```
        elif N == 1:
            return r"$\pi/2$"
        elif N == 2:
            return r"$\pi$"
        elif N % 2 > 0:
            return r"${0}\pi/2$".format(N)
        else:
            return r"${0}\pi$".format(N // 2)

    ax.xaxis.set_major_formatter(plt.FuncFormatter(format_func))
    fig
```

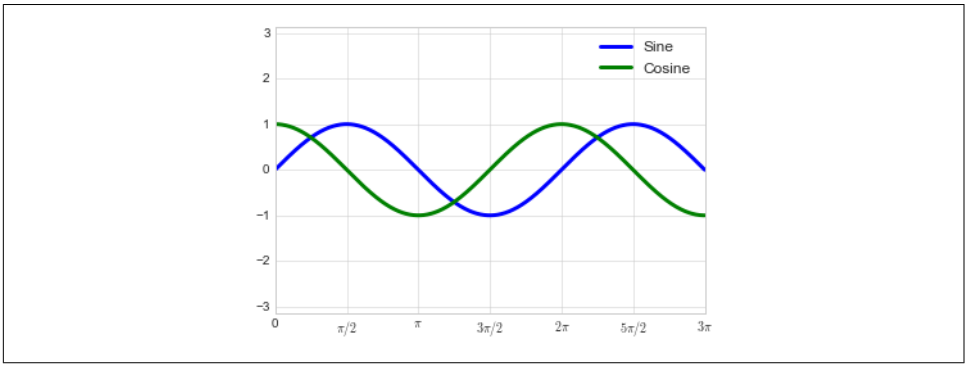

Figure 4-80. Ticks with custom labels

This is much better! Notice that we've made use of Matplotlib's LaTeX support, specified by enclosing the string within dollar signs. This is very convenient for display of mathematical symbols and formulae; in this case, `"$\pi$"` is rendered as the Greek character π.

The `plt.FuncFormatter()` offers extremely fine-grained control over the appearance of your plot ticks, and comes in very handy when you're preparing plots for presentation or publication.

Summary of Formatters and Locators

We've mentioned a couple of the available formatters and locators. We'll conclude this section by briefly listing all the built-in locator and formatter options. For more information on any of these, refer to the docstrings or to the Matplotlib online documentation. Each of the following is available in the `plt` namespace:

Locator class	Description
`NullLocator`	No ticks
`FixedLocator`	Tick locations are fixed
`IndexLocator`	Locator for index plots (e.g., where `x = range(len(y))`)

Locator class	Description
`LinearLocator`	Evenly spaced ticks from `min` to `max`
`LogLocator`	Logarithmically ticks from `min` to `max`
`MultipleLocator`	Ticks and range are a multiple of base
`MaxNLocator`	Finds up to a max number of ticks at nice locations
`AutoLocator`	(Default) `MaxNLocator` with simple defaults
`AutoMinorLocator`	Locator for minor ticks

Formatter class	Description
`NullFormatter`	No labels on the ticks
`IndexFormatter`	Set the strings from a list of labels
`FixedFormatter`	Set the strings manually for the labels
`FuncFormatter`	User-defined function sets the labels
`FormatStrFormatter`	Use a format string for each value
`ScalarFormatter`	(Default) Formatter for scalar values
`LogFormatter`	Default formatter for log axes

We'll see additional examples of these throughout the remainder of the book.

Customizing Matplotlib: Configurations and Stylesheets

Matplotlib's default plot settings are often the subject of complaint among its users. While much is slated to change in the 2.0 Matplotlib release, the ability to customize default settings helps bring the package in line with your own aesthetic preferences.

Here we'll walk through some of Matplotlib's runtime configuration (`rc`) options, and take a look at the newer *stylesheets* feature, which contains some nice sets of default configurations.

Plot Customization by Hand

Throughout this chapter, we've seen how it is possible to tweak individual plot settings to end up with something that looks a little bit nicer than the default. It's possible to do these customizations for each individual plot. For example, here is a fairly drab default histogram (Figure 4-81):

```
In[1]: import matplotlib.pyplot as plt
       plt.style.use('classic')
       import numpy as np

       %matplotlib inline
```