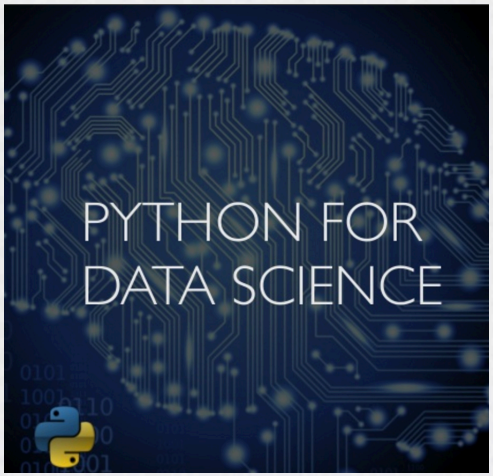
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS



Python for Data Science

Lecture 14 (04/25, 04/27): Visualization in Python

Decision, Operations & Information Technologies
Robert H. Smith School of Business
Spring, 2016

Roadmap

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Introduction: Matplotlib
- Simple plot
- Figures, subplots, axes and ticks
- Other types of plots: examples

Introduction

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Matplotlib is the most used Python package for 2D-graphics.

- **from matplotlib import pyplot as plt**
- **OR**
- **import matplotlib.pyplot as plt**

Simple plot

- In this section, we want to draw the cosine and sine functions on the same plot.
- Starting from the default settings, we will enrich the figure step by step to make it nicer

Get the data ready


```
import numpy as np

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C = np.cos(X)
S = np.sin(X)
```

X is now a numpy array with 256 values ranging from -π to +π (included). C is the cosine (256 values) and S is the sine (256 values).

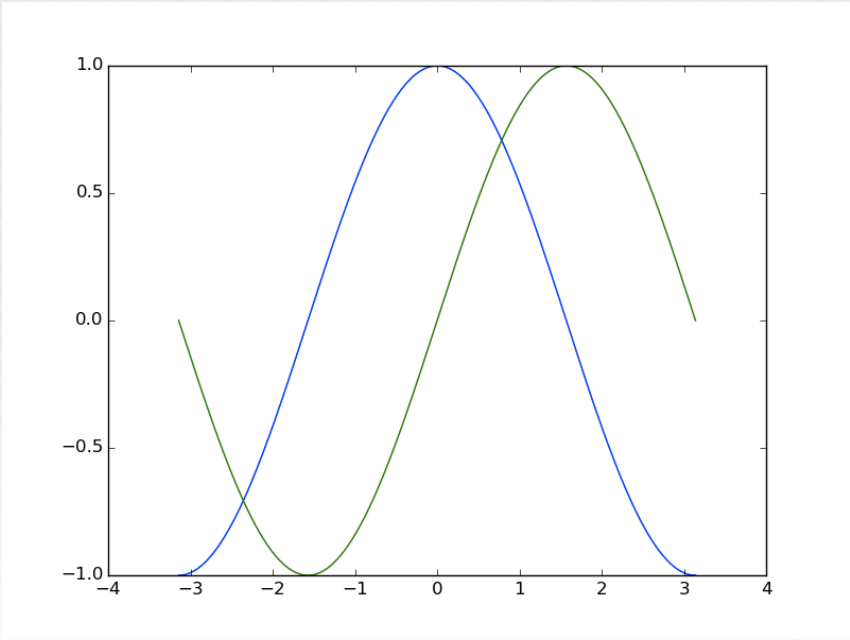
Plot()

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- **matplotlib** comes with a set of default settings that allow customizing all kinds of properties. You can control the defaults of almost every property in matplotlib:
  - figure size
  - dpi
  - line width
  - color and style, axis and grid properties
  - text and font properties
  - And so on.

Plot()


```
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C = np.cos(X)
S = np.sin(X)

plt.plot(X, C)
plt.plot(X, S)
plt.show()
```



Changing colors and line widths

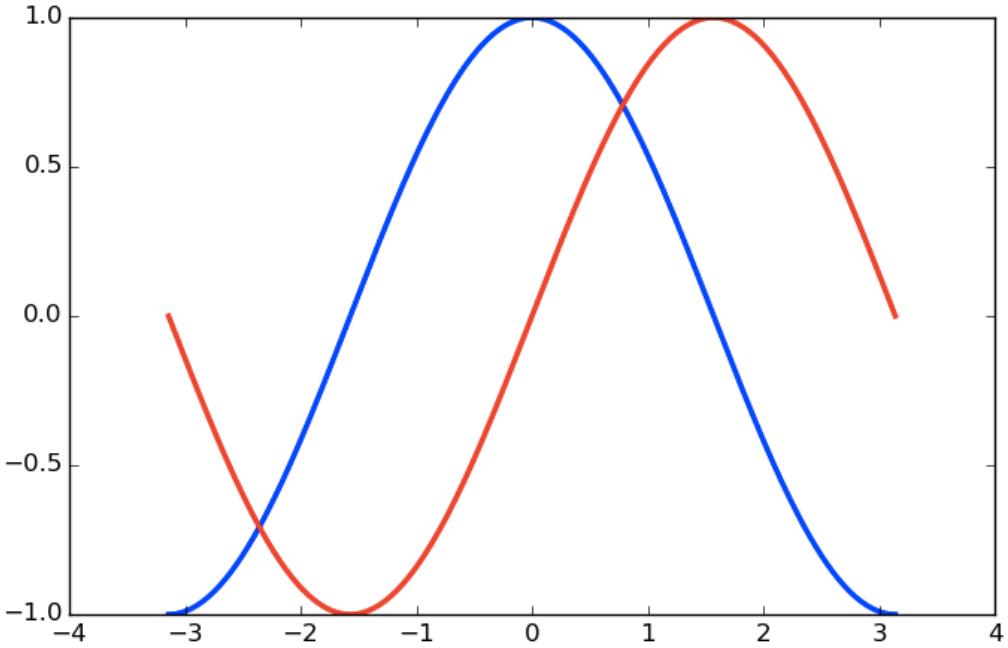
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- We want to have the cosine in blue and the sine in red and a slightly thicker line for both.
- We'll also slightly alter the figure size to make it more horizontal.

Changing colors and line widths

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
...
plt.figure(figsize=(10, 6), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
```



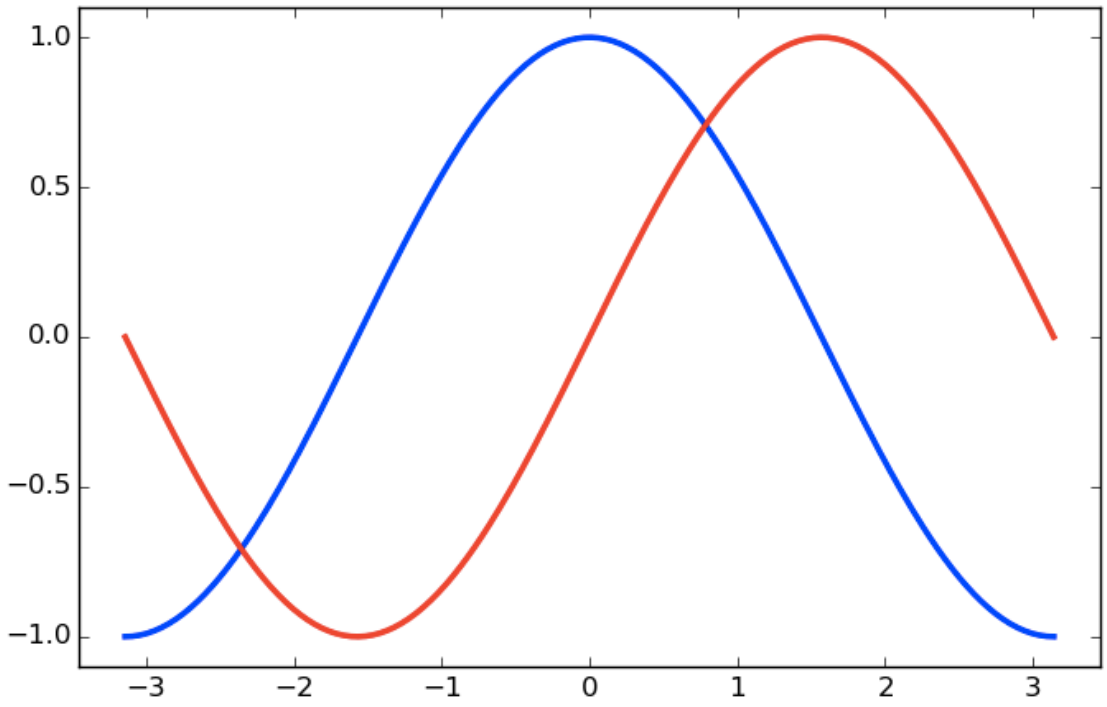
Setting limits

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Current limits of the figure are a bit too tight and we want to make some space in order to clearly see all data points.

Setting limits


```
...
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
```

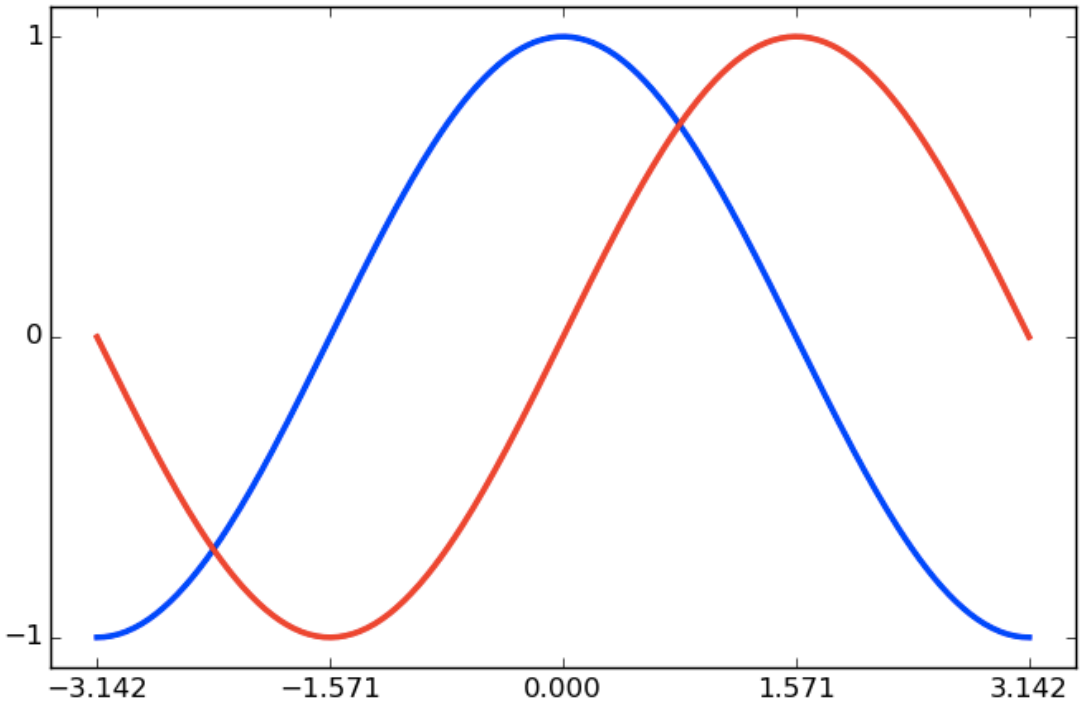


Setting ticks

- Current ticks are not ideal because they do not show the interesting values (+/-π,+/-π/2) for sine and cosine. We'll change them such that they show only these values.

Setting ticks


```
...
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.yticks([-1, 0, +1])
```

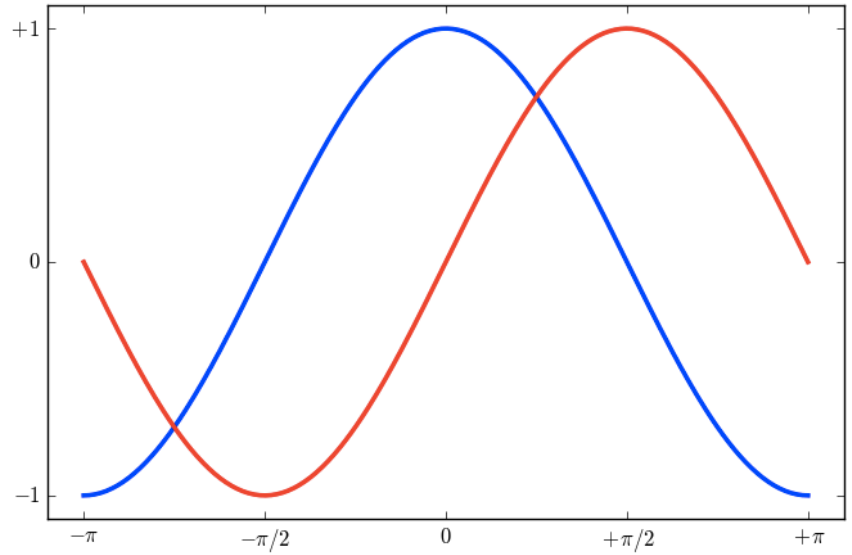


Setting tick labels

- Ticks are now properly placed but their label is not very explicit. We could guess that 3.142 is π but it would be better to make it explicit.
- When we set tick values, we can also provide a corresponding label in the second argument list.
- Note that we'll use latex to allow for nice rendering of the label.

Setting tick labels


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



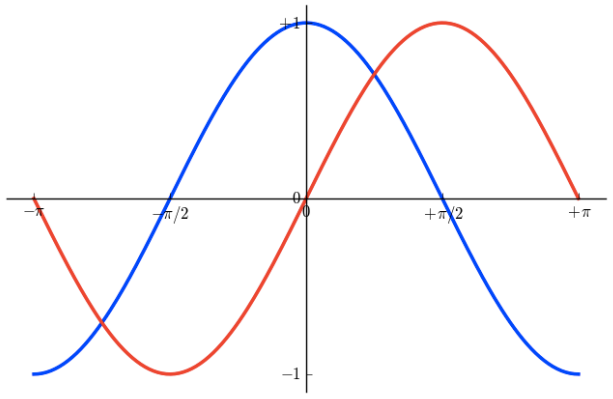
Moving spines

- Spines are the lines connecting the axis tick marks and noting the boundaries of the data area.
- They can be placed at arbitrary positions and until now, they were on the border of the axis. We'll change that since we want to have them in the middle.
- Since there are four of them (top/bottom/left/right), we'll discard the top and right by setting their color to none and we'll move the bottom and left ones to coordinate 0 in data space coordinates.

Moving spines

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
...
ax = plt.gca()  # gca stands for 'get current axis'
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))
```



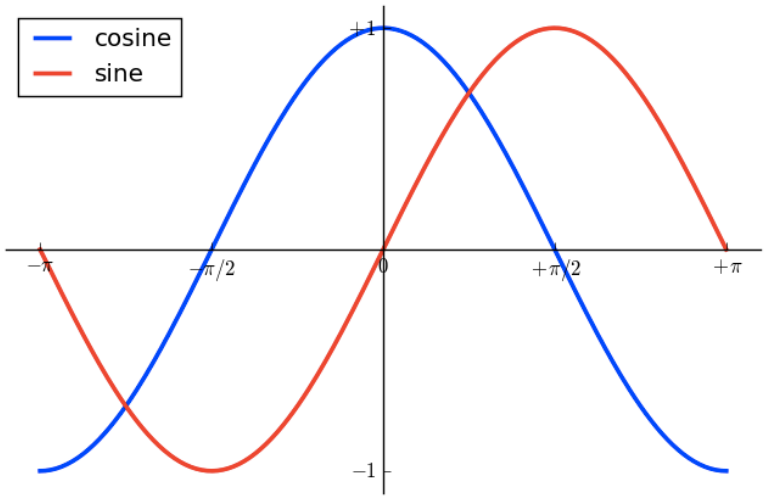
Adding a legend

- Let's add a legend in the upper left corner.
- It only requires adding the keyword argument label to the plot commands.

Adding a legend

...

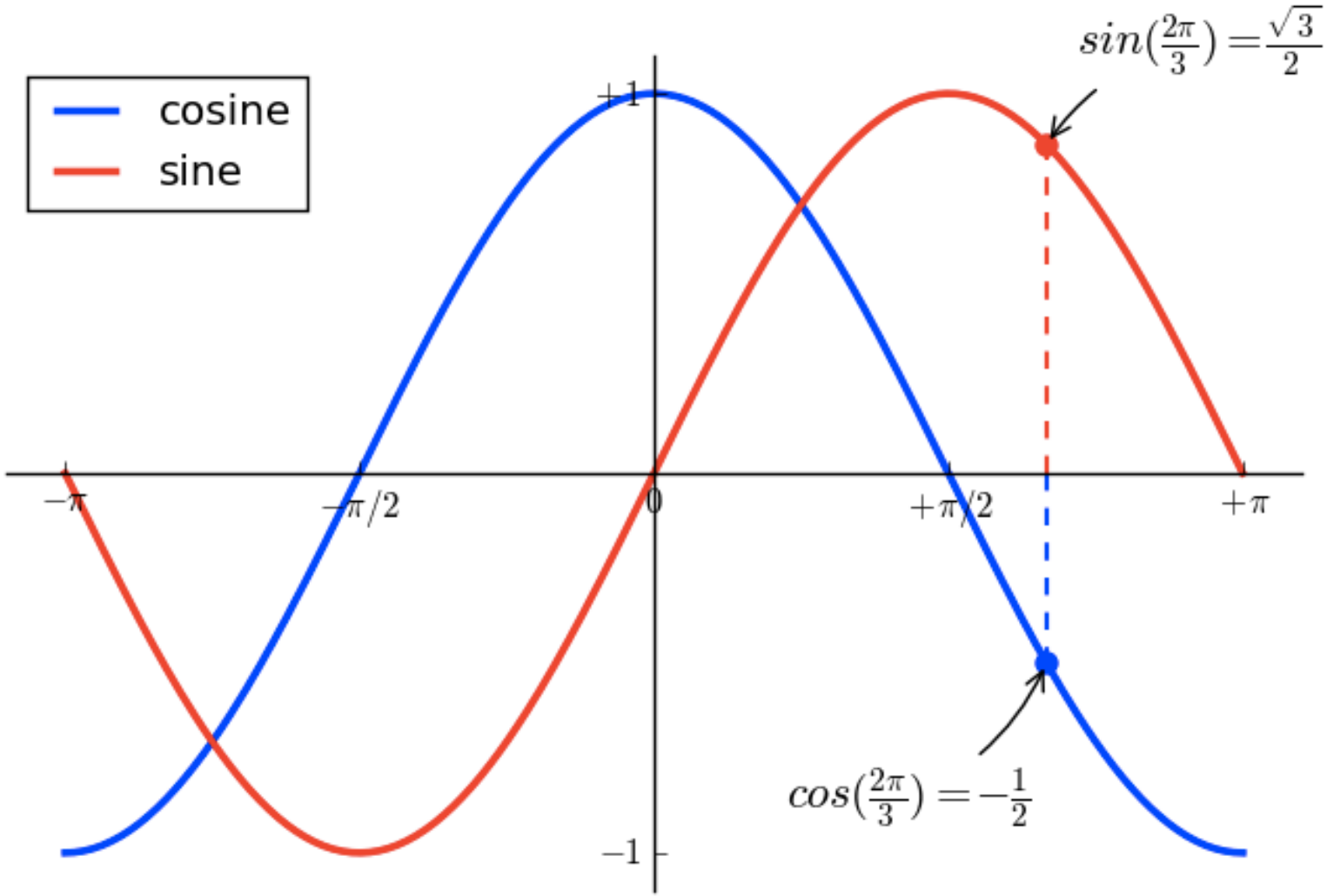
```
plt.plot(X, C, color="blue", linewidth=2.5,
linestyle="-", label="cosine")
plt.plot(X, S, color="red",  linewidth=2.5,
linestyle="-", label="sine”)
plt.legend(loc='upper left’)
```



Annotate some points

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Let's annotate some interesting points using the annotate command. We chose the $2\pi/3$ value and we want to annotate both the sine and the cosine. We'll first draw a marker on the curve as well as a straight dotted line. Then, we'll use the annotate command to display some text with an arrow.


cosine
sine
$sin(\frac{2\pi}{3}) = \frac{\sqrt{3}}{2}$
$cos(\frac{2\pi}{3}) = -\frac{1}{2}$
+1
−1
0
$-\pi$
$-\pi/2$
$+\pi/2$
$+\pi$

Annotate some points

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
…
t = 2 * np.pi / 3
plt.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyle="--")
plt.scatter([t, ], [np.cos(t), ], 50, color='blue')

plt.annotate(r'$cos(\frac{2\pi}{3})=-\frac{1}{2}$',
             xy=(t, np.cos(t)), xycoords='data',
             xytext=(-90, -50), textcoords='offset points', fontsize=16,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3, rad=.2"))

plt.plot([t, t],[0, np.sin(t)], color='red', linewidth=2.5, linestyle="--")
plt.scatter([t, ],[np.sin(t), ], 50, color='red')

plt.annotate(r'$sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',
             xy=(t, np.sin(t)), xycoords='data',
             xytext=(+10, +30), textcoords='offset points', fontsize=16,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3, rad=.2"))
```

Devils is in the details

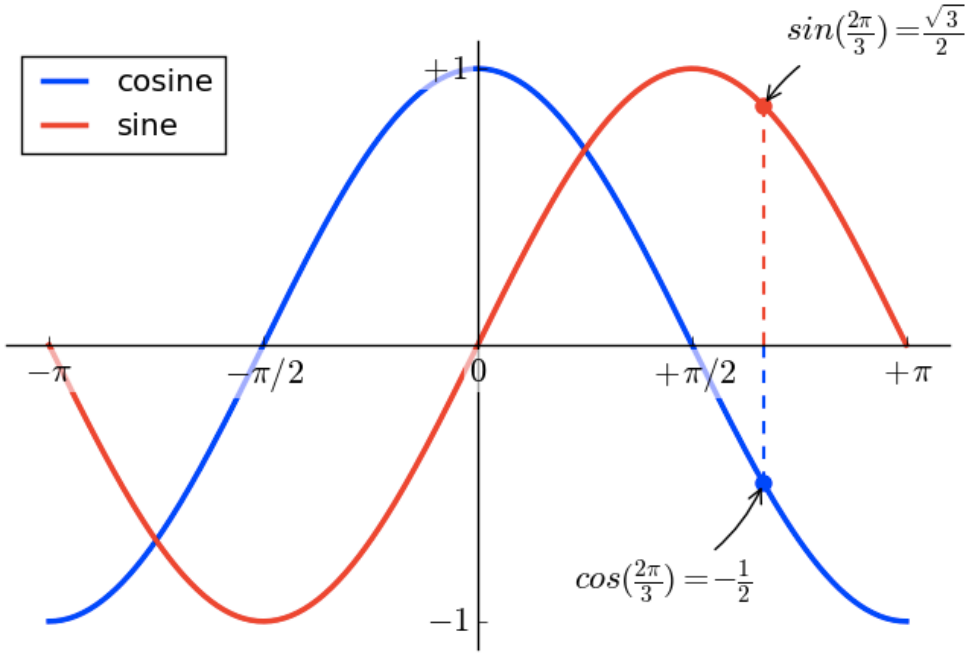
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- The tick labels are now hardly visible because of the blue and red lines. We can make them bigger and we can also adjust their properties such that they'll be rendered on a semi-transparent white background. This will allow us to see both the data and the labels.

Devil is in the details

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
...
for label in ax.get_xticklabels() + ax.get_yticklabels():
    label.set_fontsize(16)
    label.set_bbox(dict(facecolor='white', edgecolor='None',
alpha=0.65))
```



Figures, subplots, axes and ticks

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- A "**figure**" in matplotlib means the whole window in the user interfact. Within the figure there can be "**subplots**".
- A figure is the windows in the GUI that has "Figure #" as title. Figures are numbered starting from 1.
- There are several parameters that determine what the figure looks like.

Figures

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

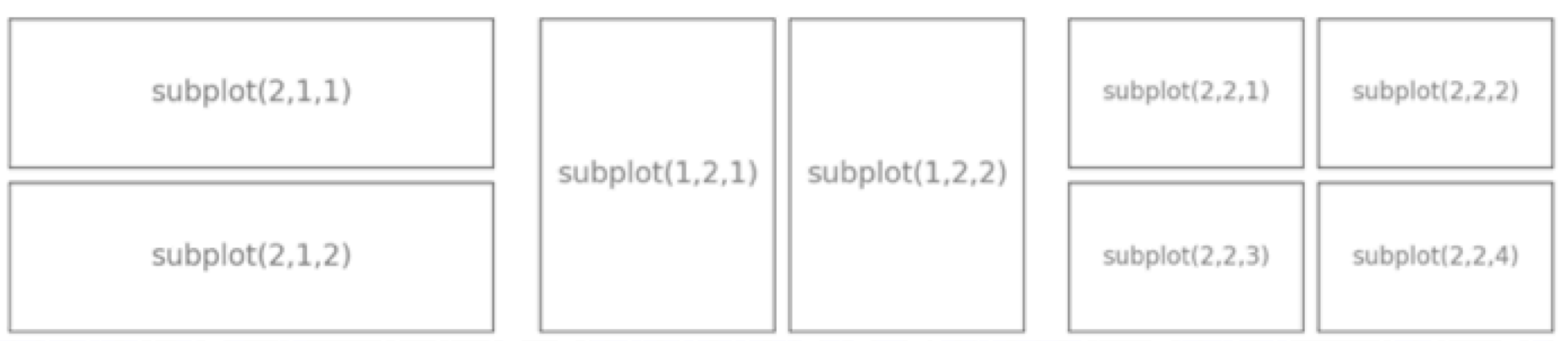
Argument	Default	Description
num	1	number of figure
figsize	figure.figsize	figure size in in inches (width, height)
dpi	figure.dpi	resolution in dots per inch
facecolor	figure.facecolor	color of the drawing background
edgecolor	figure.edgecolor	color of edge around the drawing background
frameon	True	draw figure frame or not

- You can close a figure programmatically by calling close. Depending on the argument it closes (1) the current figure (no argument), (2) a specific figure (figure number or figure instance as argument), or (3) all figures ("all" as argument).
- plt.close(1) # close figure 1

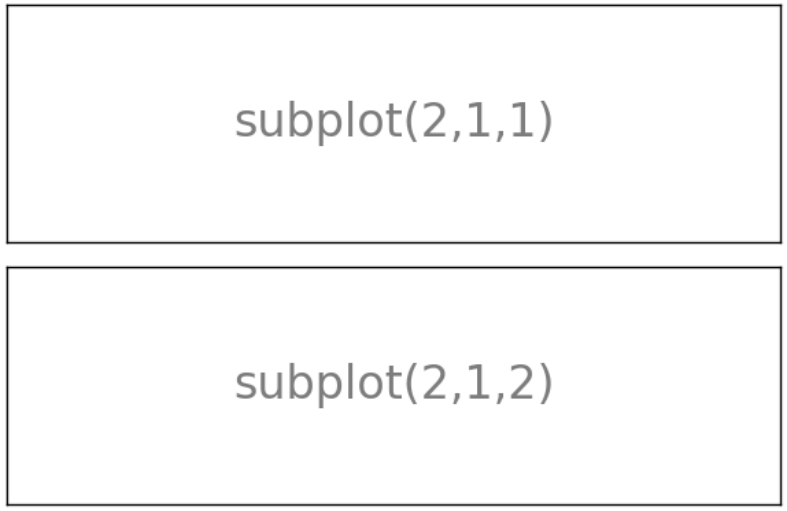
Subplots

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- With subplot you can arrange plots in a regular grid. You need to specify the number of rows and columns and the number of the plot.



Subplots

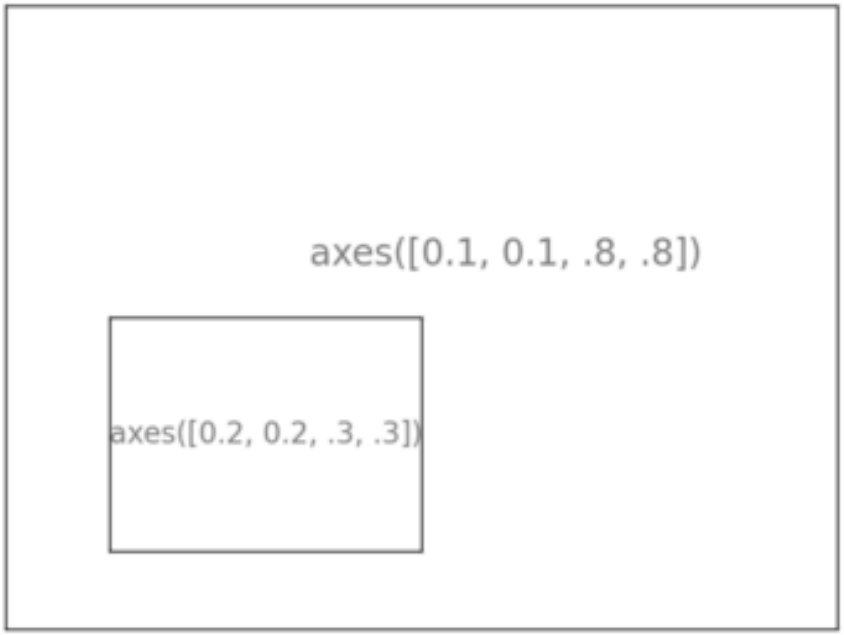


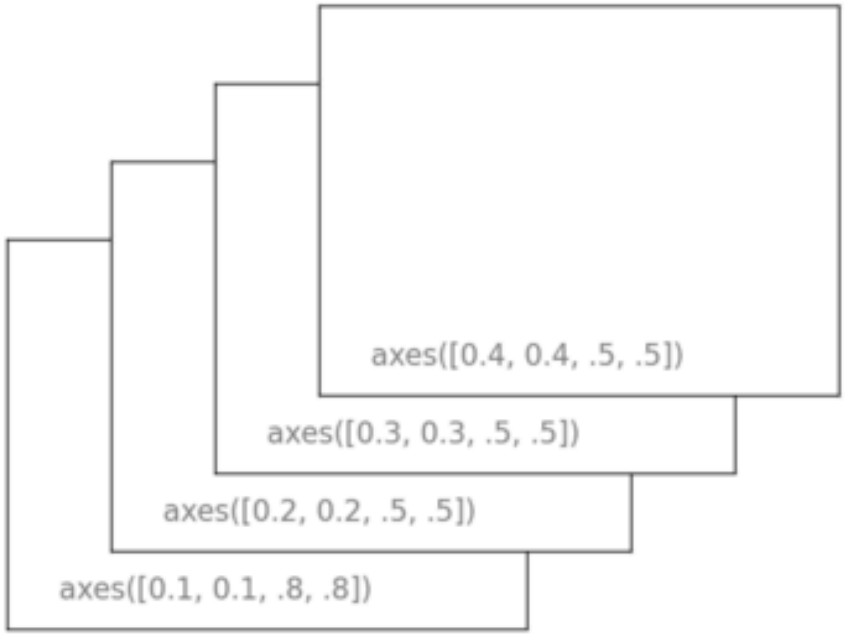

```
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))
plt.subplot(2, 1, 1)
plt.xticks(())
plt.yticks(())
plt.text(0.5, 0.5, 'subplot(2,1,1)', ha='center', va='center', size=24, alpha=.5)

plt.subplot(2, 1, 2)
plt.xticks(())
plt.yticks(())
plt.text(0.5, 0.5, 'subplot(2,1,2)', ha='center', va='center', size=24, alpha=.5)

plt.tight_layout()
plt.show()
```

Axes

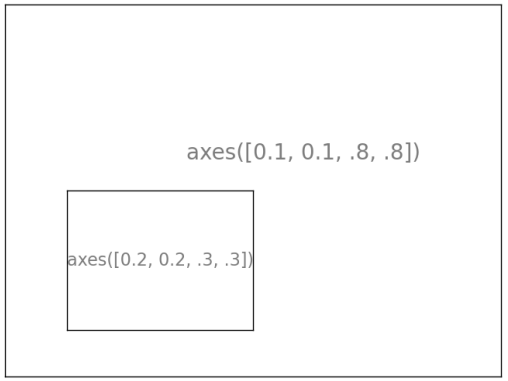
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Axes are very similar to subplots but allow placement of plots at any location in the figure. So if we want to put a smaller plot inside a bigger one we do so with axes.





Axes



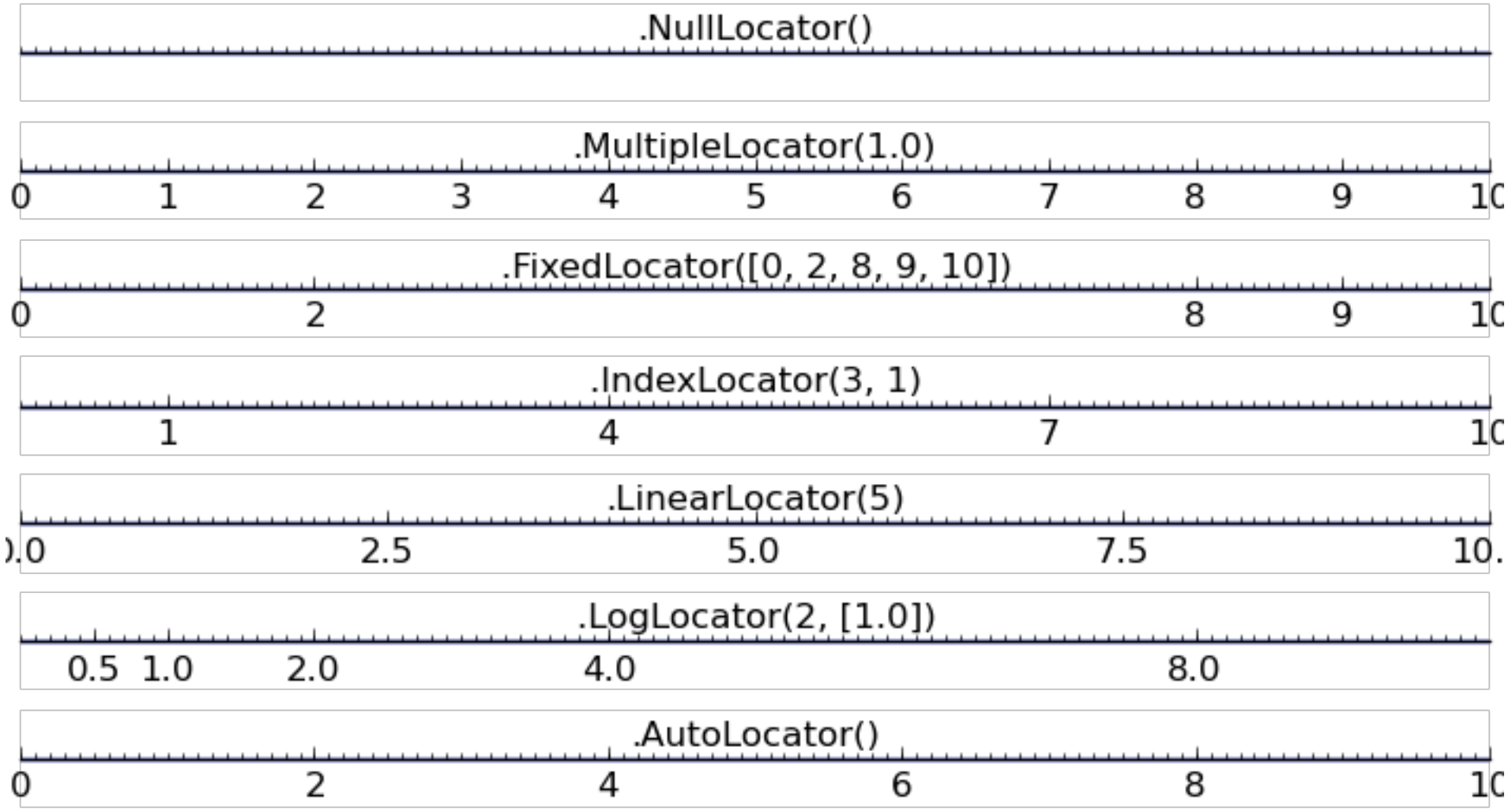

```
import matplotlib.pyplot as plt

plt.axes([.1, .1, .8, .8])
plt.xticks(())
plt.yticks(())
plt.text(.6, .6, 'axes([0.1, 0.1, .8, .8])', ha='center', va='center', size=20, alpha=.5)

plt.axes([.2, .2, .3, .3])
plt.xticks(())
plt.yticks(())
plt.text(.5, .5, 'axes([0.2, 0.2, .3, .3])', ha='center', va='center', size=16, alpha=.5)

plt.show()
```

Ticks

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- There are tick locators to specify where ticks should appear and tick formatters to give ticks the appearance you want.
- Major and minor ticks can be located and formatted independently from each other.
- Tick locators control the positions of the ticks. They are set as follows:

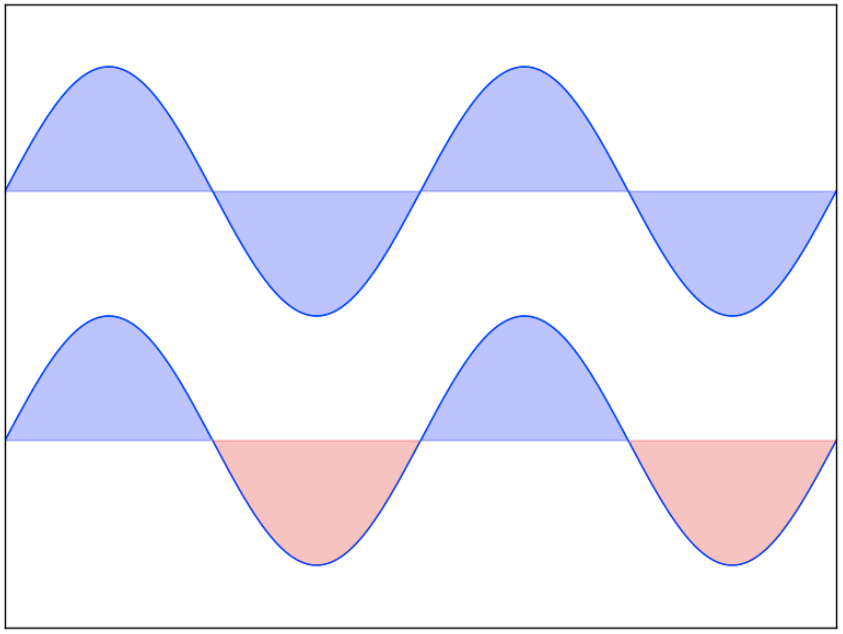
```
ax = plt.gca()
ax.xaxis.set_major_locator(locator)
```


.NullLocator()
.MultipleLocator(1.0)
0 1 2 3 4 5 6 7 8 9 10
.FixedLocator([0, 2, 8, 9, 10])
0 2 8 9 10
.IndexLocator(3, 1)
1 4 7 10
.LinearLocator(5)
0.0 2.5 5.0 7.5 10.0
.LogLocator(2, [1.0])
0.5 1.0 2.0 4.0 8.0
.AutoLocator()
0 2 4 6 8 10

Other types of plots: regular

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

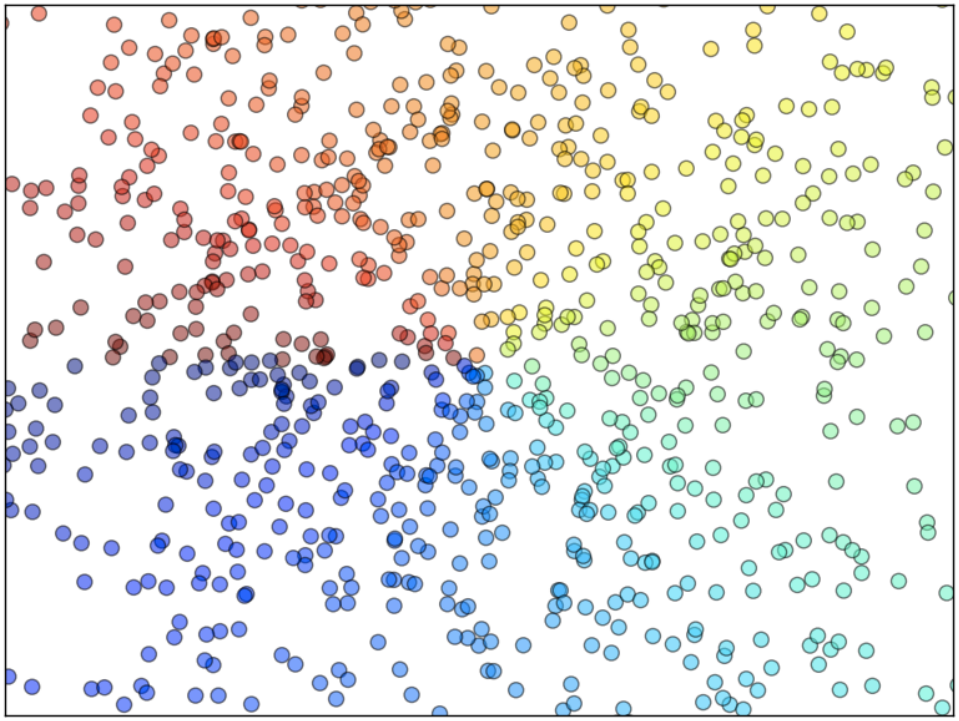
```
import numpy as np
import matplotlib.pyplot as plt

n = 256
X = np.linspace(-np.pi, np.pi, n, endpoint=True)
Y = np.sin(2 * X)

plt.axes([0.025, 0.025, 0.95, 0.95])

plt.plot(X, Y + 1, color='blue', alpha=1.00)
plt.fill_between(X, 1, Y + 1, color='blue', alpha=.25)

plt.plot(X, Y - 1, color='blue', alpha=1.00)
plt.fill_between(X, -1, Y - 1, (Y - 1) > -1, color='blue', alpha=.25)
plt.fill_between(X, -1, Y - 1, (Y - 1) < -1, color='red',  alpha=.25)

plt.xlim(-np.pi, np.pi)
plt.xticks(())
plt.ylim(-2.5, 2.5)
plt.yticks(())

plt.show()
```



Other types of plots: Scatter

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

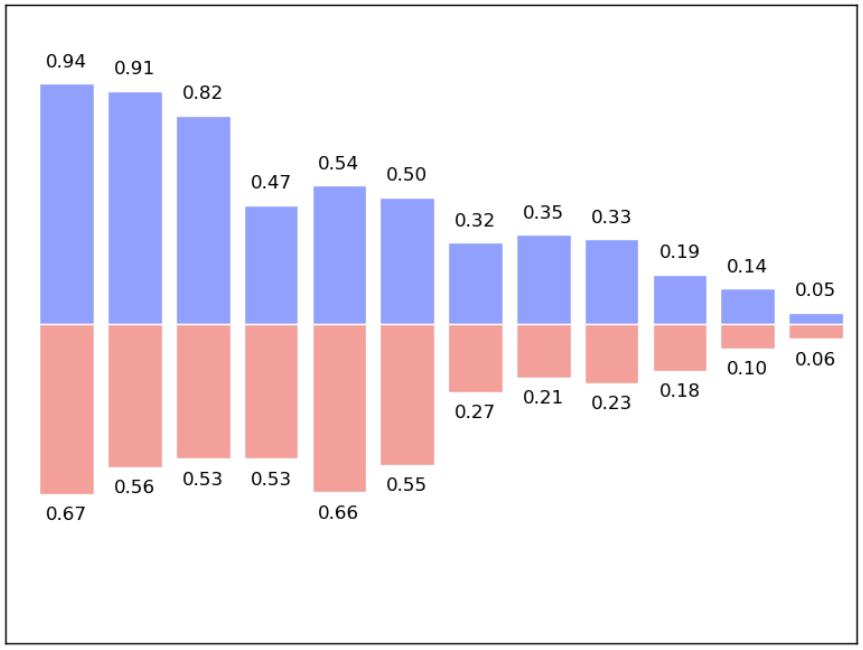
```
import numpy as np
import matplotlib.pyplot as plt

n = 1024
X = np.random.normal(0, 1, n)
Y = np.random.normal(0, 1, n)
T = np.arctan2(Y, X)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.scatter(X, Y, s=75, c=T, alpha=.5)

plt.xlim(-1.5, 1.5)
plt.xticks(())
plt.ylim(-1.5, 1.5)
plt.yticks(())

plt.show()
```



Other types of plots: Bar

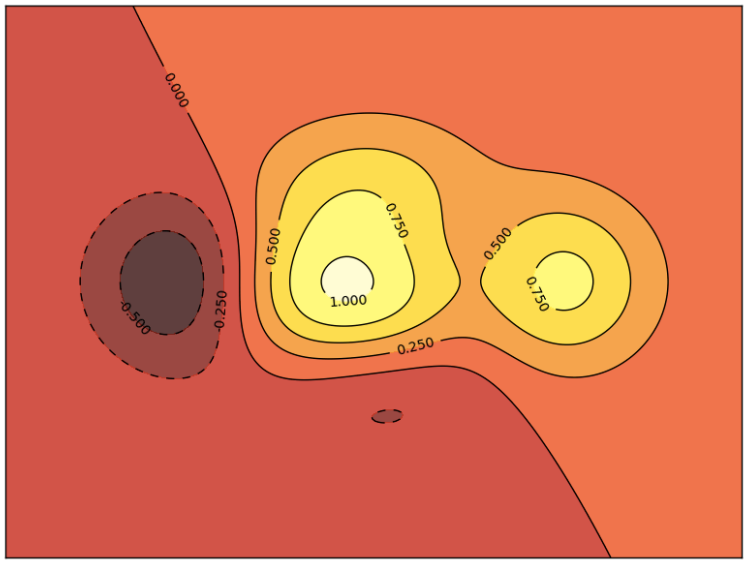
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
import numpy as np
import matplotlib.pyplot as plt

n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.bar(X, +Y1, facecolor='#9999ff', edgecolor='white')
plt.bar(X, -Y2, facecolor='#ff9999', edgecolor='white')

for x, y in zip(X, Y1):
    plt.text(x + 0.4, y + 0.05, '%.2f' % y, ha='center', va= 'bottom')

for x, y in zip(X, Y2):
    plt.text(x + 0.4, -y - 0.05, '%.2f' % y, ha='center', va= 'top')

plt.xlim(-.5, n)
plt.xticks(())
plt.ylim(-1.25, 1.25)
plt.yticks(())
plt.show()
```

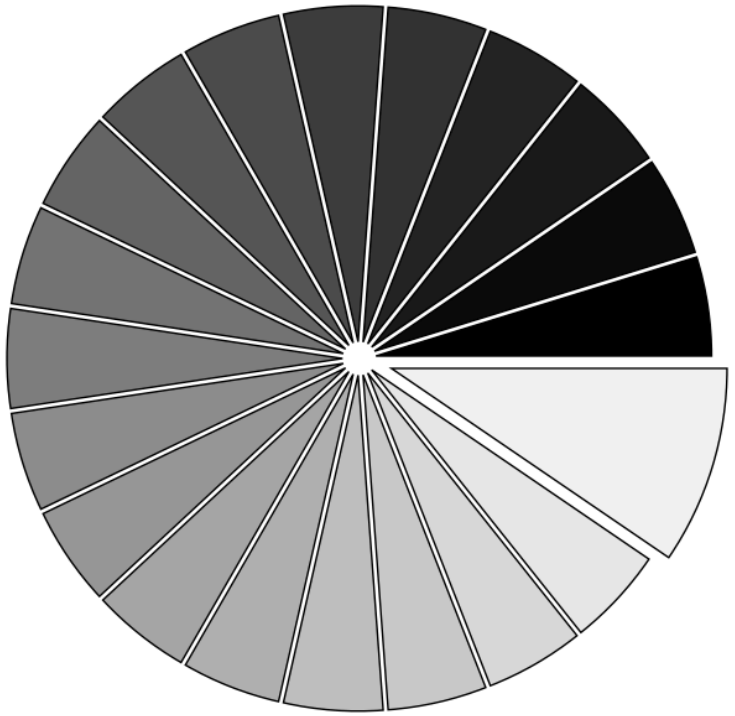


Other types of plots: Contour


```
import numpy as np
import matplotlib.pyplot as plt

def f(x,y):
    return (1 - x / 2 + x**5 + y**3) * np.exp(-x**2 -y**2)

n = 256
x = np.linspace(-3, 3, n)
y = np.linspace(-3, 3, n)
X,Y = np.meshgrid(x, y)

plt.axes([0.025, 0.025, 0.95, 0.95])

plt.contourf(X, Y, f(X, Y), 8, alpha=.75, cmap=plt.cm.hot)
C = plt.contour(X, Y, f(X, Y), 8, colors='black', linewidth=.5)
plt.clabel(C, inline=1, fontsize=10)

plt.xticks(())
plt.yticks(())
plt.show()
```

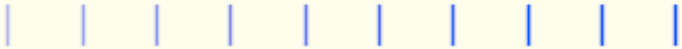
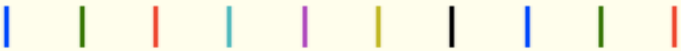
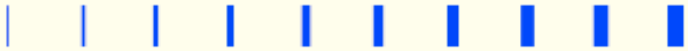
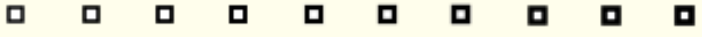


Other types of plots: Pie chart

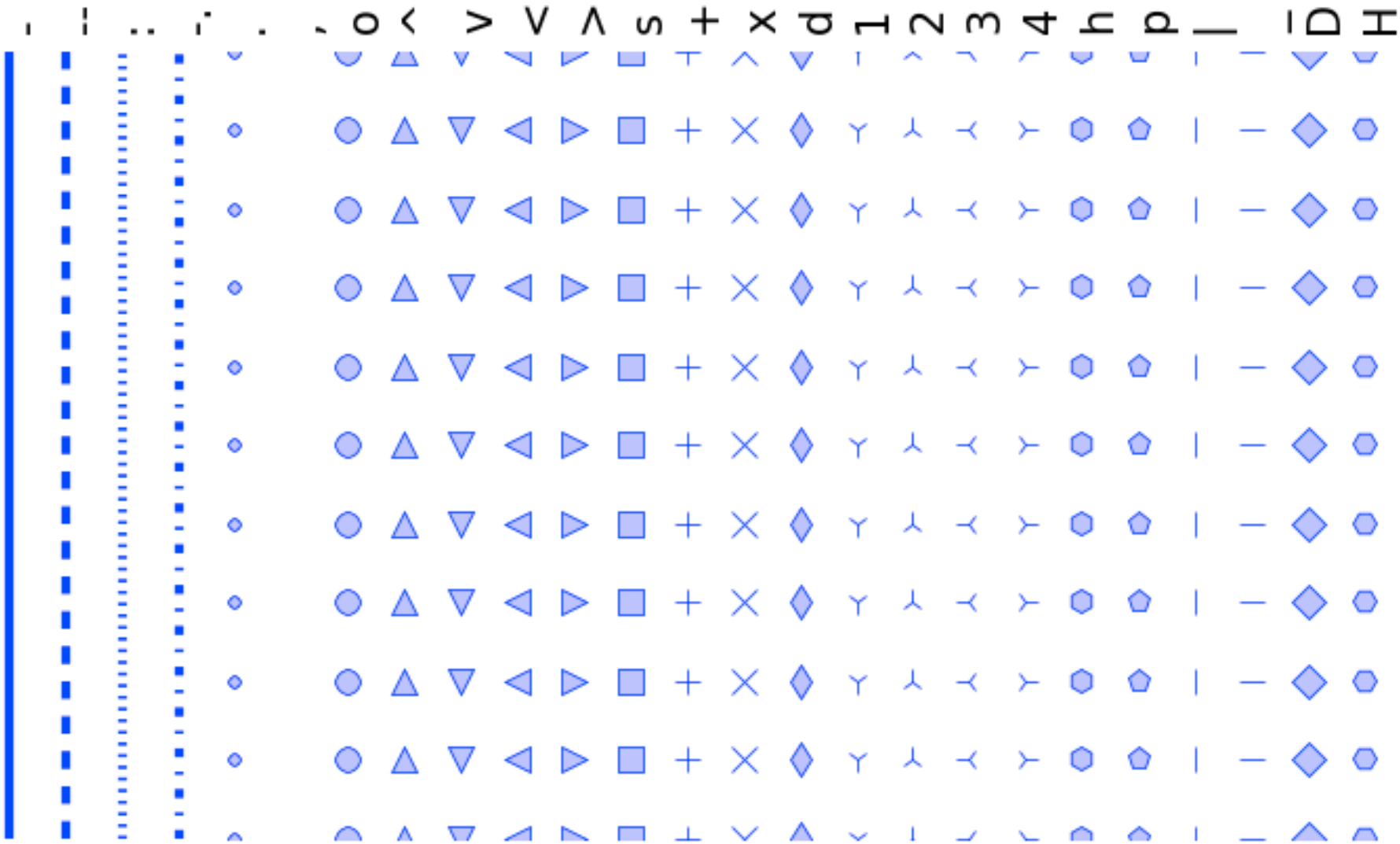


```
import numpy as np
import matplotlib.pyplot as plt

n = 20
Z = np.ones(n)
Z[-1] *= 2

plt.axes([0.025, 0.025, 0.95, 0.95])

plt.pie(Z, explode=Z*.05, colors = ['%f' % (i/float(n)) for i in range(n)])
plt.axis('equal')
plt.xticks(())
plt.yticks()

plt.show()
```

Line properties

Property	Description	Appearance
alpha (or a)	alpha transparency on 0-1 scale	
antialiased	True or False - use antialised rendering	
color (or c)	matplotlib color arg	
linestyle (or ls)	see Line properties	
linewidth (or lw)	float, the line width in points	
solid_capstyle	Cap style for solid lines	
solid_joinstyle	Join style for solid lines	
dash_capstyle	Cap style for dashes	
dash_joinstyle	Join style for dashes	
marker	see Markers	
markeredgewidth (mew)	line width around the marker symbol	
markeredgecolor (mec)	edge color if a marker is used	
markerfacecolor (mfc)	face color if a marker is used	
markersize (ms)	size of the marker in points	

Line styles



Markers