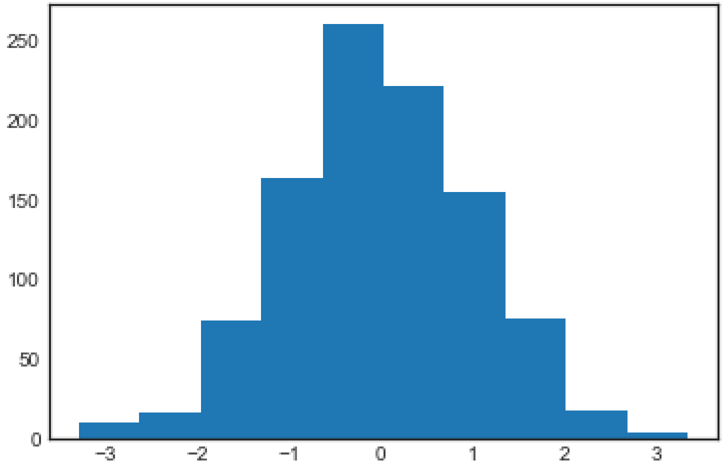
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
data=np.random.randn(1000)
plt.hist(data)
```

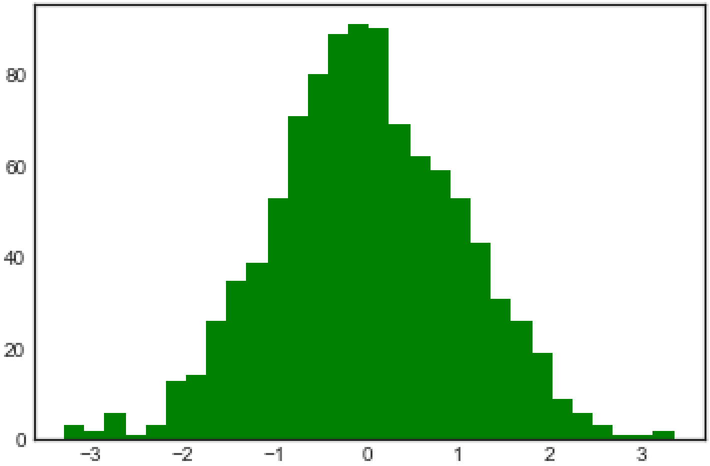
Out[1]:

```
(array([ 11.,  17.,  75., 163., 260., 221., 155.,  76.,  18.,   4.]),
 array([-3.28815987, -2.62450462, -1.96084937, -1.29719412, -0.63353888,
         0.03011637,  0.69377162,  1.35742687,  2.02108212,  2.68473736,
         3.34839261]),
 <BarContainer object of 10 artists>)
```



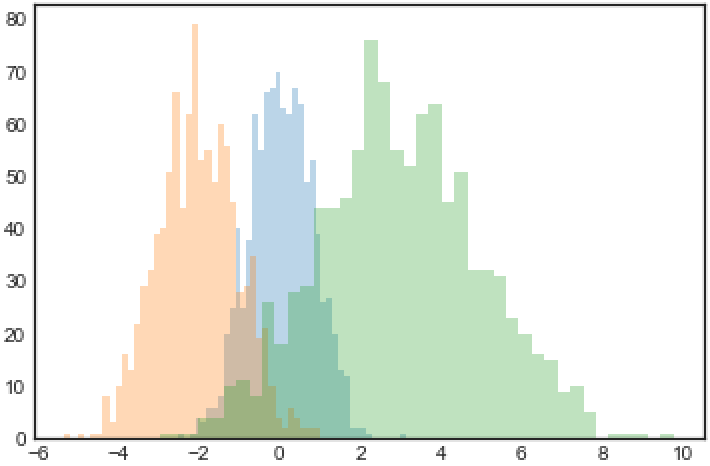
In [5]:

```
plt.hist(data, bins=30, alpha=1, histtype='stepfilled', color='green',
     edgecolor='none');
```



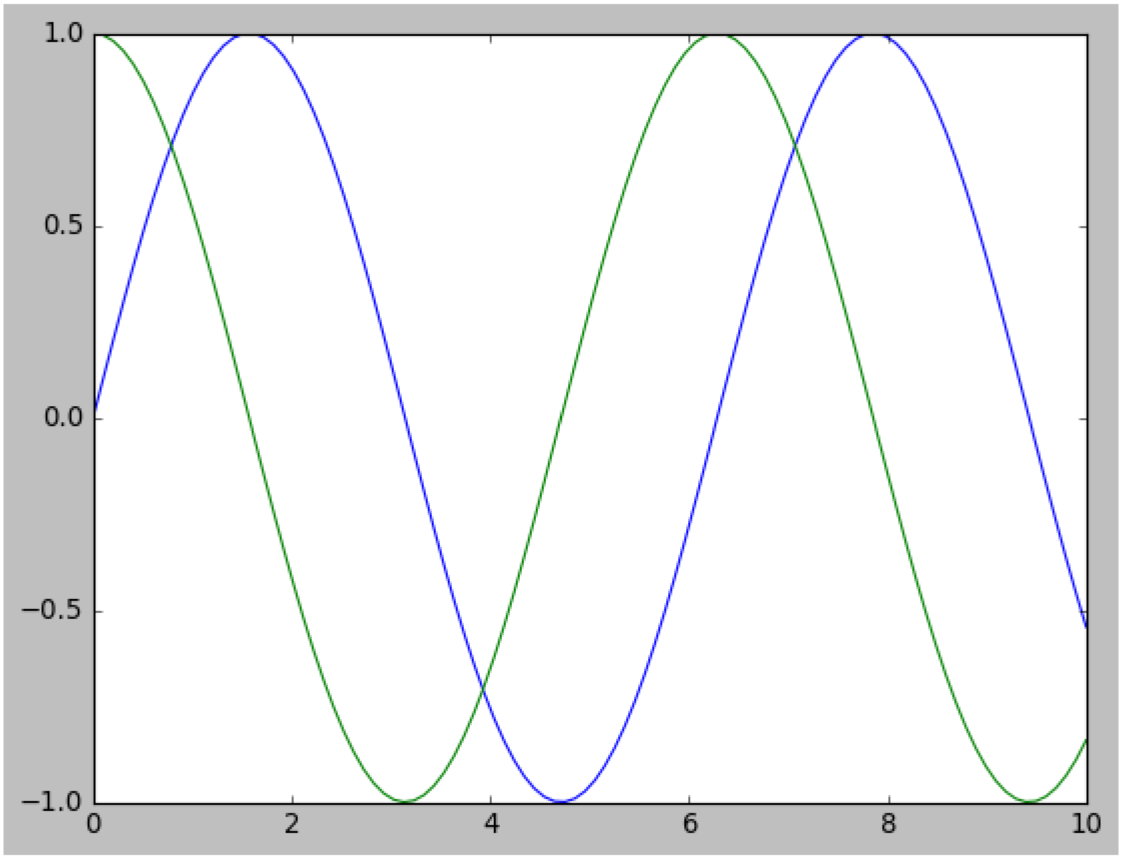
In [7]:

```
x1 = np.random.normal(0, 0.8, 1000)
x2 = np.random.normal(-2, 1, 1000)
x3 = np.random.normal(3, 2, 1000)
kwargs = dict(histtype='stepfilled', alpha=0.3, bins=40)
plt.hist(x1, **kwargs)
plt.hist(x2, **kwargs)
plt.hist(x3, **kwargs);
```



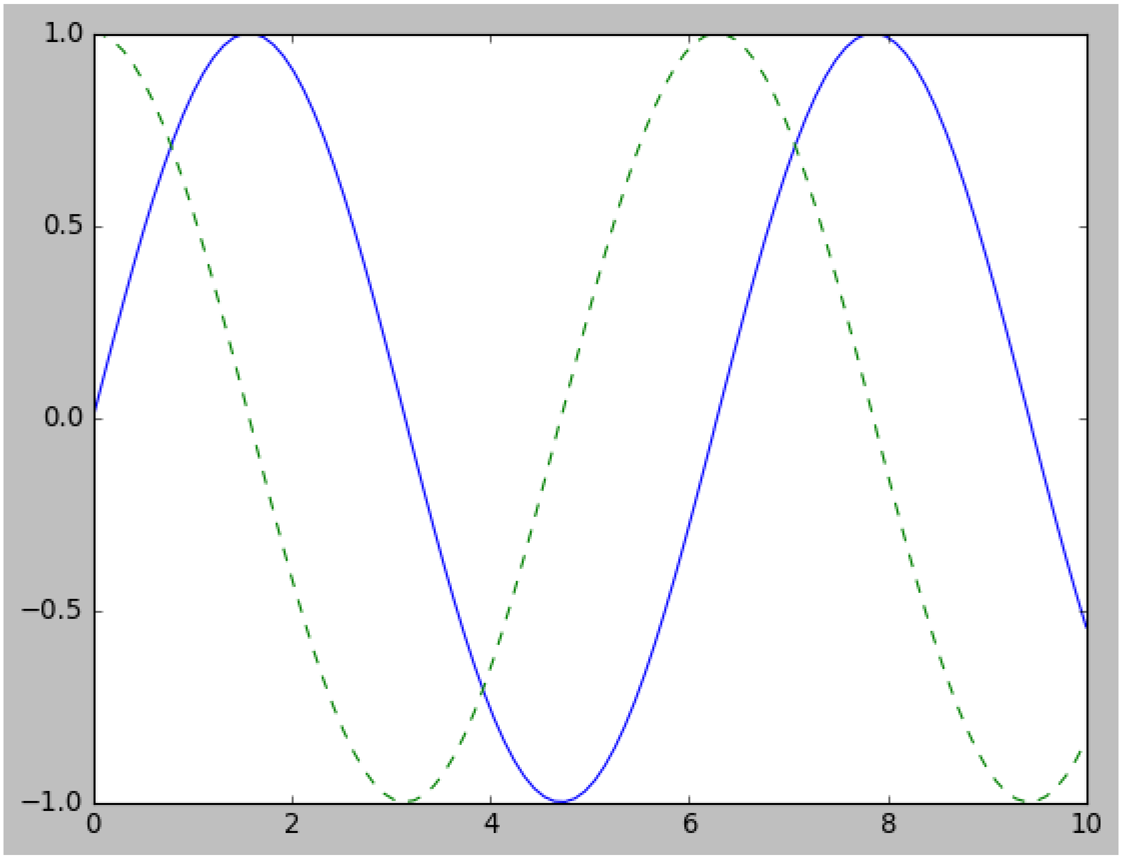
In [14]:

```
import matplotlib.pyplot as plt
import numpy as np
#plt.style.use('classic')
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.show()
```



In [18]:

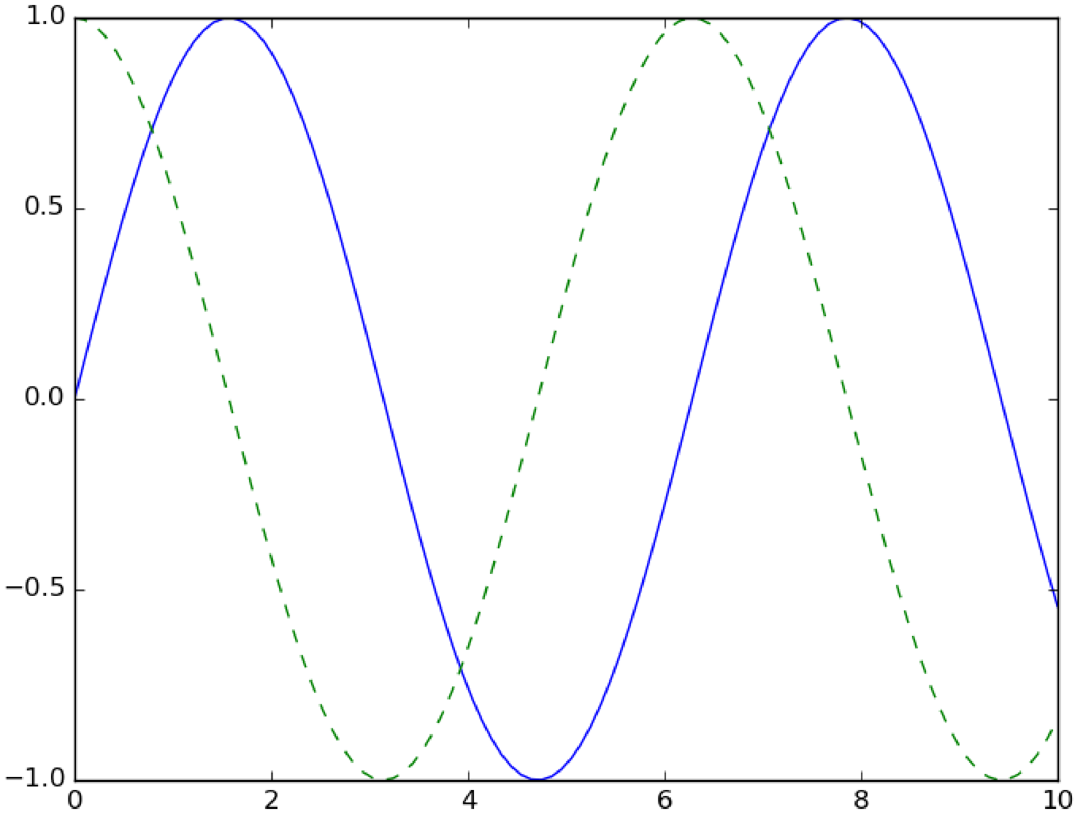
```
import numpy as np
x = np.linspace(0, 10, 100)
fig = plt.figure()
plt.plot(x, np.sin(x), '-')
plt.plot(x, np.cos(x), '--')
fig.savefig('fig1.png')
```



In [20]:

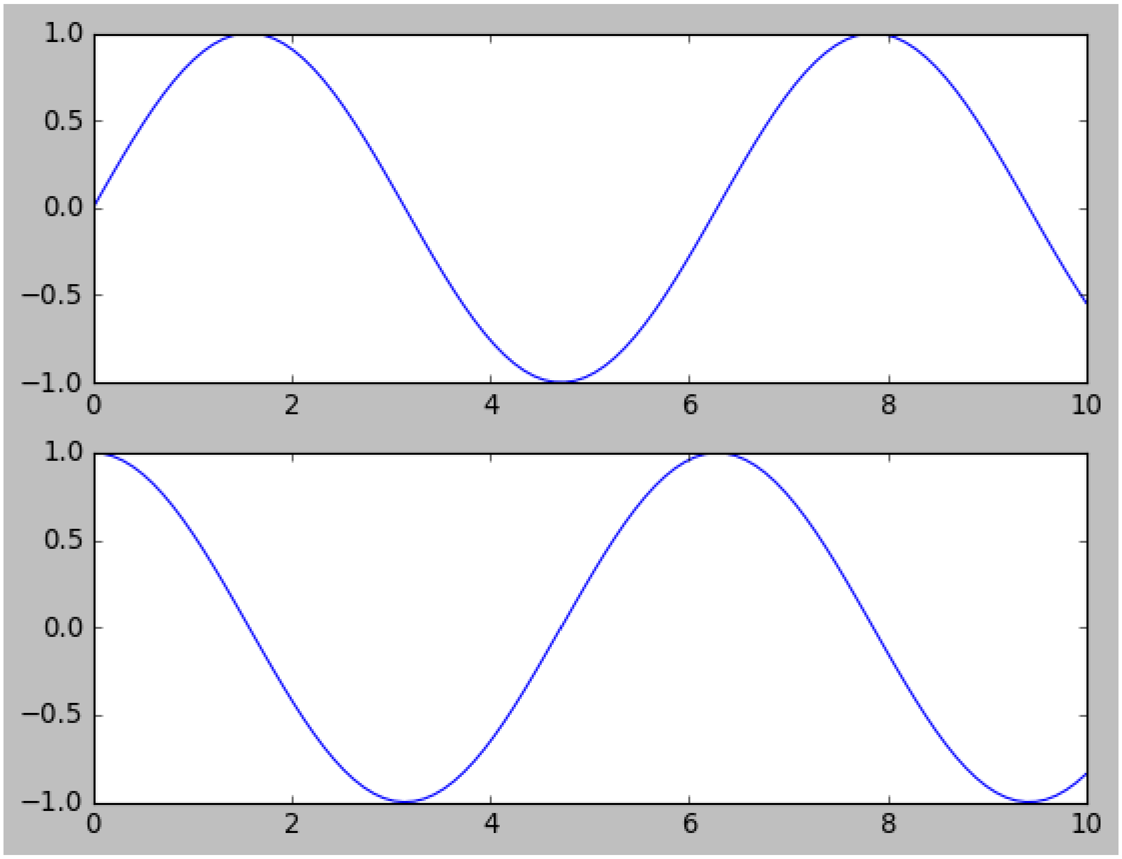
```
from IPython.display import Image
Image('fig1.png')
```

Out[20]:



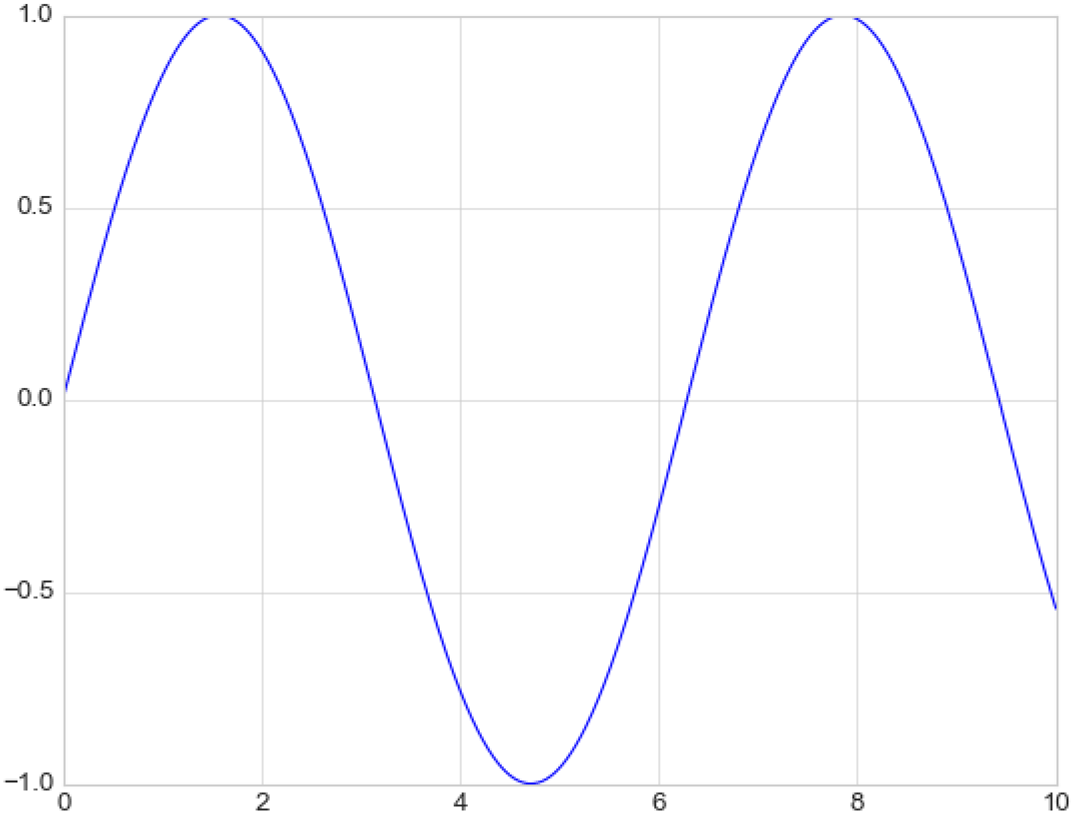
In [21]:

```
#Object-oriented interface
'''The object-oriented interface is available for these more complicated situations, an
for when you want more control over your figure
Rather than depending on some notion of an “active” figure or axes, in the object-orien
tions are methods of explicit Figure and Axes objects

'''
# First create a grid of plots
# ax will be an array of two Axes objects
fig, ax = plt.subplots(2)
# Call plot() method on the appropriate object
ax[0].plot(x, np.sin(x))
ax[1].plot(x, np.cos(x));
```



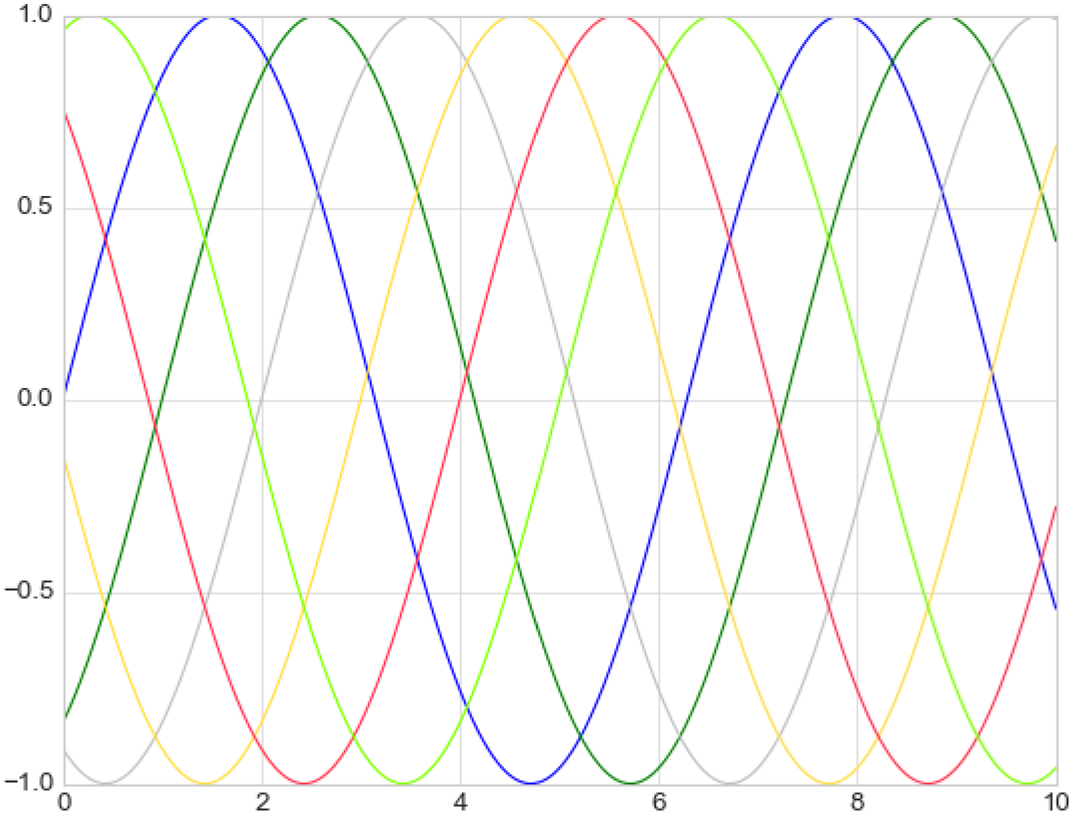
In [23]:

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x));
```



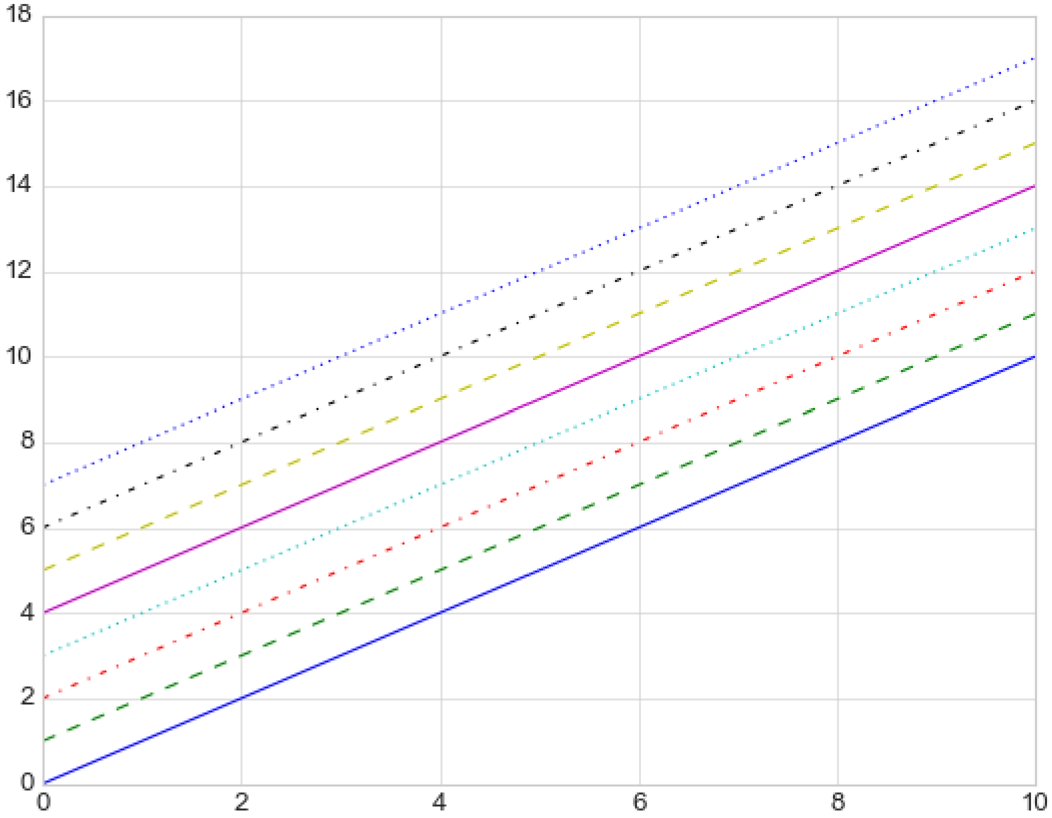
In [24]:

```
#Adjusting the Plot: Line Colors and Styles
plt.plot(x, np.sin(x - 0), color='blue') # specify color by name
plt.plot(x, np.sin(x - 1), color='g') # short color code (rgbcmyk)
plt.plot(x, np.sin(x - 2), color='0.75') # Grayscale between 0 and 1
plt.plot(x, np.sin(x - 3), color='#FFDD44') # Hex code (RRGGBB from 00 to FF)
plt.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3)) # RGB tuple, values 0 and 1
plt.plot(x, np.sin(x - 5), color='chartreuse'); # all HTML color names supported
```



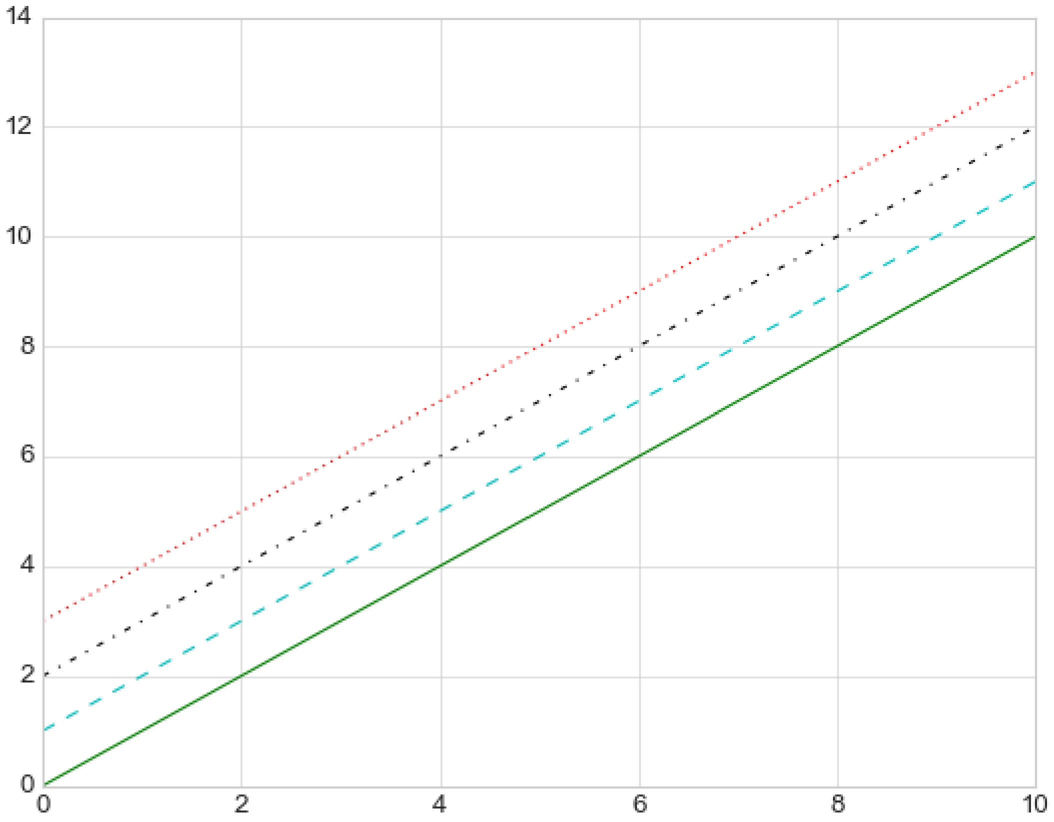
In [25]:

```
#adjust the line style using the linestyle
plt.plot(x, x + 0, linestyle='solid')
plt.plot(x, x + 1, linestyle='dashed')
plt.plot(x, x + 2, linestyle='dashdot')
plt.plot(x, x + 3, linestyle='dotted');
# For short, you can use the following codes:
plt.plot(x, x + 4, linestyle='-') # solid
plt.plot(x, x + 5, linestyle='--') # dashed
plt.plot(x, x + 6, linestyle='-.') # dashdot
plt.plot(x, x + 7, linestyle=':'); # dotted
```



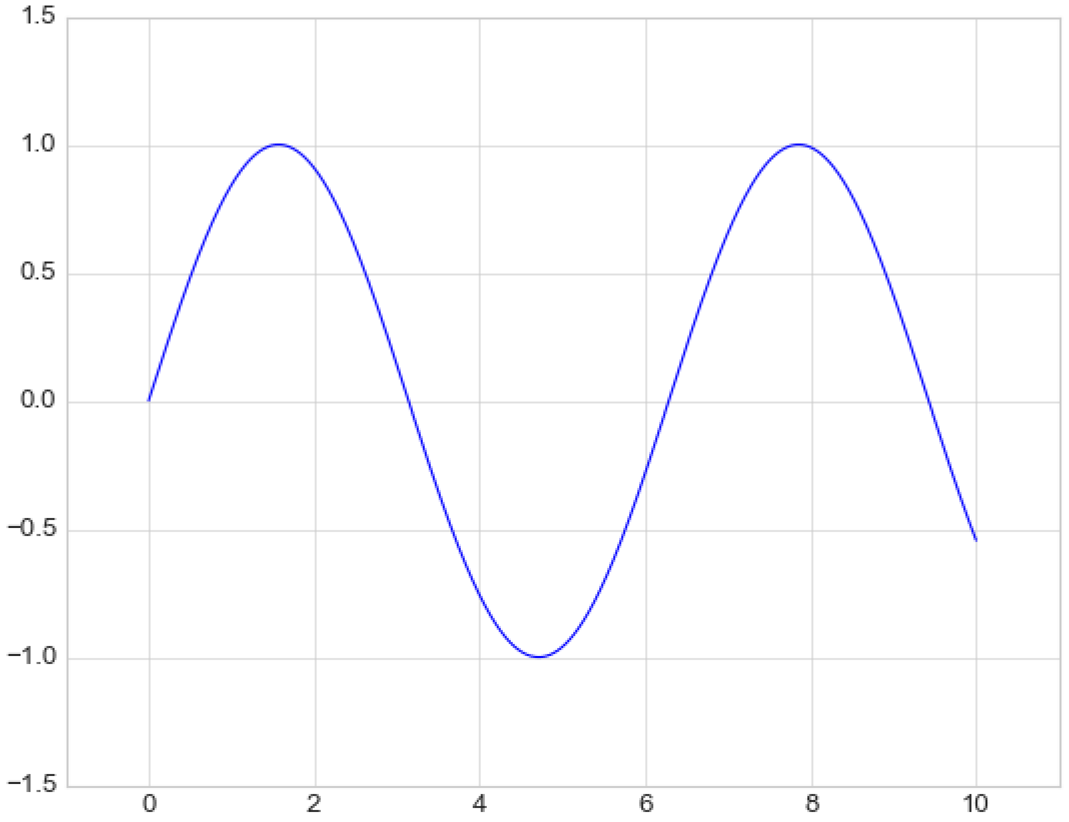
In [26]:

```
#shorthand syntax
plt.plot(x, x + 0, '-g') # solid green
plt.plot(x, x + 1, '--c') # dashed cyan
plt.plot(x, x + 2, '-.k') # dashdot black
plt.plot(x, x + 3, ':r'); # dotted red
```



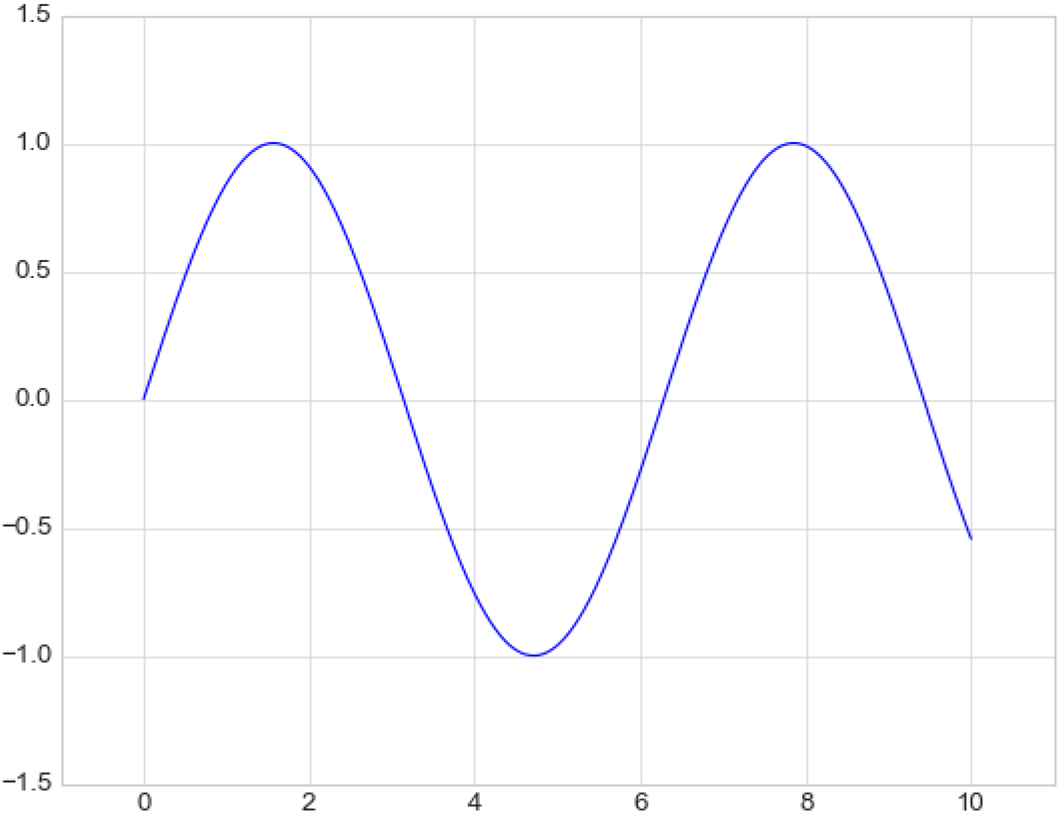
In [27]:

```
#Adjusting the Plot: Axes Limits
plt.plot(x, np.sin(x))
plt.xlim(-1, 11)
plt.ylim(-1.5, 1.5);
```



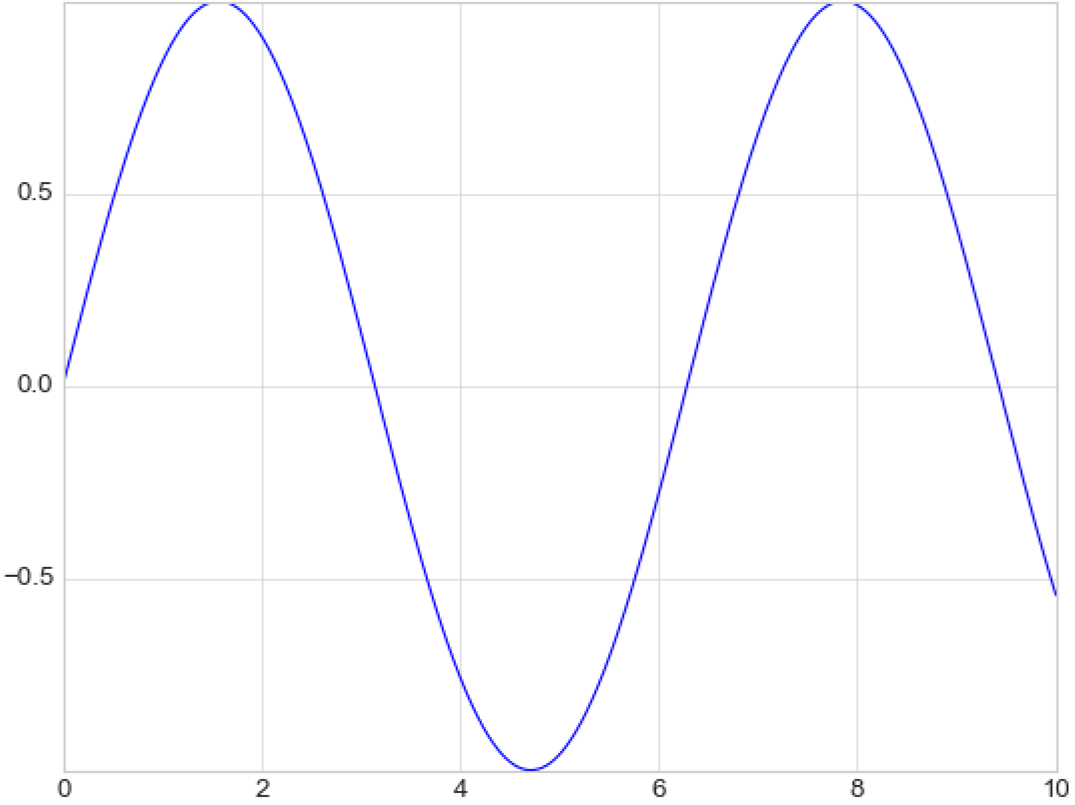
In [28]:

```
#The plt.axis() method allows you to set the x and y limits with a single call
#[xmin, xmax, ymin,ymax]
plt.plot(x, np.sin(x))
plt.axis([-1, 11, -1.5, 1.5]);
```



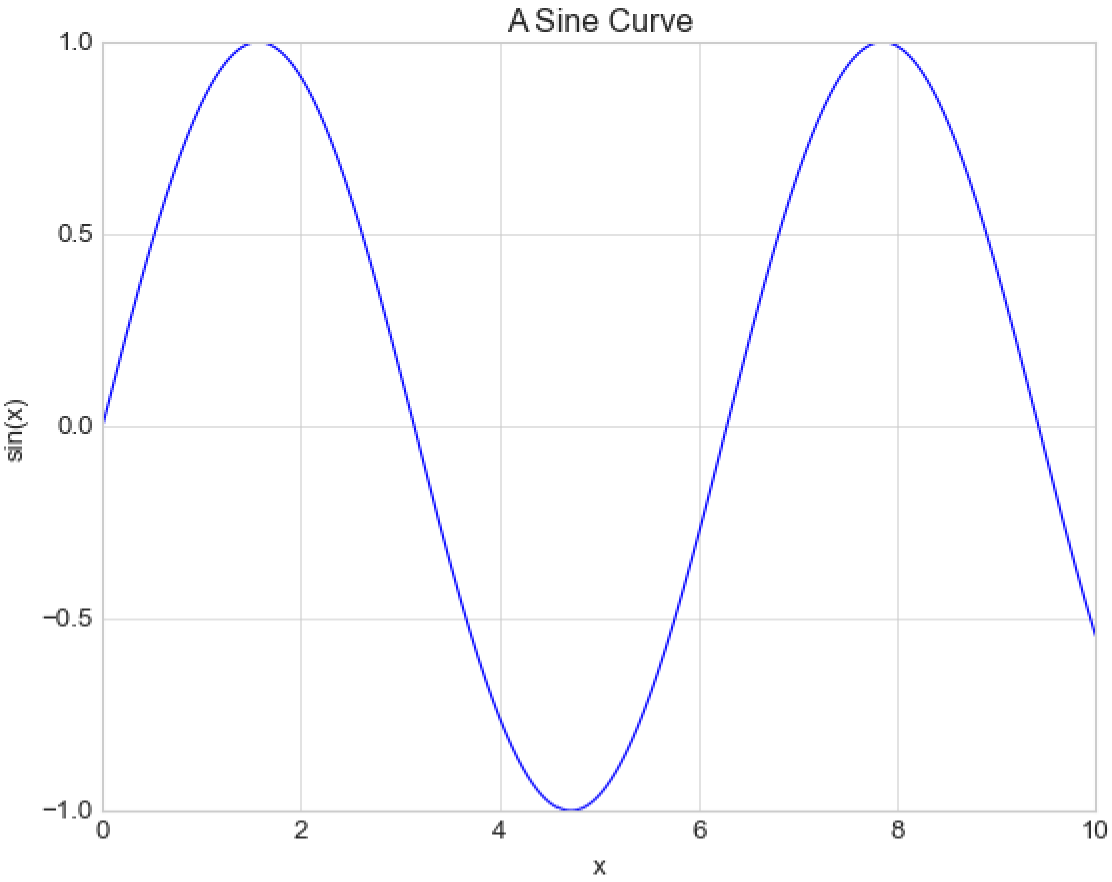
In [29]:

```
# tighten the bounds around the current plot
plt.plot(x, np.sin(x))
plt.axis('tight');
```



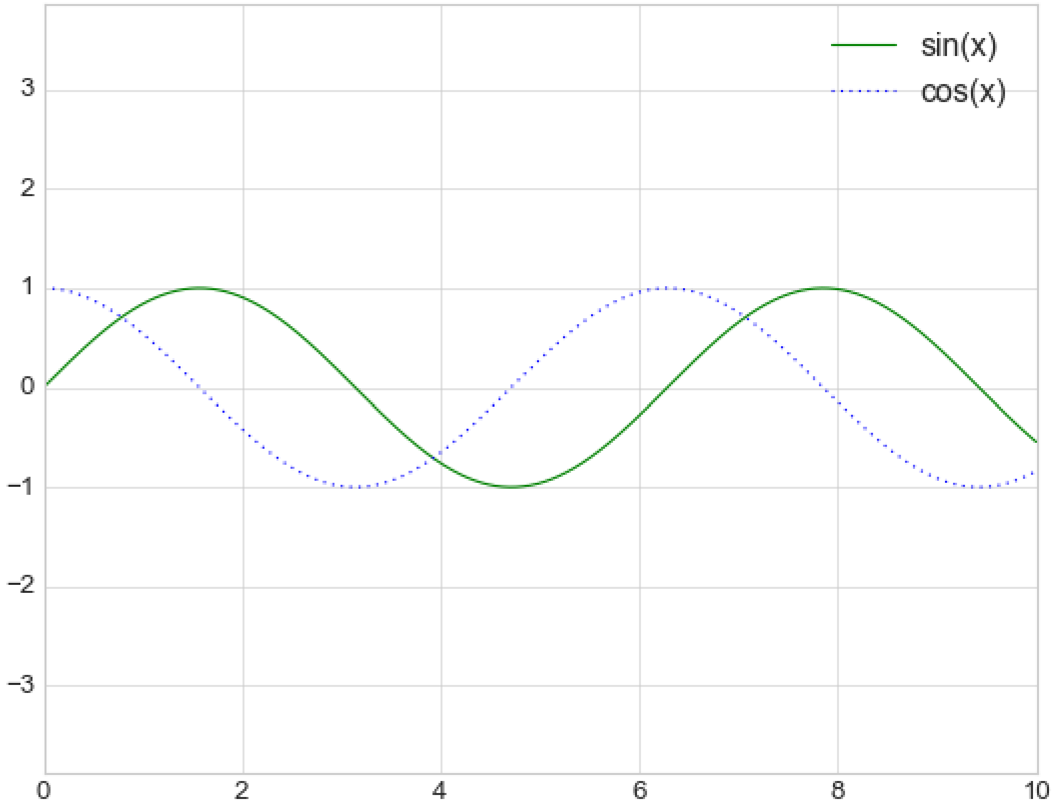
In [30]:

```
plt.plot(x, np.sin(x))
plt.title("A Sine Curve")
plt.xlabel("x")
plt.ylabel("sin(x)");
```



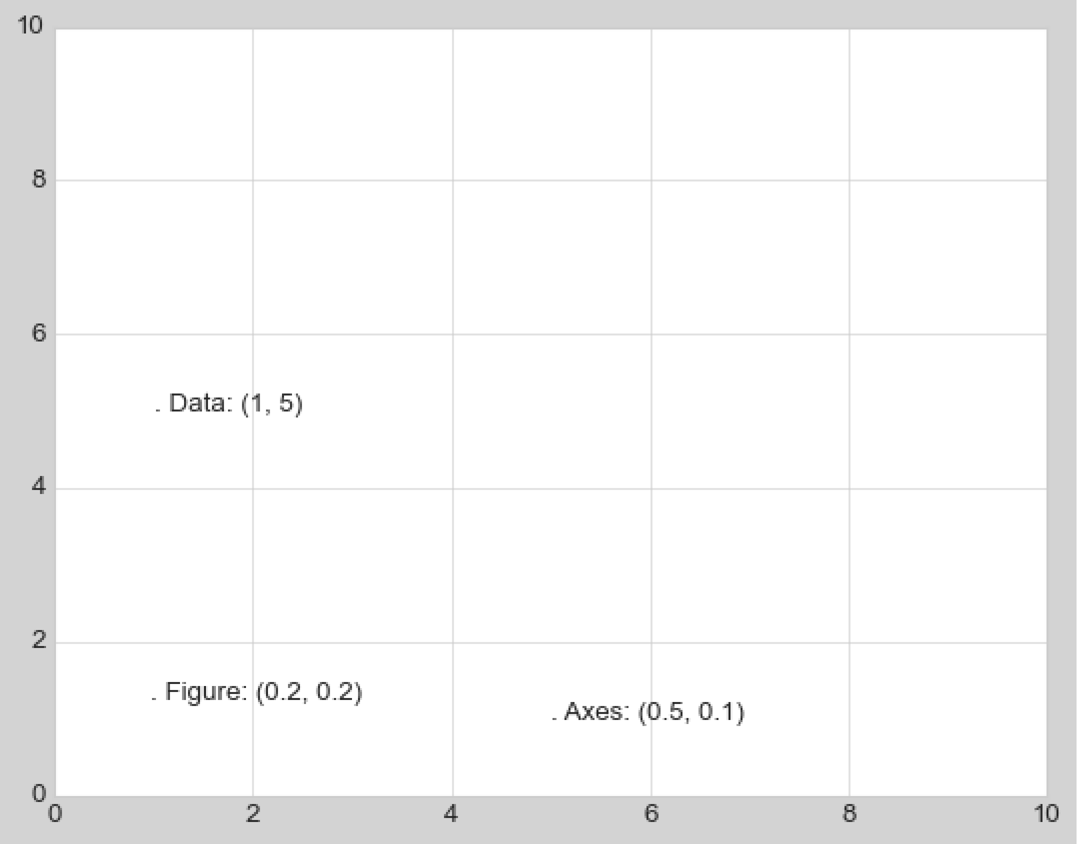
In [31]:

```
#plot legend that labels each line type
plt.plot(x, np.sin(x), '-g', label='sin(x)')
plt.plot(x, np.cos(x), ':b', label='cos(x)')
plt.axis('equal')
plt.legend();
```



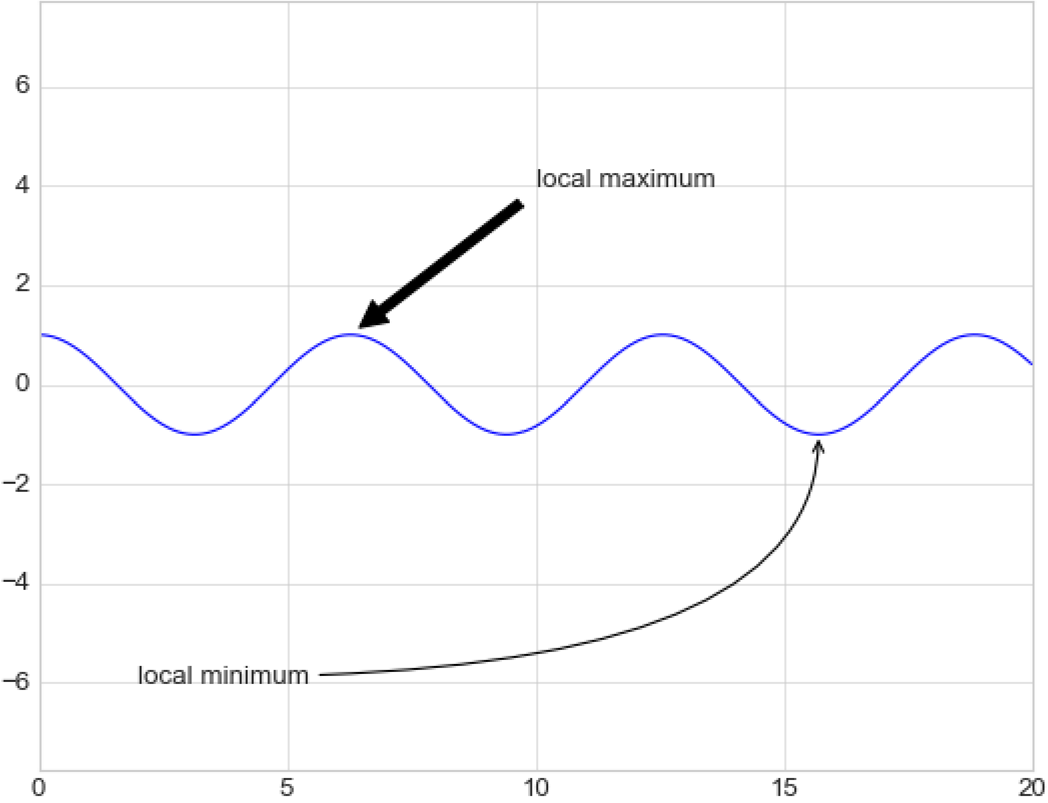
In [32]:

```
#Transforms and Text Position
'''
The transData coordinates give the usual data coordinates associated with the x- and
y-axis labels. The transAxes coordinates give the location from the bottom-left cor-
ner of the axes (here the white box) as a fraction of the axes size. The transFigure
coordinates are similar, but specify the position from the bottom left of the figure
(here the gray box) as a fraction of the figure size.


'''



fig, ax = plt.subplots(facecolor='lightgray')
ax.axis([0, 10, 0, 10])
# transform=ax.transData is the default, but we'll specify it anyway
ax.text(1, 5, ". Data: (1, 5)", transform=ax.transData)
ax.text(0.5, 0.1, ". Axes: (0.5, 0.1)", transform=ax.transAxes)
ax.text(0.2, 0.2, ". Figure: (0.2, 0.2)", transform=fig.transFigure);
```



In [33]:

```
#Arrows and Annotation
#arrow style is controlled through the arrowprops dictionary

fig, ax = plt.subplots()
x = np.linspace(0, 20, 1000)
ax.plot(x, np.cos(x))
ax.axis('equal')
ax.annotate('local maximum', xy=(6.28, 1), xytext=(10, 4),
arrowprops=dict(facecolor='black', shrink=0.05))
ax.annotate('local minimum', xy=(5 * np.pi, -1), xytext=(2, -6),
arrowprops=dict(arrowstyle="->",
connectionstyle="angle3,angleA=0,angleB=-90"));
```



In [45]:

```
import pandas as pd
births = pd.read_csv('births.csv')
print(births.head())
print(births.index)
births_by_date = births.pivot_table('births', [births.index.month, births.index.day])

fig, ax = plt.subplots(figsize=(12, 4))
births_by_date.plot(ax=ax)
# Add labels to the plot
ax.annotate("New Year's Day", xy=('2012-1-1', 4100), xycoords='data',
 xytext=(50, -30), textcoords='offset points',
 arrowprops=dict(arrowstyle="->",
 connectionstyle="arc3,rad=-0.2"))

ax.annotate("Independence Day", xy=('2012-7-4', 4250), xycoords='data',
 bbox=dict(boxstyle="round", fc="none", ec="gray"),
 xytext=(10, -40), textcoords='offset points', ha='center',
 arrowprops=dict(arrowstyle="->"))

ax.annotate('Labor Day', xy=('2012-9-4', 4850), xycoords='data', ha='center',
 xytext=(0, -20), textcoords='offset points')
ax.annotate('', xy=('2012-9-1', 4850), xytext=('2012-9-7', 4850),
 xycoords='data', textcoords='data',
 arrowprops={'arrowstyle': '|-|,widthA=0.2,widthB=0.2', })

ax.annotate('Halloween', xy=('2012-10-31', 4600), xycoords='data',
 xytext=(-80, -40), textcoords='offset points',
 arrowprops=dict(arrowstyle="fancy",
 fc="0.6", ec="none",
 connectionstyle="angle3,angleA=0,angleB=-90"))

ax.annotate('Thanksgiving', xy=('2012-11-25', 4500), xycoords='data',
 xytext=(-120, -60), textcoords='offset points',
 bbox=dict(boxstyle="round4,pad=.5", fc="0.9"),
 arrowprops=dict(arrowstyle="->",
 connectionstyle="angle,angleA=0,angleB=80,rad=20"))

ax.annotate('Christmas', xy=('2012-12-25', 3850), xycoords='data',
 xytext=(-30, 0), textcoords='offset points',
 size=13, ha='right', va="center",
 bbox=dict(boxstyle="round", alpha=0.1),
 arrowprops=dict(arrowstyle="wedge,tail_width=0.5", alpha=0.1));

# Label the axes
ax.set(title='USA births by day of year (1969-1988)',
 ylabel='average daily births')
# Format the x axis with centered month labels
ax.xaxis.set_major_locator(mpl.dates.MonthLocator())
ax.xaxis.set_minor_locator(mpl.dates.MonthLocator(bymonthday=15))
ax.xaxis.set_major_formatter(plt.NullFormatter())
ax.xaxis.set_minor_formatter(mpl.dates.DateFormatter('%h'));
ax.set_ylim(3600, 5400);
```

```
   year  month  day gender  births
0  1969      1  1.0      F    4046
1  1969      1  1.0      M    4440
2  1969      1  2.0      F    4454
3  1969      1  2.0      M    4548
4  1969      1  3.0      F    4548
RangeIndex(start=0, stop=15547, step=1)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-45-4b3760dc2cd3> in <module>
      3 print(births.head())
      4 print(births.index)
----> 5 births_by_date = births.pivot_table('births', [births.index.month, births.inde
x.day])
      6 
      7 fig, ax = plt.subplots(figsize=(12, 4))

AttributeError: 'RangeIndex' object has no attribute 'month'
```

```
In [ ]:
```