In [2]:

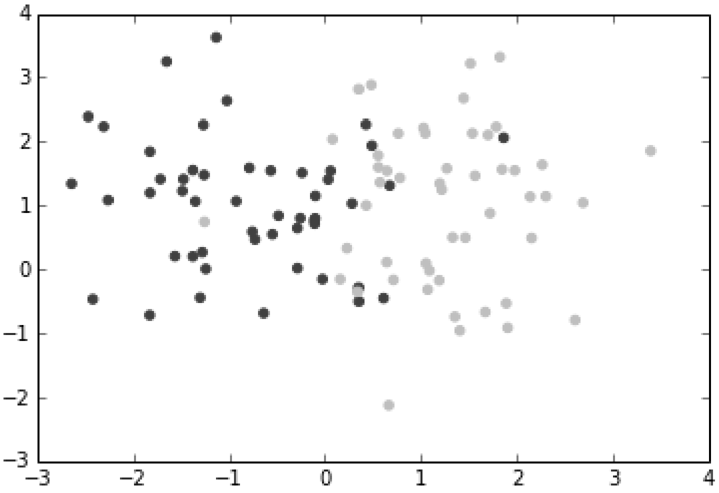
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [22]:

```
p =np.random.standard_normal((50,2))
p += np.array((-1,1)) # center the distribution at (-1,1)

q =np.random.standard_normal((50,2))
q += np.array((1,1)) #center the distribution at (-1,1)


plt.scatter(p[:,0], p[:,1], color ='.25')
plt.scatter(q[:,0], q[:,1], color = '.75')
```

Out[22]:

```
<matplotlib.collections.PathCollection at 0x71dab90>
```

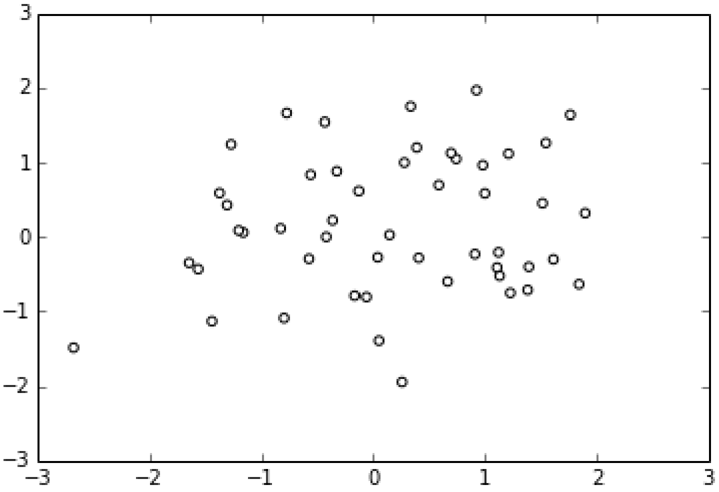


In [34]:

```
dd =np.random.standard_normal((50,2))
plt.scatter(dd[:,0], dd[:,1], color ='1.0', edgecolor ='0.0') # edge c
```

Out[34]:

```
<matplotlib.collections.PathCollection at 0x7336670>
```

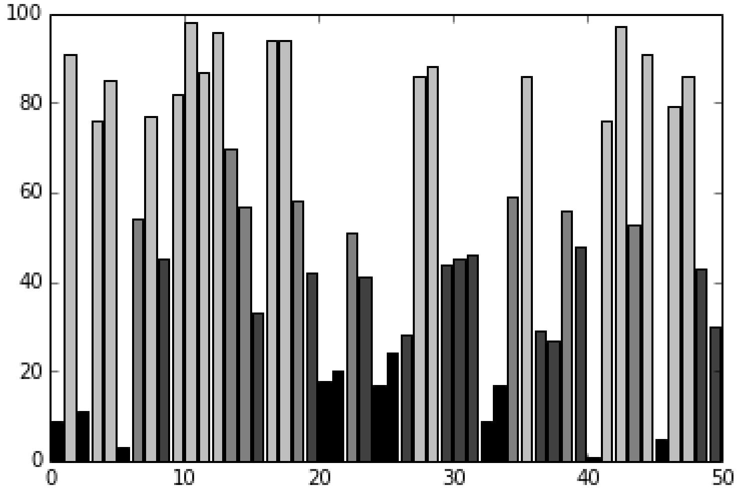


Custom Color for Bar charts,Pie charts and box plots:

The below bar graph, plots x(1 to 50) (vs) y(50 random integers, within 0-100. But you need different colors for each value. For which we create a list containing four colors(color_set). The list comprehension creates 50 different color values from color_set

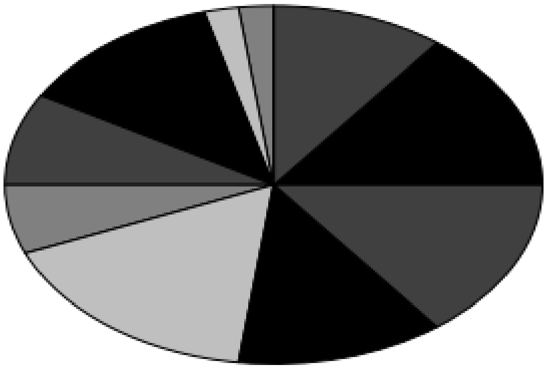
In [9]:

```
vals = np.random.random_integers(99, size =50)
color_set = ['.00', '.25', '.50','.75']
color_lists = [color_set[(len(color_set)* val) // 100] for val in vals
c = plt.bar(np.arange(50), vals, color = color_lists)
```



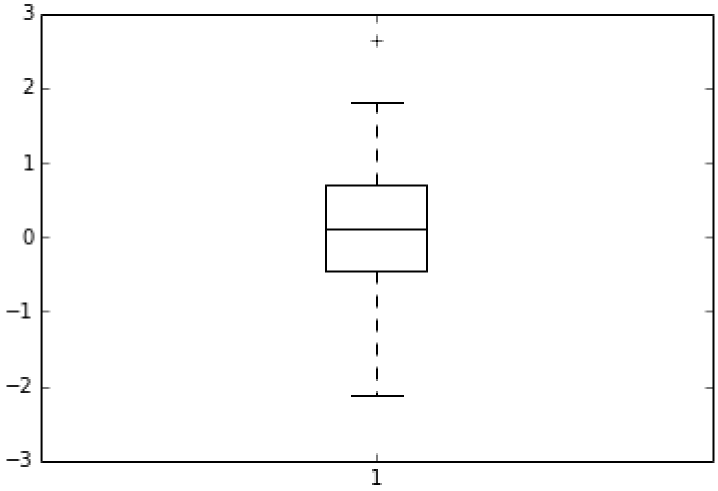
In [8]:

```
hi =np.random.random_integers(8, size =10)
color_set =['.00', '.25', '.50', '.75']
plt.pie(hi, colors = color_set)# colors attribute accepts a range of va
plt.show()
#If there are less colors than values, then pyplot.pie() will simply cy
#example, we gave a list of four colors to color a pie chart that cons
```



In [27]:

```
values = np.random.randn(100)
w = plt.boxplot(values)
for att, lines in w.iteritems():
    for l in lines:
        l.set_color('k')
```



Color Maps

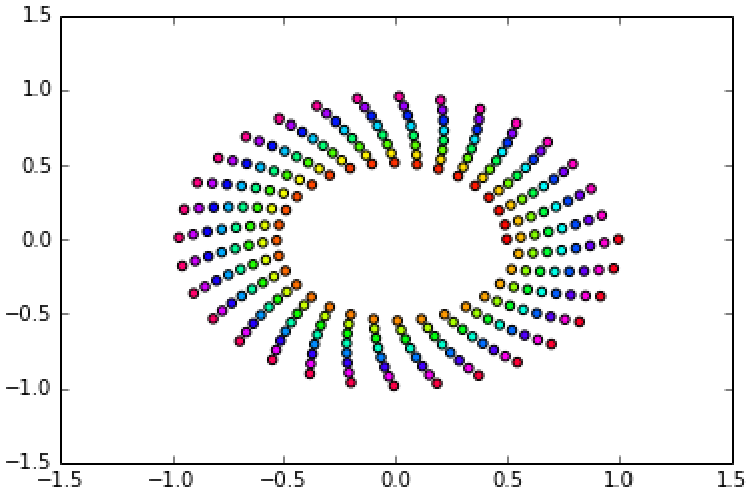
know more about hsv (http://radio.feld.cvut.cz/matlab/toolbox/images/color10.html)

In [34]:

```
# how to color scatter plots
#Colormaps are defined in the matplotib.cm module. This module provides
#functions to create and use colormaps. It also provides an exhaustive
import matplotlib.cm as cm
N = 256
angle = np.linspace(0, 8 * 2 * np.pi, N)
radius = np.linspace(.5, 1., N)
X = radius * np.cos(angle)
Y = radius * np.sin(angle)
plt.scatter(X,Y, c=angle, cmap = cm.hsv)
```

Out[34]:

```
<matplotlib.collections.PathCollection at 0x714d9f0>
```

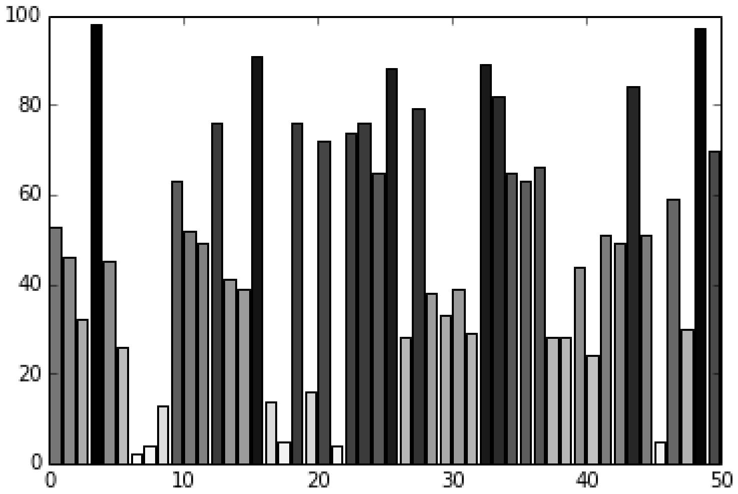


In [44]:

```
#Color in bar graphs
import matplotlib.cm as cm
vals = np.random.random_integers(99, size =50)
cmap = cm.ScalarMappable(col.Normalize(0,99), cm.binary)
plt.bar(np.arange(len(vals)),vals, color =cmap.to_rgba(vals))
```

Out[44]:

```
<Container object of 50 artists>
```



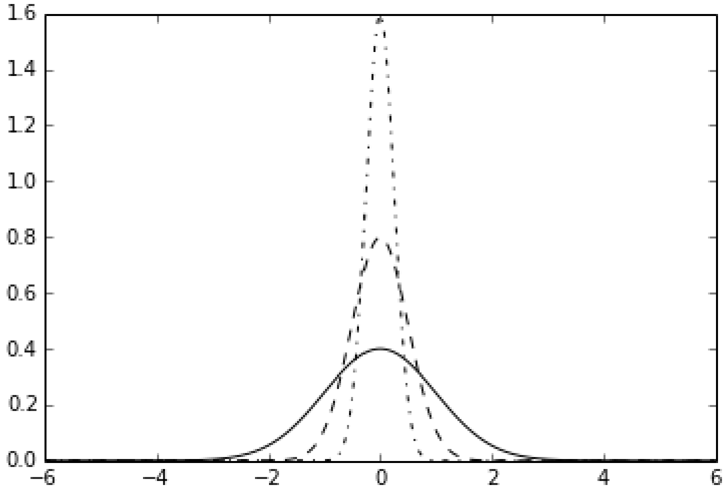
Line Styles

In [4]:

```
# I am creating 3 levels of gray plots, with different line shades


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')
```

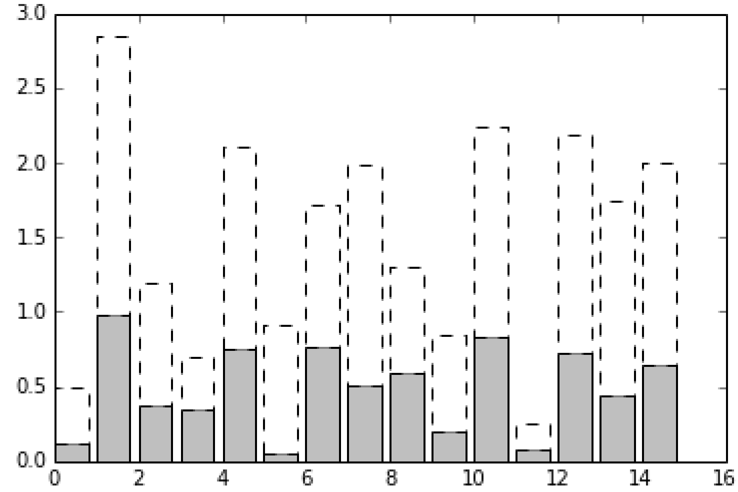
Out[4]:

```
[<matplotlib.lines.Line2D at 0x562ffb0>]
```



In [12]:

```
N = 15
A = np.random.random(N)
B= np.random.random(N)
X = np.arange(N)
plt.bar(X, A, color ='.75')
plt.bar(X, A+B , bottom = A, color ='W', linestyle ='dashed') # plot a
plt.show()
```

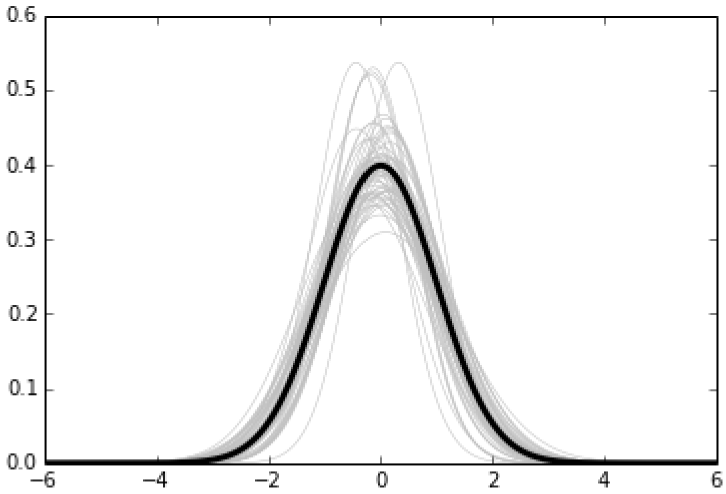


In [20]:

```
def gf(X, mu, sigma):
    a = 1. / (sigma * np.sqrt(2. * np.pi))
    b = -1. / (2. * sigma ** 2)
    return a * np.exp(b * (X - mu) ** 2)

X = np.linspace(-6, 6, 1024)
for i in range(64):
    samples = np.random.standard_normal(50)
    mu,sigma = np.mean(samples), np.std(samples)
    plt.plot(X, gf(X, mu, sigma), color = '.75', linewidth = .5)

plt.plot(X, gf(X, 0., 1.), color ='.00', linewidth = 3.)
```

Out[20]:

```
[<matplotlib.lines.Line2D at 0x59fbab0>]
```



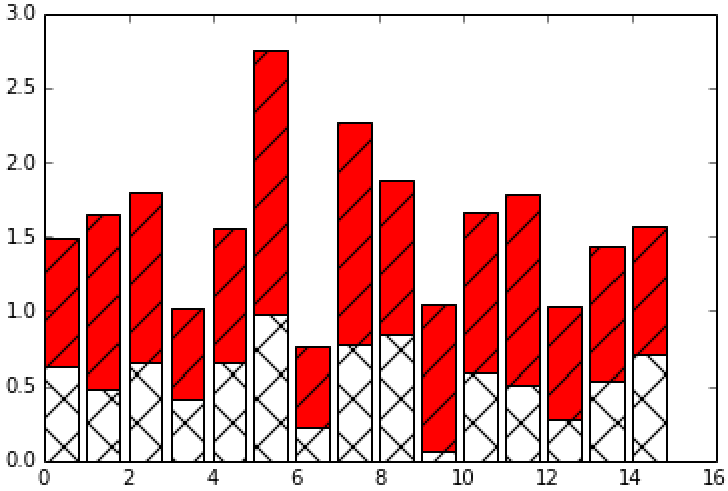
Fill surfaces with pattern

In [27]:

```
N = 15
A = np.random.random(N)
B= np.random.random(N)
X = np.arange(N)
plt.bar(X, A, color ='w', hatch ='x')
plt.bar(X, A+B,bottom =A, color ='r', hatch ='/')

# some other hatch attributes are :
#/
#\
#|
#-
#+
#x
#o
#O
#.
#*
```

Out[27]:

```
<Container object of 15 artists>
```



Marker styles

In [29]:

```
cd C:\Users\tk\Desktop\Matplot
```

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

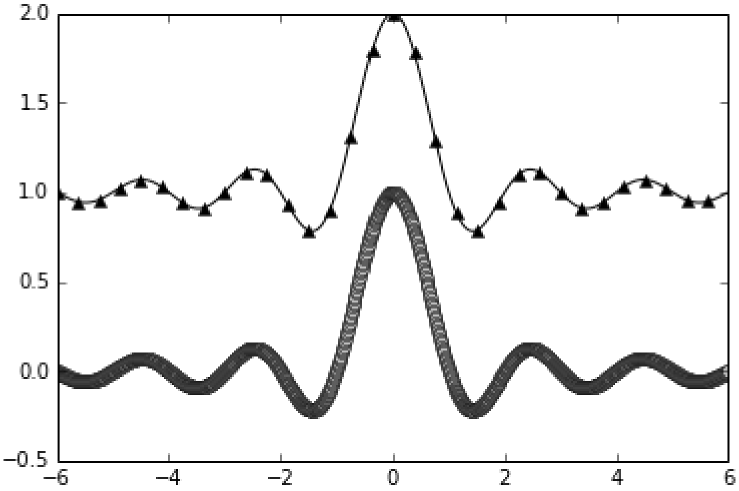
Come back to this section later

In [14]:

```
X= np.linspace(-6,6,1024)
Ya =np.sinc(X)

Yb = np.sinc(X) +1

plt.plot(X, Ya, marker ='o', color ='.75')
plt.plot(X, Yb, marker ='^', color='.00', markevery= 32)# this one mark
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x7063150>]
```

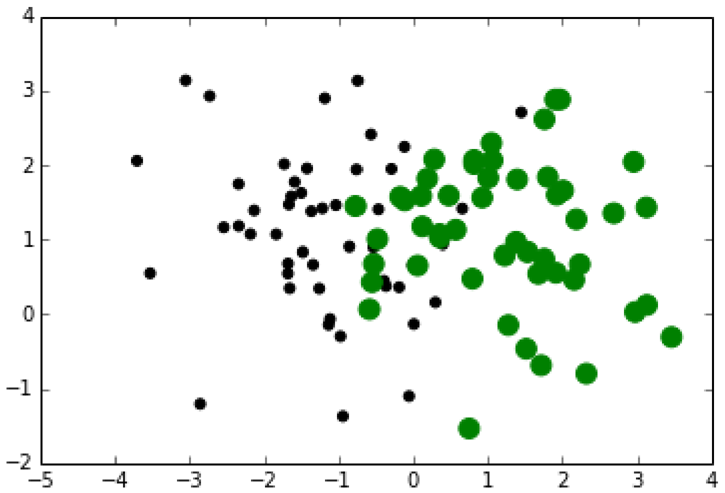


In [31]:

```
# Marker Size
A = np.random.standard_normal((50,2))
A += np.array((-1,1))

B = np.random.standard_normal((50,2))
B += np.array((1, 1))

plt.scatter(A[:,0], A[:,1], color ='k', s =25.0)
plt.scatter(B[:,0], B[:,1], color ='g', s = 100.0) # size of the marke
```

Out[31]:

```
<matplotlib.collections.PathCollection at 0x7d015f0>
```



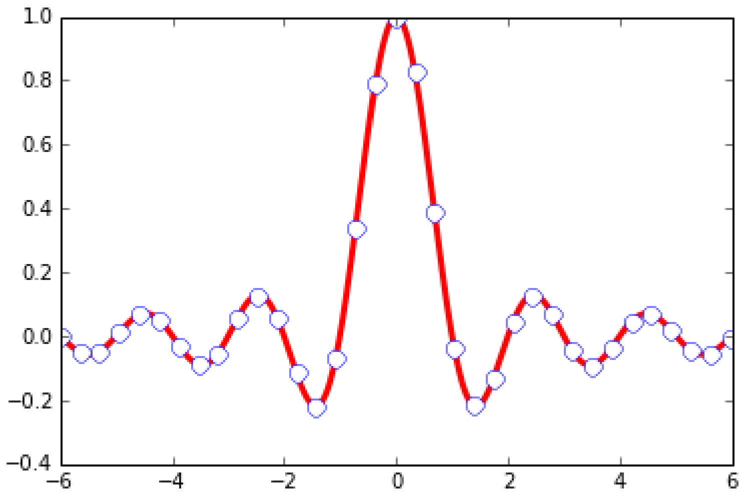
Own Marker Shapes- come back to this later

In [65]:

```
# more about markers
X =np.linspace(-6,6, 1024)
Y =np.sinc(X)
plt.plot(X,Y, color ='r', marker ='o', markersize =9, markevery = 30, r
```

Out[65]:

```
[<matplotlib.lines.Line2D at 0x84c9750>]
```

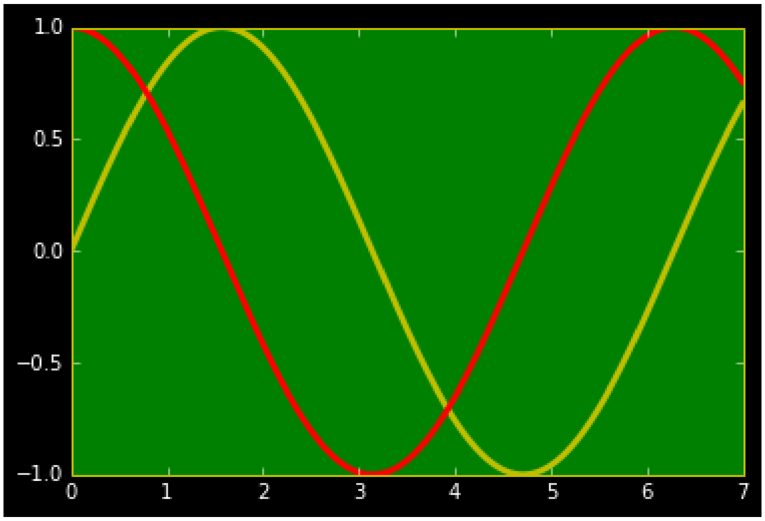


In [20]:

```
import matplotlib as mpl
mpl.rc('lines', linewidth =3)
mpl.rc('xtick', color ='w') # color of x axis numbers
mpl.rc('ytick', color = 'w') # color of y axis numbers
mpl.rc('axes', facecolor ='g', edgecolor ='y') # color of axes
mpl.rc('figure', facecolor ='.00',edgecolor ='w') # color of figure
mpl.rc('axes', color_cycle = ('y','r')) # color of plots
x = np.linspace(0, 7, 1024)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
```

Out[20]:

```
[<matplotlib.lines.Line2D at 0x7b0fb70>]
```


1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5
6
7