

```
In[1]: %matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
import numpy as np
```

plt.axes: Subplots by Hand

The most basic method of creating an axes is to use the `plt.axes` function. As we've seen previously, by default this creates a standard axes object that fills the entire figure. `plt.axes` also takes an optional argument that is a list of four numbers in the figure coordinate system. These numbers represent [*bottom*, *left*, *width*, *height*] in the figure coordinate system, which ranges from 0 at the bottom left of the figure to 1 at the top right of the figure.

For example, we might create an inset axes at the top-right corner of another axes by setting the *x* and *y* position to 0.65 (that is, starting at 65% of the width and 65% of the height of the figure) and the *x* and *y* extents to 0.2 (that is, the size of the axes is 20% of the width and 20% of the height of the figure). [Figure 4-59](#) shows the result of this code:

```
In[2]: ax1 = plt.axes() # standard axes
ax2 = plt.axes([0.65, 0.65, 0.2, 0.2])
```

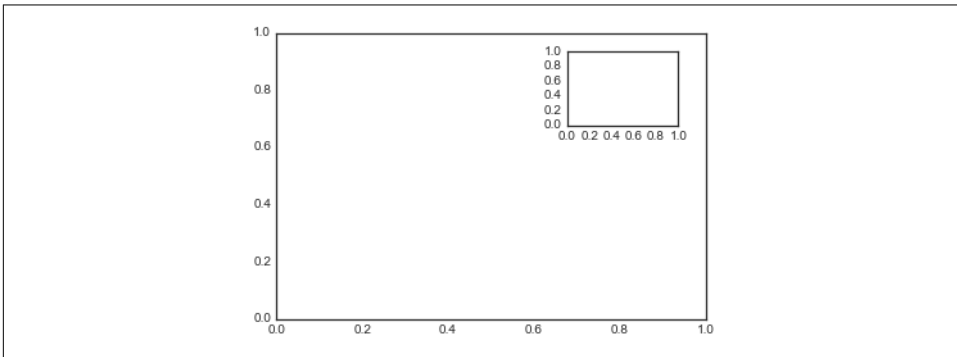


Figure 4-59. Example of an inset axes

The equivalent of this command within the object-oriented interface is `fig.add_axes()`. Let's use this to create two vertically stacked axes ([Figure 4-60](#)):

```
In[3]: fig = plt.figure()
ax1 = fig.add_axes([0.1, 0.5, 0.8, 0.4],
                  xticklabels=[], ylim=(-1.2, 1.2))
ax2 = fig.add_axes([0.1, 0.1, 0.8, 0.4],
                  ylim=(-1.2, 1.2))

x = np.linspace(0, 10)
ax1.plot(np.sin(x))
ax2.plot(np.cos(x));
```

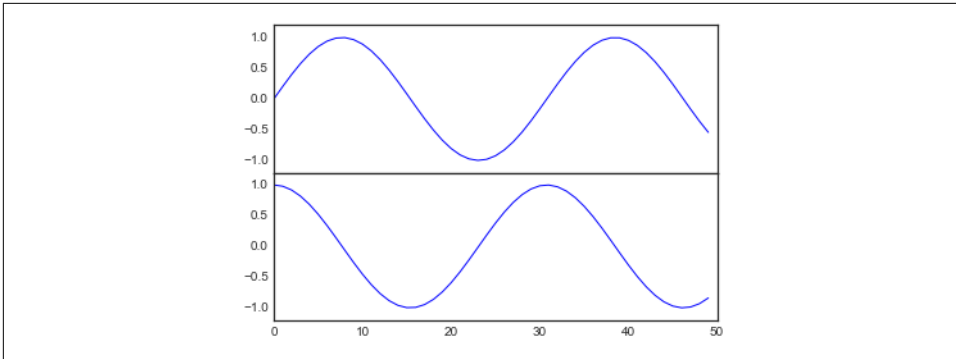


Figure 4-60. Vertically stacked axes example

We now have two axes (the top with no tick labels) that are just touching: the bottom of the upper panel (at position 0.5) matches the top of the lower panel (at position $0.1 + 0.4$).

plt.subplot: Simple Grids of Subplots

Aligned columns or rows of subplots are a common enough need that Matplotlib has several convenience routines that make them easy to create. The lowest level of these is `plt.subplot()`, which creates a single subplot within a grid. As you can see, this command takes three integer arguments—the number of rows, the number of columns, and the index of the plot to be created in this scheme, which runs from the upper left to the bottom right (Figure 4-61):

```
In[4]: for i in range(1, 7):
        plt.subplot(2, 3, i)
        plt.text(0.5, 0.5, str((2, 3, i)),
                 fontsize=18, ha='center')
```

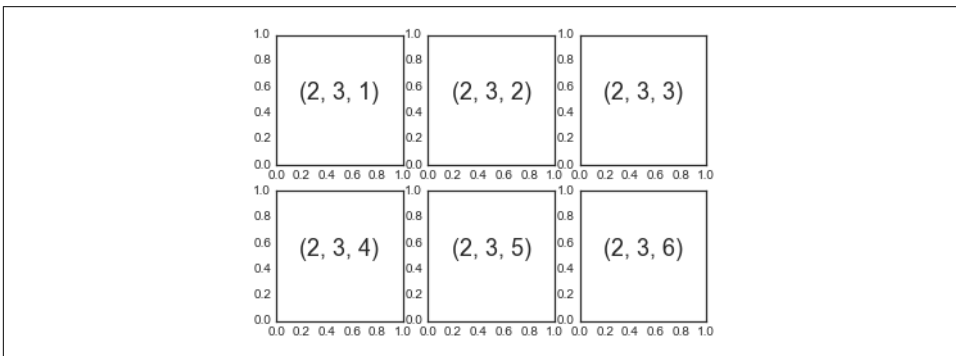


Figure 4-61. A `plt.subplot()` example