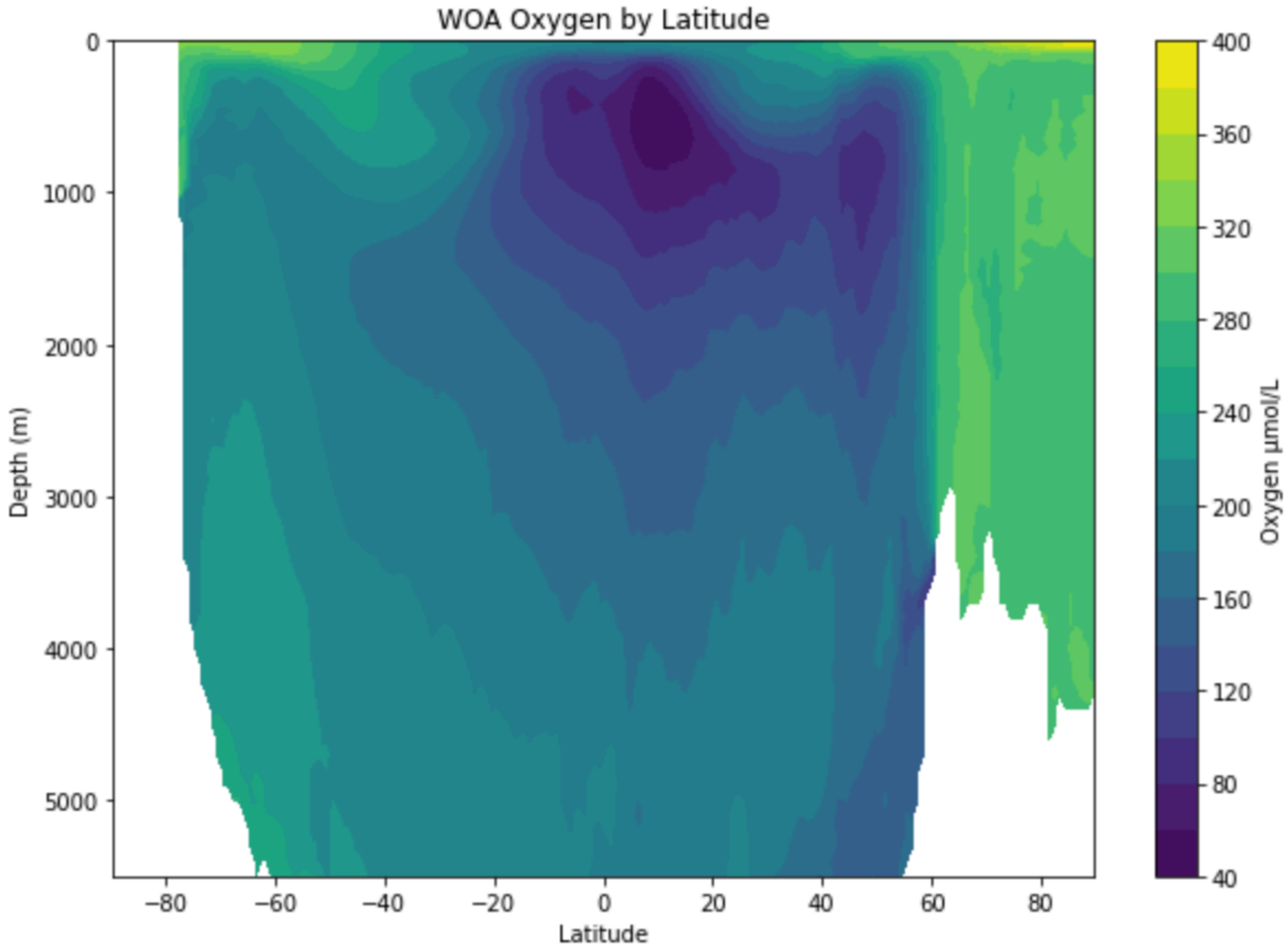
```
plt.contourf(x, y, z, levels= , cmap= )
```

Contourf plots fill the spaces between the contour lines using the colormap.

Contour plots - example

WOA oxygen dataset

```
fig = plt.figure(figsize=(10,7))
ax = plt.gca()

cntr = plt.contourf(lat, depth, o_data, levels=20)
c = plt.colorbar(cntr, ax=ax)
c.set_label('Oxygen µmol/L')

ax.invert_yaxis()
plt.xlabel('Latitude')
plt.ylabel('Depth (m)')
plt.title('WOA Oxygen by Latitude')
```



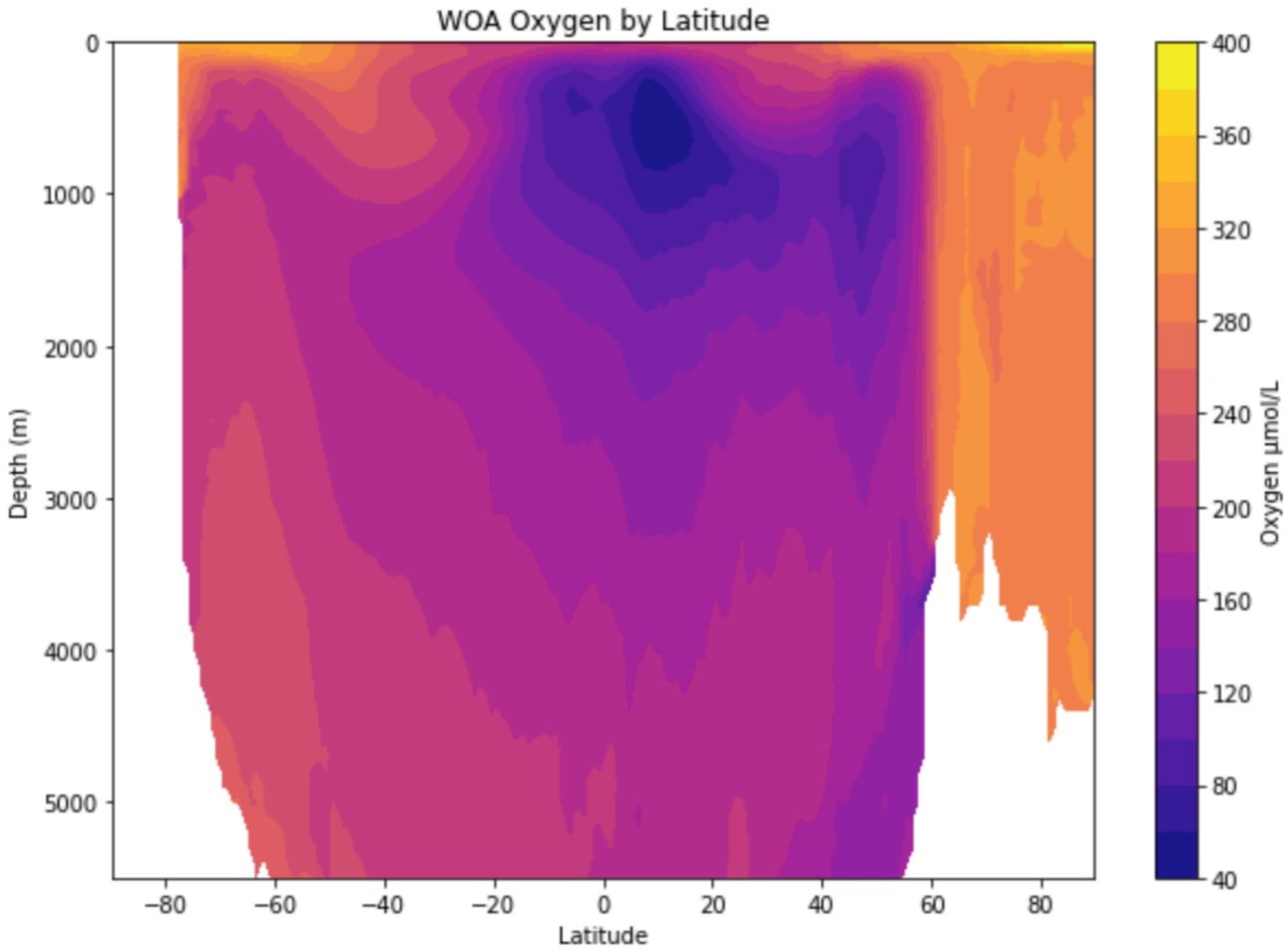
Now we need our colorbar back!

Contour plots - example

WOA oxygen dataset

```
fig = plt.figure(figsize=(10,7))
ax = plt.gca()

cntr = plt.contourf(lat, depth, o_data, levels=20,cmap='plasma')
c = plt.colorbar(cntr, ax=ax)
c.set_label('Oxygen µmol/L')

ax.invert_yaxis()
plt.xlabel('Latitude')
plt.ylabel('Depth (m)')
plt.title('WOA Oxygen by Latitude')
```

Change the color using the cmap argument

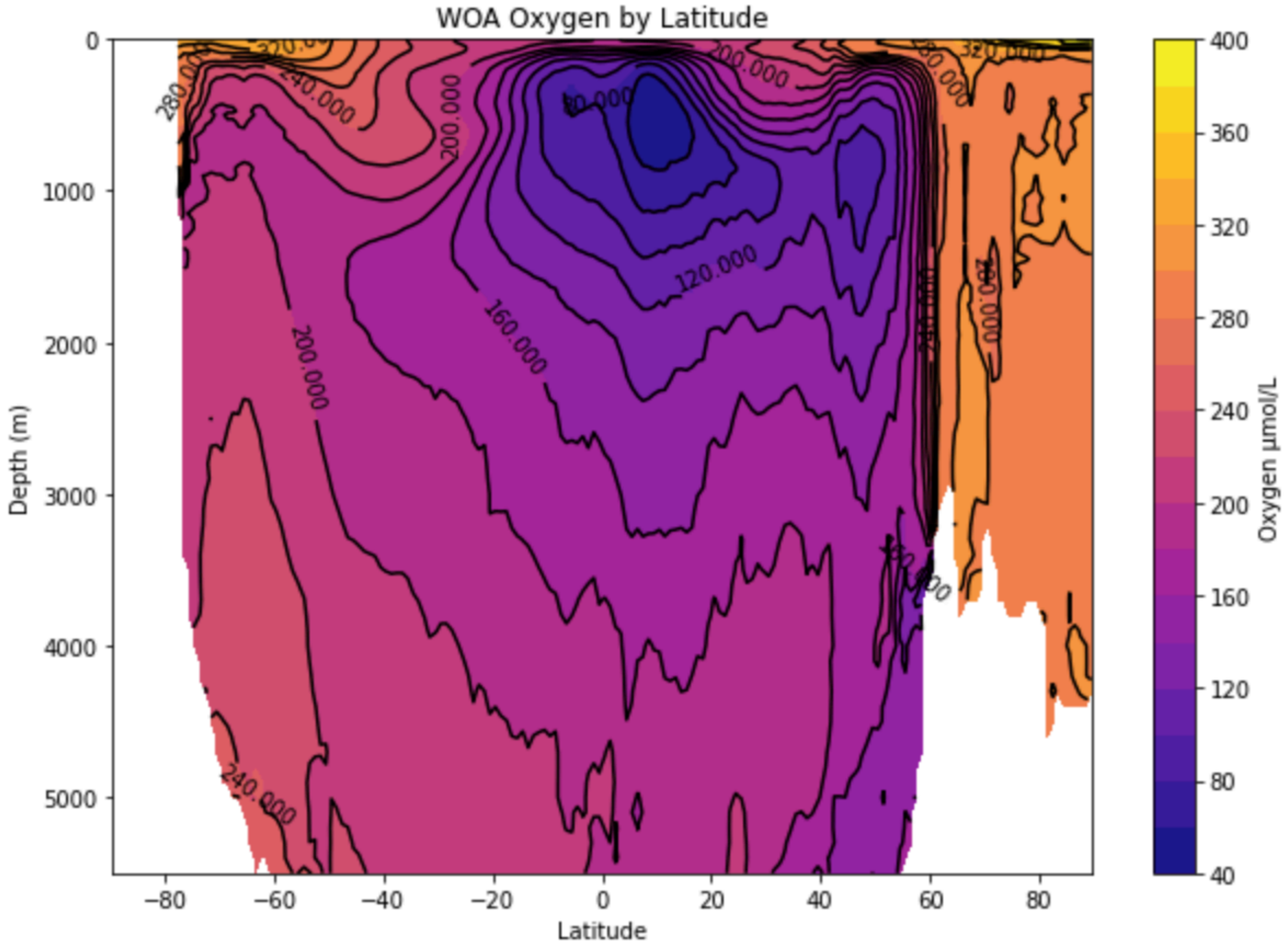


Contour plots - example

WOA oxygen dataset

```
fig = plt.figure(figsize=(10,7))
ax = plt.gca()

cntr = plt.contourf(lat, depth, o_data, levels=20,cmap='plasma')
cntr_lines = plt.contour(lat, depth, o_data, levels=20,colors='k')
c = plt.colorbar(cntr, ax=ax)
c.set_label('Oxygen µmol/L')

ax.invert_yaxis()
plt.xlabel('Latitude')
plt.ylabel('Depth (m)')
plt.title('WOA Oxygen by Latitude')

plt.clabel(cntr_lines, levels=cntr.levels[::2])
```

Put labels on the filled plot by plotting another contour on top and using clabel

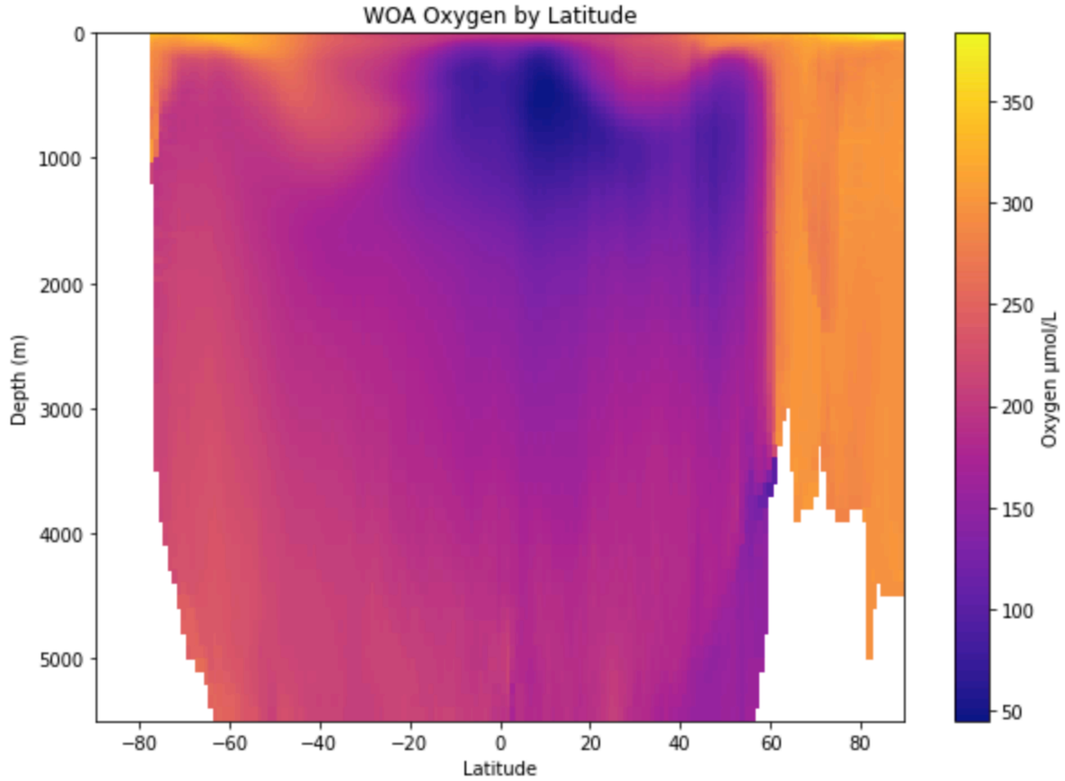


Pseudo-color plots

```
plt.pcolormesh(x, y, z, cmap= , linewidths= , edgecolor= )
```

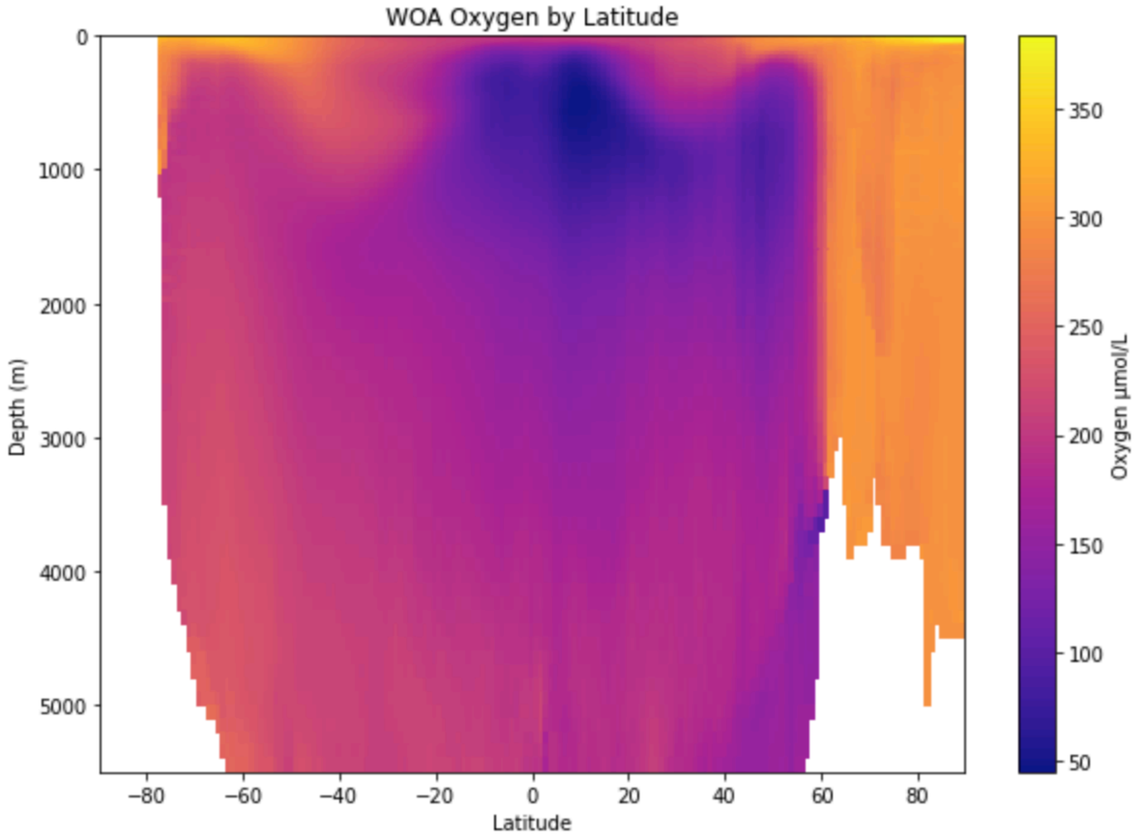
These are all optional formatting arguments

```
fig = plt.figure(figsize=(10,7))
ax = plt.gca()

cntr = plt.pcolormesh(lat, depth, o_data, cmap='plasma')
c = plt.colorbar(cntr, ax=ax)
c.set_label('Oxygen µmol/L')

ax.invert_yaxis()
plt.xlabel('Latitude')
plt.ylabel('Depth (m)')
plt.title('WOA Oxygen by Latitude')
```



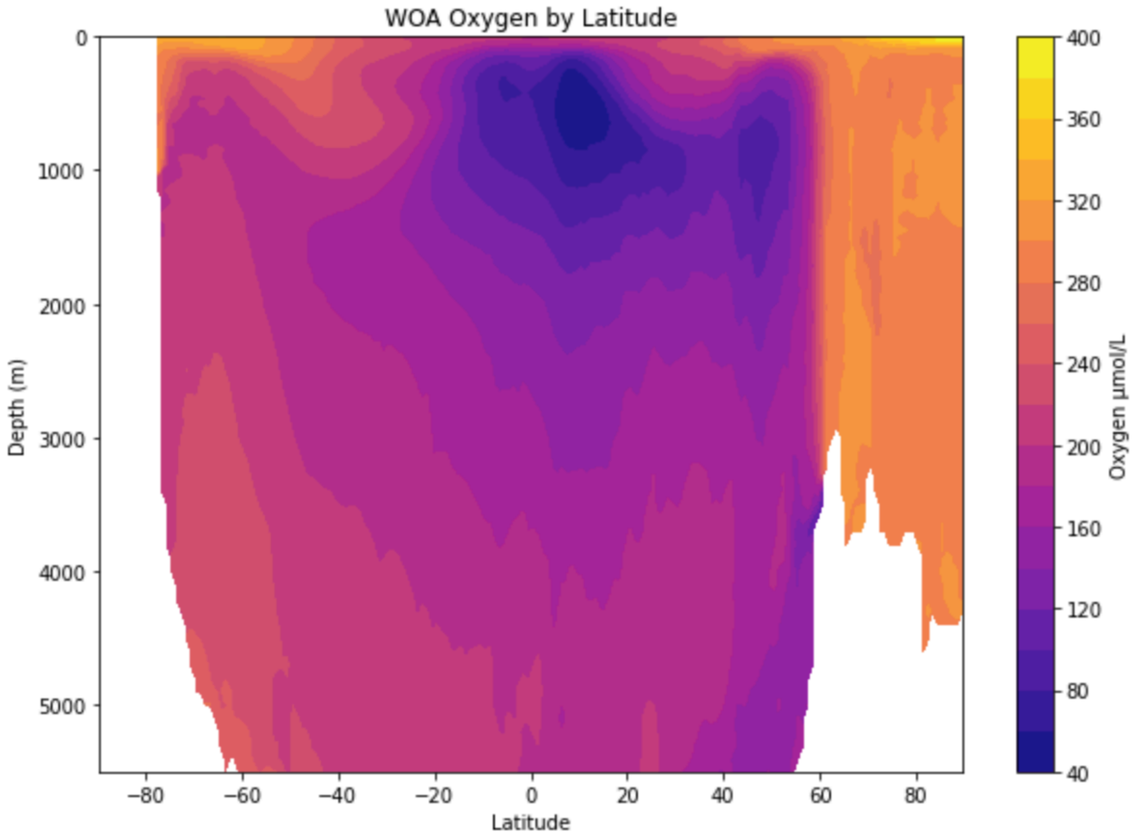
Difference between `contourf` and `pcolormesh`

`pcolormesh()`



Grid point estimates
Better for noisy data

`contourf()`



Interpolate between isolines
Better for large scale patterns

What we'll cover in this lesson

1. Review of plotting concepts

2. 2-D plotting

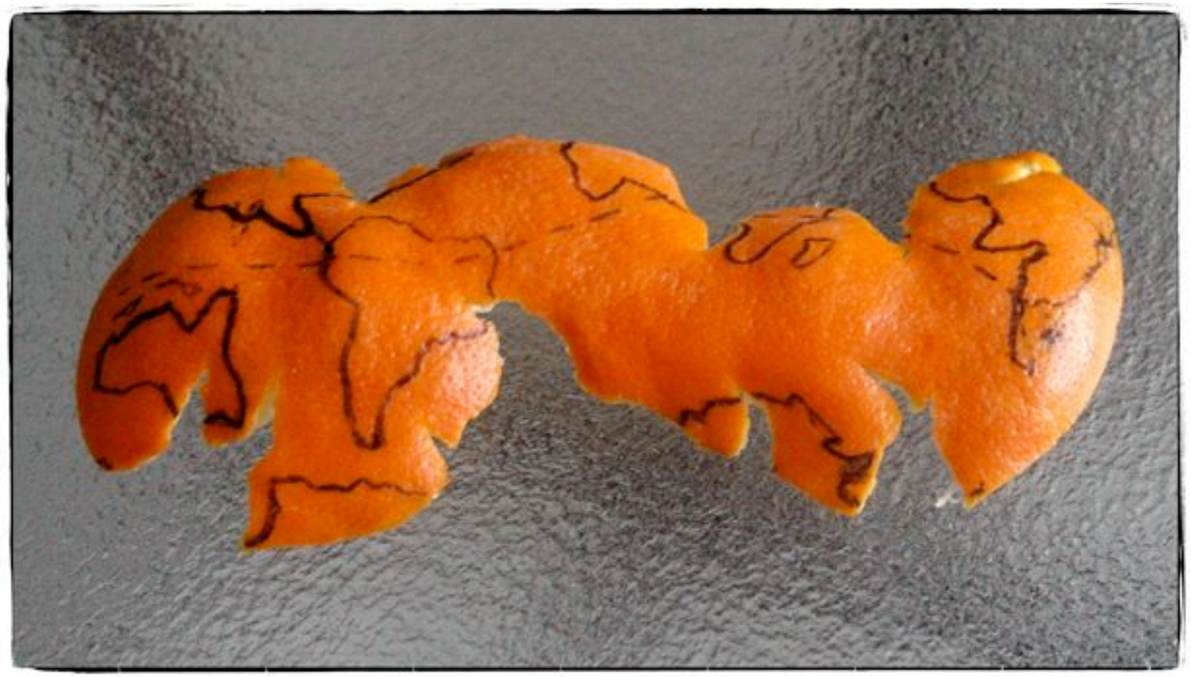
3. **Mapping with Cartopy**

Every map is wrong in some way...

https://www.scienceabc.com/social-science/what-is-wrong-with-all-our-maps-mercator-maps.html

Every map is wrong in some way...

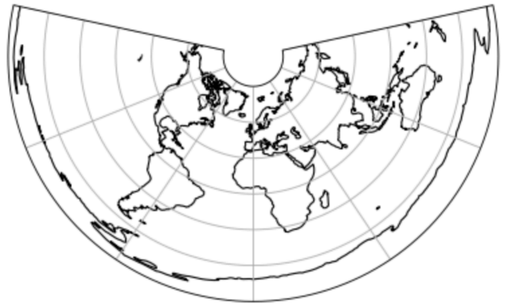
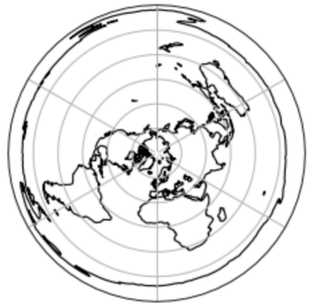
The surface of a sphere is different to a 2D surface, so we have to cut the sphere somewhere.

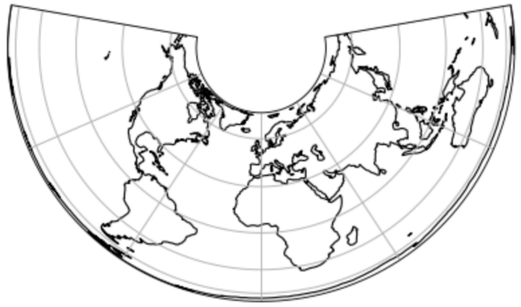


The surface cannot be represented on a plane without distortion.

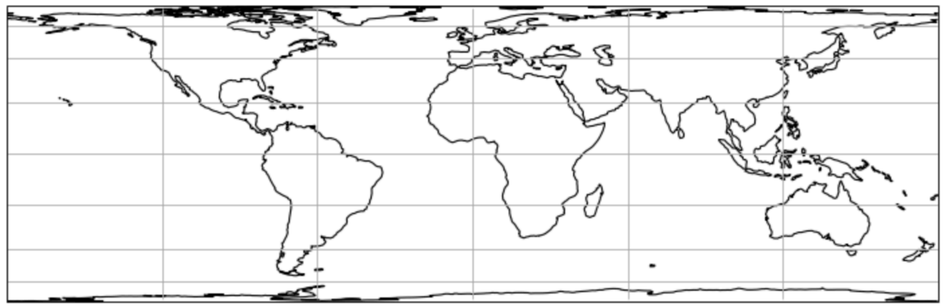
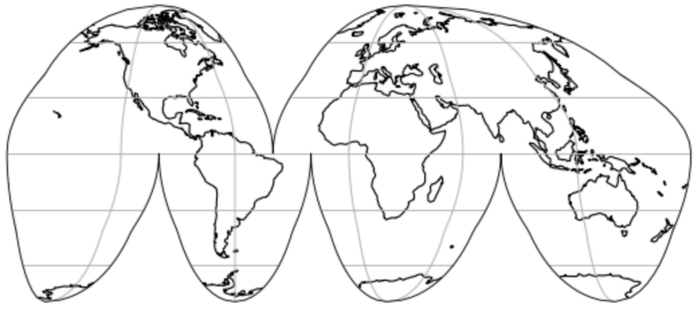
https://rabernat.github.io/research_computing_2018/maps-with-cartopy.html

Using projections to visualize geographic data

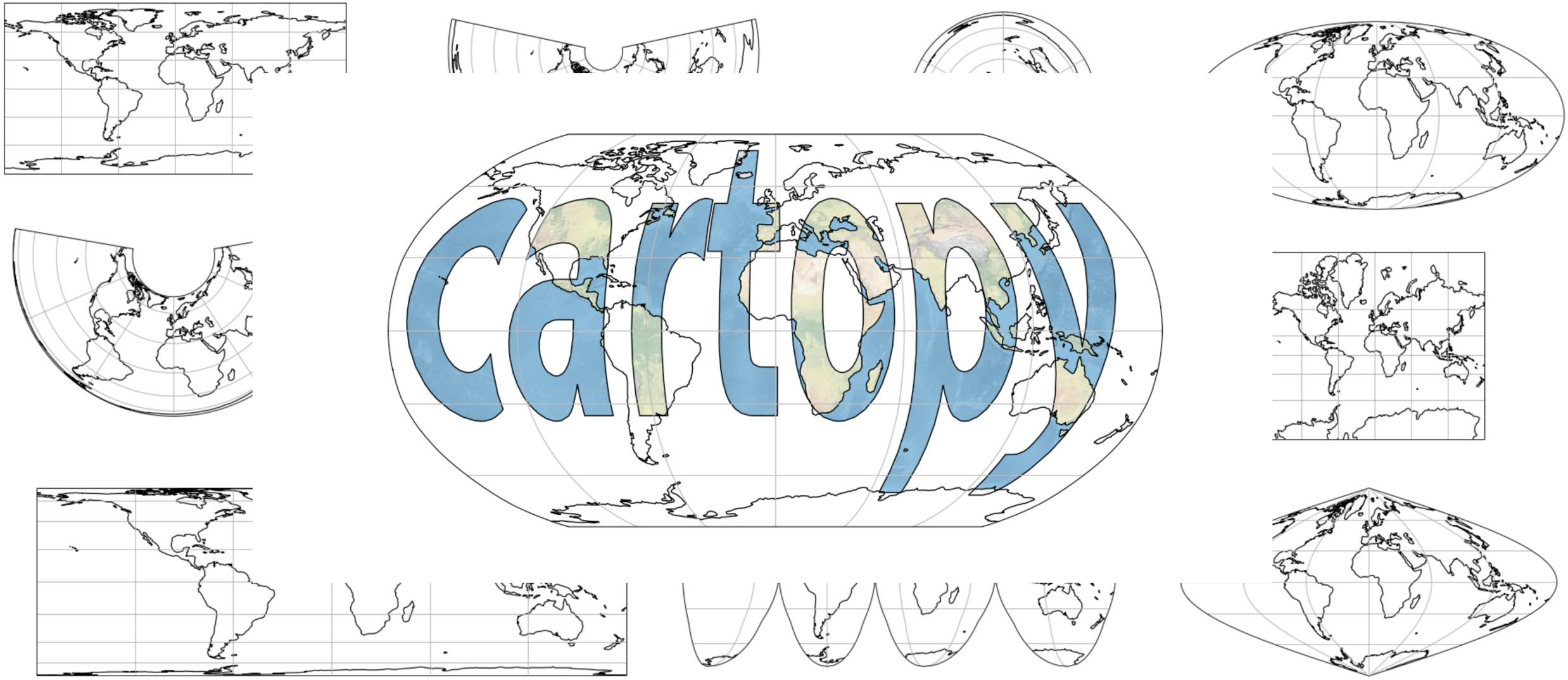









Using projections to visualize geographic data



Using projections to visualize geographic data

<class 'cartopy.crs.PlateCarree'>

<class 'cartopy.crs.Mercator'>

<class 'cartopy.crs.InterruptedGoodeHomolosine'>

<class 'cartopy.crs.Robinson'>

<class 'cartopy.crs.Orthographic'>

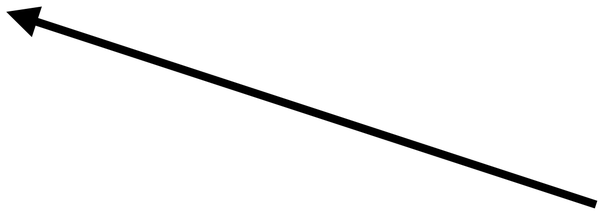
Full list of projections available:

https://scitools.org.uk/cartopy/docs/latest/crs/projections.html

Loading `cartopy`

Cartopy is not readily available in the Google Colab environment, so we need to load it into our notebook before importing.

```
!apt-get -qq install python-cartopy python3-cartopy
!pip uninstall -y shapely
!pip install shapely --no-binary shapely
```



You should only have to run
these lines once per Colab notebook.

Loading `cartopy`

Instead of importing all of Cartopy, we will only be importing select parts

```
import cartopy.crs as ccrs
import cartopy.feature as cfeature
from cartopy.mpl.gridliner import LONGITUDE_FORMATTER, LATITUDE_FORMATTER
```

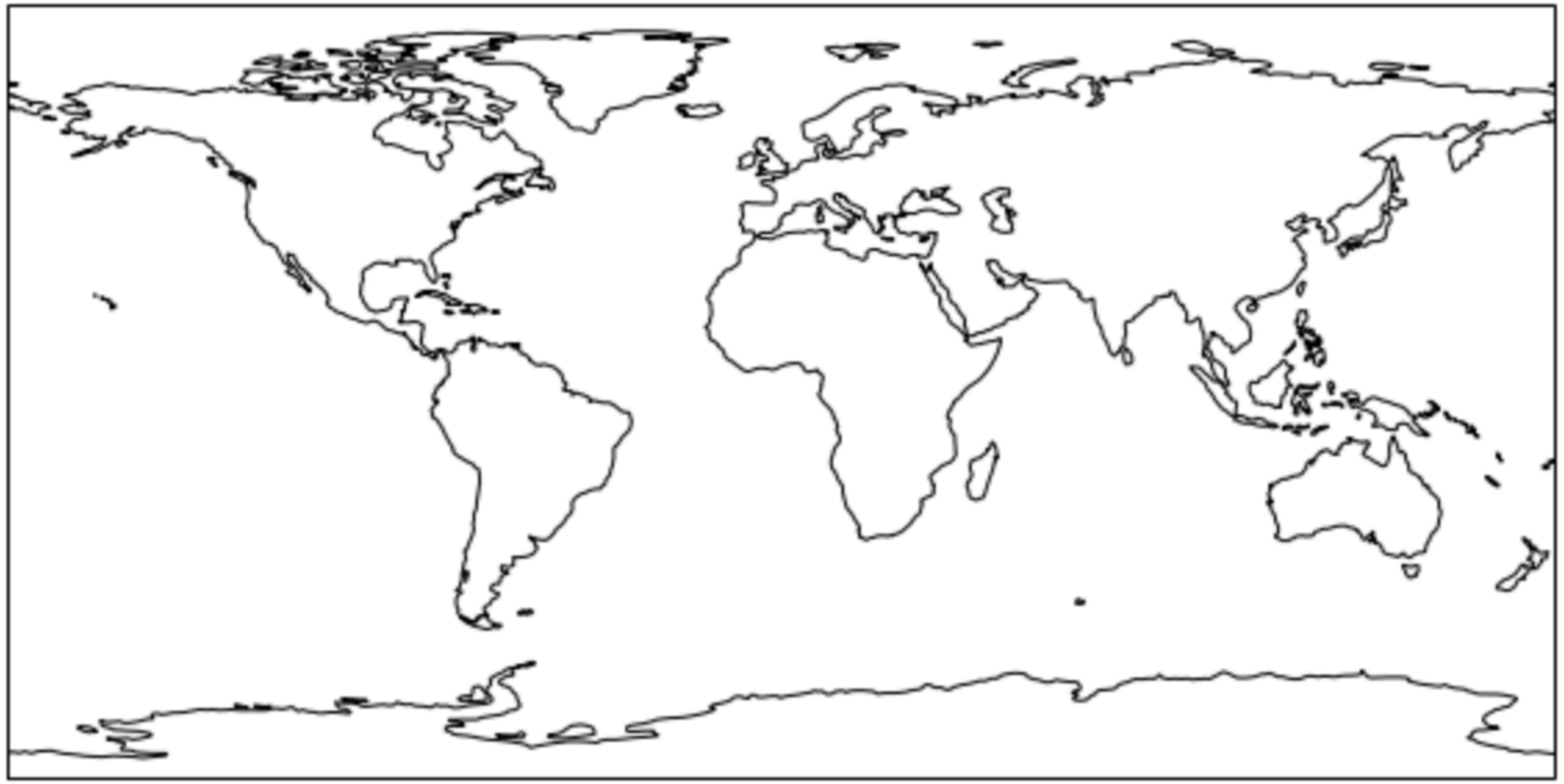
crs - helps with projections
feature - creates map features (e.g. land, rivers, borders)
gridliner - formats our axis and tick labels

Let's start with a coast

```
1     fig = plt.figure(figsize=(10,10))
2     ax = plt.axes(projection=ccrs.PlateCarree())
3     ax.coastlines(resolution='110m',color='k')
```

Optional arguments

```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7f043ec2c780>
```

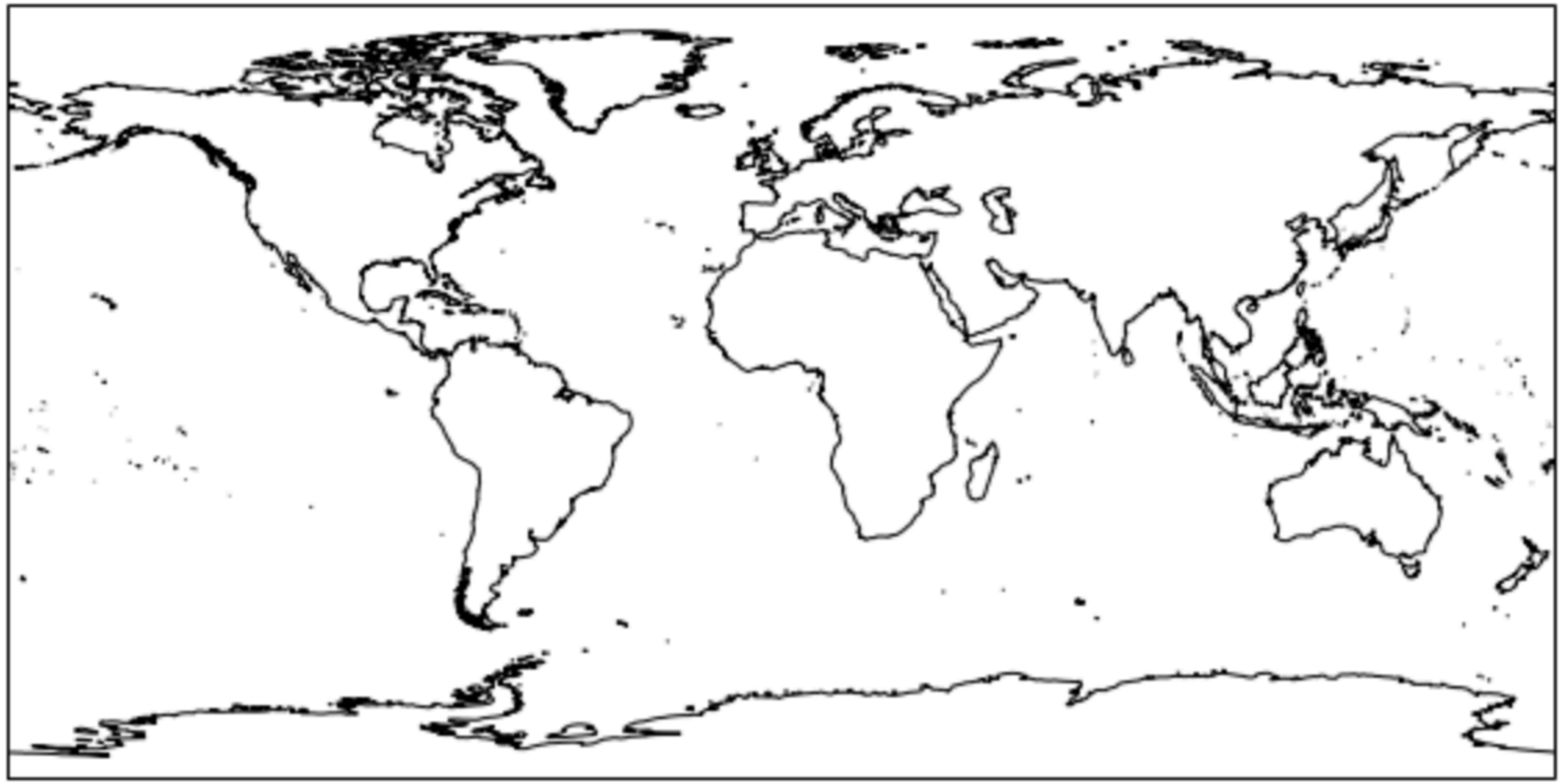


Low resolution: 110m

Let's start with a coast

```
1     fig = plt.figure(figsize=(10,10))
2     ax = plt.axes(projection=ccrs.PlateCarree())
3     ax.coastlines(resolution='50m',color='k')
```

```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7f043d9a77b8>
```

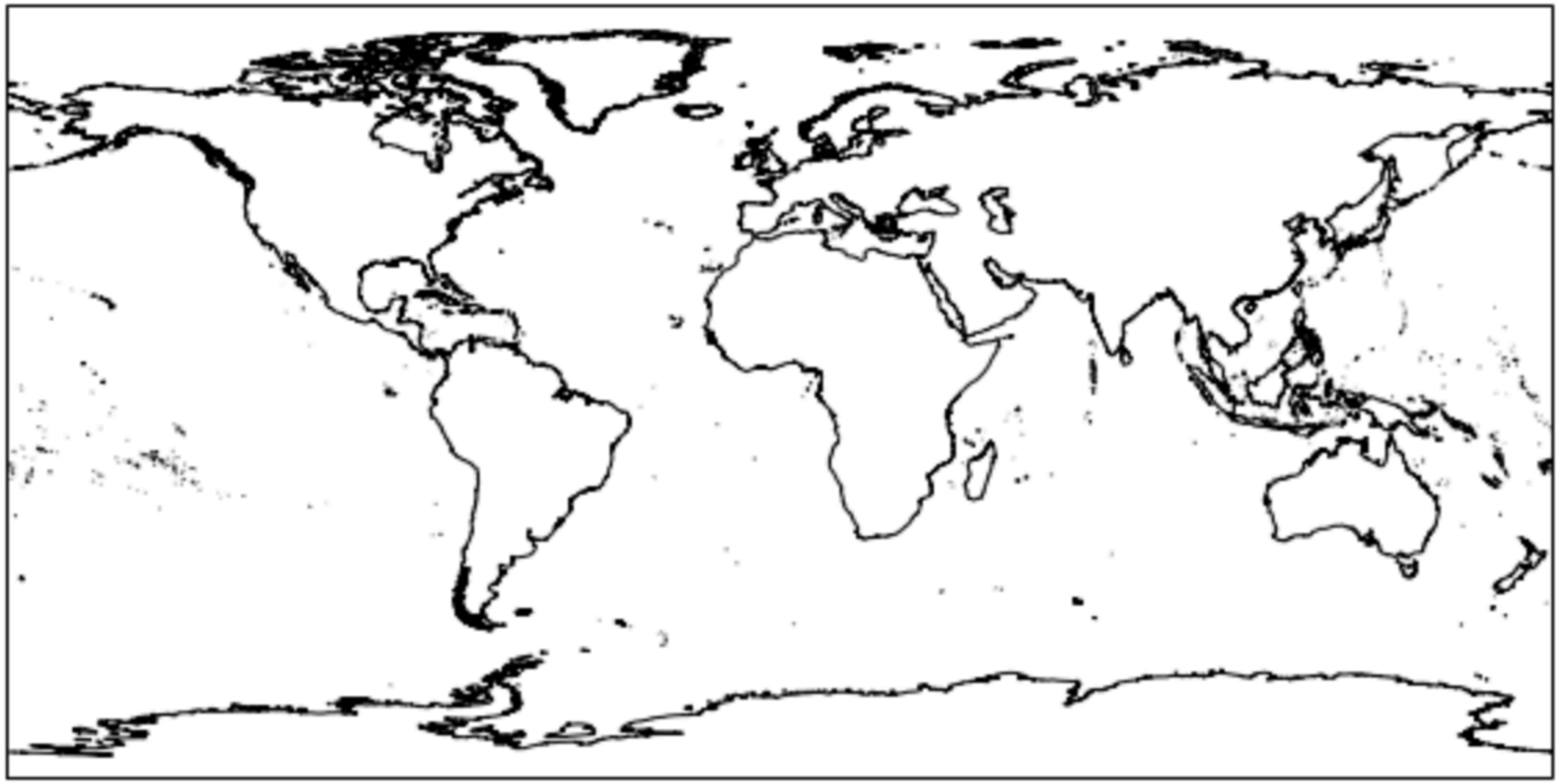


Med resolution: 50m

Let's start with a coast

```
1     fig = plt.figure(figsize=(10,10))
2     ax = plt.axes(projection=ccrs.PlateCarree())
3     ax.coastlines(resolution='10m',color='k')
```

```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7f043d5c1400>
```



High resolution: 10m

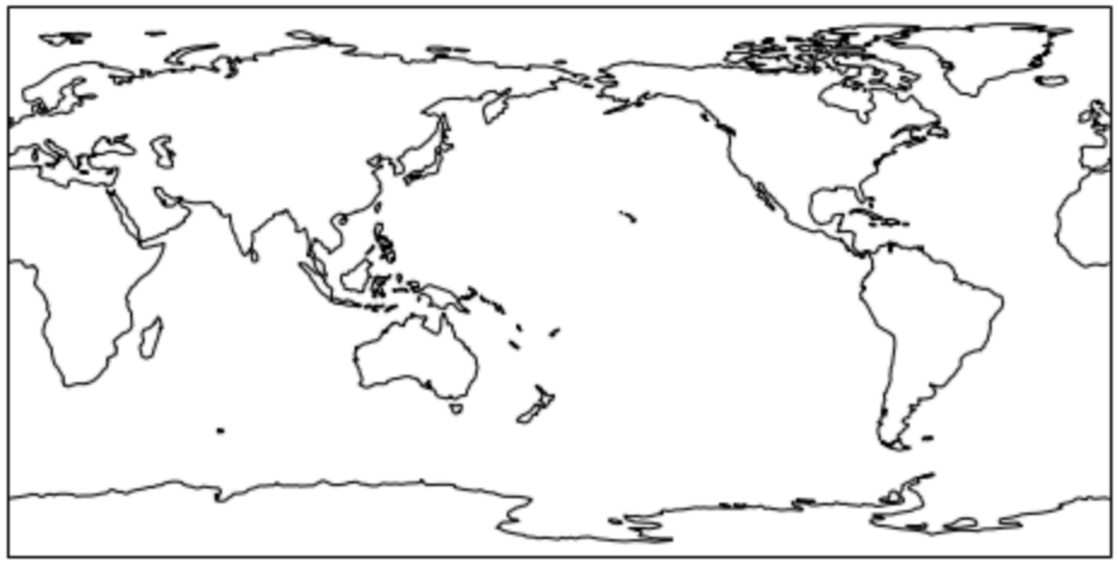
Let's start with a coast

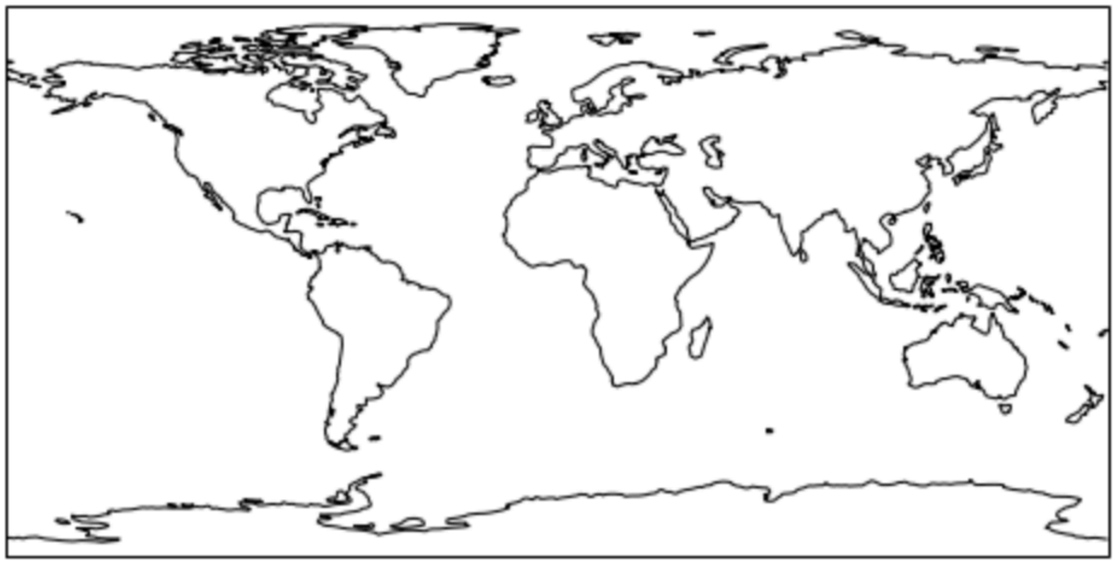
```
1 fig= plt.figure(figsize=(18,12))
2
3 ax1 = fig.add_subplot(1,2,1,projection=ccrs.PlateCarree(central_longitude=180))
4 ax1.coastlines()
5
6 ax2 = fig.add_subplot(1,2,2,projection=ccrs.PlateCarree())
7 ax2.coastlines()
```

changes map center

fig.add_subplot allows you to add more axes on a figure after it's already created

```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7f0430f6c208>
```



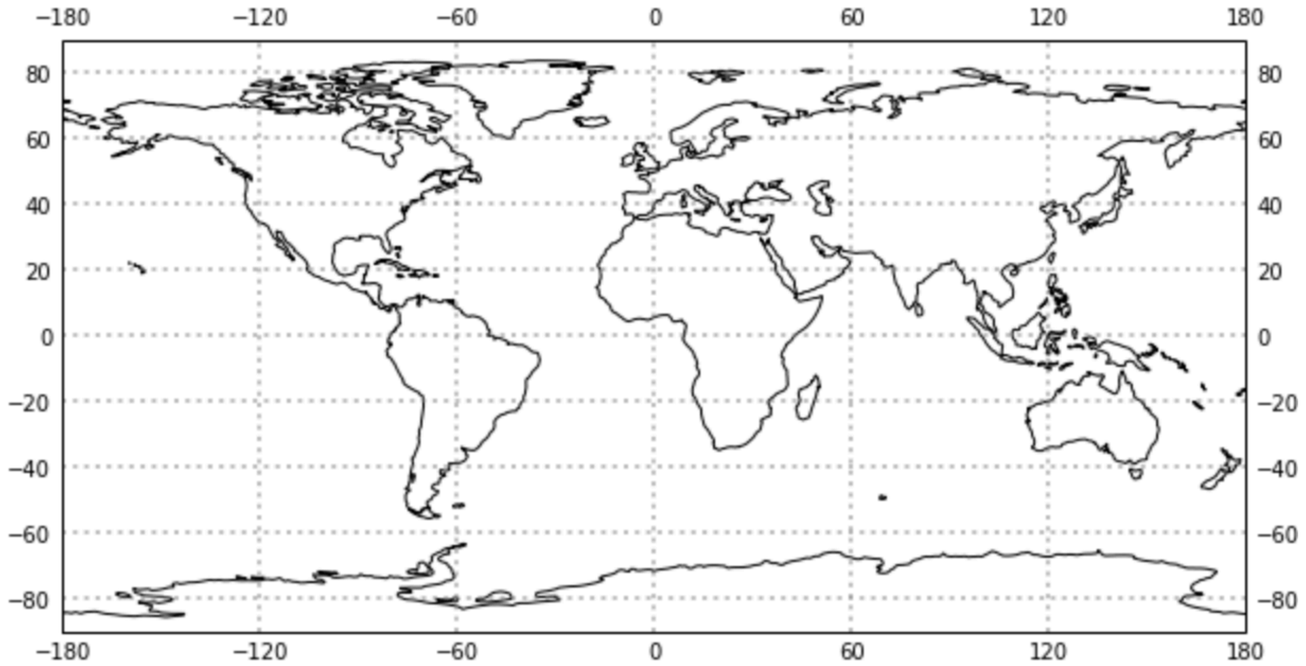


Let's add a grid

```
fig = plt.figure(figsize=(10,10))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution='110m',color='k')

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True,
                  linewidth=2, color='gray', alpha=0.5, linestyle=':')

```

Gridlines arguments

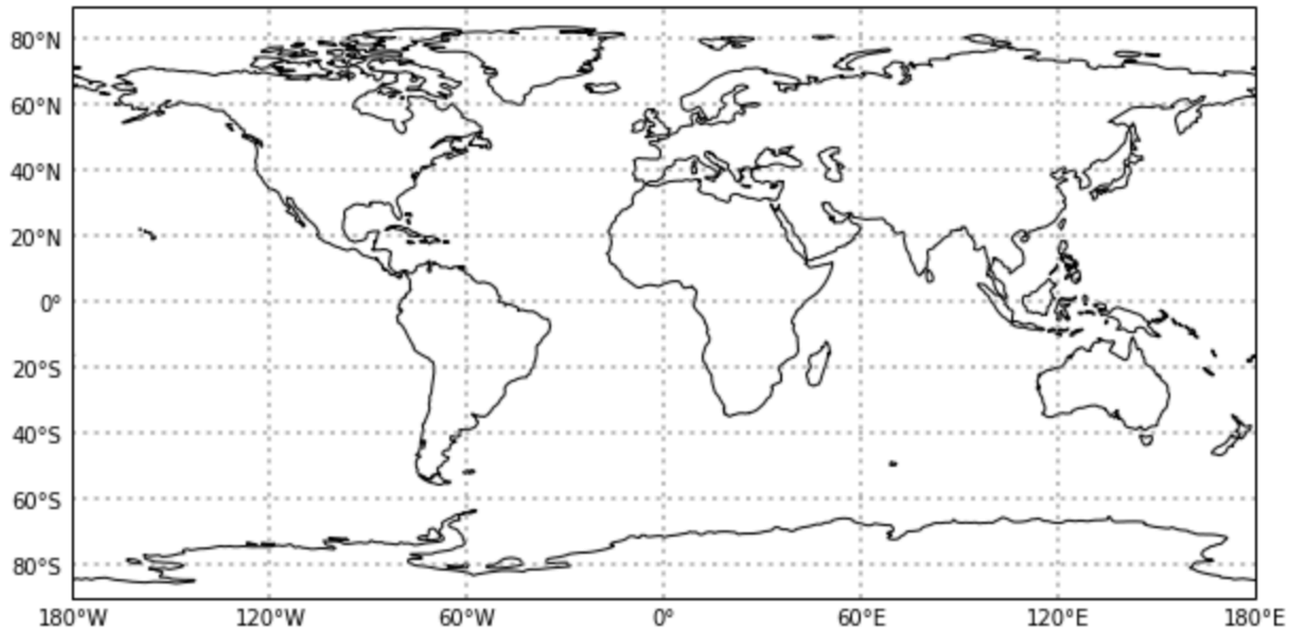
- Projection
- draw_labels
- Line formatting (lw, c, alpha, ls, etc.)



Let's format our grid

```
fig = plt.figure(figsize=(10,10))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution='110m',color='k')

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True,
                  linewidth=2, color='gray', alpha=0.5, linestyle=':')

gl.xlabels_top = False
gl.ylabels_right = False
gl.xformatter = LONGITUDE_FORMATTER
gl.yformatter = LATITUDE_FORMATTER
```

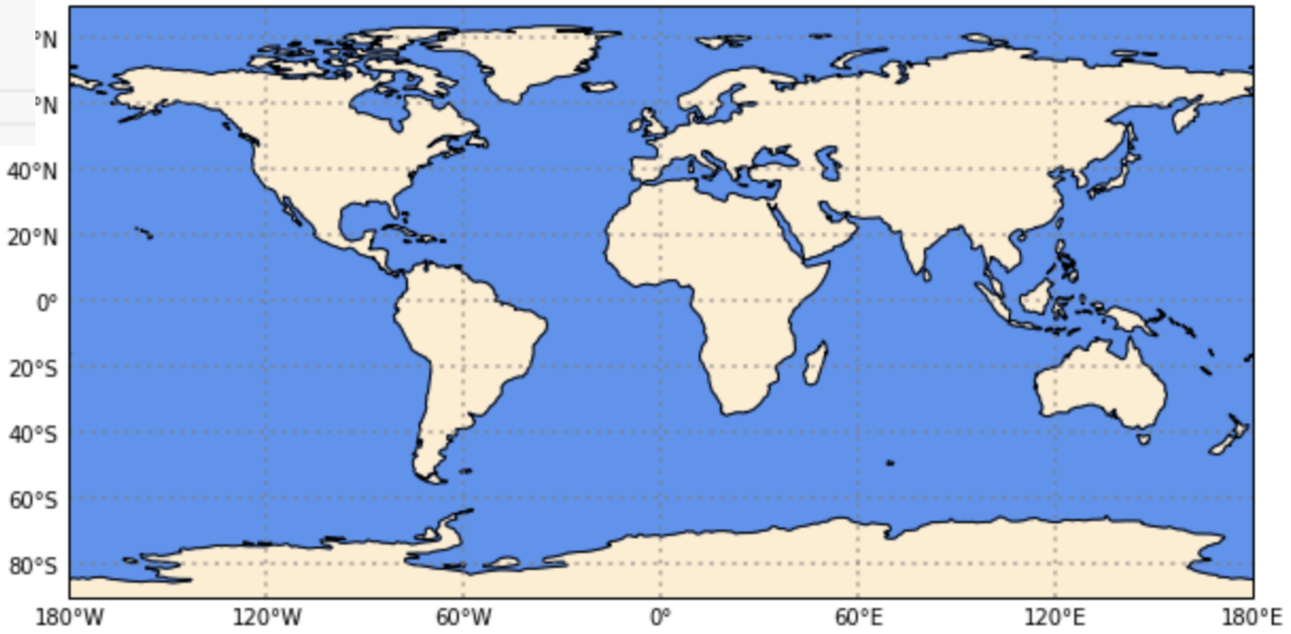
- Removed the upper and right labels
- Formatted the tick labels into ° E/W



Let's add some features

```
fig = plt.figure(figsize=(10,10))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution='110m',color='k')

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True,
                  linewidth=2, color='gray', alpha=0.5, linestyle=':')

gl.xlabels_top = False
gl.ylabels_right = False
gl.xformatter = LONGITUDE_FORMATTER
gl.yformatter = LATITUDE_FORMATTER

ax.add_feature(cfeature.LAND, color='papayawhip')
ax.add_feature(cfeature.OCEAN, color='cornflowerblue')
```

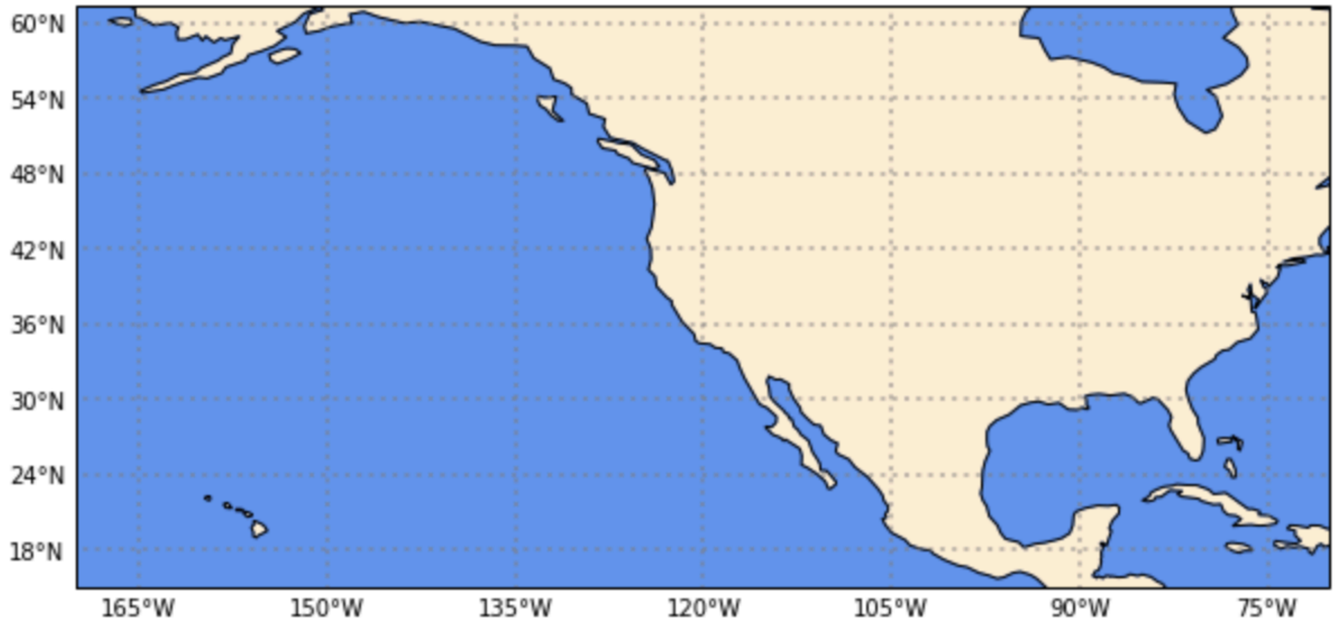
- Add features using cfeature (110m resolution)
- Add color arguments



Let's zoom in

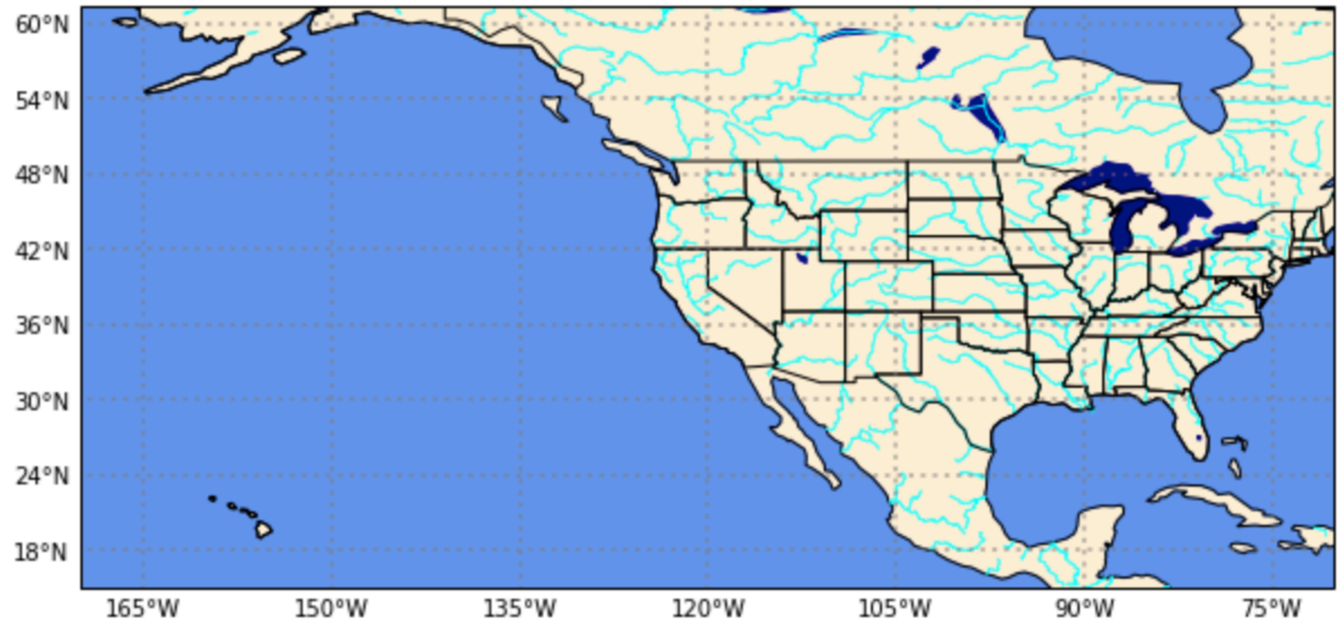
```
fig = plt.figure(figsize=(10,10))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution='110m',color='k')

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True,
                  linewidth=2, color='gray', alpha=0.5, linestyle=':')

gl.xlabels_top = False
gl.ylabels_right = False
gl.xformatter = LONGITUDE_FORMATTER
gl.yformatter = LATITUDE_FORMATTER

ax.add_feature(cfeature.LAND, color='papayawhip')
ax.add_feature(cfeature.OCEAN, color='cornflowerblue')

extent = [-170,-70,15,50]
ax.set_extent(extent)

```

ax.set_extent([x1, x2, y1, y2])



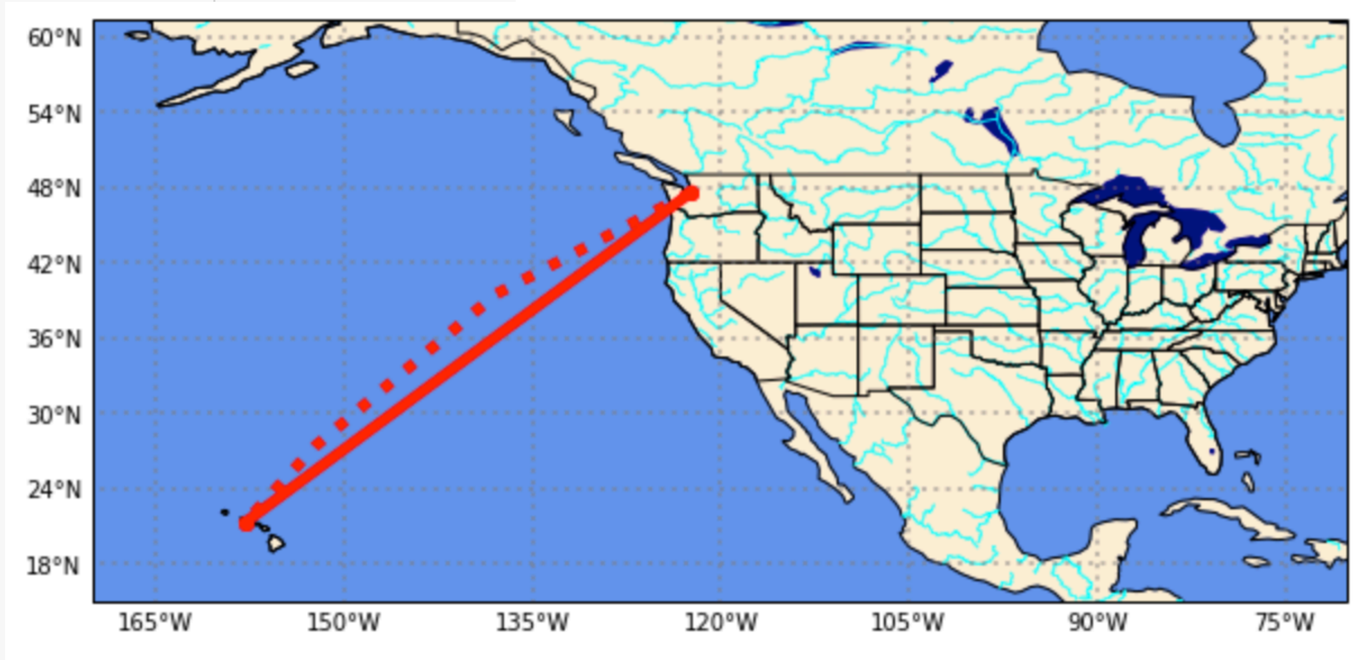
And add more features

```
fig = plt.figure(figsize=(10,10))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution='110m',color='k')

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True,
                  linewidth=2, color='gray', alpha=0.5, linestyle=':')

gl.xlabels_top = False
gl.ylabels_right = False
gl.xformatter = LONGITUDE_FORMATTER
gl.yformatter = LATITUDE_FORMATTER

ax.add_feature(cfeature.LAND, color='papayawhip')
ax.add_feature(cfeature.OCEAN, color='cornflowerblue')

extent = [-170,-70,15,50]
ax.set_extent(extent)

ax.add_feature(cfeature.LAKES,color='navy')
ax.add_feature(cfeature.RIVERS.with_scale('10m'),edgecolor='aqua')
ax.add_feature(cfeature.STATES,edgecolor='k')
```



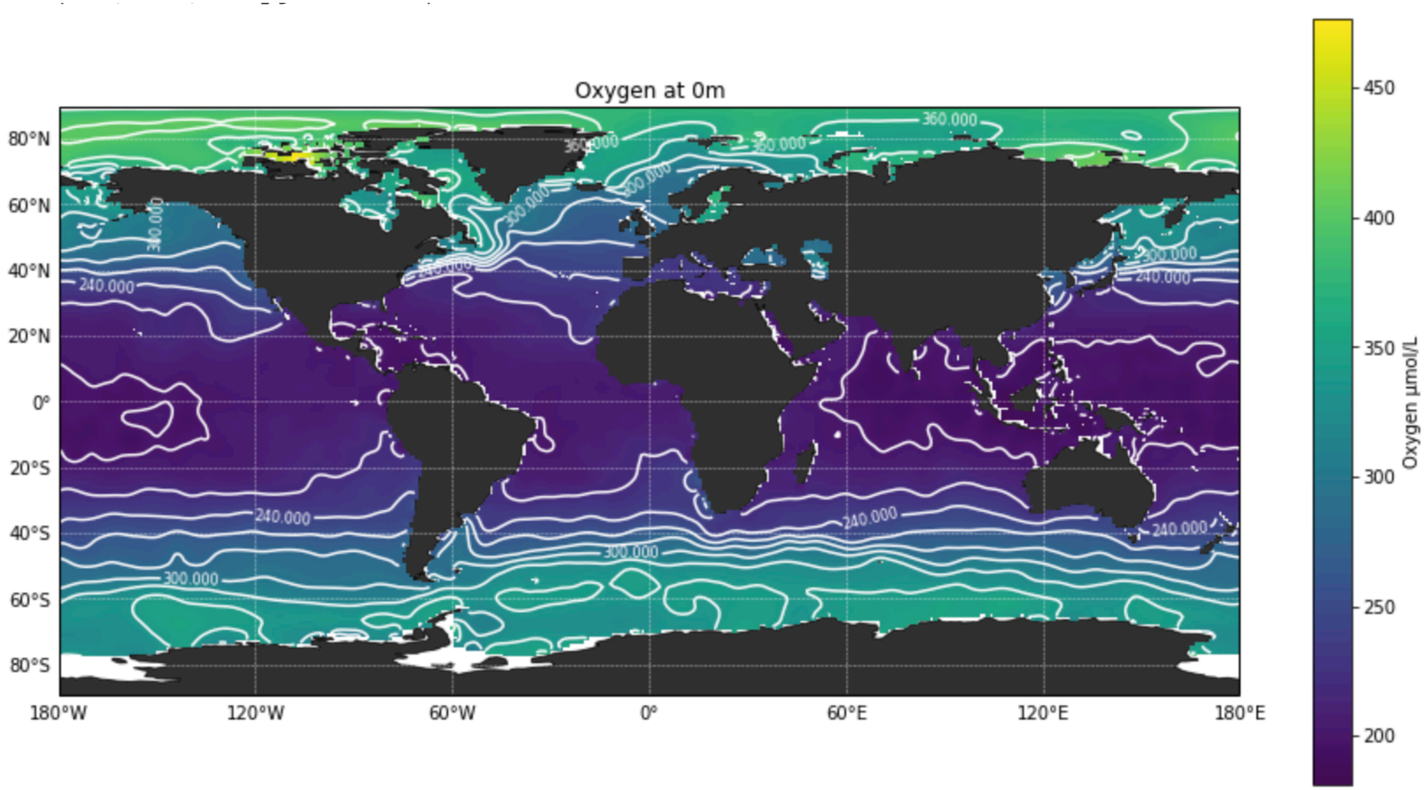
Let's put data on the map

```
fig = plt.figure(figsize=(10,10))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution='110m',color='k')

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True,
                  linewidth=2, color='gray', alpha=0.5, linestyle=':')

gl.xlabels_top = False
gl.ylabels_right = False
gl.xformatter = LONGITUDE_FORMATTER
gl.yformatter = LATITUDE_FORMATTER

ax.add_feature(cfeature.LAND, color='papayawhip')
ax.add_feature(cfeature.OCEAN, color='cornflowerblue')

extent = [-170,-70,15,50]
ax.set_extent(extent)

ax.add_feature(cfeature.LAKES,color='navy')
ax.add_feature(cfeature.RIVERS.with_scale('10m'),edgecolor='aqua')
ax.add_feature(cfeature.STATES,edgecolor='k')

seattle = [-122.332, 47.606]
honolulu = [-157.8583, 21.3069]

ax.plot([seattle[0],honolulu[0]],[seattle[1],honolulu[1]],'ro-',lw=5,transform=ccrs.PlateCarree())
ax.plot([seattle[0],honolulu[0]],[seattle[1],honolulu[1]],'ro:',lw=5,transform=ccrs.Geodetic())
```



Combining 2-d plotting and mapping

```
# Using our Oxygen data
# Change the path to wherever you have the data
filepath = '/content/drive/My Drive/Data_folder/woa18_oxy.nc'

# Read the data - times are in a bad format, so don't parse them
oxy = xr.open_dataset(filepath,decode_times=False)

# Take the average of the data along longitudes and select the only time level
# Make numpy arrays for lat, depth, and o_data
lat = oxy['lat'].values
lon = oxy['lon'].values
o_data = oxy['o_an'].sel(depth=0,method='nearest').isel(time=0).values

fig = plt.figure(figsize=(15,8))
ax = plt.axes(projection=ccrs.PlateCarree())

pcm = ax.pcolormesh(lon,lat,o_data,transform=ccrs.PlateCarree())
cntr = ax.contour(lon,lat,o_data,levels=17,transform=ccrs.PlateCarree(),colors='w')
ax.clabel(cntr,levels=cntr.levels[::3],colors='w',fontsize=8)

c = plt.colorbar(pcm, ax=ax)
c.set_label('Oxygen µmol/L')

ax.add_feature(cfeature.LAND, color='k',alpha=0.8)

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True,
                  linewidth=0.5, color='w', alpha=0.7, linestyle='--')
gl.xlabels_top = False
gl.ylabels_right = False
gl.xformatter = LONGITUDE_FORMATTER
gl.yformatter = LATITUDE_FORMATTER

ax.set_title('Oxygen at 0m')
```



Extra resources for this lesson

Cartopy tutorial: https://rabernat.github.io/research_computing_2018/maps-with-cartopy.html

Cartopy+Matplotlib: https://scitools.org.uk/cartopy/docs/latest/matplotlib/geoaxes.html#cartopy.mpl.geoaxes.GeoAxes.coastlines

Cartopy grids: https://scitools.org.uk/cartopy/docs/latest/matplotlib/gridliner.html

Contourf: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.contourf.html

Contour: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.contour.html

Pcolormesh: https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.pcolormesh.html

Mapping problems: https://www.scienceabc.com/social-science/what-is-wrong-with-all-our-maps-mercator-maps.html