

Figure 4-11. Controlling colors and styles with the shorthand syntax

These single-character color codes reflect the standard abbreviations in the RGB (Red/Green/Blue) and CMYK (Cyan/Magenta/Yellow/black) color systems, commonly used for digital color graphics.

There are many other keyword arguments that can be used to fine-tune the appearance of the plot; for more details, I'd suggest viewing the docstring of the `plt.plot()` function using IPython's help tools (see [“Help and Documentation in IPython”](#) on page 3).

## Adjusting the Plot: Axes Limits

Matplotlib does a decent job of choosing default axes limits for your plot, but sometimes it's nice to have finer control. The most basic way to adjust axis limits is to use the `plt.xlim()` and `plt.ylim()` methods (Figure 4-12):

```
In[9]: plt.plot(x, np.sin(x))

plt.xlim(-1, 11)
plt.ylim(-1.5, 1.5);
```

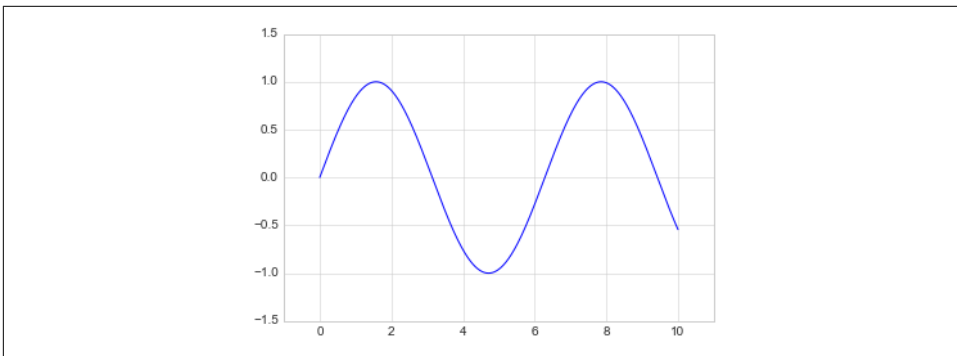


Figure 4-12. Example of setting axis limits

If for some reason you'd like either axis to be displayed in reverse, you can simply reverse the order of the arguments (Figure 4-13):

```
In[10]: plt.plot(x, np.sin(x))

plt.xlim(10, 0)
plt.ylim(1.2, -1.2);
```

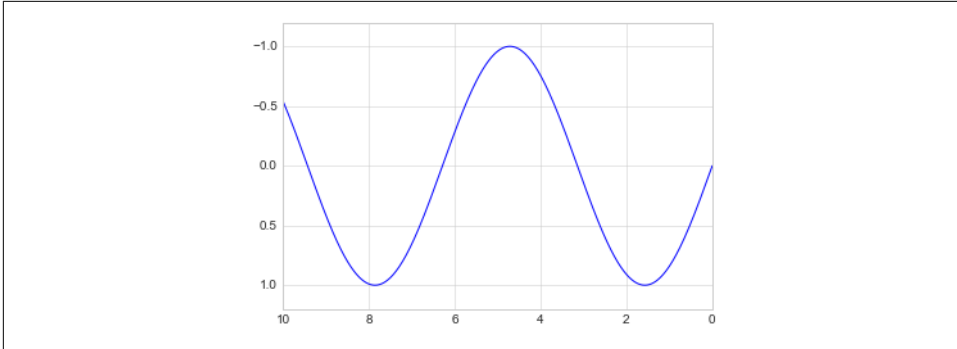


Figure 4-13. Example of reversing the y-axis

A useful related method is `plt.axis()` (note here the potential confusion between *axes* with an *e*, and *axis* with an *i*). The `plt.axis()` method allows you to set the x and y limits with a single call, by passing a list that specifies `[xmin, xmax, ymin, ymax]` (Figure 4-14):

```
In[11]: plt.plot(x, np.sin(x))
plt.axis([-1, 11, -1.5, 1.5]);
```

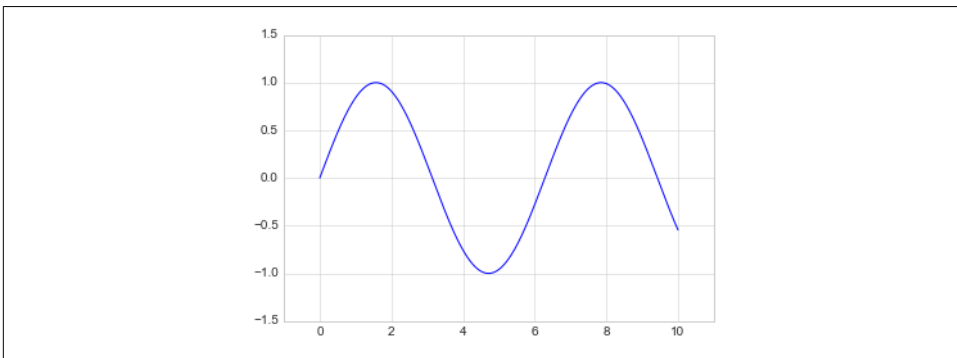


Figure 4-14. Setting the axis limits with `plt.axis`

The `plt.axis()` method goes even beyond this, allowing you to do things like automatically tighten the bounds around the current plot (Figure 4-15):

```
In[12]: plt.plot(x, np.sin(x))
plt.axis('tight');
```

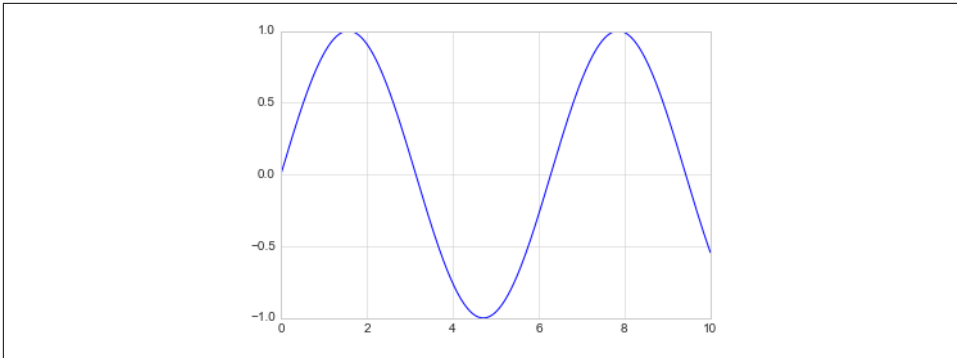


Figure 4-15. Example of a “tight” layout

It allows even higher-level specifications, such as ensuring an equal aspect ratio so that on your screen, one unit in x is equal to one unit in y (Figure 4-16):

```
In[13]: plt.plot(x, np.sin(x))
plt.axis('equal');
```

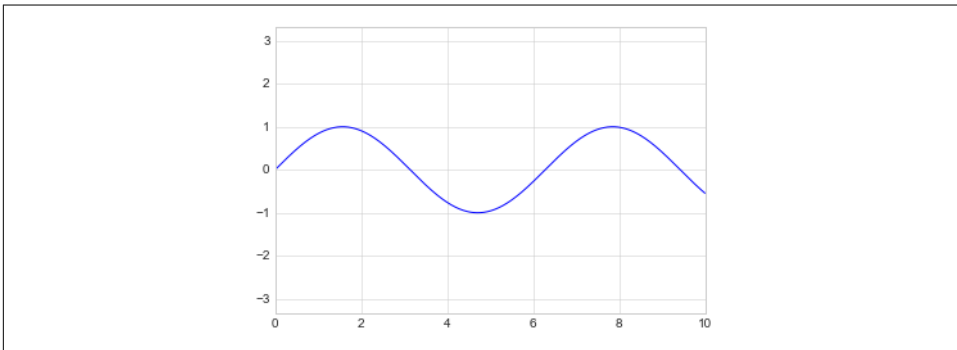


Figure 4-16. Example of an “equal” layout, with units matched to the output resolution

For more information on axis limits and the other capabilities of the `plt.axis()` method, refer to the `plt.axis()` docstring.

## Labeling Plots

As the last piece of this section, we’ll briefly look at the labeling of plots: titles, axis labels, and simple legends.

Titles and axis labels are the simplest such labels—there are methods that can be used to quickly set them (Figure 4-17):

```
In[14]: plt.plot(x, np.sin(x))
plt.title("A Sine Curve")
```