Matplotlib II

9.29.24

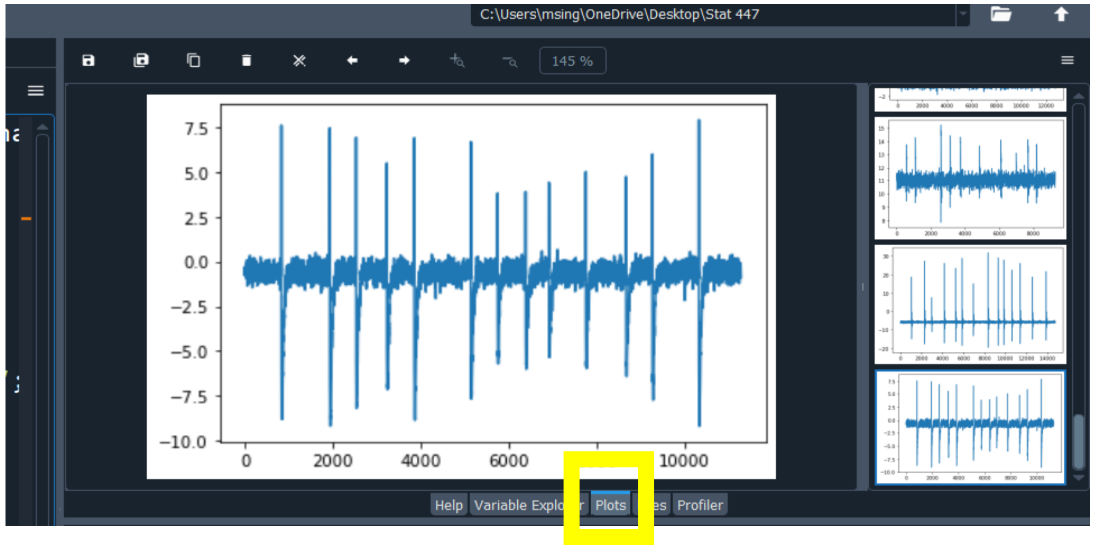
Learning Objectives

import matplotlib.pyplot as **plt**

- Some common plot-types
- Adding details to plots
- Programmatic Chart Generation

Viewing Python Graphics: Spyder

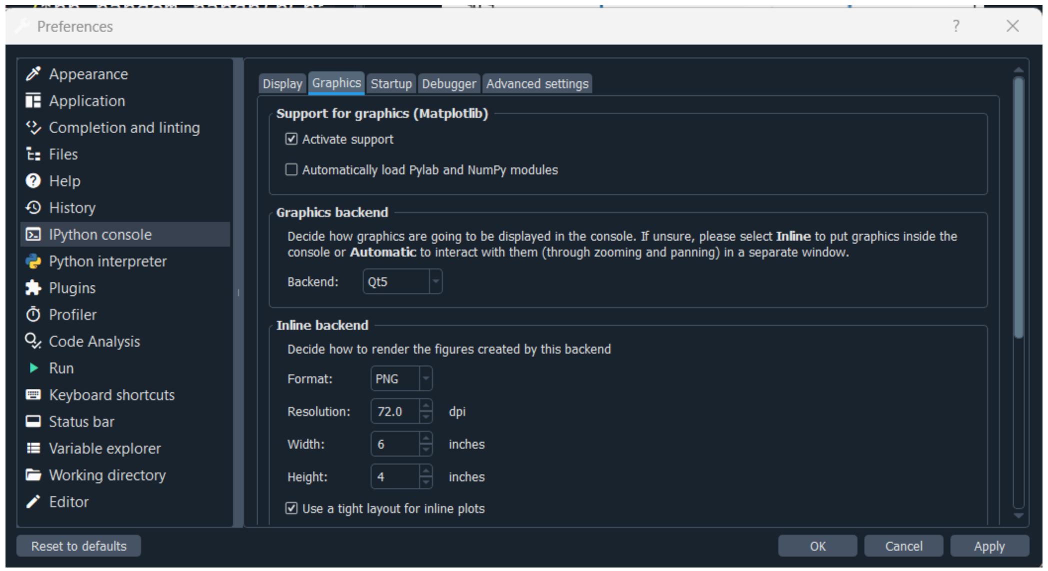
- Plot Panel of Spyder: Plots appear as tabs (Inline backend)
- You can undock into a separate window, but not very interactive



Viewing Python Graphics: Spyder

- Qt backend: separate windows, can pan, zoom etc.
- Preferences->iPython Console->Graphics->Graphics backend

    Switch to Qt5 (or whatever version you have)



Support for graphics (Matplotlib)

☑ Activate support

☐ Automatically load Pylab and NumPy modules

Graphics backend

Decide how graphics are going to be displayed in the console. If unsure, please select **Inline** to put graphics inside the console or **Automatic** to interact with them (through zooming and panning) in a separate window.

Backend: Qt5

Inline backend

Decide how to render the figures created by this backend

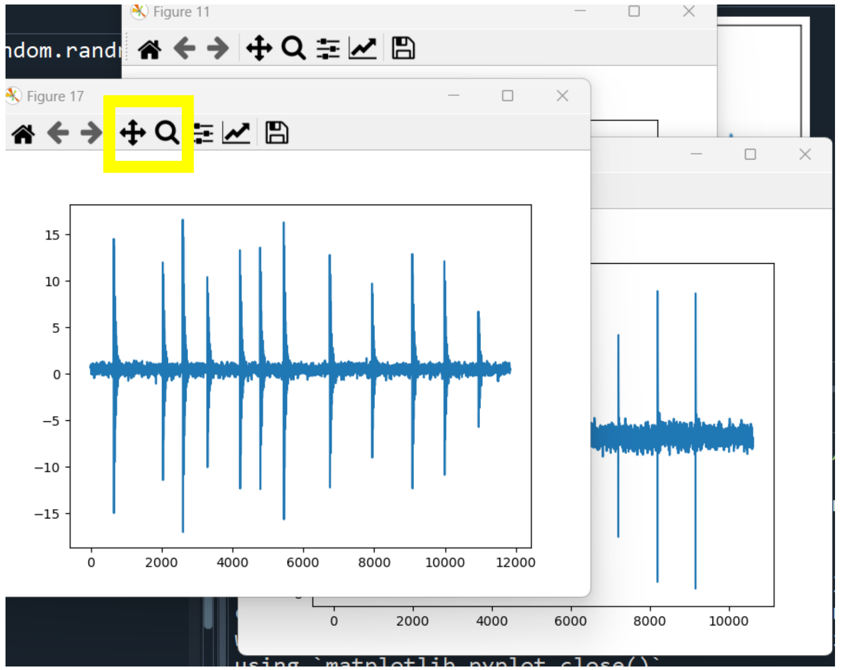
Format: PNG

Resolution: 72.0 dpi

Width: 6 inches

Height: 4 inches

☑ Use a tight layout for inline plots



Viewing Python Graphics: Jupyter

- Standard—inline graphics

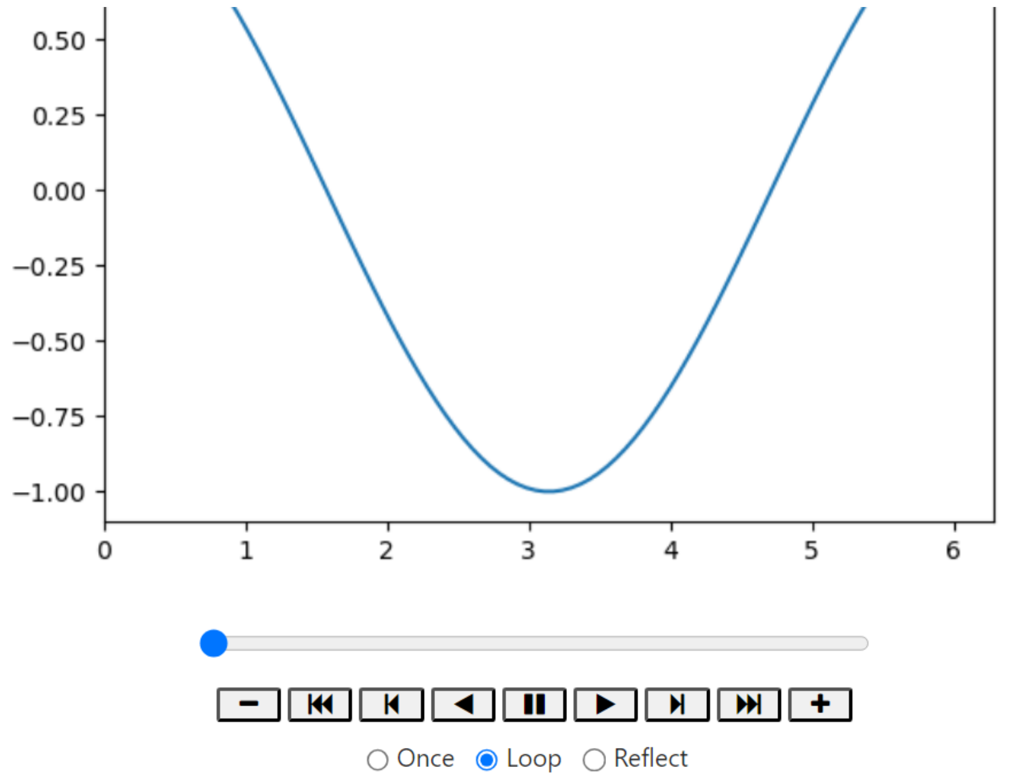
Can add qt: popout/interactive figures

- %matplotlib qt

Go back to inline:

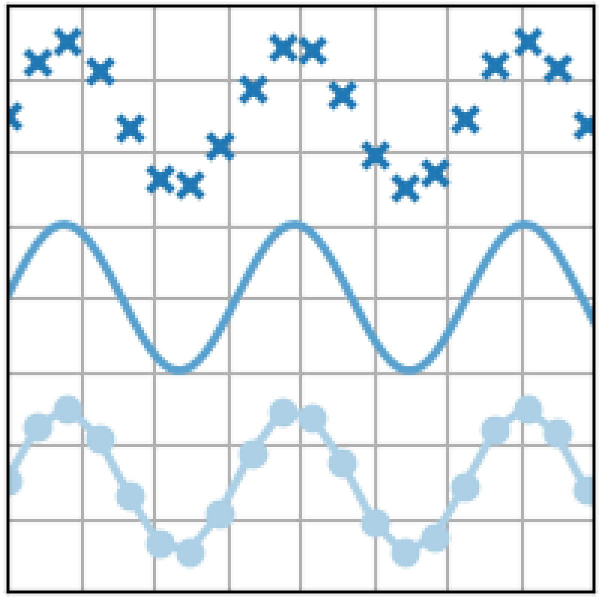
- %matplotlib inline

Can use HTML for animations:

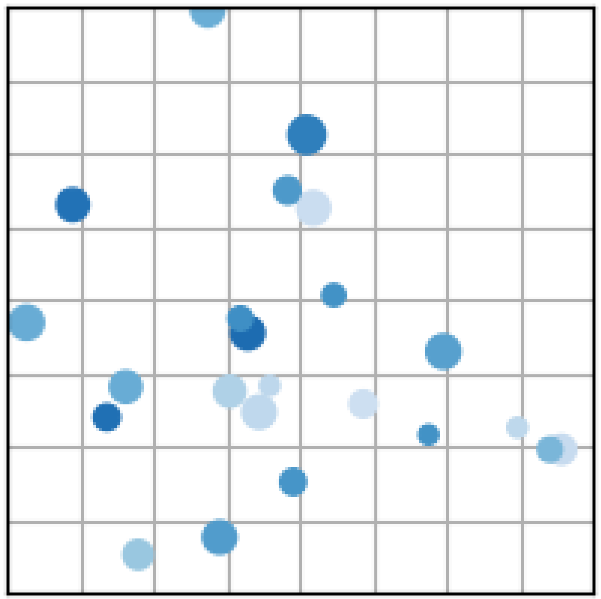


Basic Plot Types

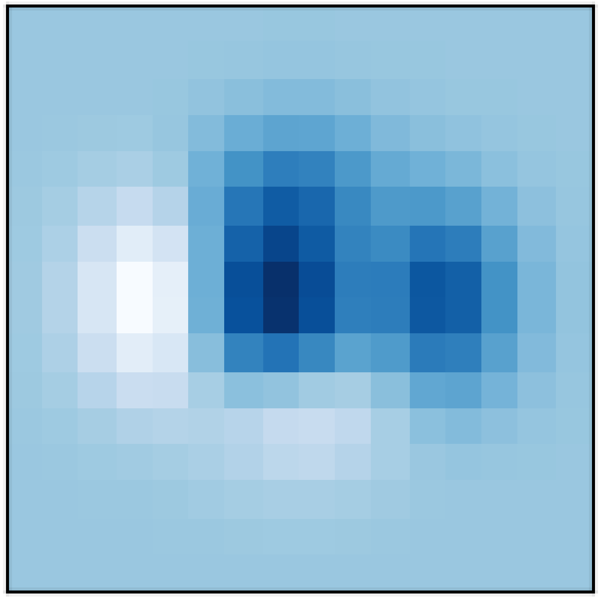
plot(x, y)



scatter(x, y)



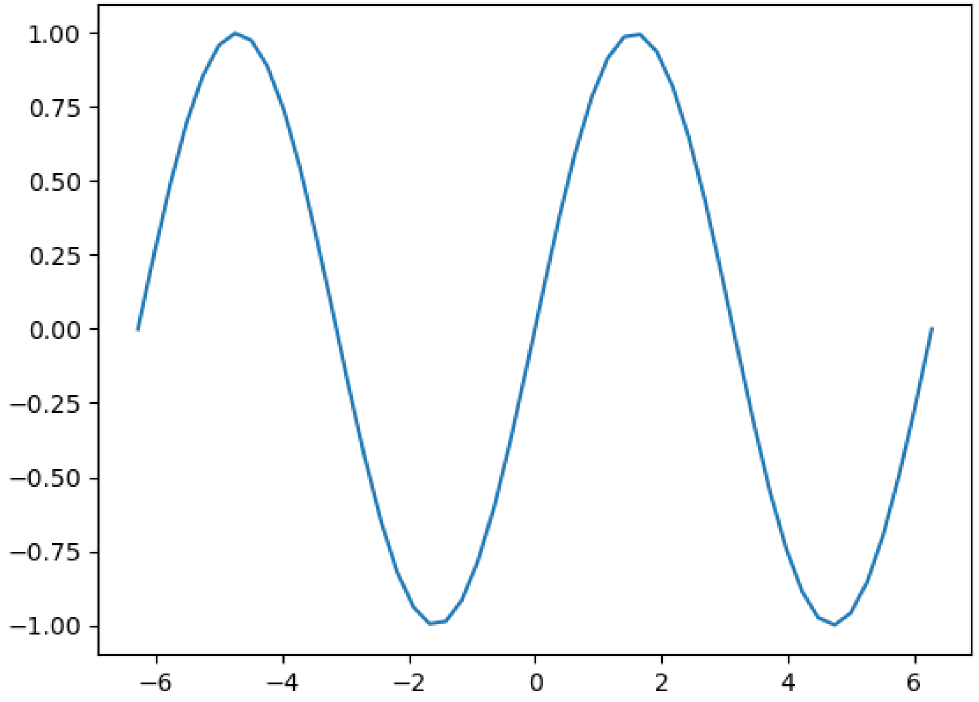
imshow(Z)



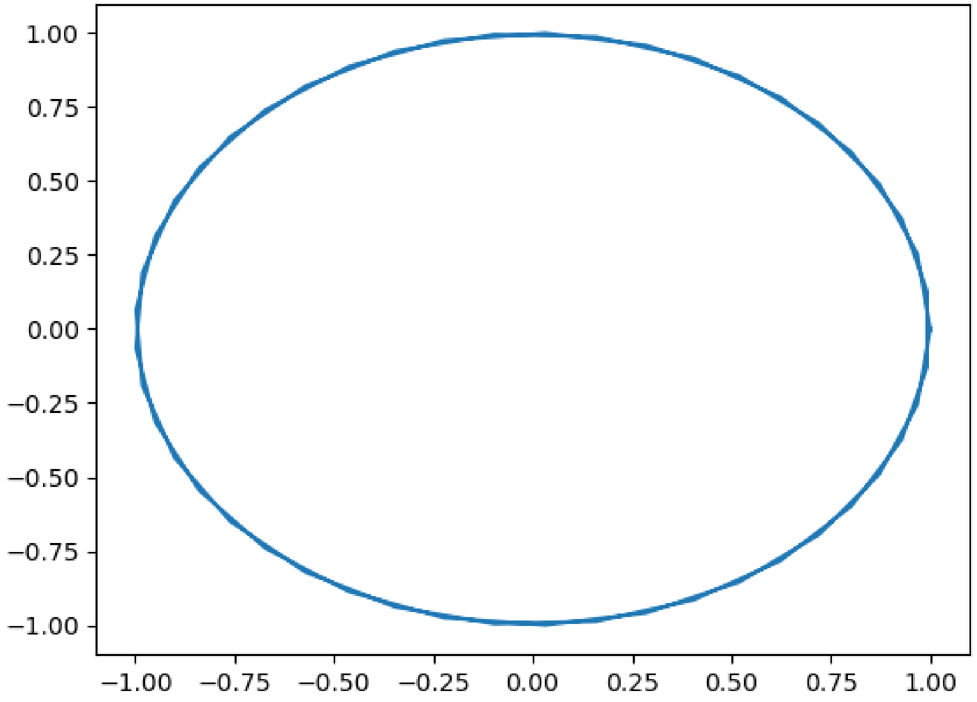
Line Plots

plot(x , y)   or plot(y): (assumes 1:n for x)

```
Xval=np.pi*np.linspace(-2,2,50)
plt.plot(Xval,np.sin(Xval))
```



```
Xval=np.pi*np.linspace(-2,2,50)
plt.plot(np.cos(Xval),np.sin(Xval))
```



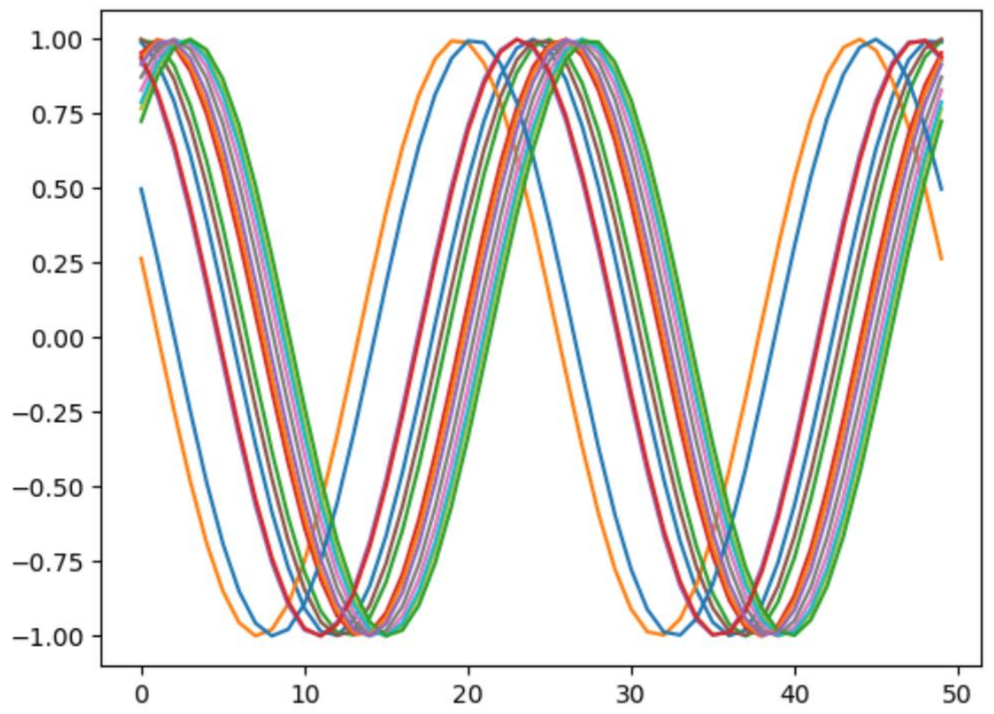
Multiple Line Plots

plot(x,Z), **plot**(A,Z) or **plot**(Z) for matrices A,Z, or shared vector x

Lines sorted by column

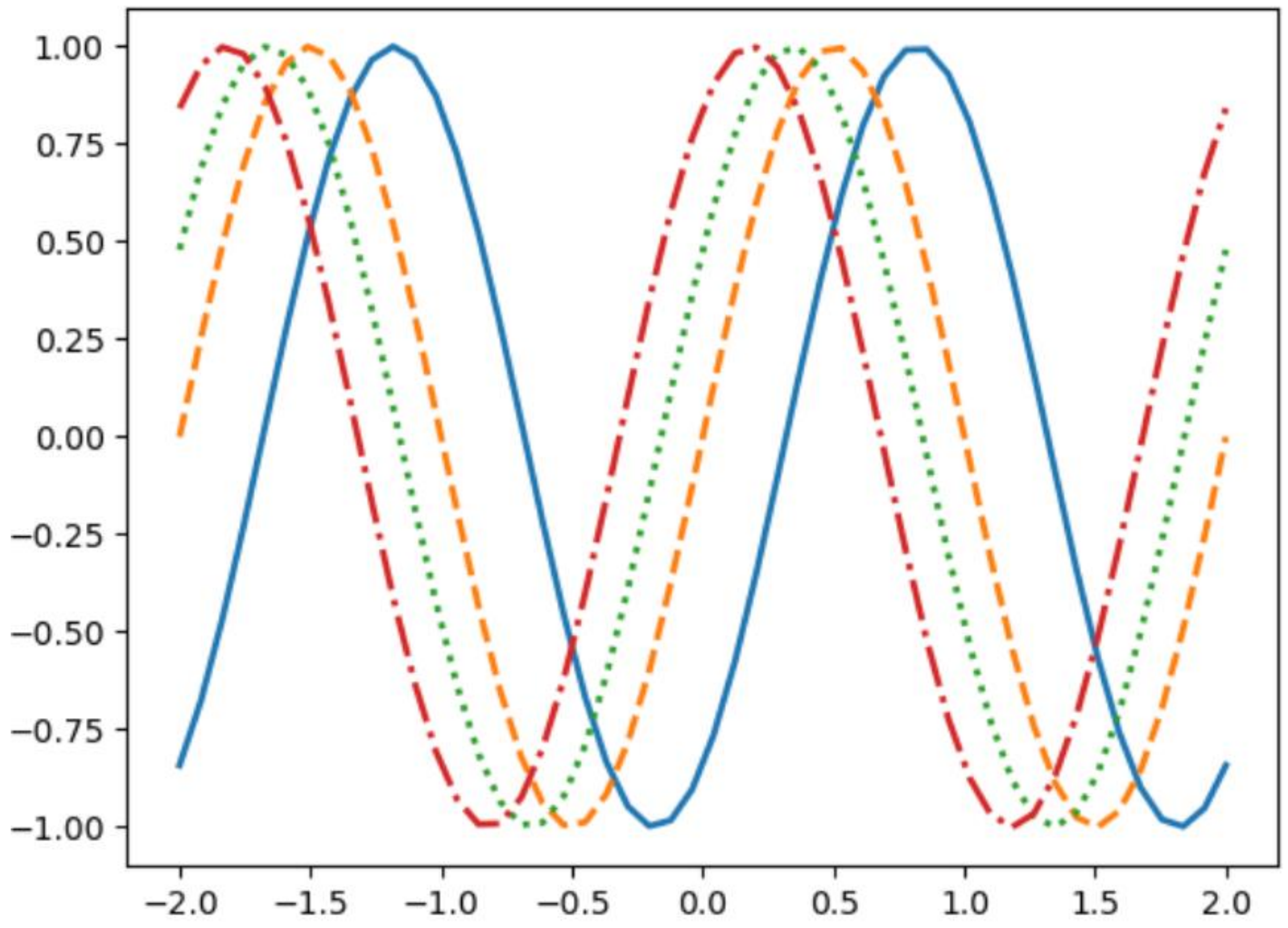
50 x 15 after .T = 15 lines

```
plt.plot((np.cos(np.random.randn(15,1)+Xval)).T)
```



Options: Line Style

```
## This is solid (def.)
plt.plot(x,y0,'-')
## This is dashed
plt.plot(x,y1,'--')
## This is dotted
plt.plot(x,y2,':')
## This is dash-dot
plt.plot(x,y3,'-.')
```



Options: Line appearance

Line width or lw

```
plt.plot(x,y,lw=2)
```

Color (name or rgb-values)

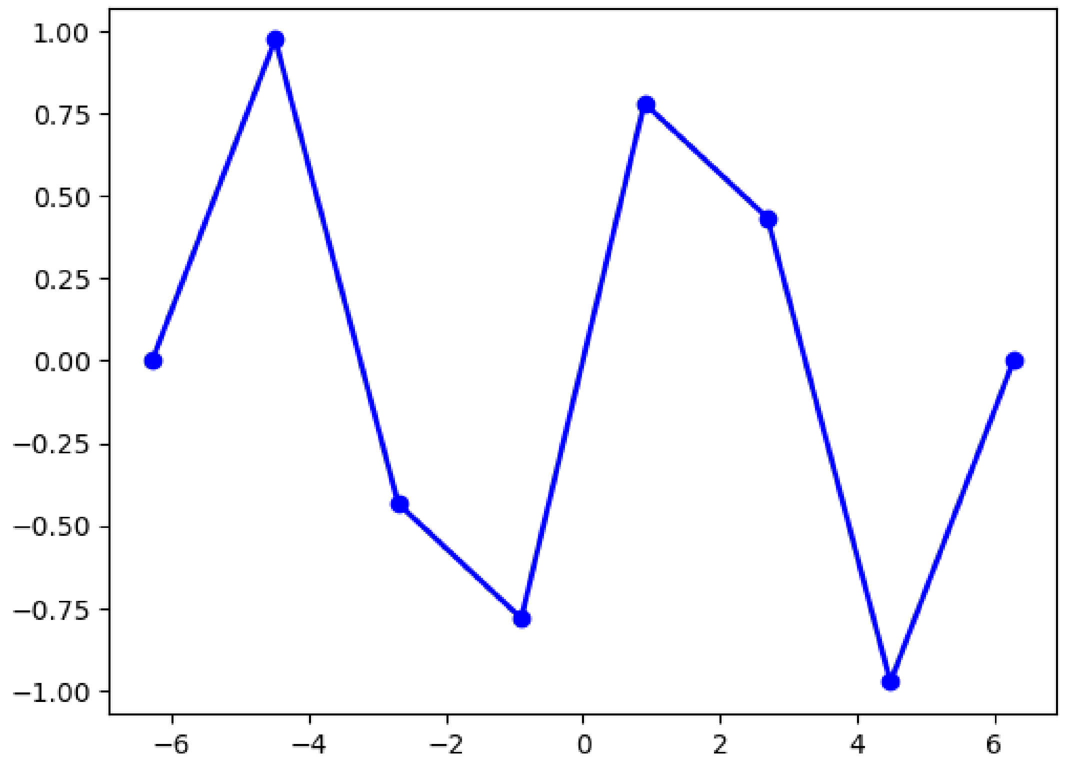
```
plt.plot(x,y,c='r')
plt.plot(x,y,c=[1,0,0])
```

Markers

(o=circle, ^=triangle, s=square, *, x)

```
plt.plot(x,y,'o')
```

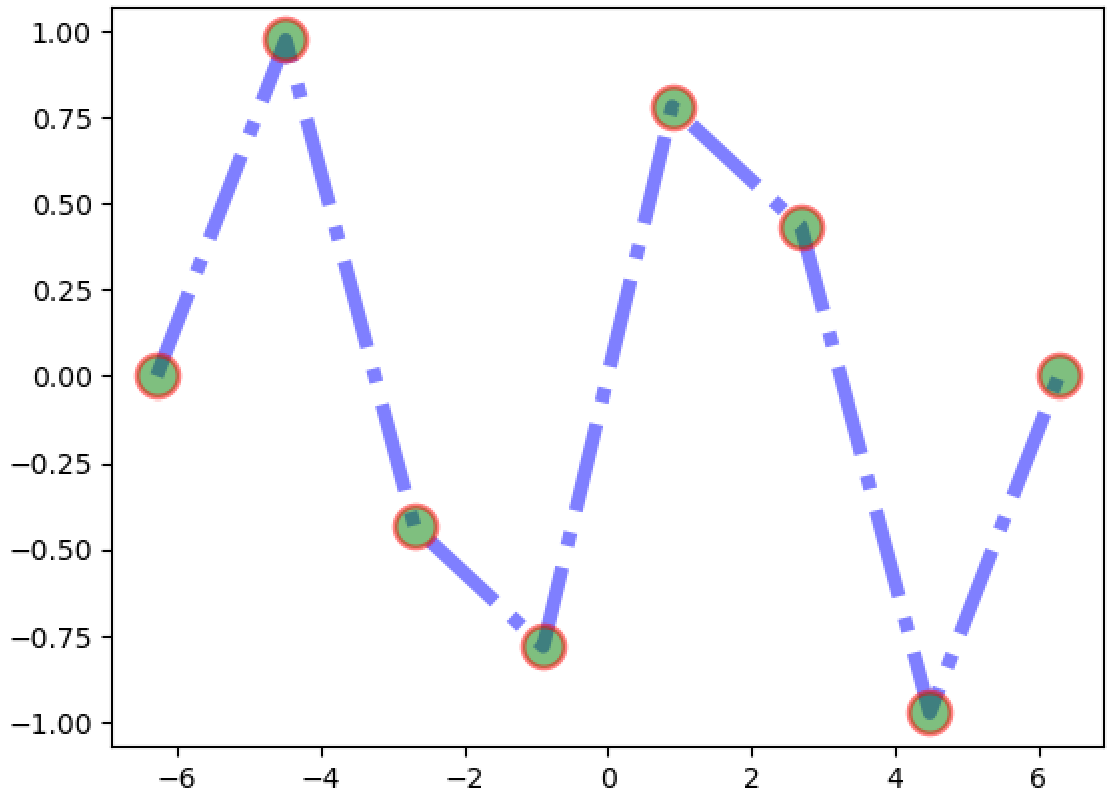
```
plt.plot(x,y,'-ob',lw=2)
```



More options

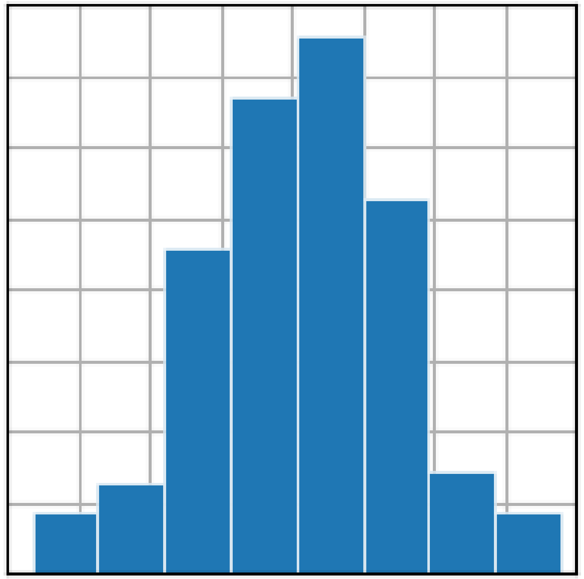
```
plt.plot(x,y,'-.bo',lw=5,ms=15,
          alpha=.5,mfc='g',mec='r',mew=2)
```

- **alpha**: transparency
- **ms**: marker size
- **mfc**: marker face color
- **mec**: marker edge color
- **mew**: marker edge width

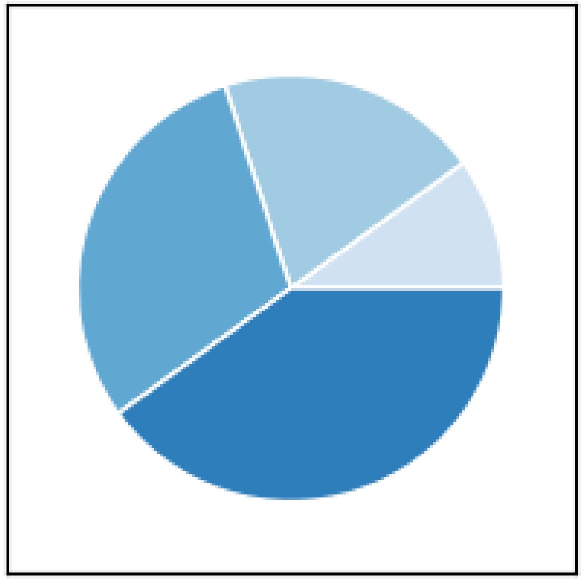


Statistical Plots

hist(x,bins=)

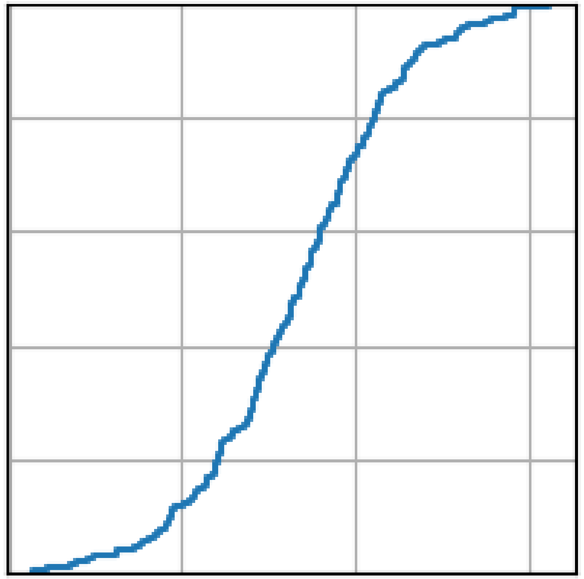


pie(x)



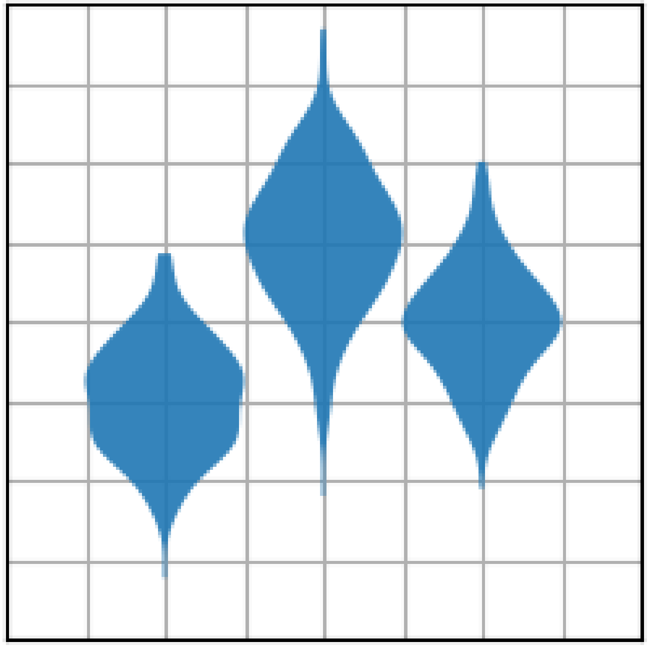
(empirical CDF)

ecdf(x)

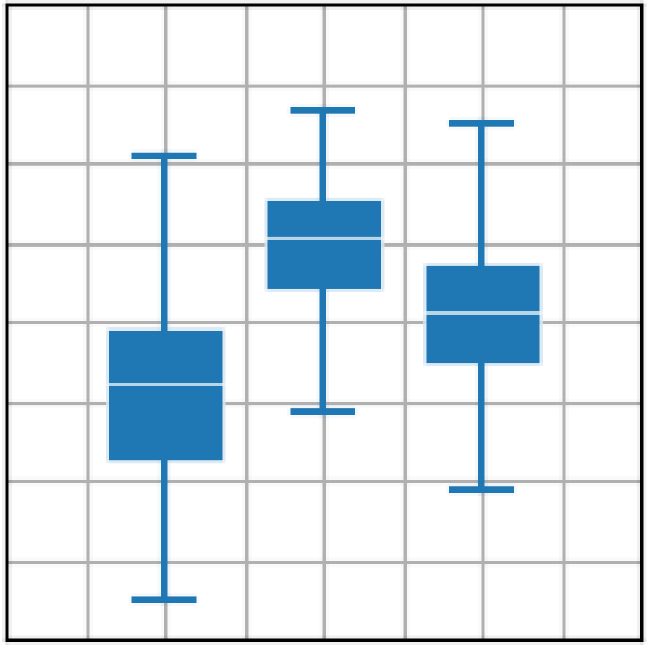


Statistical Plots

violinplot(Z) or violinplot([x,y,z])



boxplot(Z) or boxplot([x,y,z])



errorbar (x,y,xerr,yerr)

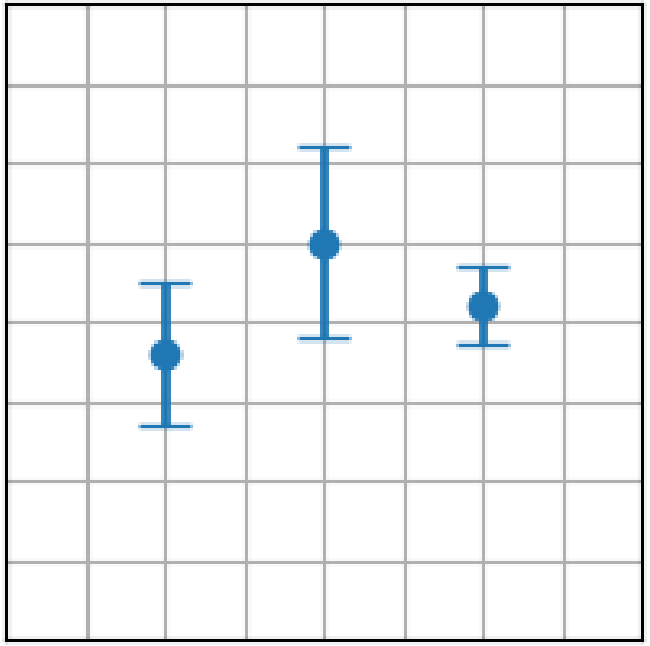
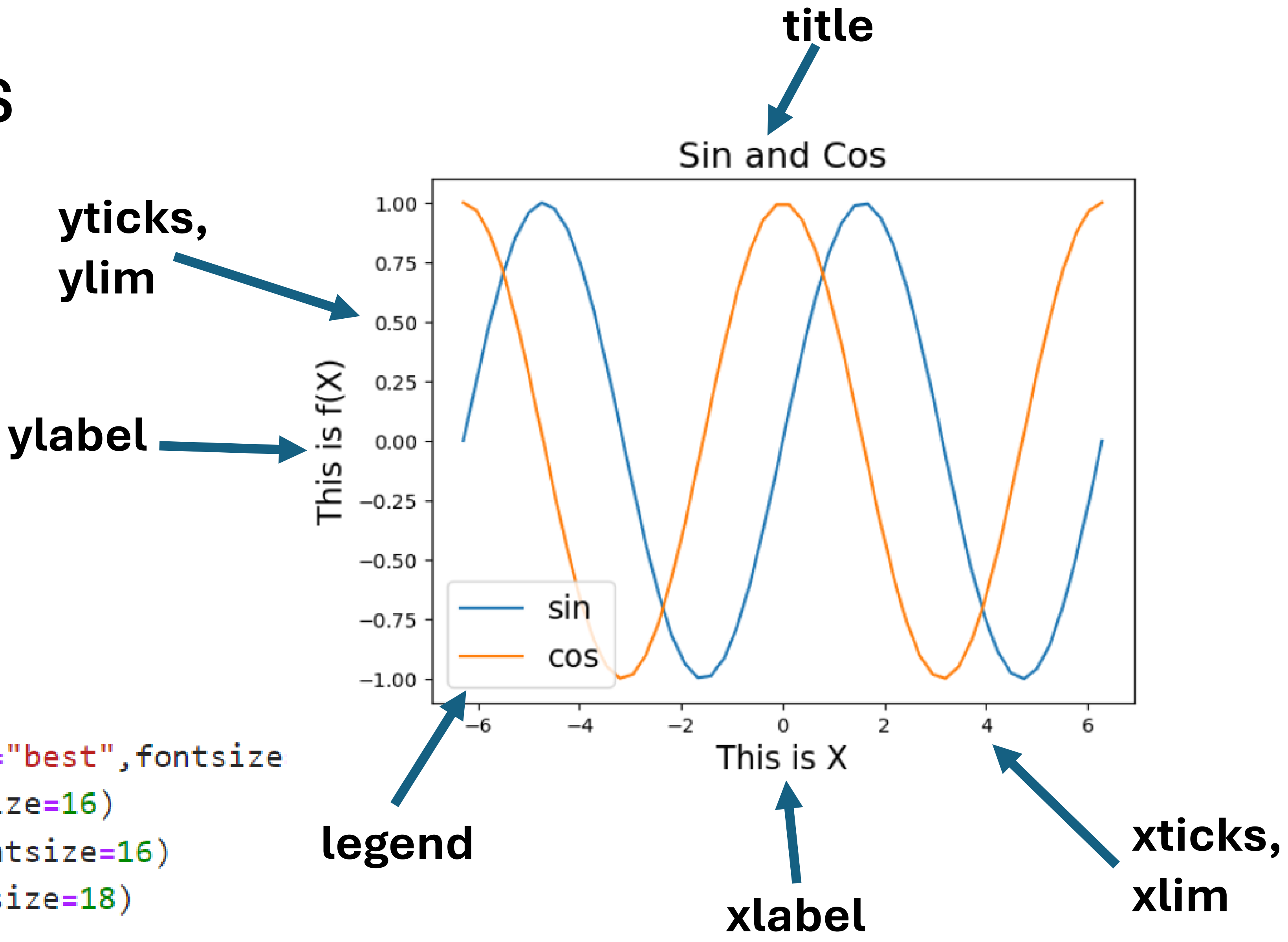


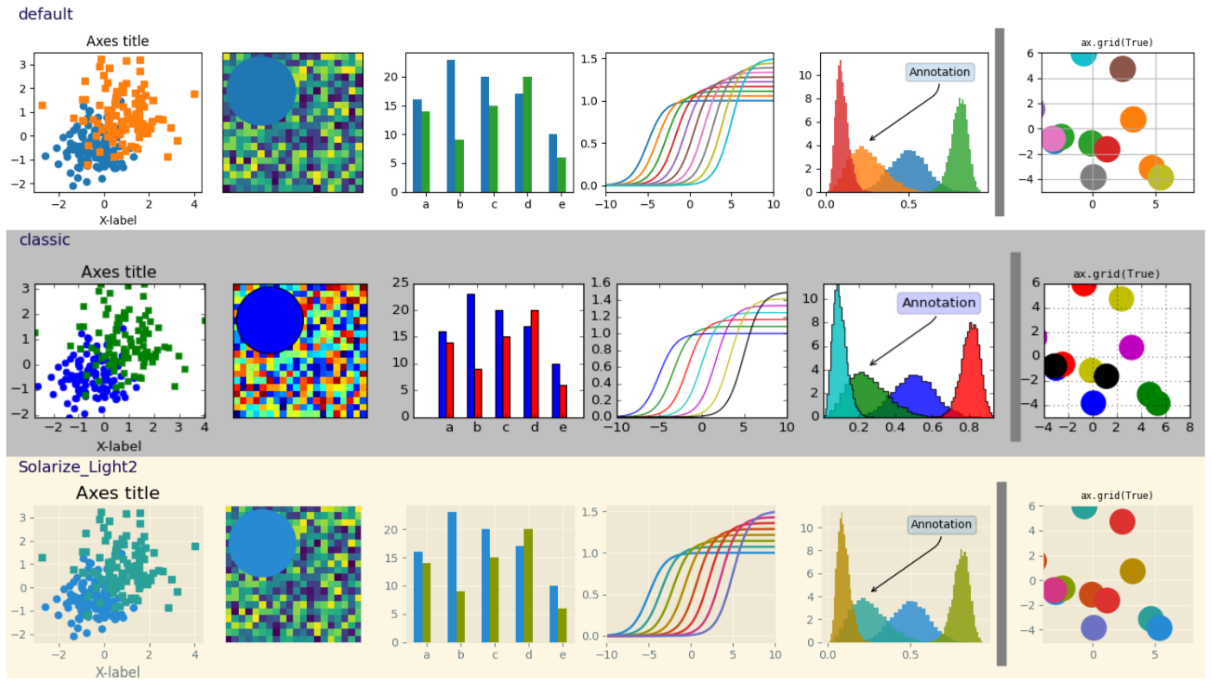
Figure Parts

```
x=np.linspace(-2,2,50)*np.pi
plt.figure();
plt.plot(x,np.sin(x))
plt.plot(x,np.cos(x))
plt.legend(["sin","cos"],loc="best",fontsize
plt.xlabel("This is X",fontsize=16)
plt.ylabel("This is f(X)",fontsize=16)
plt.title("Sin and Cos",fontsize=18)
```



Plotting with style

- A bunch of prebuilt styles, colormaps
- https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html
- https://matplotlib.org/stable/users/explain/colors/colormaps.html



```
plt.style.use('ggplot')
```

Onto Jupyter for the remainder…

Learning Objectives

import matplotlib.pyplot as **plt**

- Some common plot-types
- Adding details to plots
- Programmatic Chart Generation

Fin