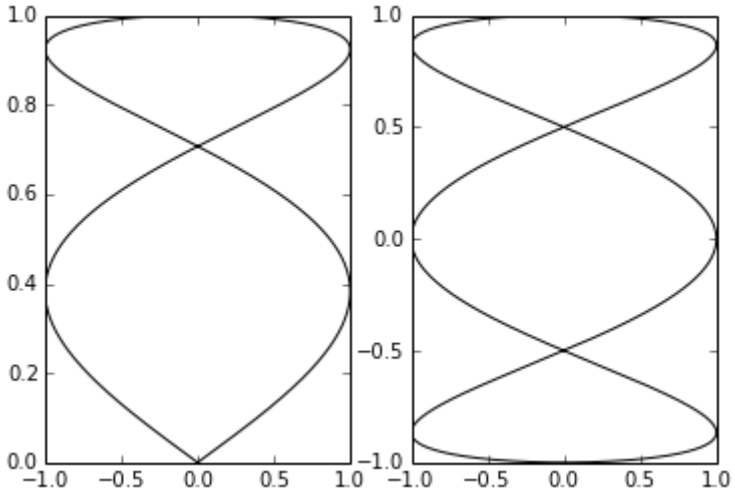
Working with figures

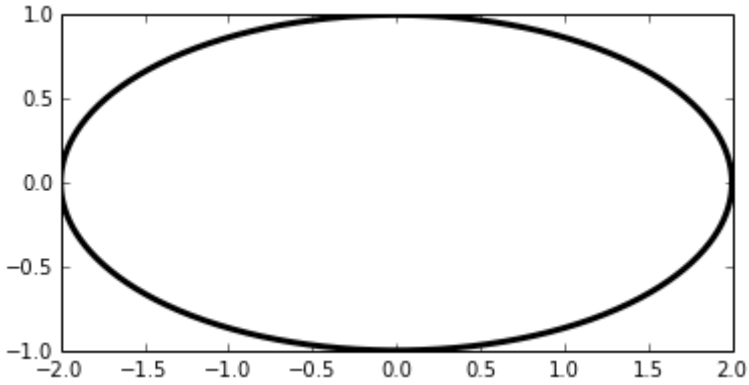
In [4]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [5]:
```
T = np.linspace(-np.pi, np.pi, 1024) #
fig, (ax0, ax1) = plt.subplots(ncols =2)
ax0.plot(np.sin(2 * T), np.cos(0.5 * T), c = 'k')
ax1.plot(np.cos(3 * T), np.sin(T), c = 'k')
plt.show()
```



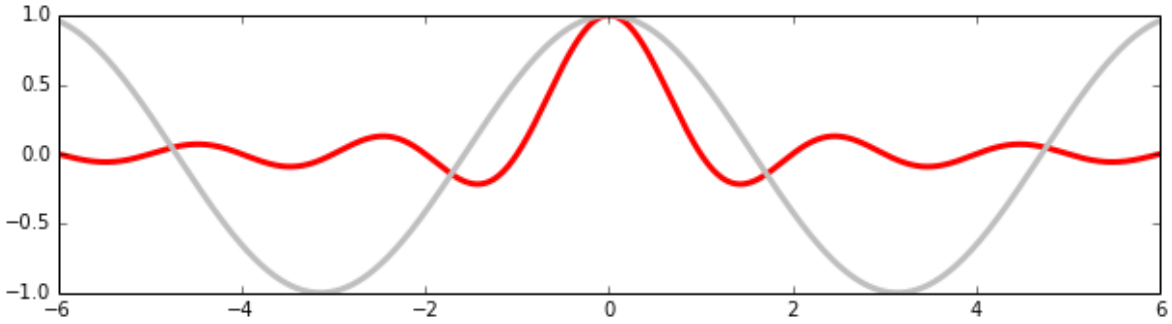
Setting aspect ratio

In [7]:
```
T = np.linspace(0, 2 * np.pi, 1024)
plt.plot(2. * np.cos(T), np.sin(T), c = 'k', lw = 3.)
plt.axes().set_aspect('equal') # remove this line of code and see how t
plt.show()
```



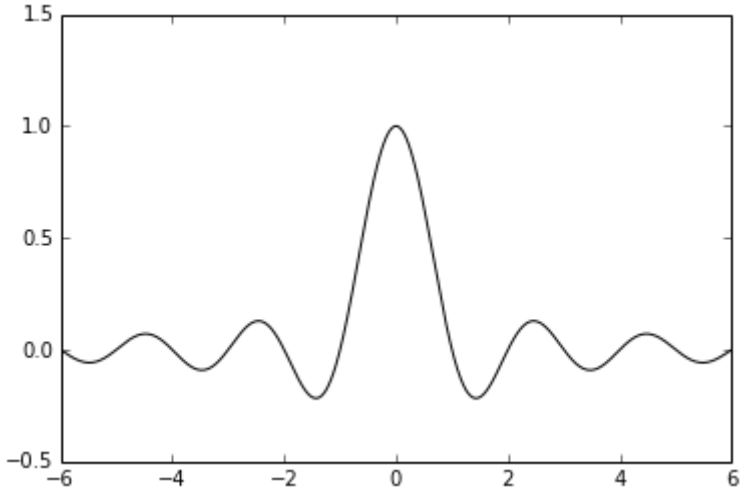
In [12]:

```
X = np.linspace(-6, 6, 1024)
Y1, Y2 = np.sinc(X), np.cos(X)
plt.figure(figsize=(10.24, 2.56)) #sets size of the figure
plt.plot(X, Y1, c='r', lw = 3.)
plt.plot(X, Y2, c='.75', lw = 3.)
plt.show()
```

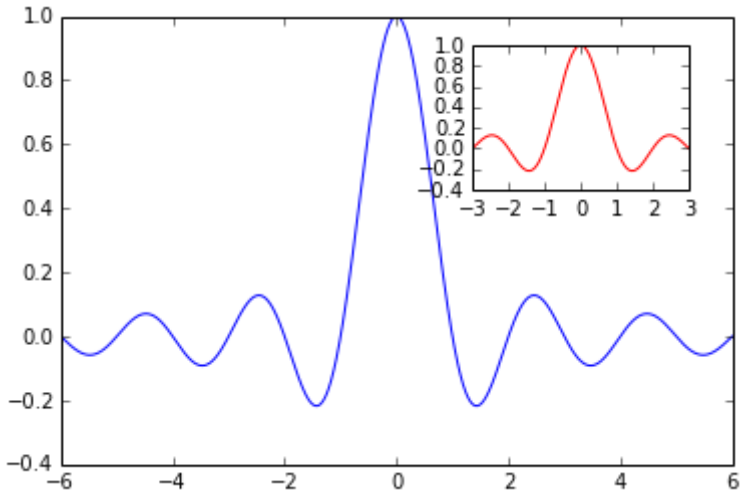


In [8]:

```
X = np.linspace(-6, 6, 1024)
plt.ylim(-.5, 1.5)
plt.plot(X, np.sinc(X), c = 'k')
plt.show()
```



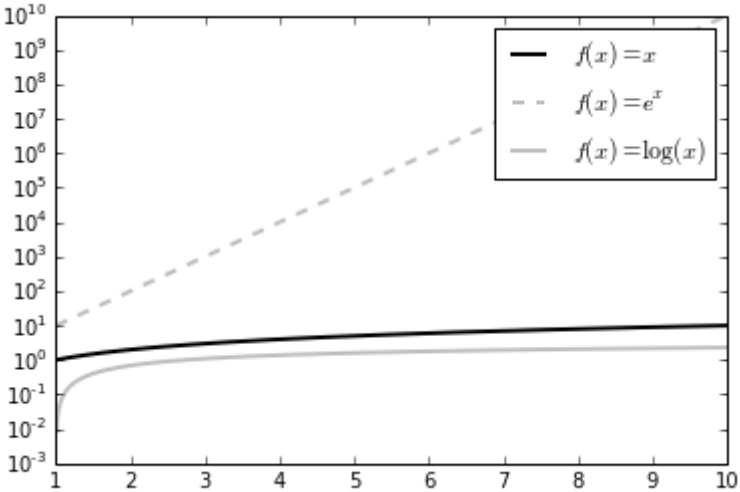
In [16]:
```
X = np.linspace(-6, 6, 1024)
Y = np.sinc(X)
X_sub = np.linspace(-3, 3, 1024)#coordinates of subplot
Y_sub = np.sinc(X_sub) # coordinates of sub plot
plt.plot(X, Y, c = 'b')
sub_axes = plt.axes([.6, .6, .25, .25])# coordinates, length and width
sub_axes.plot(X_detail, Y_detail, c = 'r')
plt.show()
```



Log Scale

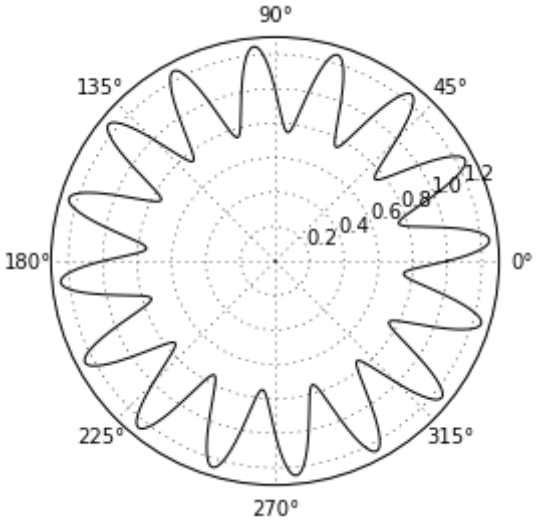
In [20]:
```
X = np.linspace(1, 10, 1024)
plt.yscale('log') # set y scale as log. we would use plot.xscale()
plt.plot(X, X, c = 'k', lw = 2., label = r'$f(x)=x$')
plt.plot(X, 10 ** X, c = '.75', ls = '--', lw = 2., label = r'$f(x)=e^x$')
plt.plot(X, np.log(X), c = '.75', lw = 2., label = r'$f(x)=\log(x)$')
plt.legend()
plt.show()

#The logarithm base is 10 by default, but it can be changed with the op
```

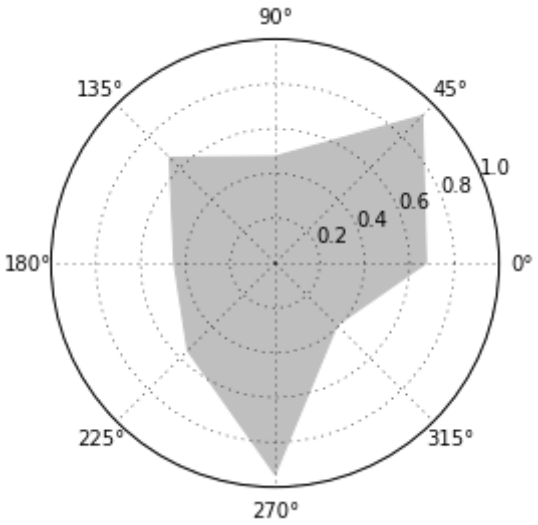


Polar Coordinates

In [23]:
```
T = np.linspace(0 , 2 * np.pi, 1024)
plt.axes(polar = True) # show polar coordinates
plt.plot(T, 1. + .25 * np.sin(16 * T), c= 'k')
plt.show()
```

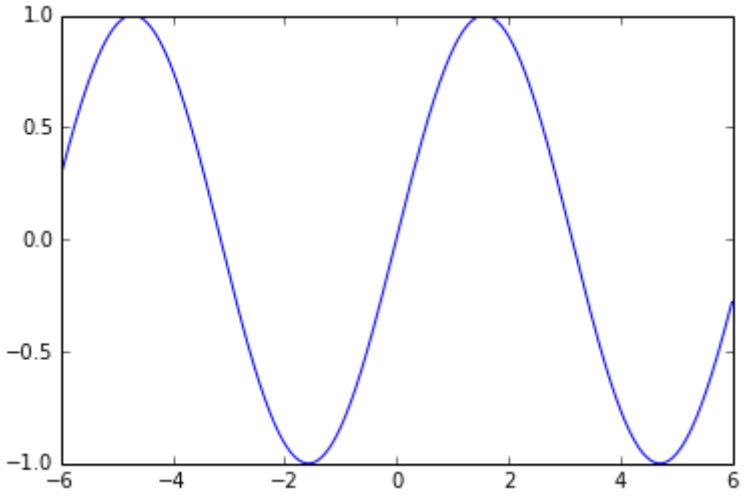


In [25]:
```
import matplotlib.patches as patches # import patch module from matplot
ax = plt.axes(polar = True)
theta = np.linspace(0, 2 * np.pi, 8, endpoint = False)
radius = .25 + .75 * np.random.random(size = len(theta))
points = np.vstack((theta, radius)).transpose()
plt.gca().add_patch(patches.Polygon(points, color = '.75'))
plt.show()
```



In [2]:

```
x = np.linspace(-6,6,1024)
y= np.sin(x)
plt.plot(x,y)
plt.savefig('bigdata.png', c= 'y', transparent = True) #savefig functi
# will create a file named bigdata.png. Its resolution will be 800 x 6
```



In [3]:

```
theta =np.linspace(0, 2 *np.pi, 8)
points =np.vstack((np.cos(theta), np.sin(theta))).T
plt.figure(figsize =(6.0, 6.0))
plt.gca().add_patch(plt.Polygon(points, color ='r'))
plt.axis('scaled')
plt.grid(True)
plt.savefig('pl.png', dpi =300) # try 'pl.pdf', pl.svg'
#dpi is dots per inch. 300*8 x 6*300 = 2400 x 1800 pixels
```