

Figure 4-92. An empty three-dimensional axes

With this 3D axes enabled, we can now plot a variety of three-dimensional plot types. Three-dimensional plotting is one of the functionalities that benefits immensely from viewing figures interactively rather than statically in the notebook; recall that to use interactive figures, you can use `%matplotlib notebook` rather than `%matplotlib inline` when running this code.

Three-Dimensional Points and Lines

The most basic three-dimensional plot is a line or scatter plot created from sets of (x, y, z) triples. In analogy with the more common two-dimensional plots discussed earlier, we can create these using the `ax.plot3D` and `ax.scatter3D` functions. The call signature for these is nearly identical to that of their two-dimensional counterparts, so you can refer to “Simple Line Plots” on page 224 and “Simple Scatter Plots” on page 233 for more information on controlling the output. Here we’ll plot a trigonometric spiral, along with some points drawn randomly near the line (Figure 4-93):

```
In[4]: ax = plt.axes(projection='3d')

# Data for a three-dimensional line
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'gray')

# Data for three-dimensional scattered points
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
```

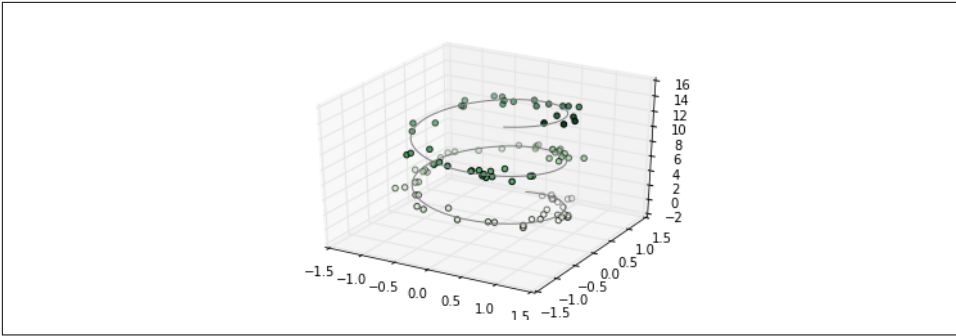


Figure 4-93. Points and lines in three dimensions

Notice that by default, the scatter points have their transparency adjusted to give a sense of depth on the page. While the three-dimensional effect is sometimes difficult to see within a static image, an interactive view can lead to some nice intuition about the layout of the points.

Three-Dimensional Contour Plots

Analogous to the contour plots we explored in “Density and Contour Plots” on page 241, `mplot3d` contains tools to create three-dimensional relief plots using the same inputs. Like two-dimensional `ax.contour` plots, `ax.contour3D` requires all the input data to be in the form of two-dimensional regular grids, with the `Z` data evaluated at each point. Here we’ll show a three-dimensional contour diagram of a three-dimensional sinusoidal function (Figure 4-94):

```
In[5]: def f(x, y):
        return np.sin(np.sqrt(x ** 2 + y ** 2))

        x = np.linspace(-6, 6, 30)
        y = np.linspace(-6, 6, 30)

        X, Y = np.meshgrid(x, y)
        Z = f(X, Y)

In[6]: fig = plt.figure()
        ax = plt.axes(projection='3d')
        ax.contour3D(X, Y, Z, 50, cmap='binary')
        ax.set_xlabel('x')
        ax.set_ylabel('y')
        ax.set_zlabel('z');
```