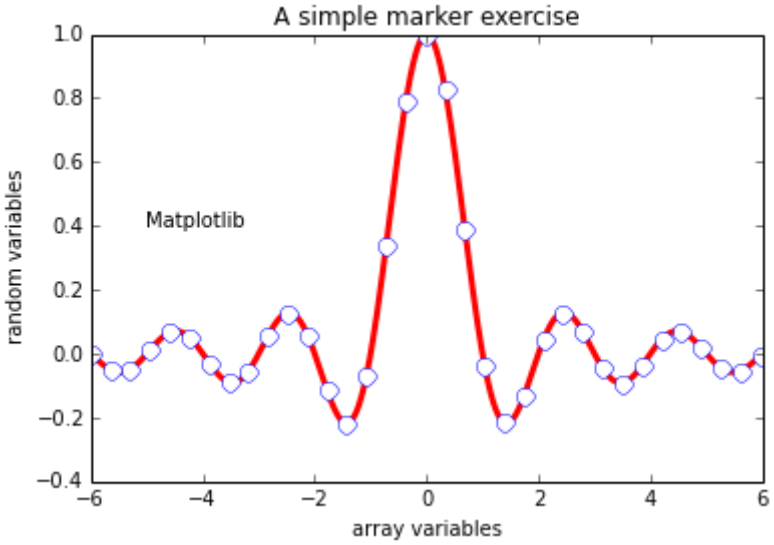
Annotation

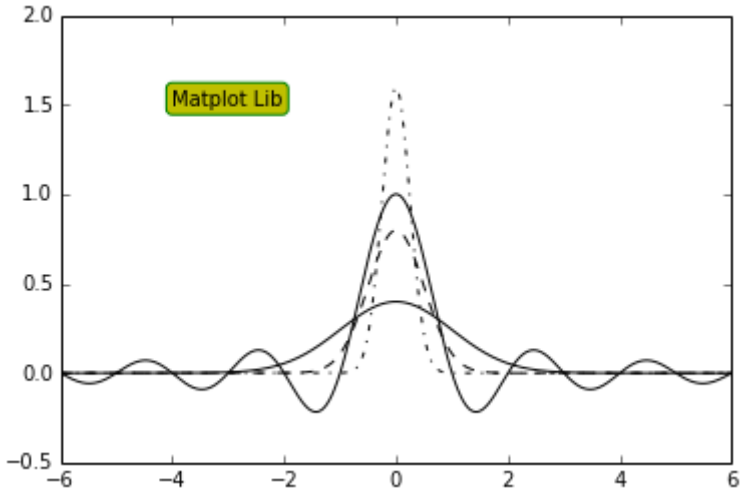
In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [28]:
```
X =np.linspace(-6,6, 1024)
Y =np.sinc(X)
plt.title('A simple marker exercise')# a title notation
plt.xlabel('array variables') # adding xlabel
plt.ylabel(' random variables') # adding ylabel
plt.text(-5, 0.4, 'Matplotlib') # -5 is the x value and 0.4 is y value
plt.plot(X,Y, color ='r', marker ='o', markersize =9, markevery = 30, r
```

Out[28]:
```
[<matplotlib.lines.Line2D at 0x84b6430>]
```



In [39]:
```
def pq(I, mu, sigma):
    a = 1. / (sigma * np.sqrt(2. * np.pi))
    b = -1. / (2. * sigma ** 2)
    return a * np.exp(b * (I - mu) ** 2)

I =np.linspace(-6,6, 1024)

plt.plot(I, pq(I, 0., 1.), color = 'k', linestyle ='solid')
plt.plot(I, pq(I, 0., .5), color = 'k', linestyle ='dashed')
plt.plot(I, pq(I, 0., .25), color = 'k', linestyle ='dashdot')

# I have created a dictinary of styles
design = {
'facecolor' : 'y', # color used for the text box
'edgecolor' : 'g',
'boxstyle' : 'round'
}
plt.text(-4, 1.5, 'Matplot Lib', bbox = design)
plt.plot(X, Y, c='k')
plt.show()

#This sets the style of the box, which can either be 'round' or 'square
#'pad': If 'boxstyle' is set to 'square', it defines the amount of pad
```



Alignment Control

The text is bound by a box. This box is used to relatively align the text to the coordinates passed to pyplot.text(). Using the verticalalignment and horizontalalignment parameters (respective shortcut equivalents are va and ha), we can control how the alignment is done.

The vertical alignment options are as follows:
'center': This is relative to the center of the textbox
'top': This is relative to the upper side of the textbox
'bottom': This is relative to the lower side of the textbox
'baseline': This is relative to the text's baseline

Horizontal alignment options are as follows:

align ='bottom' align ='baseline'

----------align = center----------

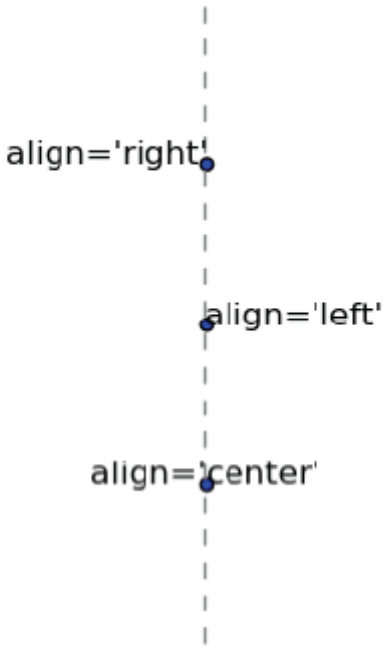
align= 'top

In [41]:
```
cd C:\Users\tk\Desktop
```

```
C:\Users\tk\Desktop
```

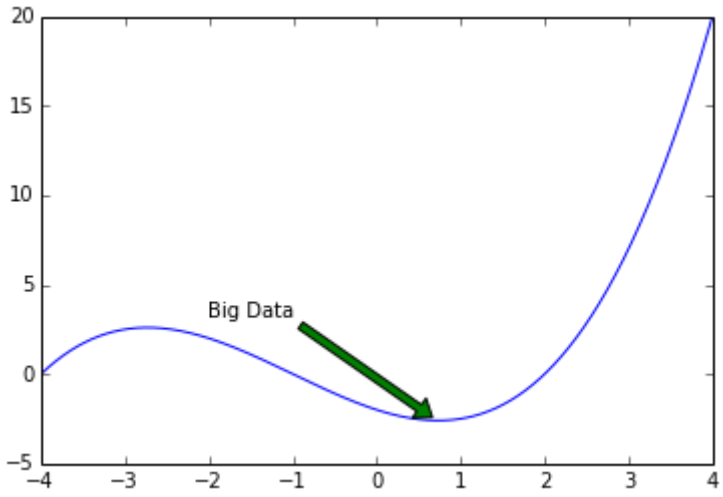
In [44]:
```
from IPython.display import Image
Image(filename='text alignment.png')
#The horizontal alignment options are as follows:
#'center': This is relative to the center of the textbox
#'left': This is relative to the left side of the textbox
#'right': This is relative to the right-hand side of the textbox
```

Out[44]:



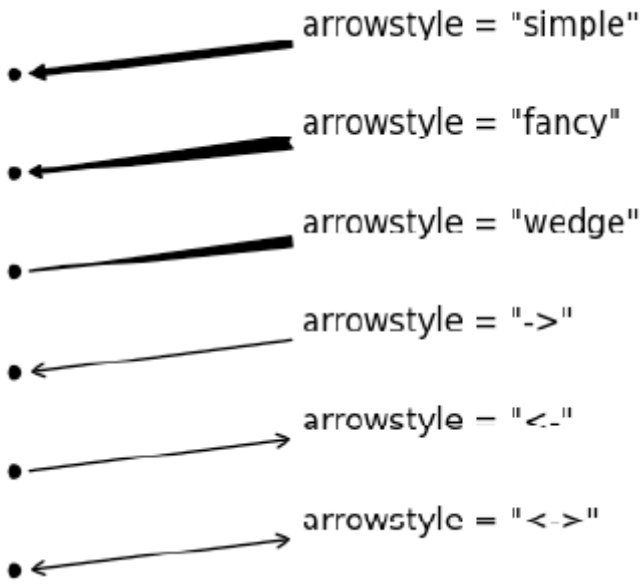
In [76]:
```
X = np.linspace(-4, 4, 1024)
Y = .25 * (X + 4.) * (X + 1.) * (X - 2.)

plt.annotate('Big Data',
ha ='center', va ='bottom',
xytext =(-1.5, 3.0), xy =(0.75, -2.7),
arrowprops ={'facecolor': 'green', 'shrink':0.05, 'edgecolor': 'black'}
plt.plot(X, Y)
```

Out[76]:
```
[<matplotlib.lines.Line2D at 0x9d1def0>]
```



In [74]:
```
#arrow styles are :

from IPython.display import Image
Image(filename='arrows.png')
```

Out[74]:



Legend properties:
'loc': This is the location of the legend. The default value is 'best', which will place it automatically. Other valid values are
'upper left', 'lower left', 'lower right', 'right', 'center left', 'center right', 'lower center', 'upper center', and 'center'.

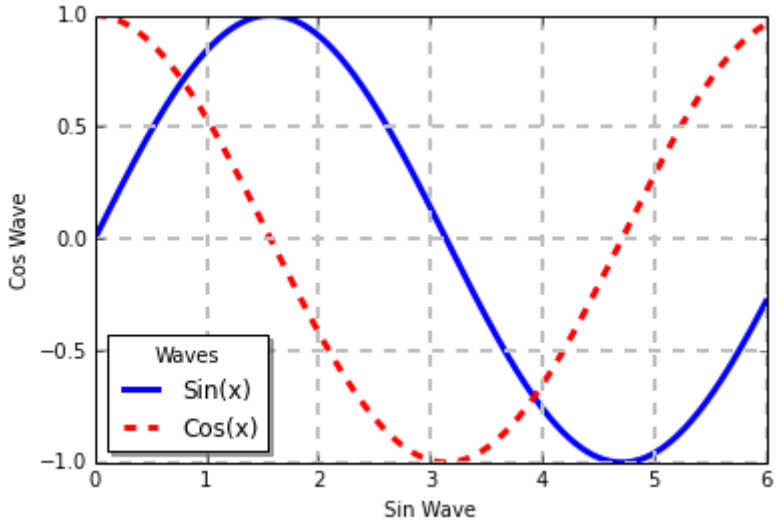
'shadow': This can be either True or False, and it renders the legend with a shadow effect.

'fancybox': This can be either True or False and renders the legend with a rounded box.

'title': This renders the legend with the title passed as a parameter.

'ncol': This forces the passed value to be the number of columns for the legend

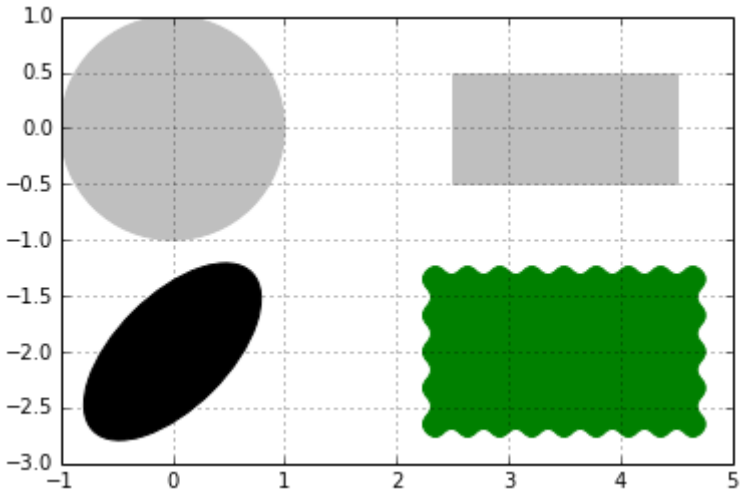
In [101]:
```
x =np.linspace(0, 6,1024)
y1 =np.sin(x)
y2 =np.cos(x)
plt.xlabel('Sin Wave')
plt.ylabel('Cos Wave')
plt.plot(x, y1, c='b', lw =3.0, label ='Sin(x)') # labels are specifie
plt.plot(x, y2, c ='r', lw =3.0, ls ='--', label ='Cos(x)')
plt.legend(loc ='best', shadow = True, fancybox = False, title ='Waves'
plt.grid(True, lw = 2, ls ='--', c='.75') # adds grid lines to the figu
plt.show()
```



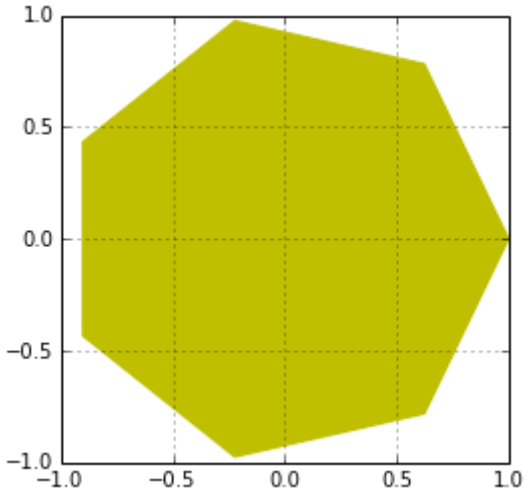
Shapes

In [4]:

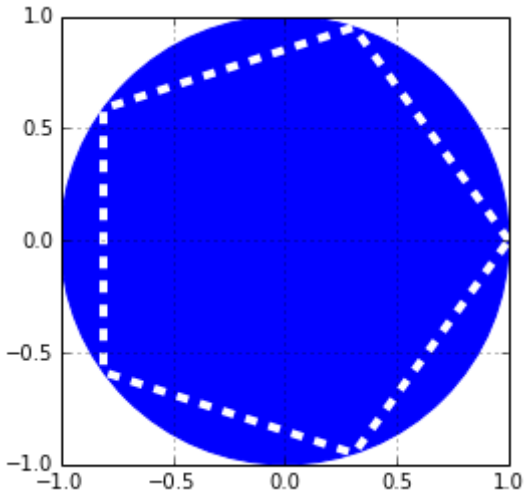
```
#Paths for several kinds of shapes are available in the matplotlib.patc
import matplotlib.patches as patches

dis = patches.Circle((0,0), radius = 1.0, color ='.75' )
plt.gca().add_patch(dis) # used to render the image.

dis = patches.Rectangle((2.5, -.5), 2.0, 1.0, color ='.75') #patches.re
plt.gca().add_patch(dis)

dis = patches.Ellipse((0, -2.0), 2.0, 1.0, angle =45, color ='.00')
plt.gca().add_patch(dis)

dis = patches.FancyBboxPatch((2.5, -2.5), 2.0, 1.0, boxstyle ='roundtoo
plt.gca().add_patch(dis)

plt.grid(True)
plt.axis('scaled') # displays the images within the prescribed axis
plt.show()

#FancyBox: This is like a rectangle but takes an additional boxstyle pa
#(either 'larrow', 'rarrow', 'round', 'round4', 'roundtooth', 'sawtooth
```



In [22]:
```
import matplotlib.patches as patches
theta = np.linspace(0, 2 * np.pi, 8) # generates an array
vertical = np.vstack((np.cos(theta), np.sin(theta))).transpose() # vert
#print vertical, print and see how the array looks
plt.gca().add_patch(patches.Polygon(vertical, color ='y'))
plt.axis('scaled')
plt.grid(True)
plt.show()

#The matplotlib.patches.Polygon()constructor takes a list of coordinate
```

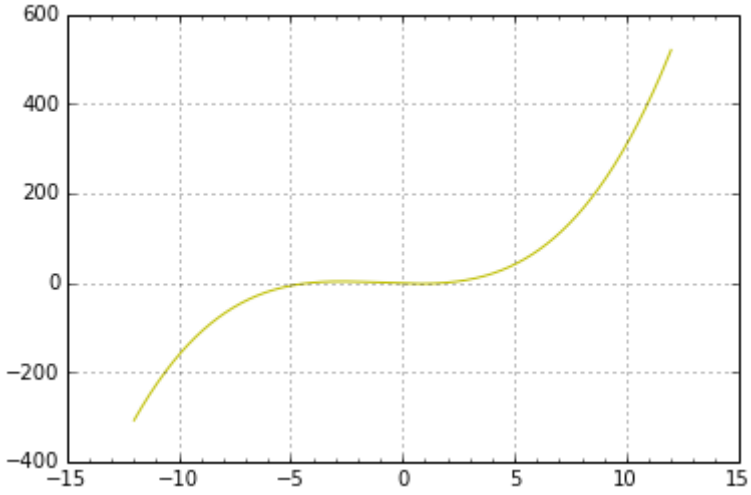


In [34]:
```
# a polygon can be imbided into a circle
theta = np.linspace(0, 2 * np.pi, 6) # generates an array
vertical = np.vstack((np.cos(theta), np.sin(theta))).transpose() # vert
#print vertical, print and see how the array looks
plt.gca().add_patch(plt.Circle((0,0), radius =1.0, color ='b'))
plt.gca().add_patch(plt.Polygon(vertical, fill =None, lw =4.0, ls ='das
plt.axis('scaled')
plt.grid(True)
plt.show()
```



Ticks in Matplotlib

In [54]:
```
#In matplotlib, ticks are small marks on both the axes of a figure
import matplotlib.ticker as ticker
X = np.linspace(-12, 12, 1024)
Y = .25 * (X + 4.) * (X + 1.) * (X - 2.)
pl =plt.axes() #the object that manages the axes of a figure
pl.xaxis.set_major_locator(ticker.MultipleLocator(5))
pl.xaxis.set_minor_locator(ticker.MultipleLocator(1))
plt.plot(X, Y, c = 'y')
plt.grid(True, which ='major') # which can take three values: minor, ma
plt.show()
```



In [59]:
```
name_list = ('Omar', 'Serguey', 'Max', 'Zhou', 'Abidin')
value_list = np.random.randint(0, 99, size = len(name_list))
pos_list = np.arange(len(name_list))
ax = plt.axes()
ax.xaxis.set_major_locator(ticker.FixedLocator((pos_list)))
ax.xaxis.set_major_formatter(ticker.FixedFormatter((name_list)))
plt.bar(pos_list, value_list, color = '.75',align = 'center')
plt.show()
```