ENGINEERING
TEXAS A&M UNIVERSITY

# Week 8

## Plotting with MatPlotLib

# Learning Objectives

ENGINEERING
TEXAS A&M UNIVERSITY

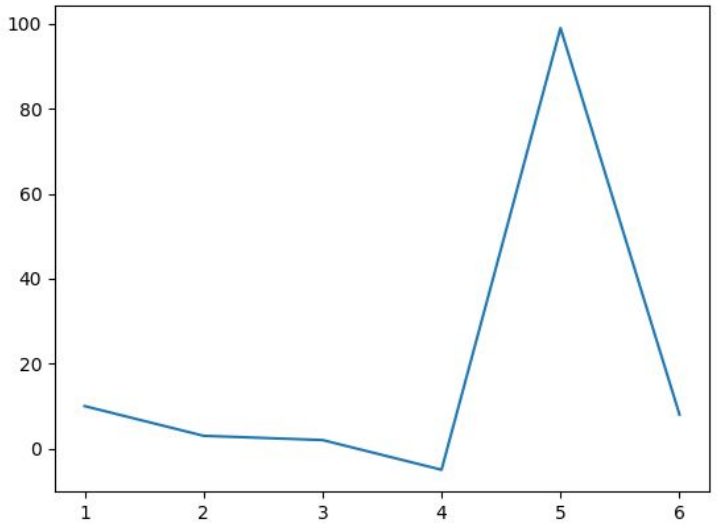
## Plotting

- Use matplotlib to create a plot for a specific range of values
- Create axis labels, titles and legends on a plot

# Creating a Plot

Here's a simple example:

```
import matplotlib.pyplot as plt
import math

x = [1, 2, 3, 4, 5, 6]
y = [10, 3, 2, -5, 99, 8]

plt.plot(x, y)
plt.show()
```

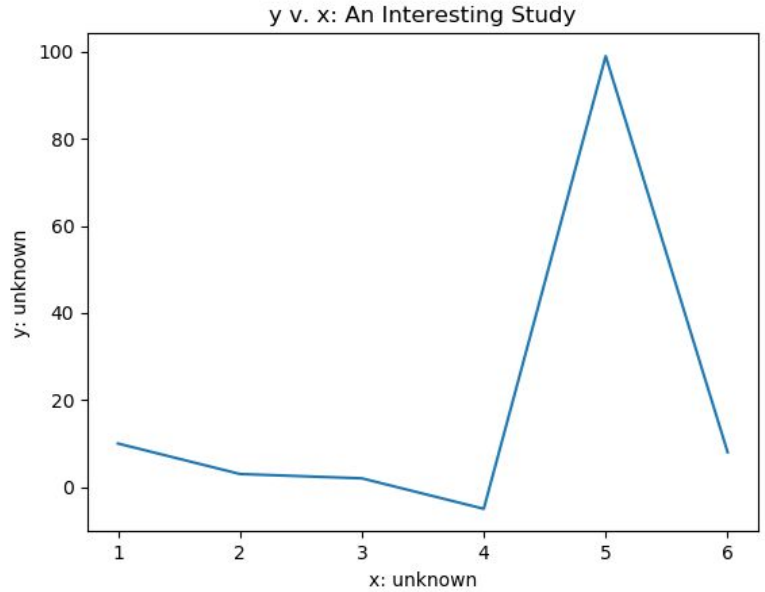



What's missing?

# Creating a Plot

Include titles and labels everywhere!

```
import matplotlib.pyplot as plt
import math

x = [1, 2, 3, 4, 5, 6]
y = [10, 3, 2, -5, 99, 8]

plt.plot(x, y)
plt.title('y v. x: An Interesting Study')
plt.xlabel('x: unknown')
plt.ylabel('y: unknown')

plt.show()
```



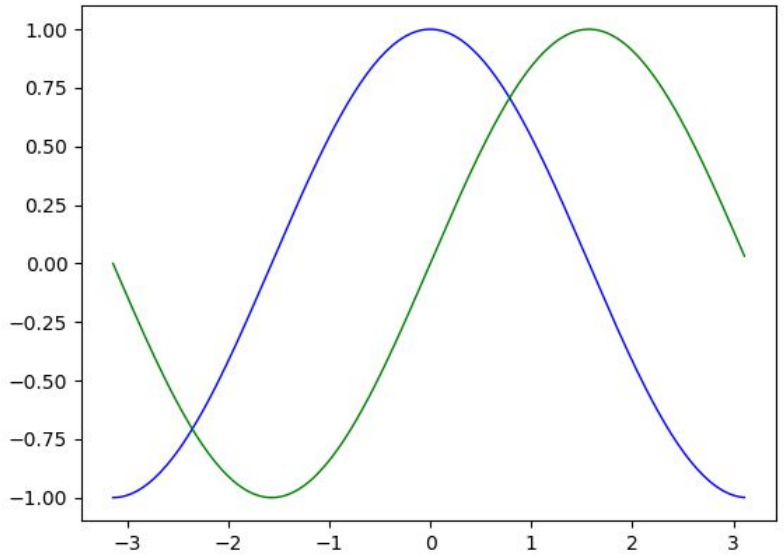
# Creating a Plot

```
import matplotlib.pyplot as plt
import math
x = []
y1 = []
y2 = []
for i in range(-100, 100):
   x.append(math.pi*i/100)
   y1.append(math.cos(math.pi*i/100))
   y2.append(math.sin(math.pi*i/100))

plt.plot(x, y1, color="blue", linewidth=1.0, linestyle="-")
plt.plot(x, y2, color="green", linewidth=1.0, linestyle="-")
plt.show()
```

More complicated - what's happening here?

# Creating a Plot

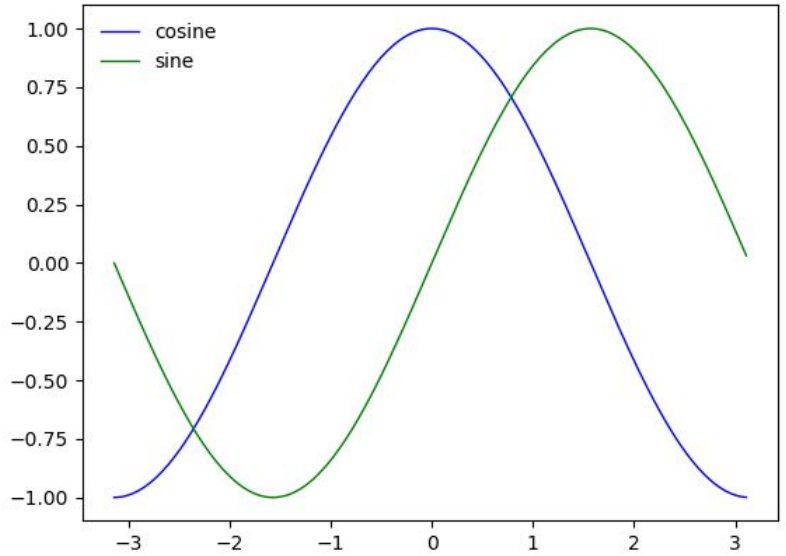
```
import matplotlib.pyplot as plt
import math
x = []
y1 = []
y2 = []
for i in range(-100, 100):
   x.append(math.pi*i/100)
   y1.append(math.cos(math.pi*i/100))#could also use math.cos(x[-1])
   y2.append(math.sin(math.pi*i/100))#could also use math.sin(x[-1])

plt.plot(x, y1, color="blue", linewidth=1.0, linestyle="-")
plt.plot(x, y2, color="green", linewidth=1.0, linestyle="-")
plt.show()
```




Let’s improve it… first a legend

# Creating a Plot - legend

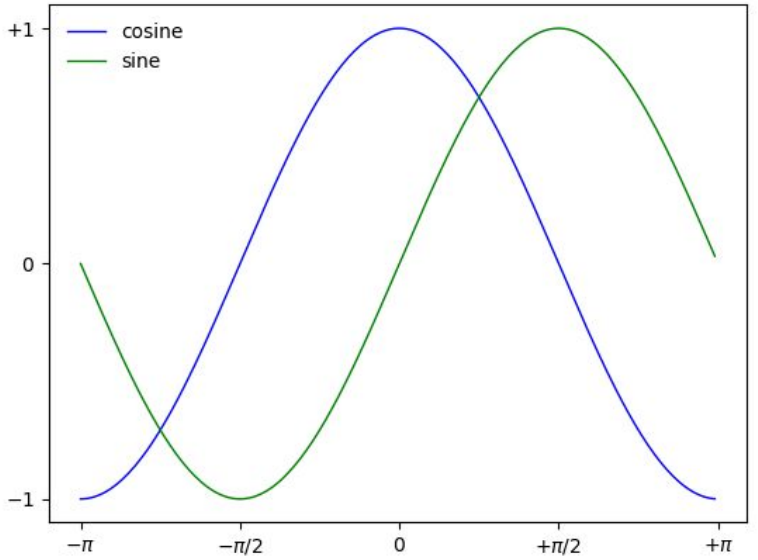
```
import matplotlib.pyplot as plt
import math
x = []
y1 = []
y2 = []
for i in range(-100, 100):
   x.append(math.pi*i/100)
   y1.append(math.cos(math.pi*i/100))
   y2.append(math.sin(math.pi*i/100))

plt.plot(x, y1, color="blue", linewidth=1.0, linestyle="-", label="cosine")
plt.plot(x, y2, color="green", linewidth=1.0, linestyle="-", label="sine")
plt.legend(loc='upper left', frameon=False)
plt.show()
```




Let’s improve it… now for the tick marks

# Creating a Plot - tick marks

```
import matplotlib.pyplot as plt
import math
x = []
y1 = []
y2 = []
for i in range(-100, 100):
   x.append(math.pi*i/100)
   y1.append(math.cos(math.pi*i/100))
   y2.append(math.sin(math.pi*i/100))

plt.plot(x, y1, color="blue", linewidth=1.0, linestyle="-", label="cosine")
plt.plot(x, y2, color="green", linewidth=1.0, linestyle="-", label="sine")
plt.legend(loc='upper left', frameon=False)

plt.xticks([-math.pi, -math.pi/2, 0, math.pi/2, math.pi],
    [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-1, 0, 1], [r'$-1$', r'$0$', r'$+1$'])

plt.show()
```

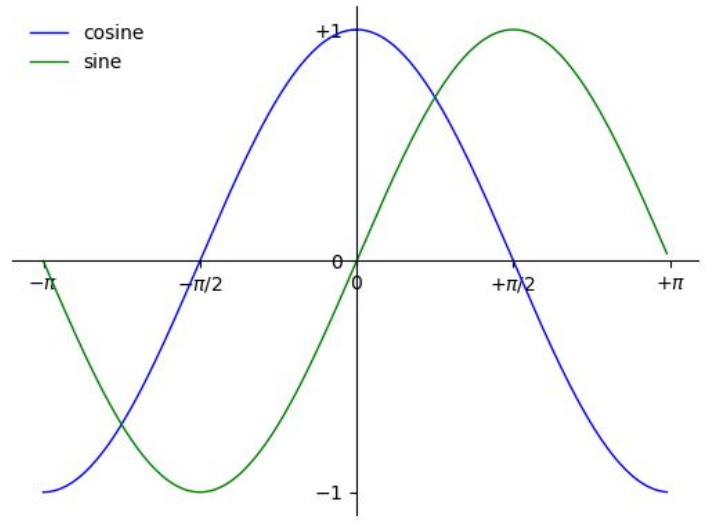



Let's improve it… now for the axes

*Note: you should know how to adjust the ticks, but using r' for raw strings is fancier than we'll test you on*

# Creating a Plot - axes

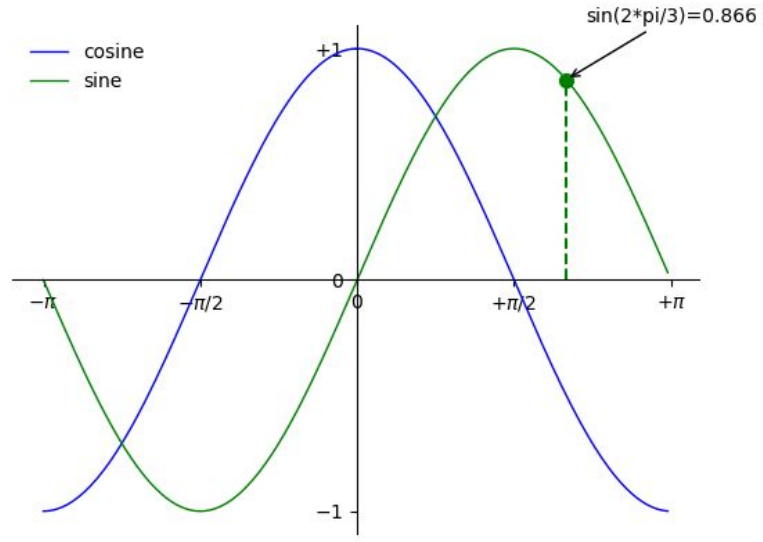
```
import matplotlib.pyplot as plt
import math
x = []
y1 = []
y2 = []
for i in range(-100, 100):
   x.append(math.pi*i/100)
   y1.append(math.cos(math.pi*i/100))
   y2.append(math.sin(math.pi*i/100))

plt.plot(x, y1, color="blue", linewidth=1.0, linestyle="-", label="cosine")
plt.plot(x, y2, color="green", linewidth=1.0, linestyle="-", label="sine")

## Note - I excluded the prior screen axis/legend commands for room ##

ax = plt.gca()
ax.spines['right'].set color('none')
ax.spines['top'].set color('none')
ax.xaxis.set ticks position('bottom')
ax.spines['bottom'].set position(('data', 0))
ax.yaxis.set ticks position('left')
ax.spines['left'].set_position(('data', 0))

plt.show()
```




*Note: gca = "get current axis"*

# Creating a Plot - annotation

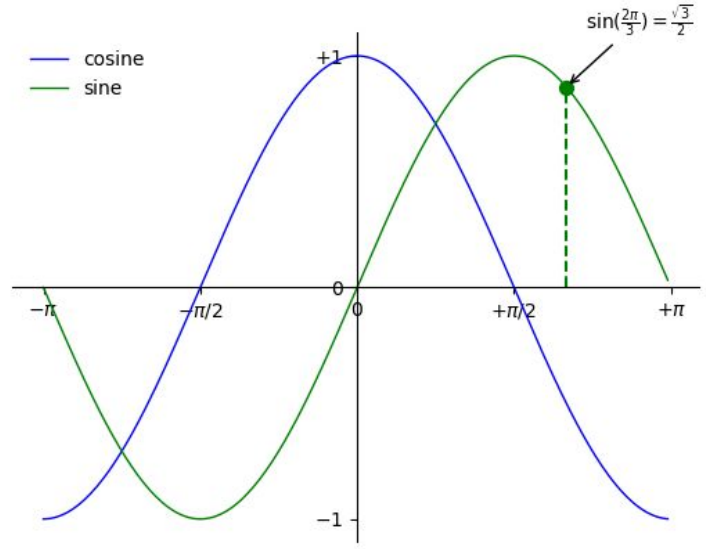
```
import matplotlib.pyplot as plt
import math
x = []
y1 = []
y2 = []
for i in range(-100, 100):
   x.append(math.pi*i/100)
   y1.append(math.cos(math.pi*i/100))
   y2.append(math.sin(math.pi*i/100))

plt.plot(x, y1, color="blue", linewidth=1.0, linestyle="-", label="cosine")
plt.plot(x, y2, color="green", linewidth=1.0, linestyle="-", label="sine")

t = 2*math.pi/3
plt.annotate('sin(2*pi/3)='+str(round(math.sin(t),3)), xy=(t, math.sin(t)),
             xycoords='data', xytext=(+10, +30), textcoords='offset points',
             fontsize=10, arrowprops=dict(arrowstyle="->"))

plt.plot([t,t],[0,math.sin(t)], color ='green', linewidth=1.5,
             linestyle="--")
plt.scatter([t,],[math.sin(t),], 50, color ='green')

plt.show()
```




*Two parts to this one - adding the dotted line, and adding the point with text annotation*

# Creating a Plot - fancier version

```
import matplotlib.pyplot as plt
import math
x = []
y1 = []
y2 = []
for i in range(-100, 100):
   x.append(math.pi*i/100)
   y1.append(math.cos(math.pi*i/100))
   y2.append(math.sin(math.pi*i/100))

plt.plot(x, y1, color="blue", linewidth=1.0, linestyle="-", label="cosine")
plt.plot(x, y2, color="green", linewidth=1.0, linestyle="-", label="sine")

t = 2*math.pi/3
plt.annotate(r'$\sin(\frac{2\pi}{3})$',
             xy=(t, math.sin(t)), xycoords='data',
             xytext=(+10, +30), textcoords='offset points', fontsize=12,
             arrowprops=dict(arrowstyle="->"))

plt.plot([t,t],[0,math.sin(t)], color ='green', linewidth=1.5,
linestyle="--")
plt.scatter([t,],[math.sin(t),], 50, color ='green')

plt.show()
```

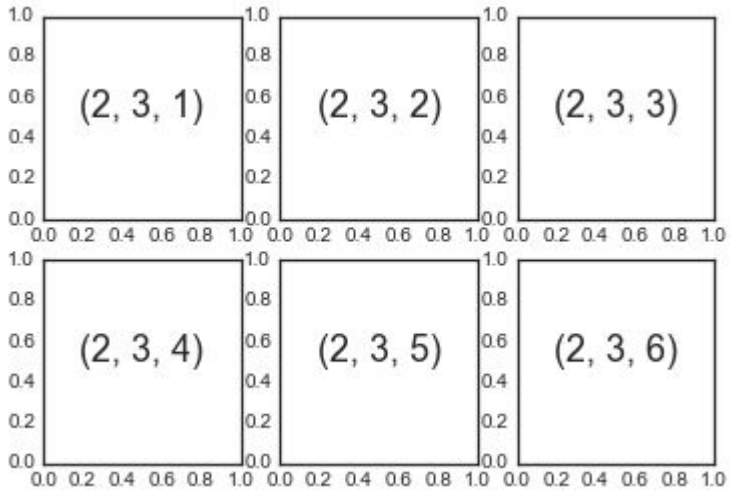



*Note: again, you're not required to know r' commands*

# Other useful options

plt.axis([x_axis_min, x_axis_max, y_axis_min, y_axis_max])

plt.subplot(2,3,1)



# Matplotlib

Remember: always create plots with titles, axis labels, etc.

A good site to reference: https://www.labri.fr/perso/nrougier/teaching/matplotlib/

Lots of examples and a tutorial: https://matplotlib.org/gallery/index.html