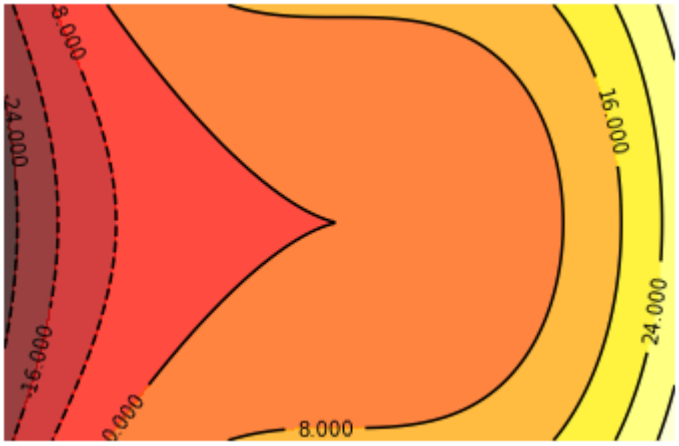
In [3]:

```
# Visualizing a Function as an Image and a Contour
# Let’s visualize a numerical function.
# Import all the needed libraries as follows:

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
# Let’s define the function as follows:
def f(x, y):
    return (x ** 3 + y ** 2)
# Let’s visualize it as an image, as follows:
n = 10
x = np.linspace(-3, 3, 8 * n)
y = np.linspace(-3, 3, 6 * n)
X, Y = np.meshgrid(x, y)
Z = f( X, Y )
plt.imshow(Z, interpolation='nearest',
           cmap = 'cool', origin='lower')
plt.axis('off')
plt.show()
```

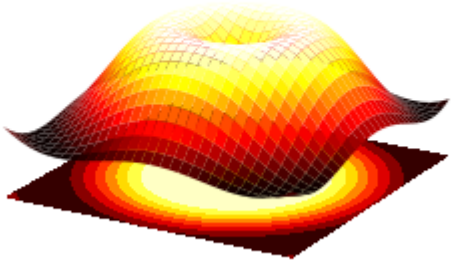
In [4]:

```
# You can visualize the function as a contour too.

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
# Let's define the function as follows:
def f(x, y):
    return (x ** 3 + y ** 2)
# Let's visualize it as an image, as follows:
n = 256
x = np.linspace(-3, 3, n)
y = np.linspace(-3, 3, n)
X, Y = np.meshgrid(x, y)
plt.contourf(X, Y, f(X, Y), 8,
             alpha = 0.75, cmap='hot')
C = plt.contour(X, Y, f(X, Y), 8,
colors='black')
plt.clabel(C, inline=1, fontsize=10)
plt.axis('off')
plt.show()
```



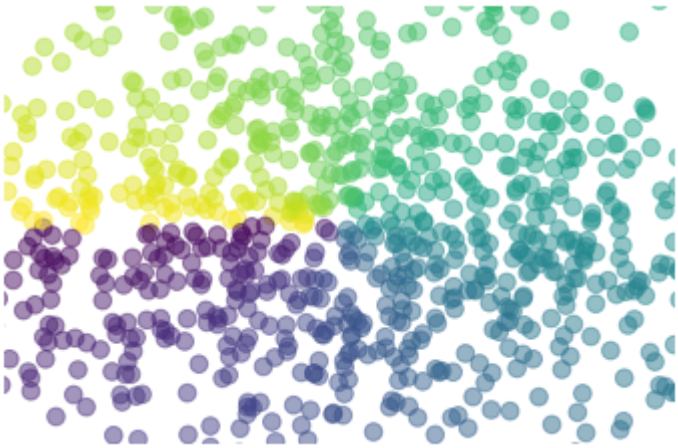
In [5]:

```
# 3D Vignettes
# You can create a 3D vignette visualization as follows:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection='3d')
X = np.arange(-4, 4, 0.25)
Y = np.arange(-4, 4, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X ** 2 + Y ** 2)
Z = np.sin(R)
ax.plot_surface(X, Y, Z, rstride=1,
                cstride=1, cmap='hot')
ax.contourf(X, Y, Z, zdir='z',
            offset=-2, cmap='hot')
ax.set_zlim(-2, 2)
plt.axis('off')
ax.set_zticks(())
plt.show()
```



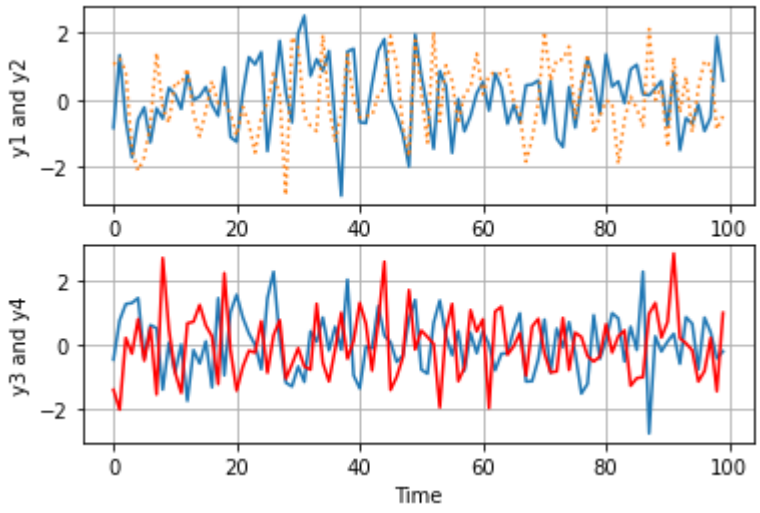
In [6]:

```
# Decorated Scatter Plots
# You can create decorated scatter plots with Matplotlib.
# You need to pass the color and size as arguments. Here's an example:

%matplotlib qt
%matplotlib inline
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)
color = np.arctan2(Y, X)
plt.scatter(X, Y, s=75, c=color, alpha=0.5)
plt.xlim(-1.5, 1.5)
plt.ylim(-1.5, 1.5)
plt.axis('off')
plt.show()
```



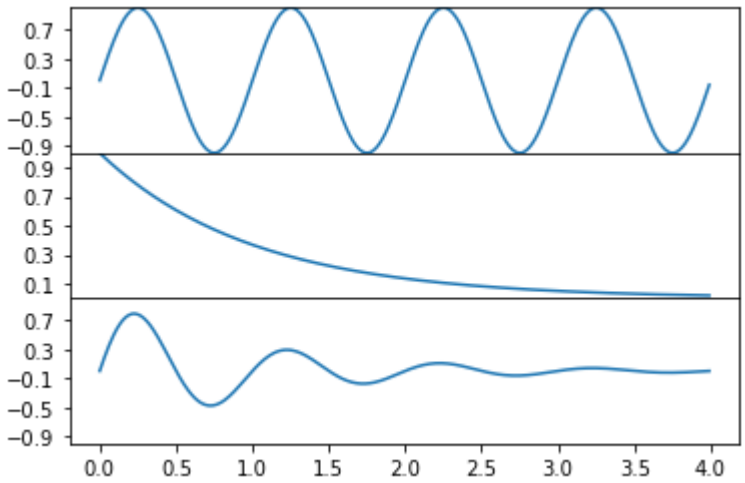
In [7]:

```
# Time Plots and Signals
# You can visualize time plots and signals as follows:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
N = 100
x = np.arange(N) # timestamps
y1 = np.random.randn(N)
y2 = np.random.randn(N)
y3 = np.random.randn(N)
y4 = np.random.randn(N)
plt.subplot(2, 1, 1)
plt.plot(x, y1)
plt.plot(x, y2, ':')
plt.grid()
plt.xlabel('Time')
plt.ylabel('y1 and y2')
plt.axis('tight')
plt.subplot(2, 1, 2)
plt.plot(x, y3)
plt.plot(x, y4, 'r')
plt.grid()
plt.xlabel('Time')
plt.ylabel('y3 and y4')
plt.axis('tight')
plt.show()
```



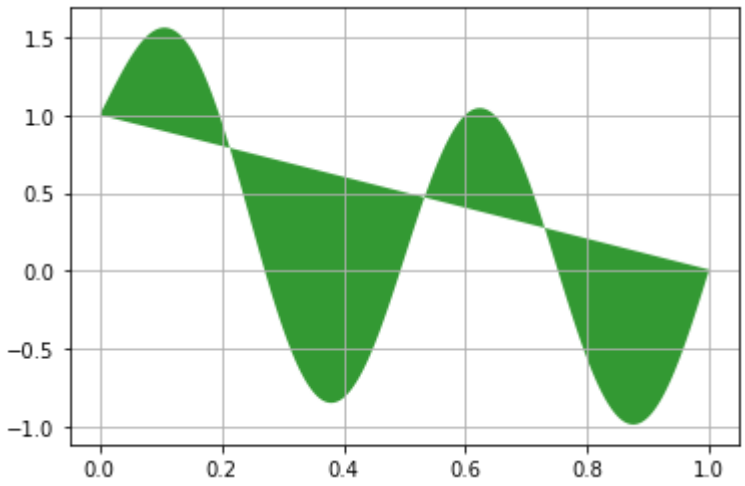
In [8]:

```
# You can also multiply two signals. In the following code example,
# we are using the same x-axis to show all three graphs.

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
f = 1
t = np.arange(0.0, 4.0, 0.01)
s1 = np.sin(2 *np.pi * f * t)
s2 = np.exp(-t)
s3 = s1 * s2
f = plt.figure()
plt.subplots_adjust(hspace=0.001)
ax1 = plt.subplot( 311 )
ax1.plot(t, s1)
plt.yticks(np.arange(-0.9, 1.0, 0.4))
plt.ylim(-1, 1)
ax2 = plt.subplot(312, sharex=ax1)
ax2.plot(t, s2)
plt.yticks(np.arange(0.1, 1.0, 0.2))
plt.ylim(0, 1)
ax3 = plt.subplot(313, sharex = ax1)
ax3.plot(t, s3)
plt.yticks(np.arange(-0.9, 1.0, 0.4))
plt.ylim(-1, 1)
xticklabels = ax1.get_xticklabels() + ax2.get_xticklabels()
plt.setp(xticklabels, visible=False)
plt.show()
```



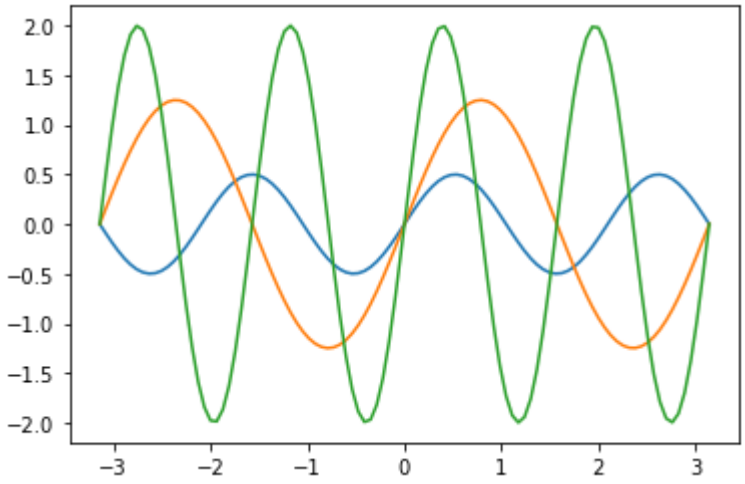
In [9]:

```
# Filled Plots
# You can fill in the empty spaces within the boundaries of plots
# as follows:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
N = 1000
x = np.linspace(0, 1, N)
y = np.sin(4 * np.pi * x) + np.exp(-5 * x)
plt.fill(x, y, 'g', alpha = 0.8)
plt.grid(True)
plt.show()
```



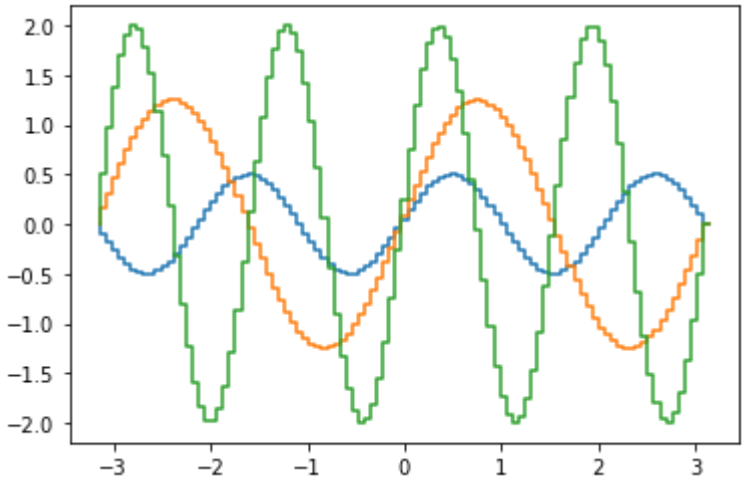
In [10]:

```
# Step Plots
# Let's visualize some sine waves first.

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
N = 100
x = np.linspace(-np.pi, np.pi, N)
y1 = 0.5 * np.sin(3*x)
y2 = 1.25 * np.sin(2*x)
y3 = 2 * np.sin(4*x)
plt.plot(x, y1, x, y2, x, y3)
plt.show()
```



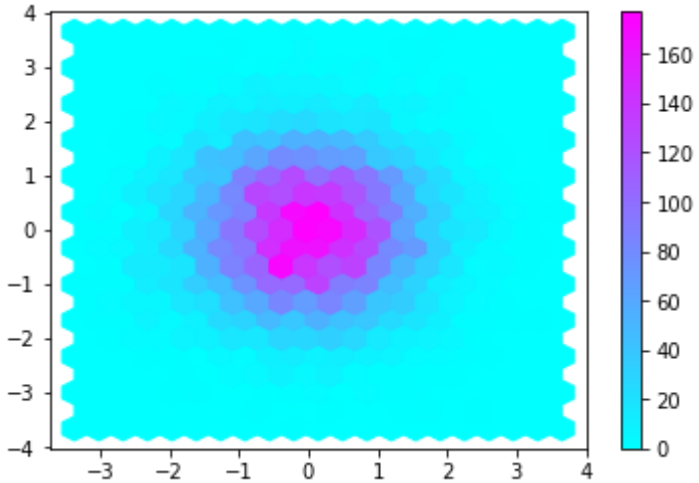
In [11]:

```
# You can show them as step plots as follows:

%matplotlib qt
%matplotlib inline
import matplotlib.pyplot as plt
plt.step(x, y1)
plt.step(x, y2)
plt.step(x, y3)
plt.show()
```



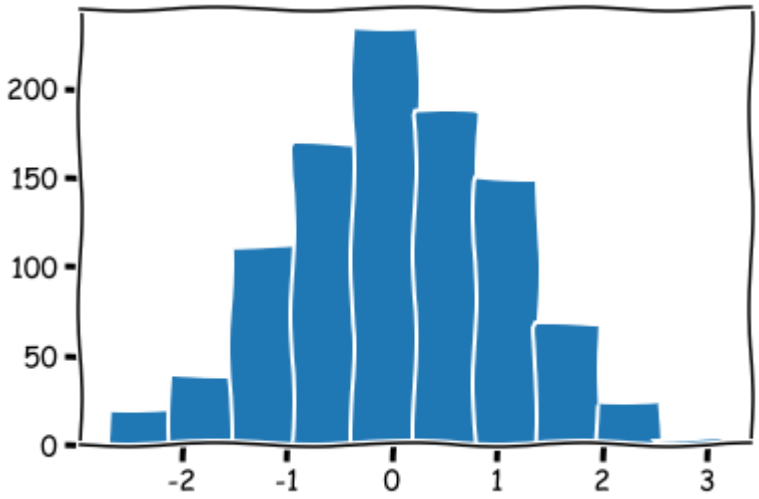
In [12]:

```
# Hexbins
# You can show data as hexbins as follows:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
x, y = np.random.normal(size=(2, 10000))
plt.hexbin(x, y, gridsize=20, cmap='cool')
plt.colorbar()
plt.show()
```



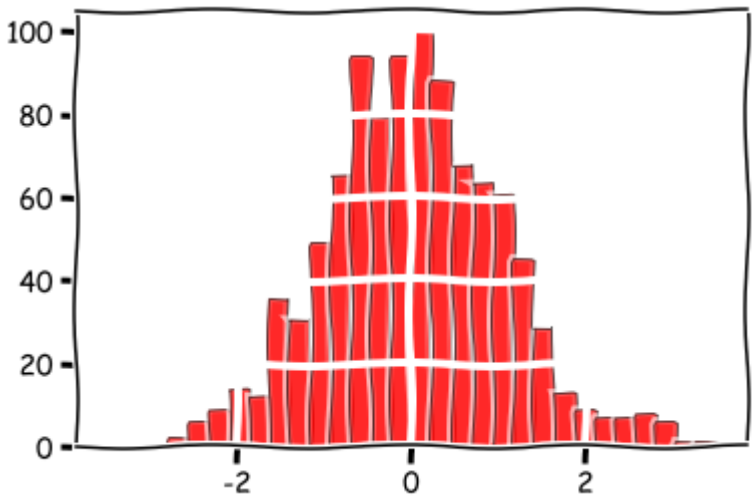
In [13]:

```
# XKCD Style
# You can visualize plots in the XKCD style.
# The XKCD is a popular a web comic.
# https://xkcd.com is the homepage of the web comic.

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
y = np.random.randn(1000)
plt.xkcd()
plt.hist(y)
plt.show()
```



In [14]:

```
# Another example is as follows:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
y = np.random.randn(1000)
plt.xkcd()
plt.hist(y, bins = 30, range=[-3.5, 3.5],
         facecolor='r',alpha=0.6,edgecolor='k')
plt.grid()
plt.show()
```



In [15]:

```
# You can visualize 2D histograms too in the same way, as shown here:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
data = np.random.randn(1000, 1000)
plt.xkcd()
plt.hist2d(data[0], data[1])
plt.show()
```