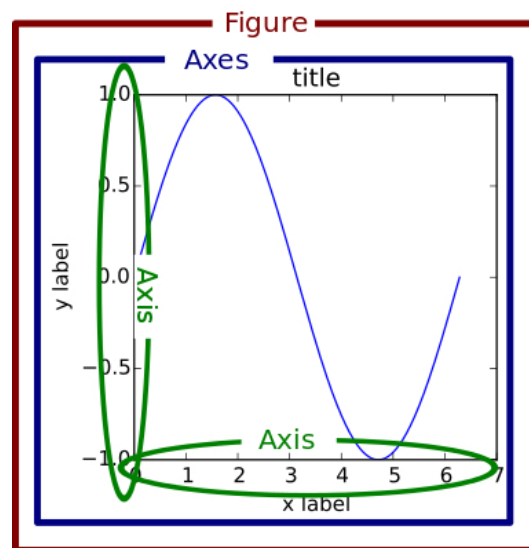# 1 Review

Last time:

- Numpy

- Questions?

- Today: Plotting with Matplotlib and Pandas

2 Matplotlib Intro

- Matplotlib (Matlab Plotting Library) is a package for creating plots and figures

- Importing matplotlib:

  ```python
  import matplotlib.pyplot as plt
  ```


- `Figure`: container for a matplotlib graphic. A figure contains multiple `Axes` objects.

- `Axes`: an individual plot. (Note: not be confused with Axis. Multiple AXIS form an AXES).

- The `Axes` contains most of the smaller objects that we care about. E.g. `title`, `xlabel`, etc...

3 Axes

Reference: https://matplotlib.org/stable/api/axes_api.html#plotting

- `subplots` method creates and returns a `Figure` and `Axes`(s). By default, `subplots` populates the `Figure` with one `Axes`. We can optionally specify the figure size.

- We use the `plot` method to plot our data. We can optionally specify a label

- `set_title`, `set_xlabel`, `set_ylabel`

- In order to display the `Figure`, we use `plt.show()`. This command finds and displays the current `Figure`

- Basic plot example:

```
fig, ax = plt.subplots(figsize=(8,6))

x = np.linspace(0, 10, 20)
y1 = 2*x+1
y2 = 3*x+1

ax.plot(x, y1, label='a line')
ax.plot(x, y2, label='a steeper line')
ax.set_title('A title!')
ax.set_xlabel('x')
ax.set_ylabel(r'$y = 2 \cdot x +1$')
fig.legend()

plt.show()
```

ax.plot format:

- `ax.plot(x, y, [format])`

Character	Color	Character	Description
'b'	Blue	'.'	Point marker
		'o'	Circle marker
'g'	Green	'x'	X marker
'r'	Red	'D'	Diamond marker
'b'	Blue	'H'	Hexagon marker
		's'	Square marker
'c'	Cyan	'+'	Plus marker
'm'	Magenta	**Character**	**Description**
'y'	Yellow	'-'	Solid line
'k'	Black	'—'	Dashed line
		'-.'	Dash-dot line
'b'	Blue	':'	Dotted line
'w'	White	'H'	Hexagon marker

- `plot` has other properties, e.g. linewidth, markersize, etc...(read the docs)

Additional Axes Methods (Demo):

- `ax.grid()`

- `ax.set_xlim(min, max)`, `ax.set_ylim(min, max)`

- `ax.xscale(value)`, `ax.yscale(value)`, e.g. `ax.xscale('log')`

- `ax.axhline(y)`, `ax.axvline(y)`

- `ax.errorbar(x, y, yerr=None, xerr=None, capsize=None)`

Other types of plots (read the docs):

- Bar and Histogram

- Pie

- Box and wisker

- etc...

4 Subplots

- Recall: we use the `subplots` function to create a `Figure`. By default, this creates one `Axes`. However, we can specify a grid of `Axes`: `ax.subplots(rows, cols)`

- `subplots` returns the figure and array of `Axes`

- Example:
  ```
  fig, ax = plt.subplots(2, 3, figsize=(8,6))
  ax[0][1].plot(x,y1)
  ax[1][0].plot(x,y2)
  plt.show()
  ```

- We can arrange the `Axes` in more complicated manners. Read the docs...

5 2D Plots

Plotting a 2d function: $z(x, y) = x + y$

X					Y					COORDINATE					Z = X + Y			
0	1	2	3		0	0	0	0		0,0	1,0	2,0	3,0		0	1	2	3
0	1	2	3		1	1	1	1		0,1	1,1	2,1	3,1		1	2	3	4
0	1	2	3		2	2	2	2		0,2	1,2	2,2	3,2		2	3	4	5

- How do we generate x, y, and z? Numpy!

```
x = np.arange(4)
y = np.arange(3)
X, Y = np.meshgrid(x,y)
Z = X + Y

>>> X
array([[0, 1, 2, 3],
       [0, 1, 2, 3],
       [0, 1, 2, 3]])
>>> Y
array([[0, 0, 0, 0],
       [1, 1, 1, 1],
       [2, 2, 2, 2]])
>>> Z
array([[0, 1, 2, 3],
       [1, 2, 3, 4],
       [2, 3, 4, 5]])
```

- Two ways to plot: `imshow` and `pcolormesh`

- `imshow(Z)`: assumes values are equally spaced (for plotting images and 2d functions).

```
        +--------+--------+
        | Z[0,0] | Z[0,1] |
        +--------+--------+
        | Z[1,0] | Z[1,1] |
        +--------+--------+
```

- `pcolormesh(X, Y, C)`: does not assumed equally spaced values. `X` and `Y` specify the quadrilateral corners. This is a mesh (cells do not have to be evenly spaced)!

```
    (X[i+1, j], Y[i+1, j])              (X[i+1, j+1], Y[i+1, j+1])
                            +--------+
                            | C[i,j] |
                            +--------+
        (X[i, j], Y[i, j])              (X[i, j+1], Y[i, j+1]),
```

- We will use `imshow`:

```
fig, ax = plt.subplots(figsize=(8, 5))
image = ax.imshow(Z)
plt.show()
```

- Color-map: key for how to map a value to a color. Matplotlib has many built-in cmaps and you can define your own.

```
# default color map
fig, ax = plt.subplots(figsize=(8, 5))
image = ax.imshow(Z)
plt.colorbar(image)
plt.show()

# example of a monotone cmap
fig, ax = plt.subplots(figsize=(8, 5))
image = ax.imshow(Z, cmap='Purples')
plt.colorbar(image)
plt.show()
```

- We can also plot contours using `ax.contour([X, Y], Z, [levels])`

6   (Optional) Example: Plotting a Dipole Potential

Note: this demo includes many optional arguments/features not covered in lecture. It's impossible cover nor remember everything. Searching the docs is an important skill!

```python
import matplotlib
x = np.linspace(−1.5, 1.5, 200)
y = np.linspace(−1, 1, 200)
X, Y = np.meshgrid(x,y)

# calculate dipole potential
R1 = np.sqrt((X−1)**2 + Y**2)
R2 = np.sqrt((X+1)**2 + Y**2)
Z = (1/R1 − 1/R2)
Z /= np.max(Z) # normalize

fig, ax = plt.subplots(figsize=(10, 7))
image = ax.imshow(Z, origin='lower', cmap='bwr',
                  extent=[np.min(x), np.max(x), np.min(y), np.max(y)],
                  norm=matplotlib.colors.SymLogNorm(0.01))
eqipots = [−0.1, −0.03, −0.01, −0.003, −0.001, 0,
           0.001, 0.003, 0.01, 0.03, 0.1]
ax.contour(X, Y, Z, eqipots, colors='k')

ax.set_title('Dipole Potential', fontsize=16)
ax.set_xlabel(r'$x$', fontsize=16)
ax.set_ylabel(r'$y$', fontsize=16)

plt.colorbar(image, fraction=.03)
plt.savefig('dipole.png', dpi=300, bbox_inches='tight')
plt.show()
```
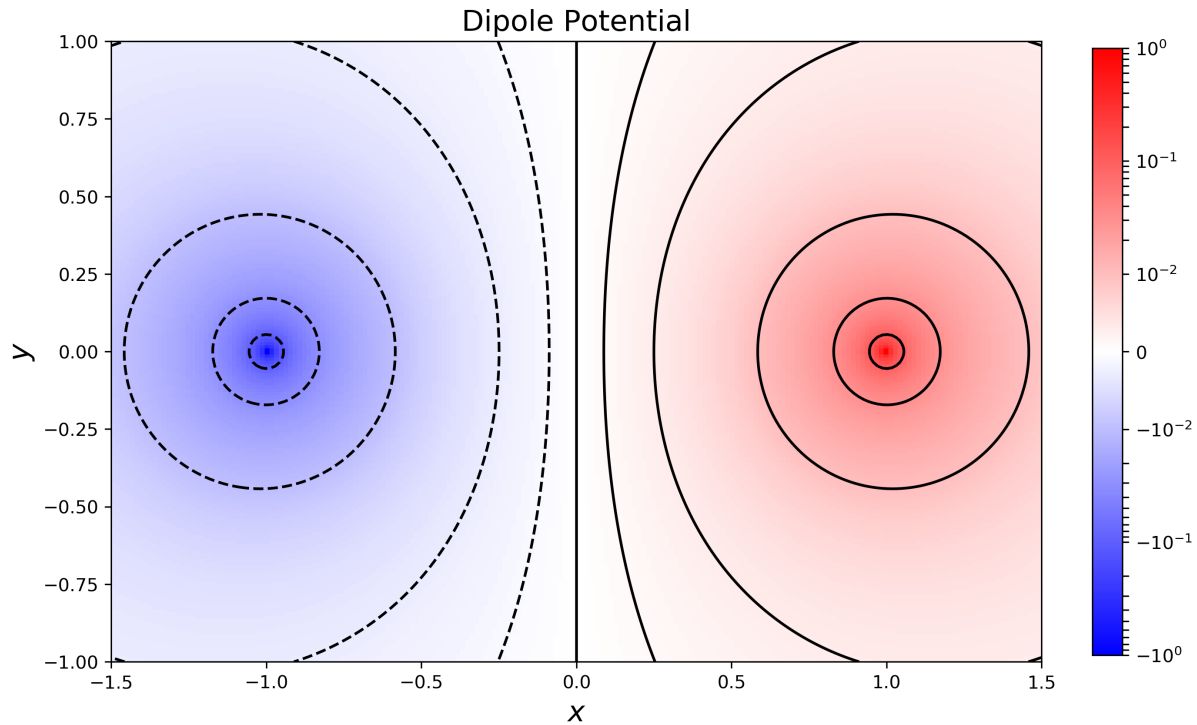
Dipole Potential

7 Save Figure

- `plt.savefig(name)`

- Recommend args: `dpi=300, bbox_inches='tight'`

- If you want to show and save figure, you must save BEFORE showing

- Example:

```
plt.savefig('fig.png',dpi=300,bbox_inches='tight')
```

8 Pandas

- Pandas is a package for reading and cleaning data. This is a topic that deserves a separate workshop. I will only show a limited functionality.

- Reading in csv/txt file from computer/online (`bb.txt` is included as a backup):

```python
import pandas as pd
# data frame
df = pd.read_csv('https://www.ocf.berkeley.edu/~yizhu/bb.txt',
                 delimiter=',')

lam = df['Lambda (nm)']
I = df['Specific Intensity (W/m^3*Sr)']
```

```
>>> data
        Lambda (nm)    Specific Intensity (W/m^3*Sr)
0      300.000000                    7.940206e+12
1      303.517588                    8.338975e+12
2      307.035176                    8.756094e+12
3      310.552764                    8.857883e+12
4      314.070352                    9.422244e+12
..            ...                             ...
195    985.929648                    9.643322e+12
196    989.447236                    9.643796e+12
197    992.964824                    9.547717e+12
198    996.482412                    9.461644e+12
199   1000.000000                    9.485265e+12

[200 rows x 2 columns]

>>>lam
0       300.000000
1       303.517588
2       307.035176
3       310.552764
4       314.070352
           ...
195     985.929648
196     989.447236
197     992.964824
198     996.482412
199    1000.000000
Name: Lambda (nm), Length: 200, dtype: float64
```

- Some common optional arguments for `pd.read_csv`: `header=None`, `encoding = 'utf8'`

9 Conclusion

This is the end of the Python lecture series. This series is by no means comprehensive, but intended to prepare you to further learn/explore Python on your own. Remember:

- Read the documentation

- Read the error tracebacks

- Google, StackOverflow, etc... is your friend

- Don't hesistate to reach out to the ULAB staff for help!