Plotting in Python with Matplotlib

Mario Alberto Quiñones Calvo

First we import the libraries we need

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

Bar Plots

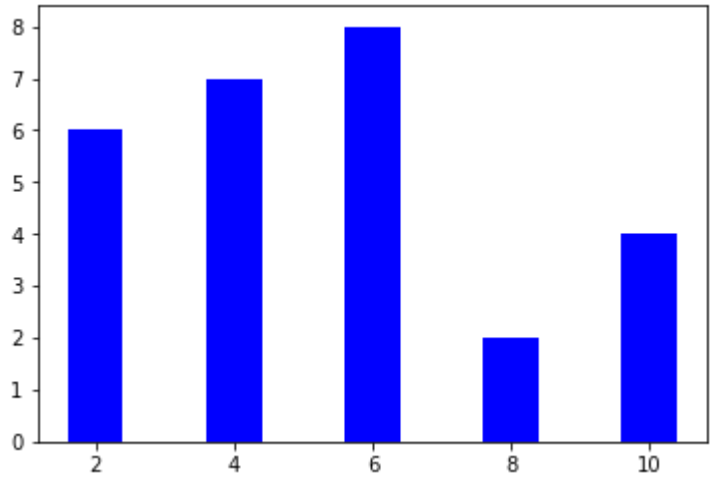
We need some data for the bar plots.

In [2]:
```
x1 = [ 2*(i+1) for i in range(5) ]
print('x1 = ' + str(x1))
y1 = [6,7,8,2,4]
print('y1 = ' + str(y1))

x2 = [ 2*(i)+1 for i in range(5) ]
print('x2 = ' + str(x2))
y2 = [7,8,2,4,2]
print('y2 = ' + str(y2))
```

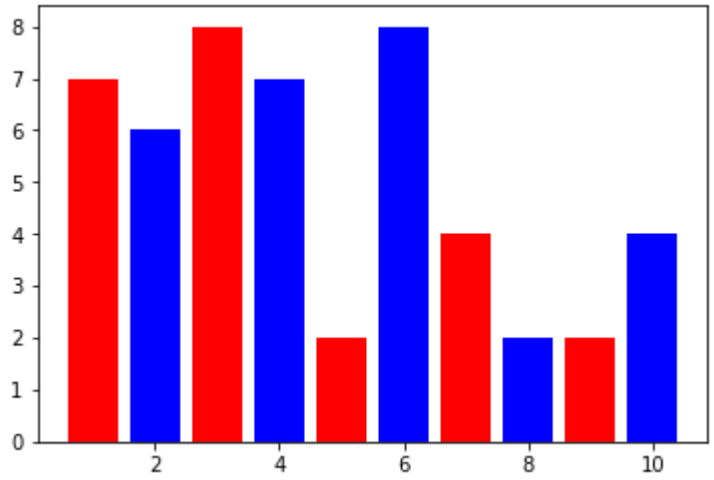
```
x1 = [2, 4, 6, 8, 10]
y1 = [6, 7, 8, 2, 4]
x2 = [1, 3, 5, 7, 9]
y2 = [7, 8, 2, 4, 2]
```

Now, we begin with the bar plot examples.

In [3]:
```
plt.bar( x1 , y1 , color = 'blue' )

plt.show()
```



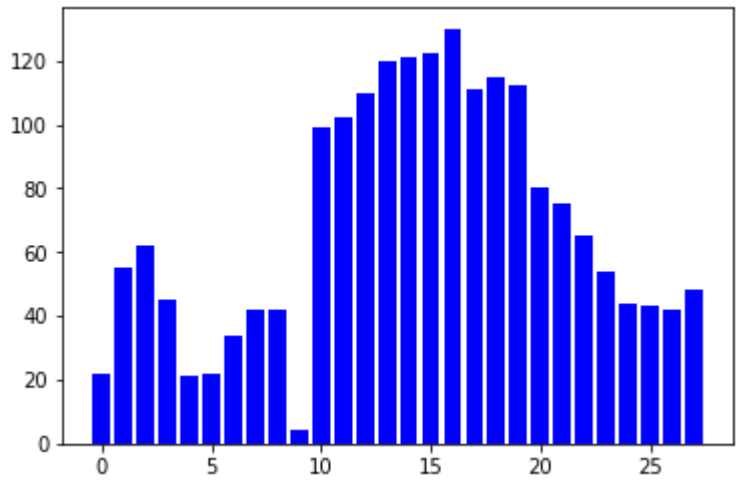
In [4]:
```
plt.bar( x1 , y1 , color = 'blue' )
plt.bar( x2 , y2 , color = 'red' )

plt.show()
```



Let's change the data for a better example.

In [5]:
```
population_ages = [22,55,62,45,21,22,34,42,42,4,99,102,110,120,121,122,130,\
                   111,115,112,80,75,65,54,44,43,42,48]
ids = [ x for x in range( len( population_ages ) ) ]
```

In [6]:
```
plt.bar( ids , population_ages , color = 'blue' )

plt.show()
```

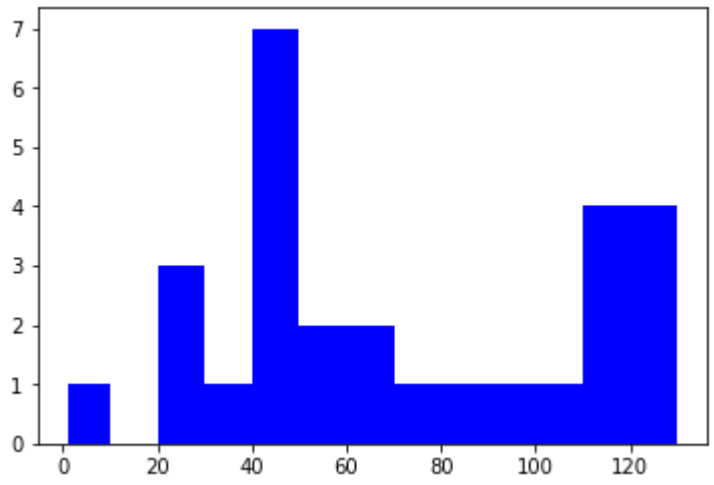


Histograms

We specify the bins and the type for the histogram we will plot.

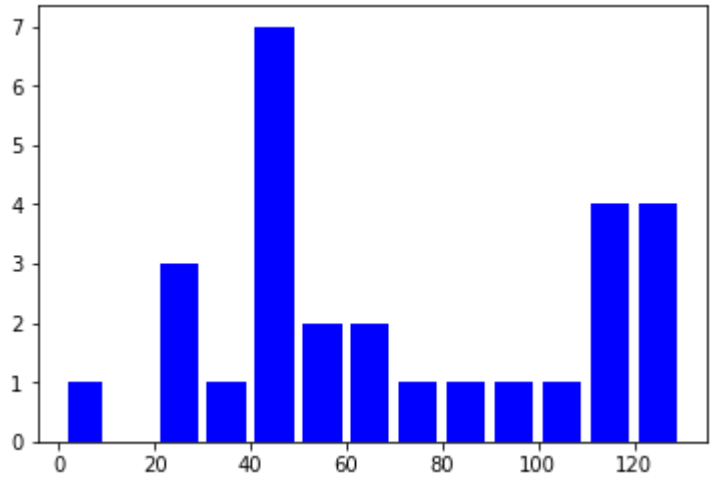
In [7]:
```
boxes = [1,10,20,30,40,50,60,70,80,90,100,110,120,130]
```

In [8]:
```
plt.hist( population_ages , bins = boxes , color = 'blue' , histtype = 'bar' )

plt.show()
```



Now, we specify the width of the bars.

In [9]:
```
plt.hist( population_ages , bins = boxes , color = 'blue' ,\
          histtype = 'bar' , rwidth = 0.8 )

plt.show()
```



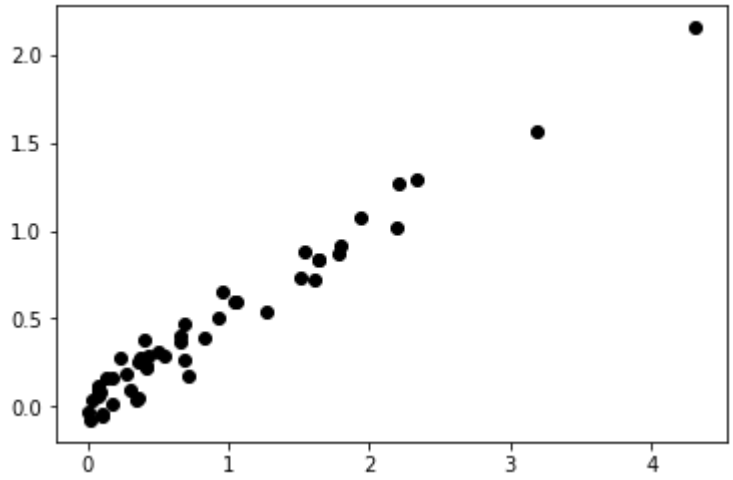
Scatter Plots

Let's change the data for these examples.

In [10]:
```
x = np.random.exponential( scale = 1 , size = 50 )
print('x = ' + str(x[0:7]) + ' ... ')
y = 0.5*x + np.random.normal( loc = 0 , scale = 0.1 , size = 50 )
print('y = ' + str(y[0:7]) + ' ... ')
```

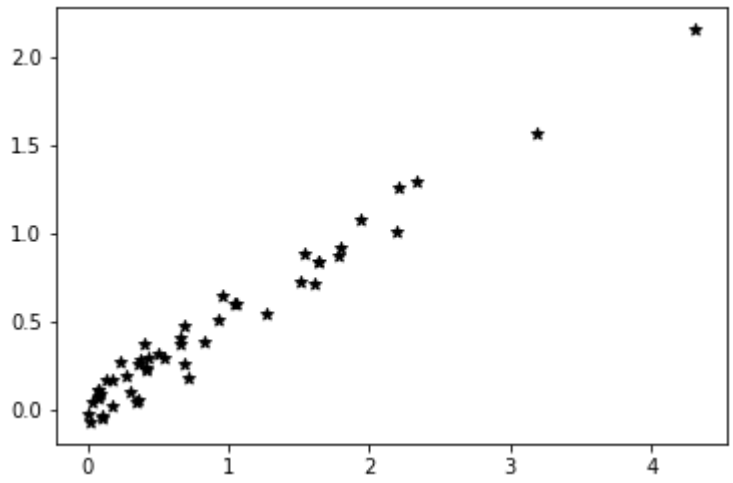
```
x = [ 0.0755029   1.77833533  1.05886375  0.68989103  0.31175846  0.92815316
  0.17154336] ...
y = [ 0.07868616  0.86675092  0.59590017  0.47386122  0.09273417  0.50613886
  0.02004814] ...
```

In [11]:
```
plt.scatter( x , y , color = 'black' )

plt.show()
```



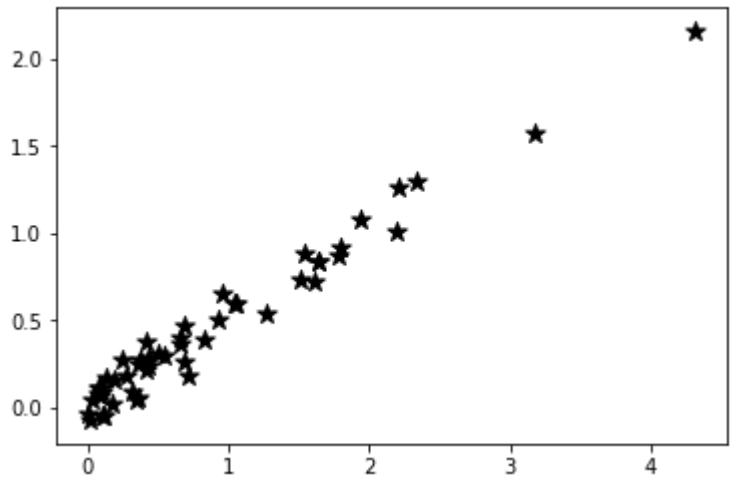
Now, we change the marker for the scatter plot.

In [12]:
```
plt.scatter( x , y , color = 'black' , marker = '*' )

plt.show()
```



We can, also, change the size of the markers.

In [13]:
```
plt.scatter( x , y , color = 'black' , marker = '*' , s = 100 )

plt.show()
```



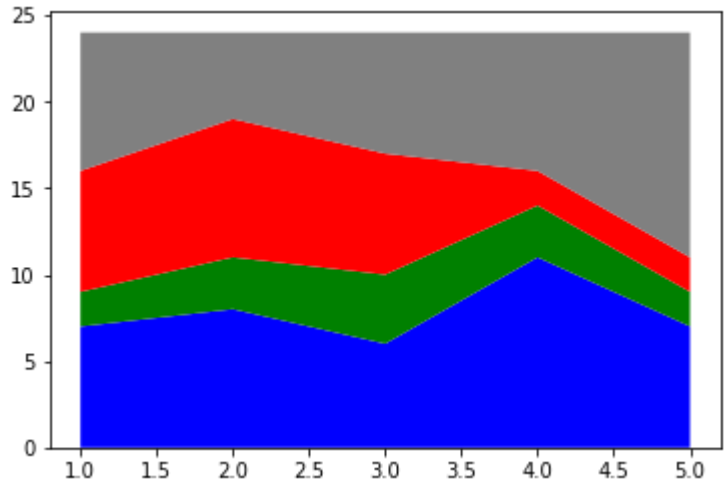
Stack Plots

Let's change the data for these examples.

In [14]:
```
days = [1,2,3,4,5]

sleeping = [7,8,6,11,7]
eating = [2,3,4,3,2]
working = [7,8,7,2,2]
playing = [8,5,7,8,13]
```

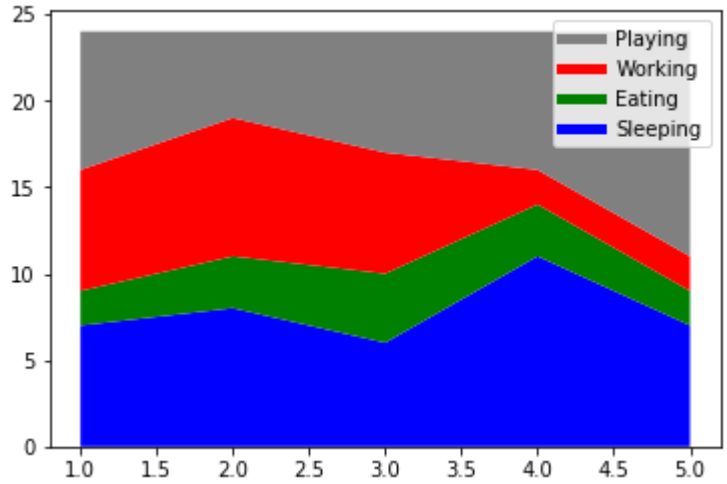
We can specify the colors of our variables in a stack plot.

In [15]:
```
plt.stackplot( days ,\
               sleeping , eating , working , playing ,\
               colors = ['blue','green','red','grey'] )

plt.show()
```



We can, also, add a legend for the variables. This is an odd way to add a legend.

In [16]:
```
plt.stackplot( days ,\
               sleeping , eating , working , playing ,\
               colors = ['blue','green','red','grey'] )
plt.plot( [] , [] , color = 'grey' , label = 'Playing' , linewidth = 5 )
plt.plot( [] , [] , color = 'red' , label = 'Working' , linewidth = 5 )
plt.plot( [] , [] , color = 'green' , label = 'Eating' , linewidth = 5 )
plt.plot( [] , [] , color = 'blue' , label = 'Sleeping' , linewidth = 5 )

plt.legend()
plt.show()
```



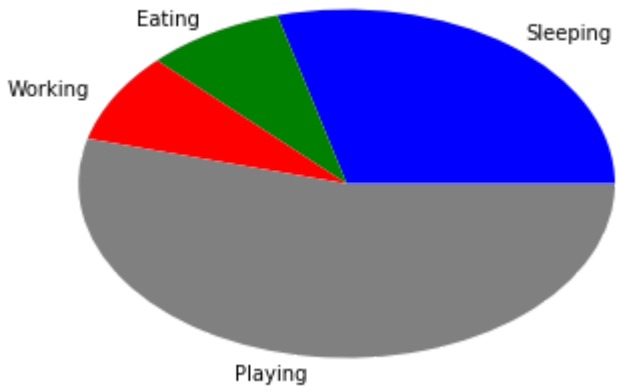
Pie Plots

Let's get some data for these examples.

In [17]:
```
slices = [7,2,2,13]
```

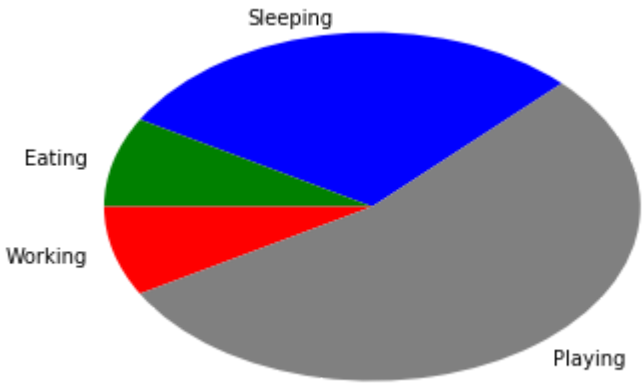
We can specify the labels and the colors of our variables in a pie plot.

In [18]:
```
activities = ['Sleeping','Eating','Working','Playing']
mycolors = ['blue','green','red','grey']

plt.pie( slices , labels = activities , colors = mycolors )

plt.show()
```



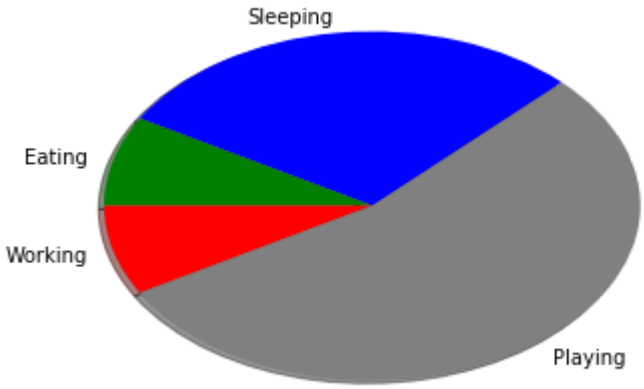
We can, also, specify the starting angle for our pie plot.

In [19]:
```
plt.pie( slices , labels = activities , colors = mycolors , startangle = 45 )

plt.show()
```



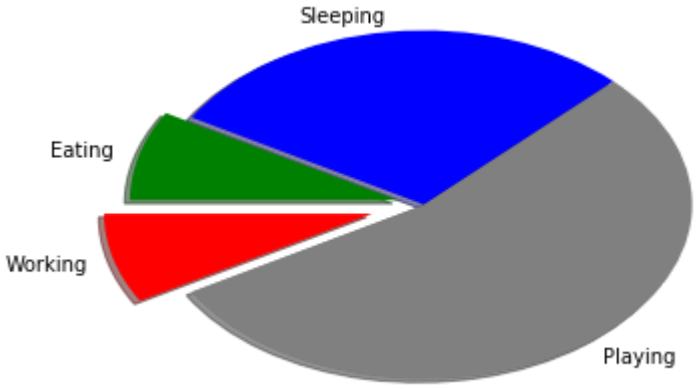
We can add a 3d effect to our pie plot with a shadow.

In [20]:
```
plt.pie( slices , labels = activities , colors = mycolors ,\
         startangle = 45 , shadow = True )

plt.show()
```



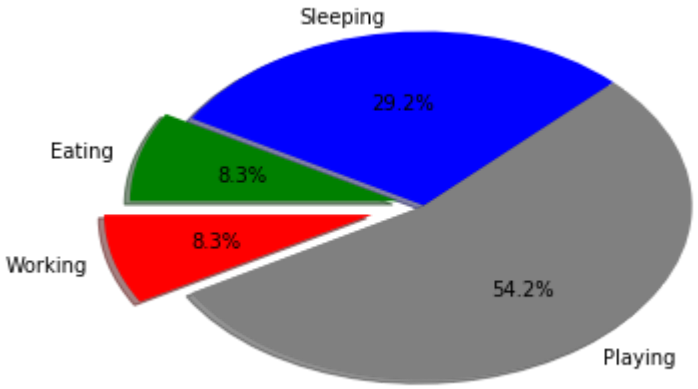
We can emphasize some slices in our pie plot.

In [21]:
```
plt.pie( slices , labels = activities , colors = mycolors ,\
         startangle = 45 , shadow = True , explode = (0,0.1,0.2,0) )

plt.show()
```



We can add a percentage for each variable in our pie plot.

In [22]:
```
plt.pie( slices , labels = activities , colors = mycolors ,\
         startangle = 45 , shadow = True ,\
         explode = (0,0.1,0.2,0) , autopct = '%1.1f%%' )

plt.show()
```



Line Plots and Subplots

Basic customizations

Let's get some data for these examples.

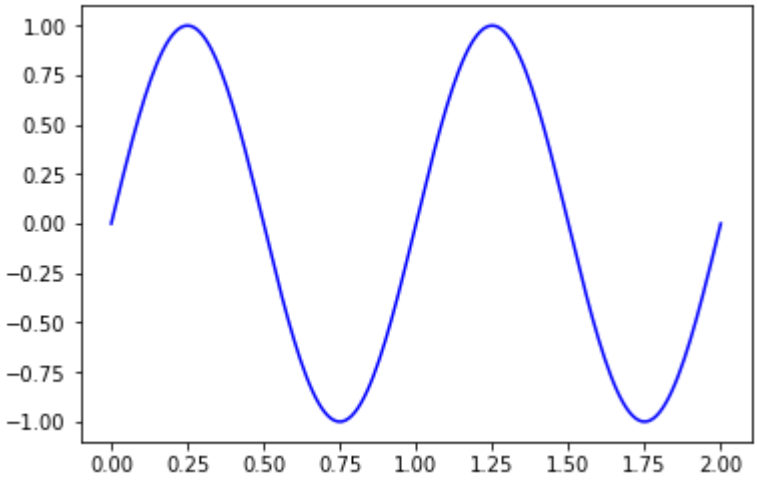
In [23]:
```
x1 = np.arange( 0.0 , 2.01 , 0.01 )
y1 = np.sin( 2 * np.pi * x1 )
y2 = 1.5 * np.sin( 4 * np.pi * x1 )
y3 = 3 * np.sin( 8 * np.pi * x1 )
```

Note:

From now on, we will create a reference for the figure on which we work, and a reference for the subplots that we will create on our figure.

We begin with a simple plot.

In [24]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )

plt.show()
```



Note:

We created the reference for the figure in case we have to use it.

The figure is like the whole space where you can create subplots on, and its like a matrix composed of spaces for the subplots.

Each subplot is like a sub-matrix of the figure.

We specify the "dimensions" of the figure, the "coordinates" of each subplot, and the "dimensions" of each subplot with the **.subplot2grid()** command.

The **.subplot2grid(param1 , param2 , rowspan , colspan)** specification is the following:

param1 is the tuple of the "dimensions" of the figure; i.e., the number of row spaces (on the left) and the number of column spaces (on the right) that the figure will be composed of.

param2 is the tuple of the "starting point coordinates" of the subplot in the figure matrix; in other words, the "localization" coordinates of the subplot in the figure. These indexes start from cero.

The following two parameters are the "dimensions" of the subplot.

rowspan is the number of "row spaces" that the subplot will use.

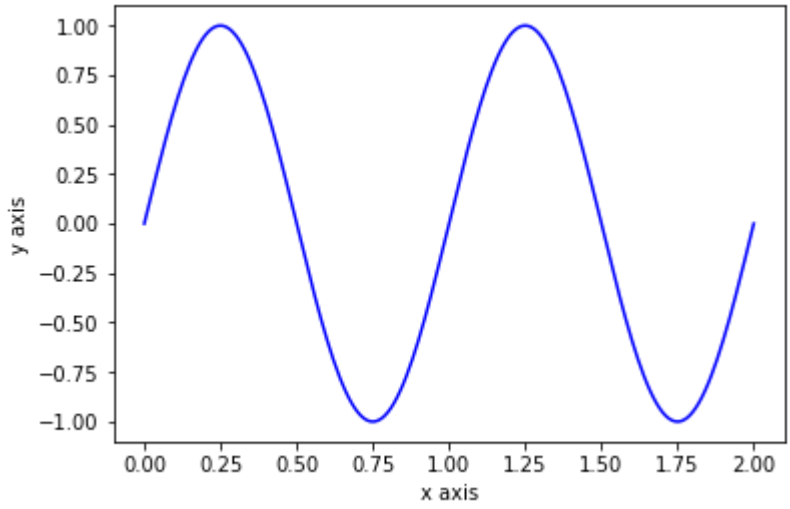
colspan is the number of "column spaces" that the subplot will use.

In the example above, **ax1** is the subplot on a **figure of 1 by 1** that **starts on the first row and frist column**, and **ax1 is a 1 by 1 subplot**.

For the moment, we will continue to use a 1 by 1 figure and a 1 by 1 subplot.

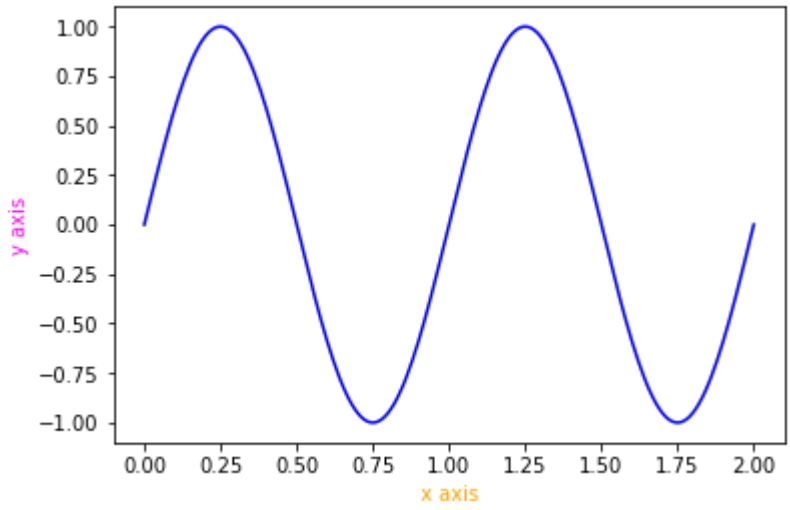
We can add some axis labels to our plot.

In [25]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )

plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```



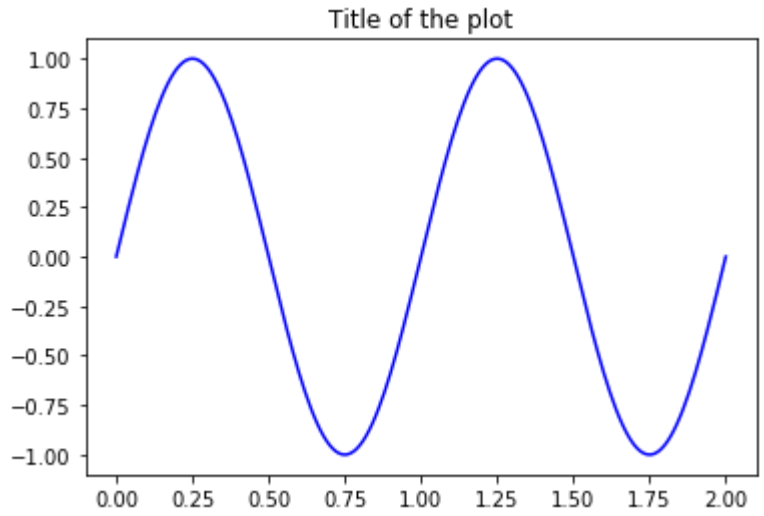
We can, also, change the color of the axis labels of our plot.

In [26]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.xaxis.label.set_color('orange')
ax1.yaxis.label.set_color('magenta')

plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```



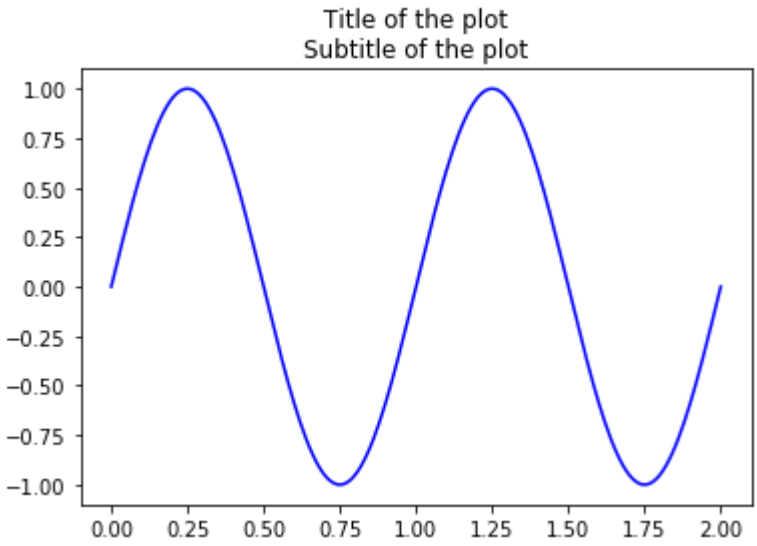
We can add a title to our plot.

In [27]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )

plt.title('Title of the plot')
plt.show()
```



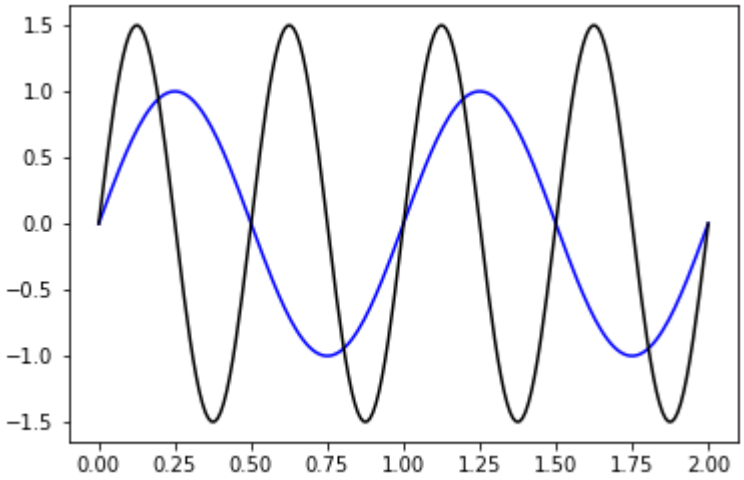
We can, also, add a subtitle to the plot. This is an odd way to add a subtitle to a plot.

In [28]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )

plt.title('Title of the plot\nSubtitle of the plot')
plt.show()
```



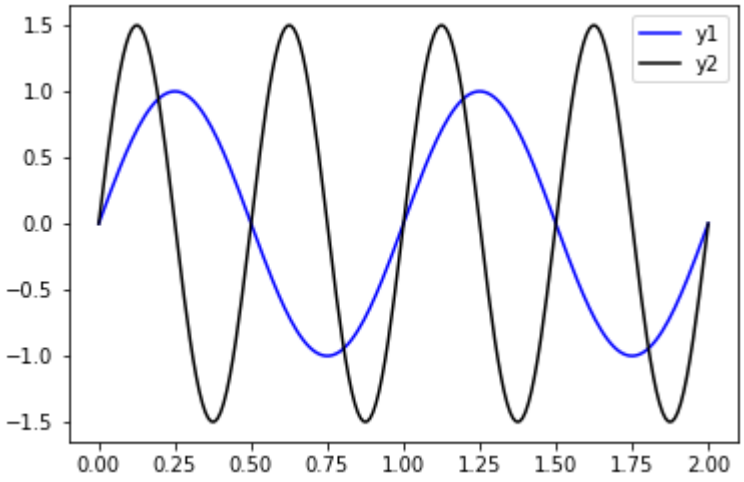
We can plot multiple times in one subplot.

In [29]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.plot( x1 , y2 , color = 'black' )

plt.show()
```



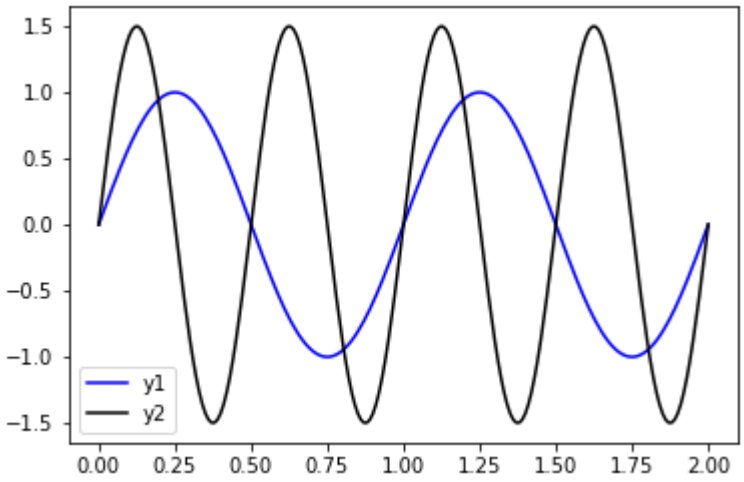
Besides the color of each line, we can add a legend to clarify the plot.

In [30]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' , label = 'y1' )
ax1.plot( x1 , y2 , color = 'black' , label = 'y2' )
ax1.legend()

plt.show()
```



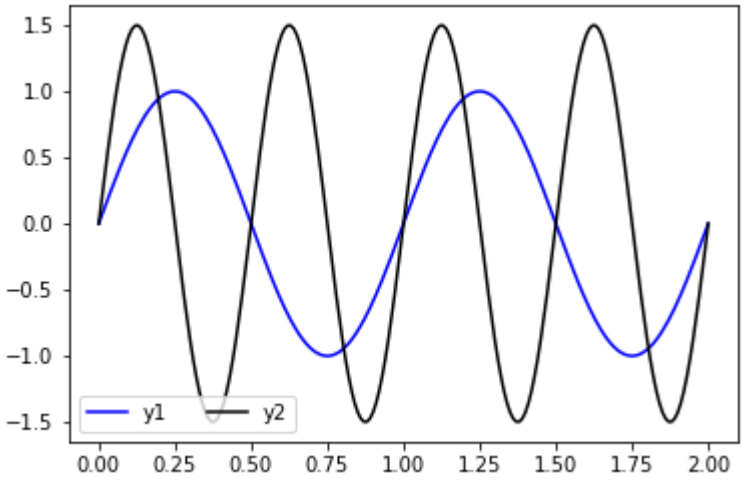
We can, also, specify the location of such legend.

In [31]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' , label = 'y1' )
ax1.plot( x1 , y2 , color = 'black' , label = 'y2' )
ax1.legend( loc = 3 )

plt.show()
```



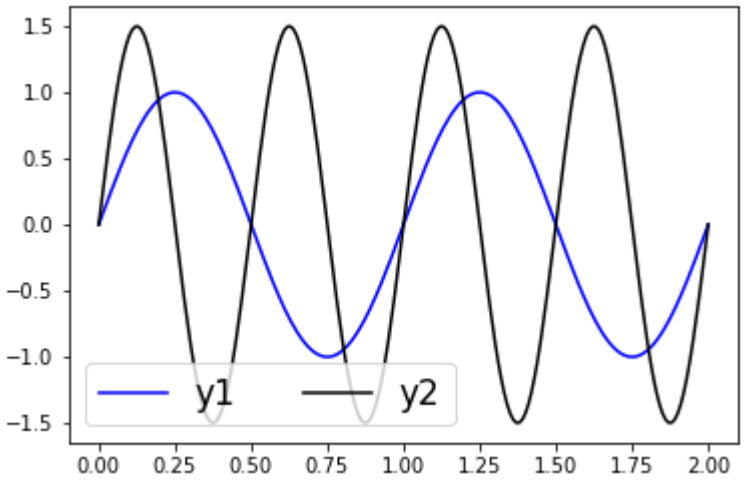
We can, also, specify the shape of the legend.

In [32]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' , label = 'y1' )
ax1.plot( x1 , y2 , color = 'black' , label = 'y2' )
ax1.legend( loc = 3 , ncol = 2 )

plt.show()
```



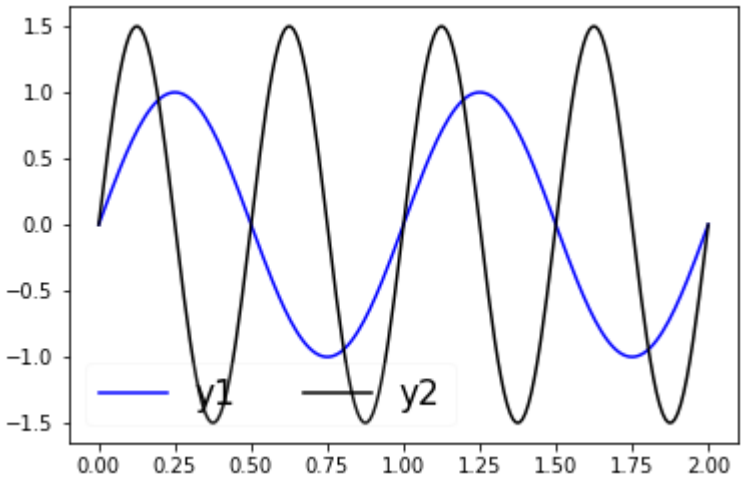
We can, also, specify the size of the legend.

In [33]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' , label = 'y1' )
ax1.plot( x1 , y2 , color = 'black' , label = 'y2' )
ax1.legend( loc = 3 , ncol = 2 , prop = {'size':17} )

plt.show()
```



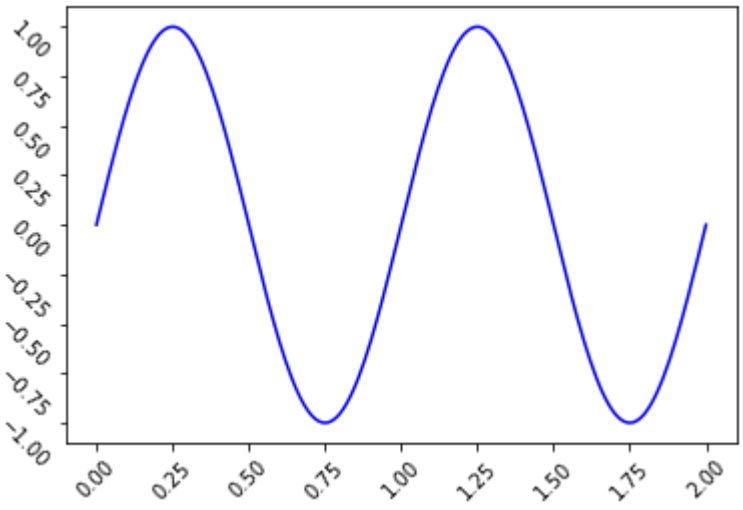
We can, also, change the transparency of the frame of the legend.

In [34]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' , label = 'y1' )
ax1.plot( x1 , y2 , color = 'black' , label = 'y2' )
leg = ax1.legend( loc = 3 , ncol = 2 , prop = {'size':17} )
leg.get_frame().set_alpha(0.1)

plt.show()
```



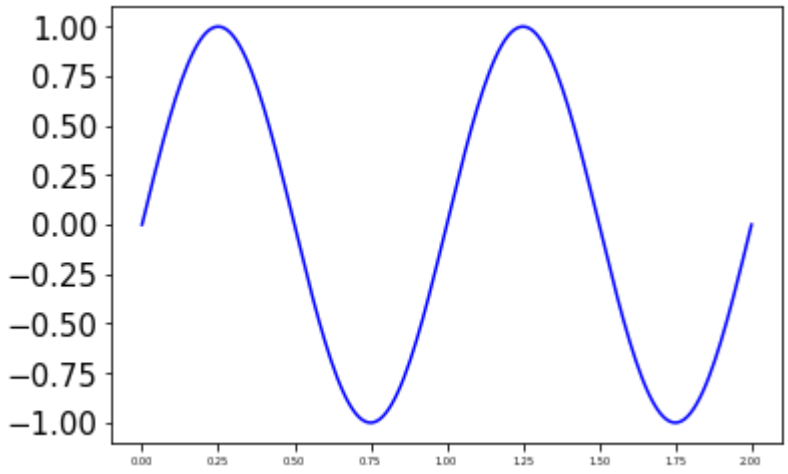
We can modify the labels for the ticks in each axis. First, we will rotate them.

In [35]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
for label in ax1.xaxis.get_ticklabels() :
    label.set_rotation(45)
for label in ax1.yaxis.get_ticklabels() :
    label.set_rotation(-45)

plt.show()
```



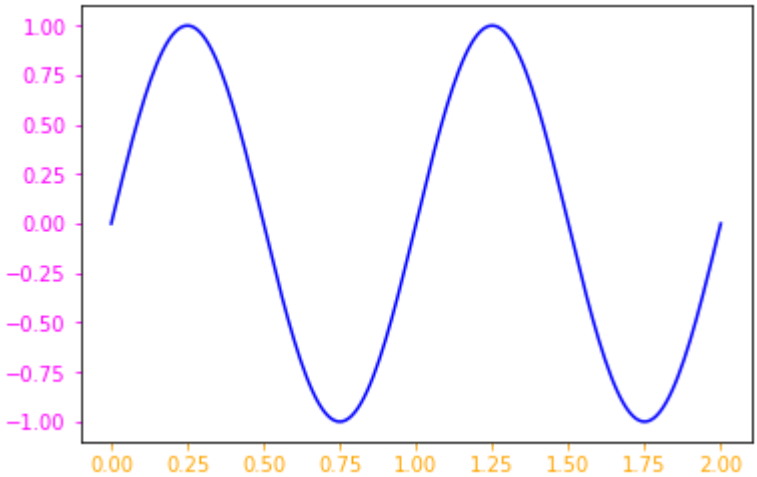
Now, we change the size of the labels for the ticks in each axis.

In [36]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
for label in ax1.xaxis.get_ticklabels() :
    label.set_fontsize(5)
for label in ax1.yaxis.get_ticklabels() :
    label.set_fontsize(15)

plt.show()
```



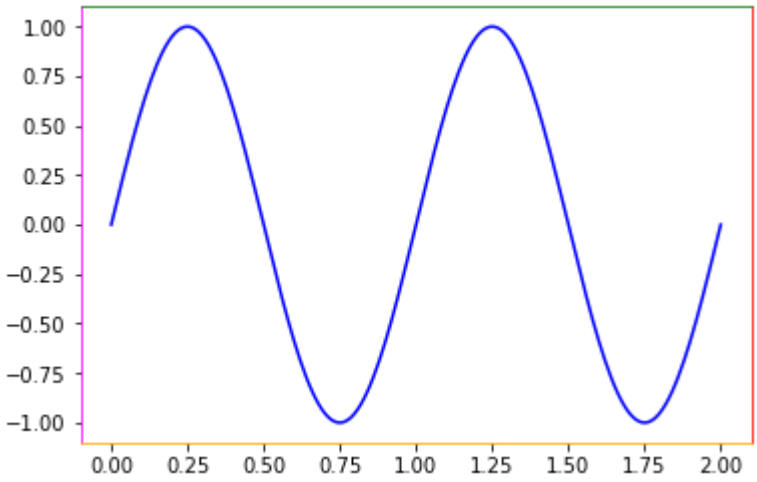
Now, we change the color of the labels for the ticks in each axis.

In [37]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.tick_params( axis = 'x' , colors = 'orange' )
ax1.tick_params( axis = 'y' , colors = 'magenta' )

plt.show()
```



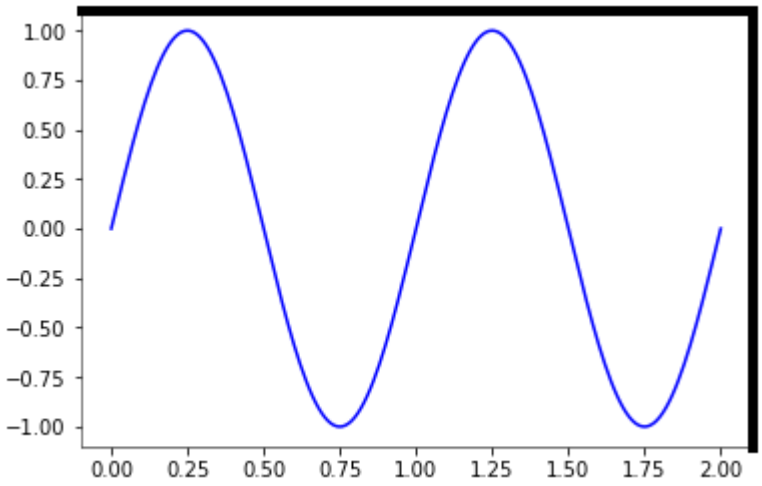
Now, we change the colors of the margin lines of our plot.

In [38]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.spines['left'].set_color('magenta')
ax1.spines['right'].set_color('red')
ax1.spines['bottom'].set_color('orange')
ax1.spines['top'].set_color('green')

plt.show()
```



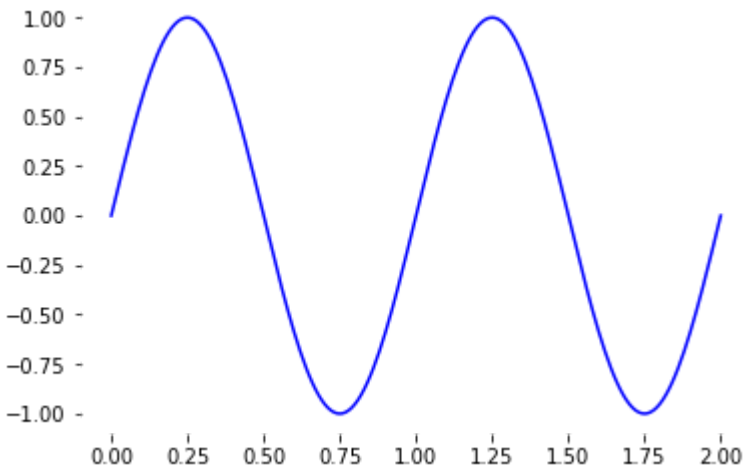
Now, we change the widths of the margin lines of our plot.

In [39]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.spines['left'].set_linewidth(0.5)
ax1.spines['right'].set_linewidth(5)
ax1.spines['bottom'].set_linewidth(0.5)
ax1.spines['top'].set_linewidth(5)

plt.show()
```



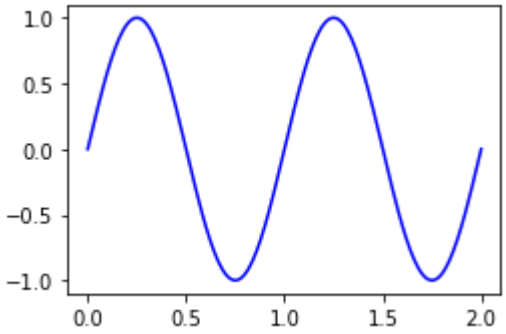
Now, we remove the margin lines of our plot.

In [40]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.spines['left'].set_visible(False)
ax1.spines['right'].set_visible(False)
ax1.spines['bottom'].set_visible(False)
ax1.spines['top'].set_visible(False)

plt.show()
```

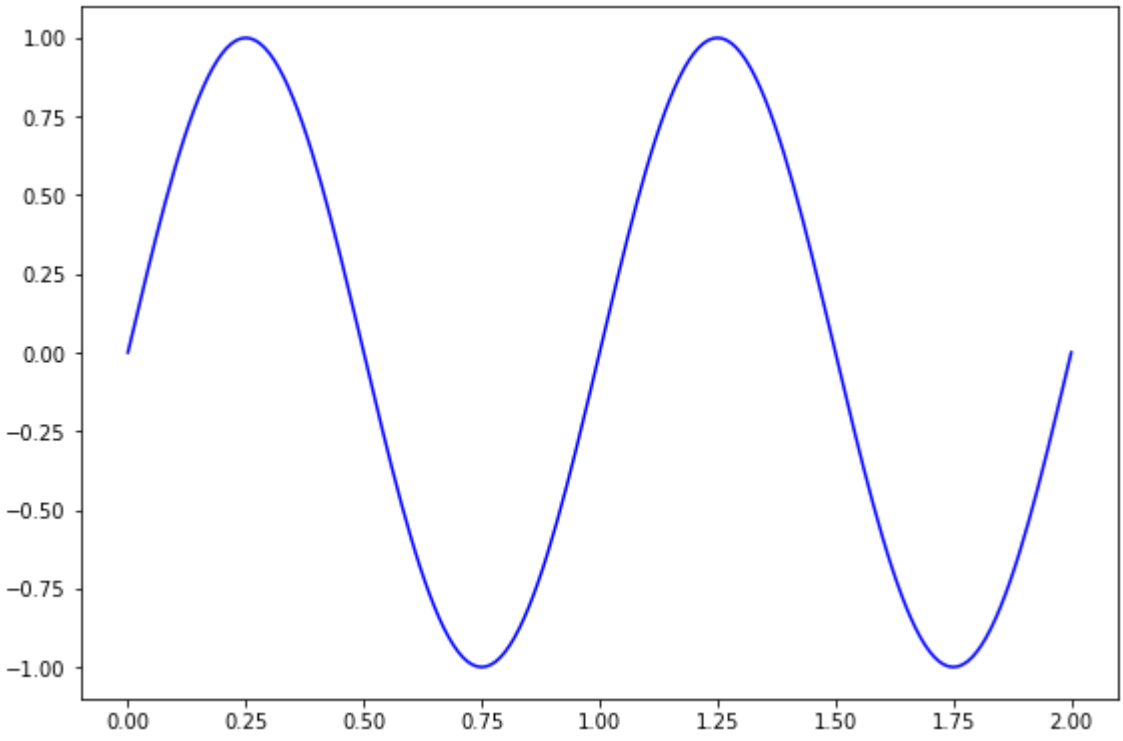


We can adjust the display window spaces for the subplots.

In [41]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )

plt.subplots_adjust( left = 0.21 , bottom = 0.21 , right = 0.71 ,\
                     top = 0.71 , wspace = 0 , hspace = 0 )
plt.show()
```

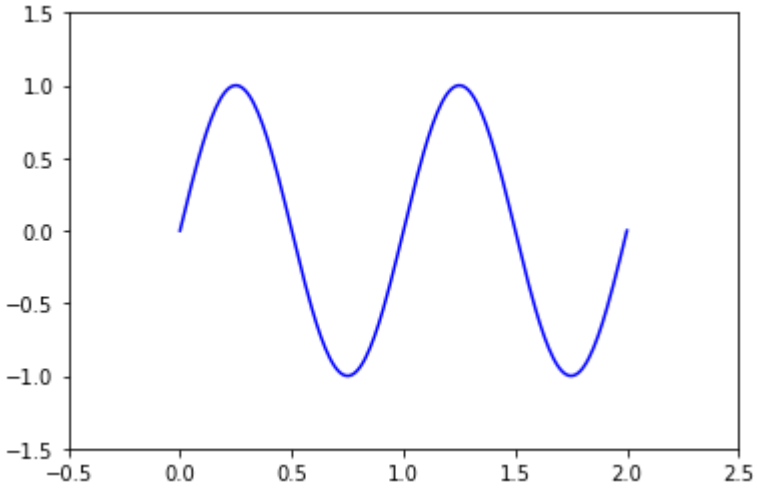


In [42]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )

plt.subplots_adjust( left = -0.1 , bottom = -0.1 , right = 1.1 ,\
                     top = 1.1 , wspace = 0 , hspace = 0 )
plt.show()
```



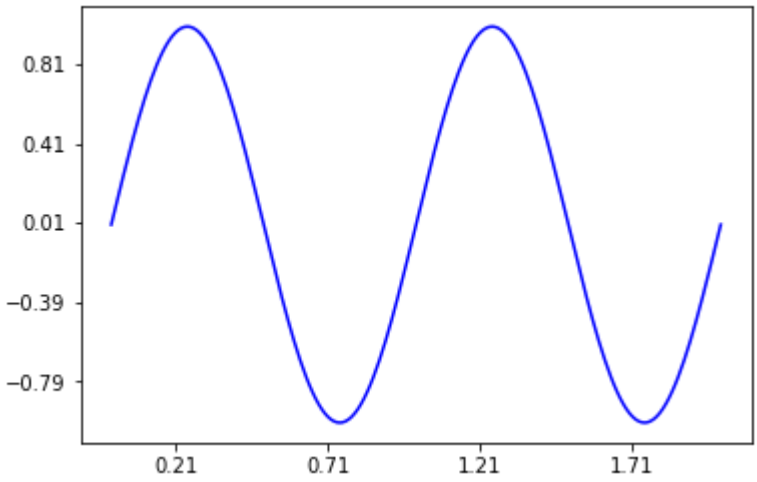
We can, also, specify the range of each axis in our plot.

In [43]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.set_xlim( -0.5 , 2.5 )
ax1.set_ylim( -1.5 , 1.5 )

plt.show()
```



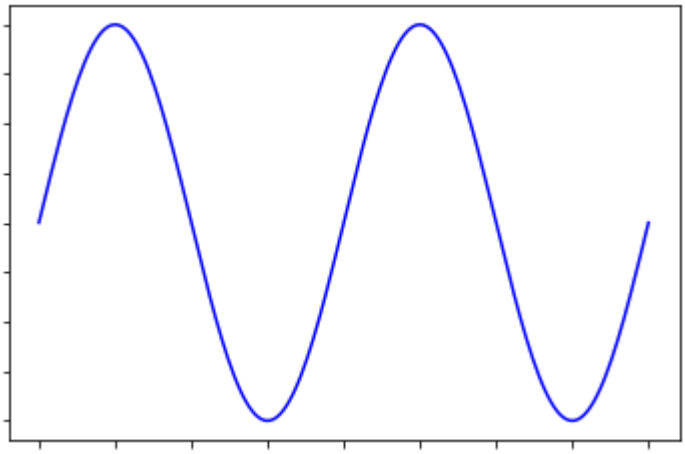
We can, also, specify the ticks in each axis of our plot.

In [44]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.set_xticks( np.arange( 0.21 , 1.72 , 0.5 ) )
ax1.set_yticks( np.arange( -0.79 , 0.82 , 0.4 ) )

plt.show()
```



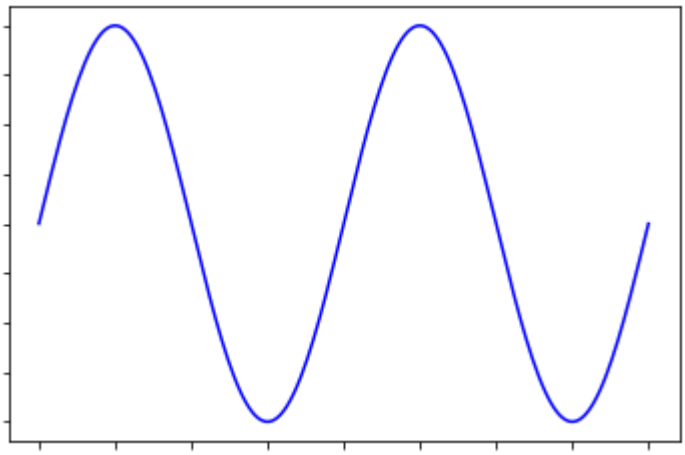
We can, also, hide the ticks labels in each axis of our plot.

In [45]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.axes.xaxis.set_ticklabels( [] )
ax1.axes.yaxis.set_ticklabels( [] )

plt.show()
```



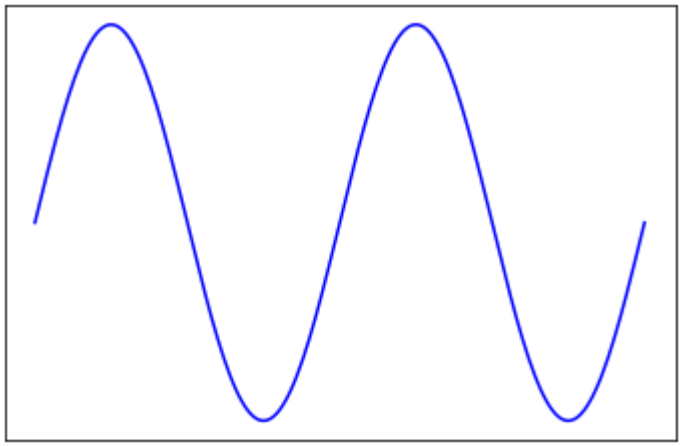
Here is a second way to get the same result.

In [46]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )

plt.setp( ax1.get_xticklabels() , visible = False )
plt.setp( ax1.get_yticklabels() , visible = False )
plt.show()
```



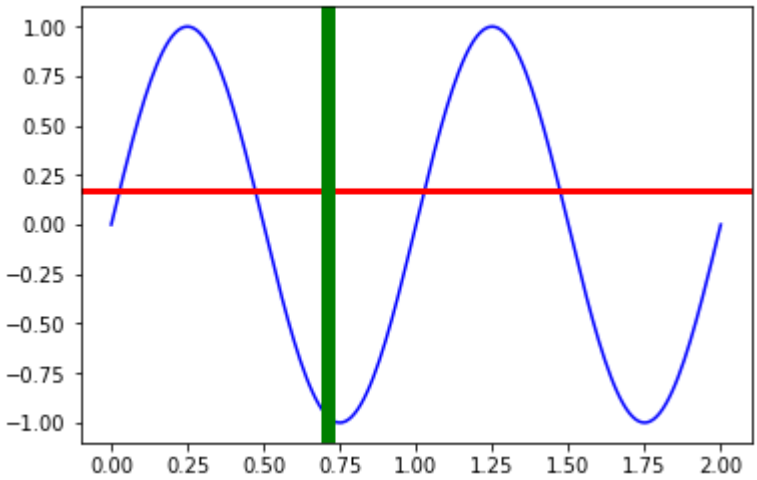
We can, also, remove the ticks from one or both axes in our plot.

In [47]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.set_xticks( [] )
ax1.set_yticks( [] )

plt.show()
```



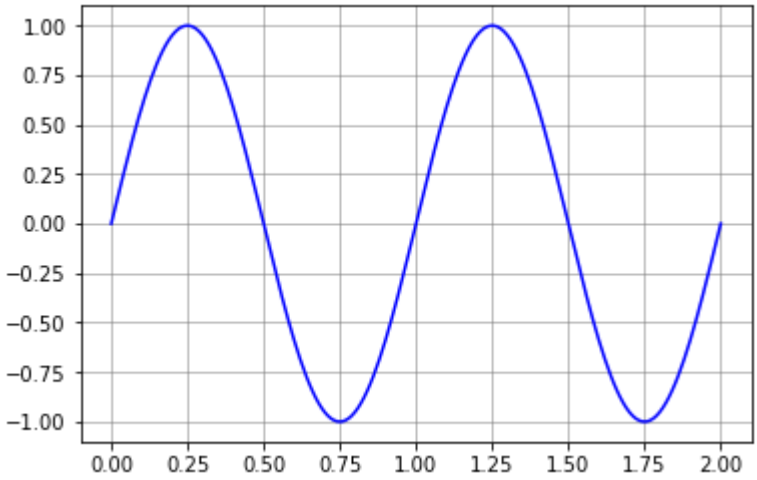
We can, also, add some horizontal and vertical lines to our plot.

In [48]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.axhline( 0.17 , color = 'red' , linewidth = 3 )
ax1.axvline( 0.71 , color = 'green' , linewidth = 7 )

plt.show()
```

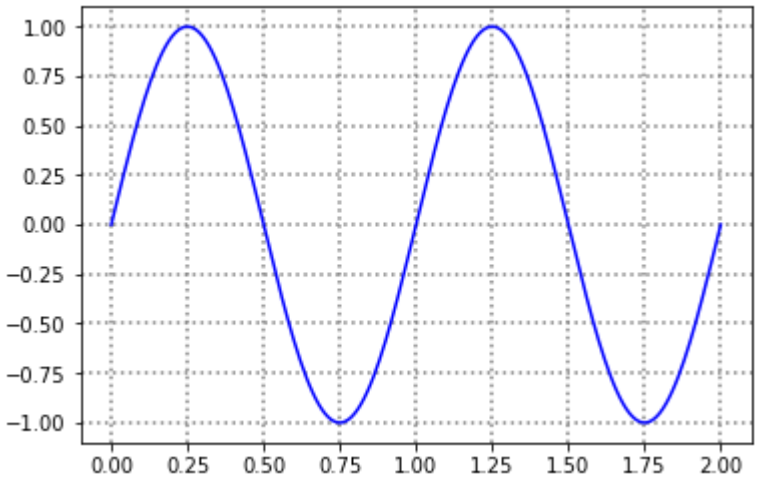


We can, also, add a grid to our plot.

In [49]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.grid( True , color = 'grey' , linestyle = 'solid' , linewidth = 0.5 )

plt.show()
```

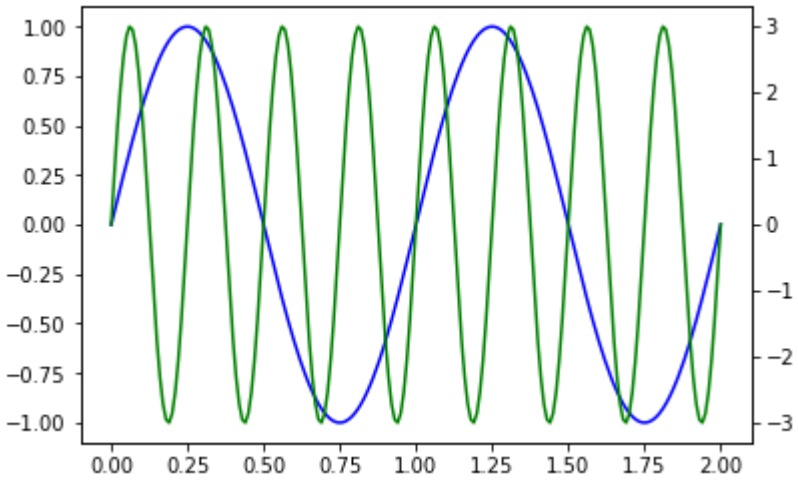


In [50]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.grid( True , color = 'grey' , linestyle = 'dotted' , linewidth = 1.5 )

plt.show()
```



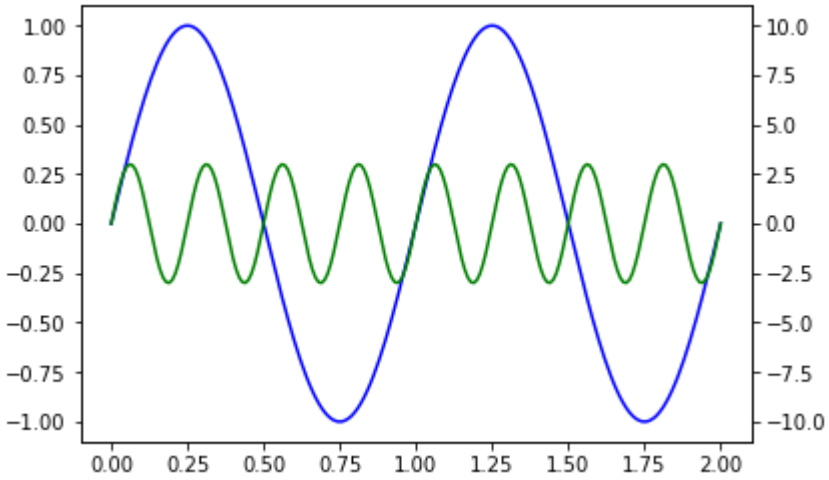
We can, also, create an additional plot with a second vertical axis in the same subplot.

In [51]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )
ax1v = ax1.twinx()

ax1.plot( x1 , y1 , color = 'blue' )
ax1v.plot( x1 , y3 , color = 'green' )

plt.show()
```



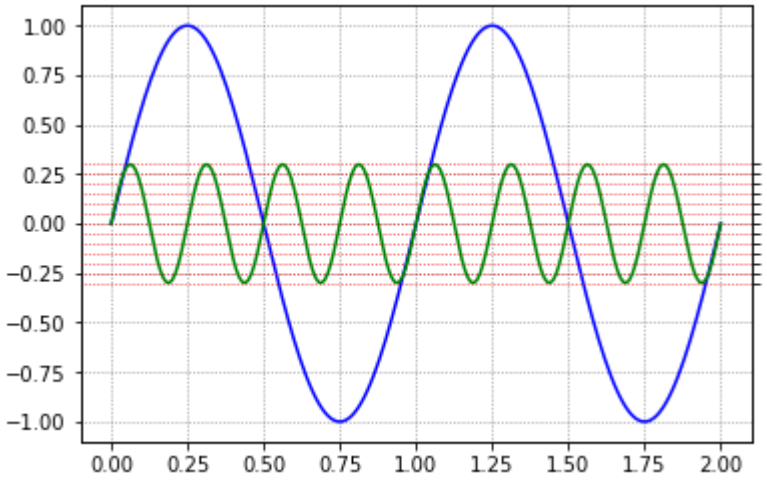
In this case, we have to specify, the same way as before, the axis properties for one or both vertical axes to get the result we want.

In [52]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )
ax1v = ax1.twinx()

ax1.plot( x1 , y1 , color = 'blue' )
ax1v.plot( x1 , y3 , color = 'green' )
ax1v.set_ylim( -11 , 11 )

plt.show()
```



Finally, we can create two grids in the same subplot.

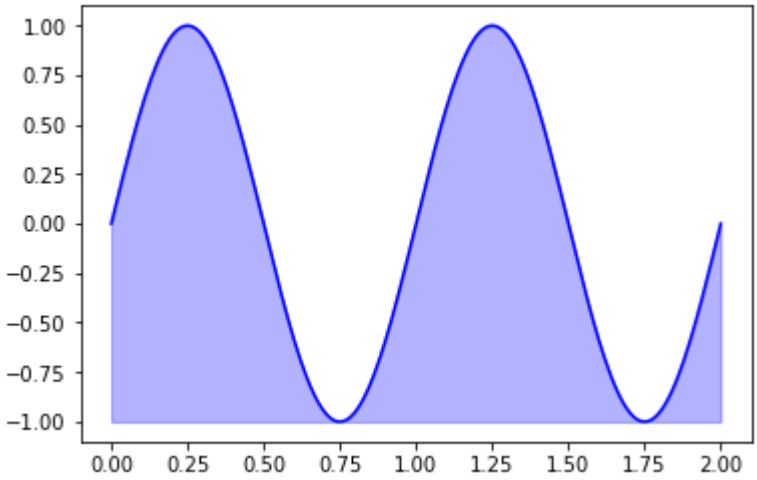
In [53]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )
ax1v = ax1.twinx()

ax1.plot( x1 , y1 , color = 'blue' )
ax1.grid( True , color = 'grey' , linestyle = 'dotted' , linewidth = 0.7 )

ax1v.plot( x1 , y3 , color = 'green' )
ax1v.set_ylim( -11 , 11 )
ax1v.set_yticks( np.arange( -3 , 3.01 , 0.5 ) )
ax1v.axes.yaxis.set_ticklabels( [] )
ax1v.grid( True , color = 'red' , linestyle = 'dotted' , linewidth = 0.7 )

plt.show()
```



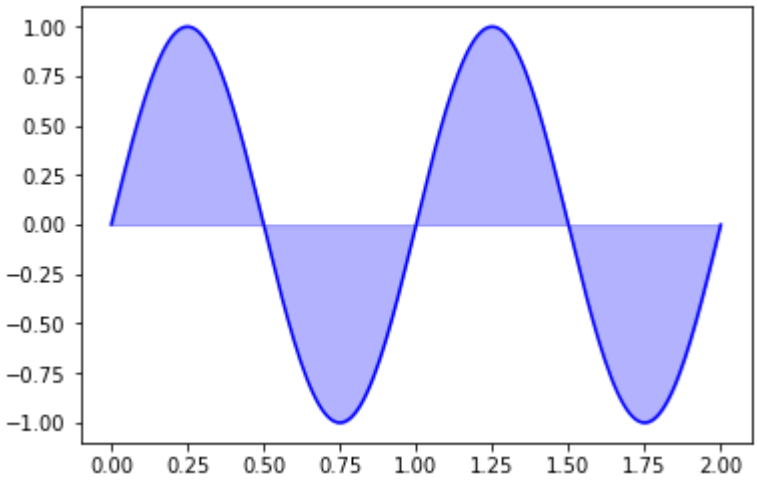
Fill Between Curves

We can add a fill between some constant and our curve to our plot.

In [54]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.fill_between( x1 , y1 , -1 , color = 'blue' , alpha = 0.3 )

plt.show()
```

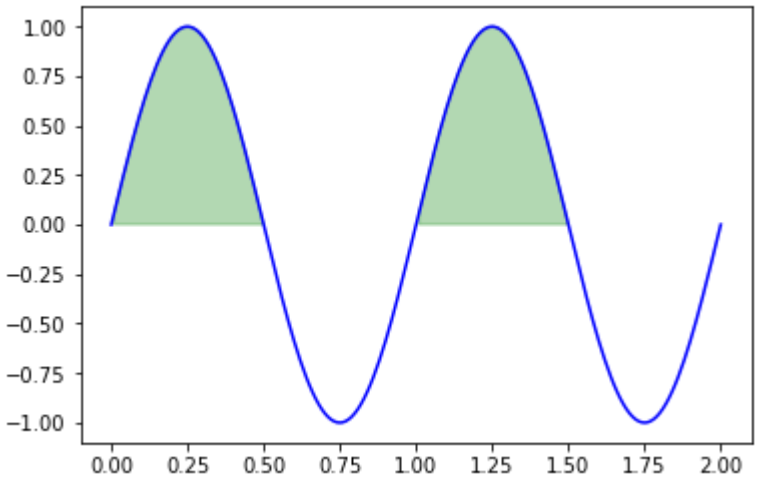


In [55]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.fill_between( x1 , y1 , 0 , color = 'blue' , alpha = 0.3 )

plt.show()
```

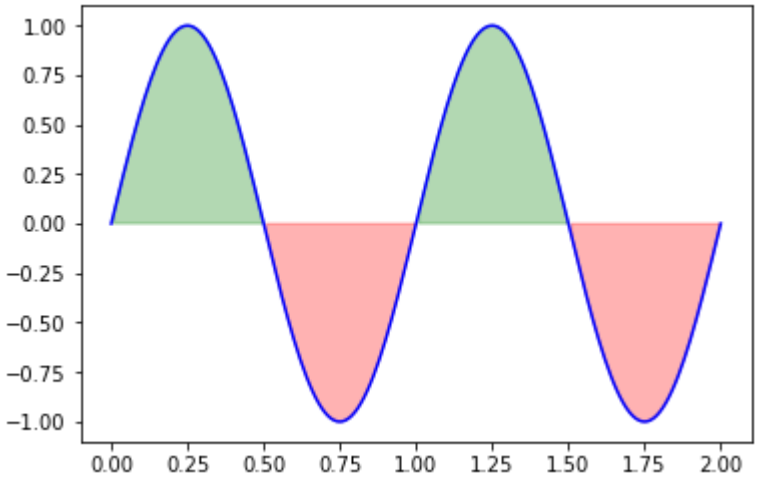


We can, also, specify where the fill will be made with a logical array.

In [56]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.fill_between( x1 , y1 , 0 , where = 0 < y1 , color = 'green' , alpha = 0.3
 )

plt.show()
```

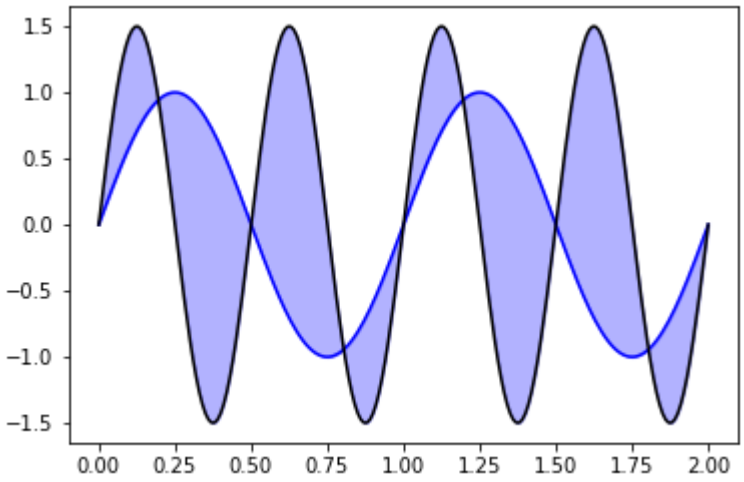


In [57]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.fill_between( x1 , y1 , 0 , where = 0 < y1 , color = 'green' , alpha = 0.3
 )
ax1.fill_between( x1 , y1 , 0 , where = y1 < 0 , color = 'red' , alpha = 0.3 )

plt.show()
```

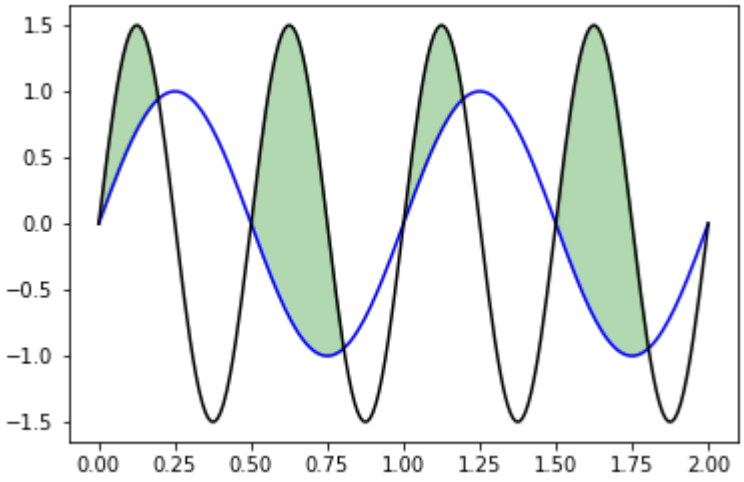


Finally, we can do the same with some additional non-constant curve.

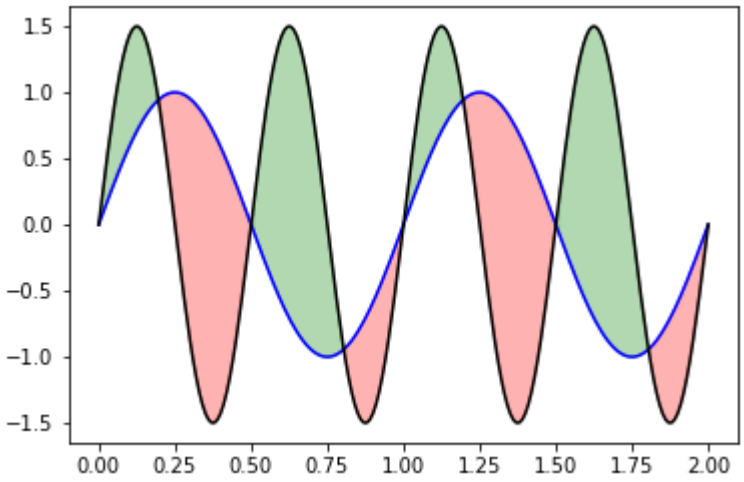
In [58]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.plot( x1 , y2 , color = 'black' )
ax1.fill_between( x1 , y1 , y2 , color = 'blue' , alpha = 0.3 )

plt.show()
```



In [59]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.plot( x1 , y2 , color = 'black' )
ax1.fill_between( x1 , y1 , y2 , where = y1<y2 , color = 'green' , alpha = 0.3
 )

plt.show()
```



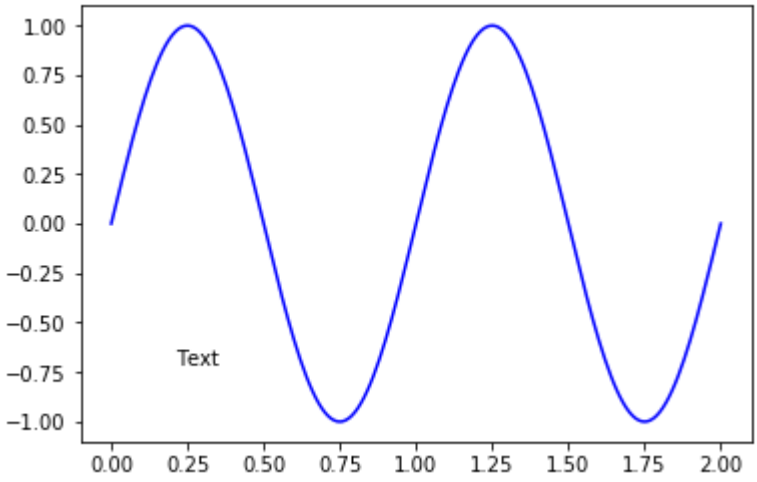
In [60]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.plot( x1 , y2 , color = 'black' )
ax1.fill_between( x1 , y1 , y2 , where = y1<y2 , color = 'green' , alpha = 0.3
 )
ax1.fill_between( x1 , y1 , y2 , where = y2<y1 , color = 'red' , alpha = 0.3 )

plt.show()
```



Annotations and text on the plot

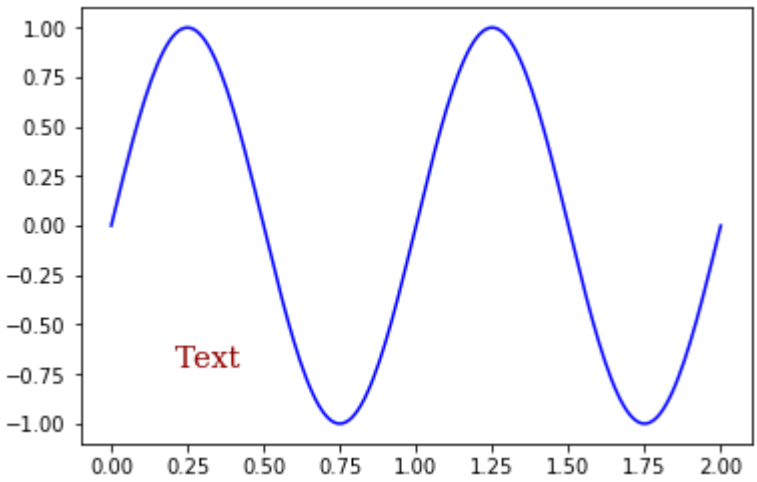
We can add text to our plot by specifying the coordinates for it's location.

In [61]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.text( 0.21 , -0.71 , 'Text' )

plt.show()
```



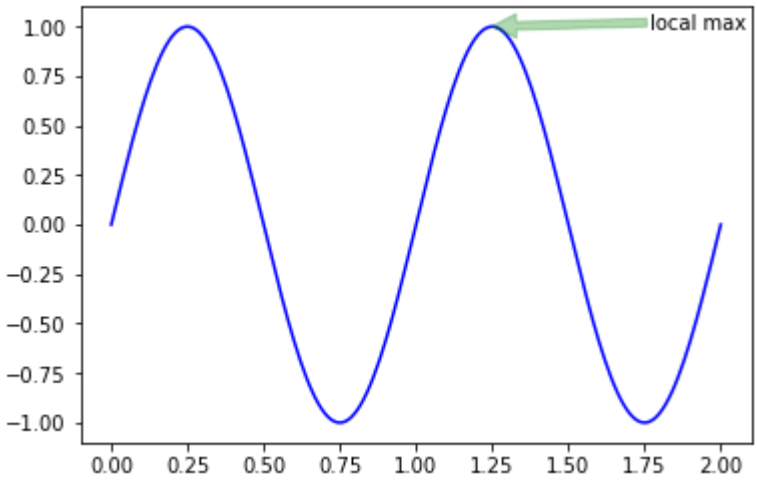
We can, also, specify the properties of the text with a dictionary.

In [62]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

text_properties = { 'family' : 'serif' ,
                    'color' : 'darkred' ,
                    'size' : 15 }

ax1.plot( x1 , y1 , color = 'blue' )
ax1.text( 0.21 , -0.71 , 'Text' , fontdict = text_properties )

plt.show()
```

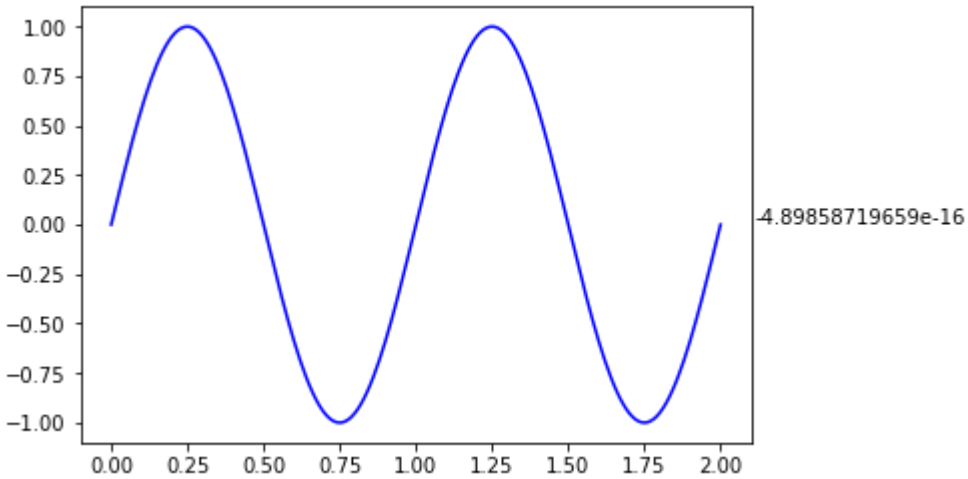


We can, also, add annotations that reference a point in the plot.

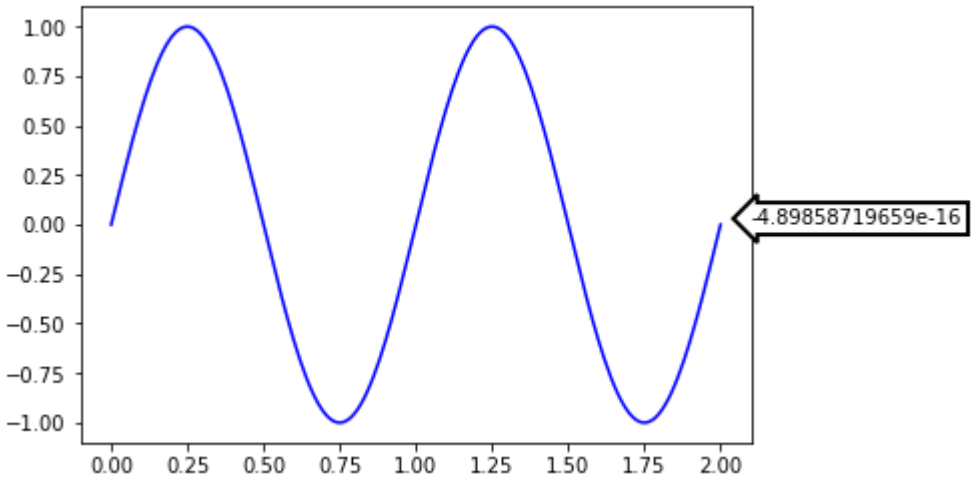
In [63]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.annotate( 'local max' , (1.25,1) , xytext = (0.85,0.95) ,\
              textcoords = 'axes fraction' ,\
              arrowprops = dict( color = 'green' , alpha = 0.3 ) )

plt.show()
```



In [64]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.annotate( str(y1[-1]) , (x1[-1],y1[-1]) , xytext = (x1[-1]+0.11,y1[-1]) )

plt.show()
```



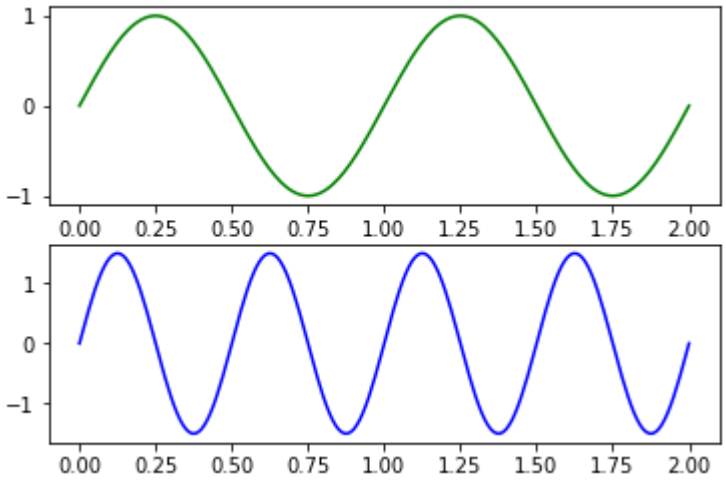
In [65]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

text_box_properties = dict( boxstyle = 'larrow' , facecolor = 'white' ,
                            edgecolor = 'black' , linewidth = 2 )

ax1.plot( x1 , y1 , color = 'blue' )
ax1.annotate( str(y1[-1]) , (x1[-1],y1[-1]) , xytext = (x1[-1]+0.1,y1[-1]) ,\
              bbox = text_box_properties )

plt.show()
```



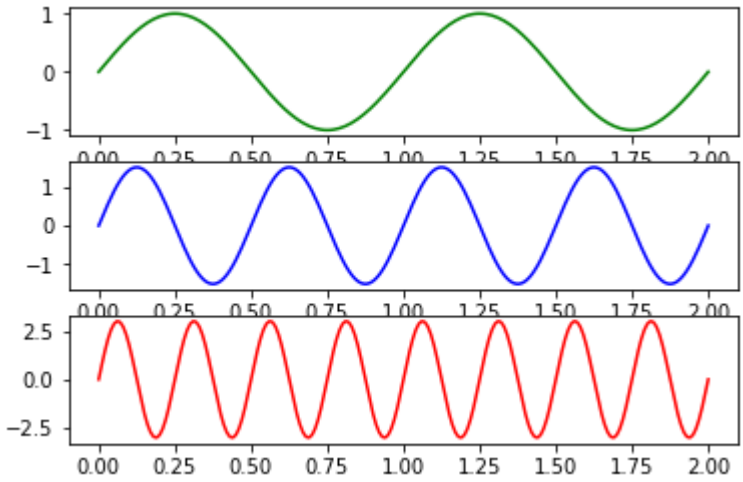
Multiple subplots

As we have said before, we can create multiple subplots in one figure.

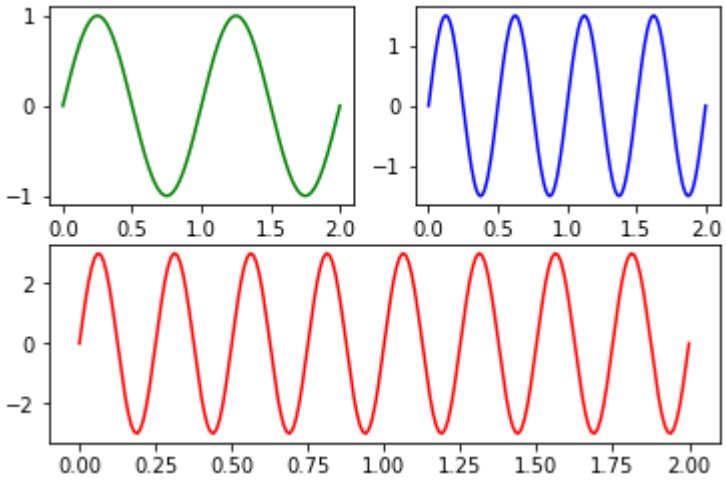
In [66]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (2,1) , (0,0) , rowspan = 1 , colspan = 1 )
ax2 = plt.subplot2grid( (2,1) , (1,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'green' )
ax2.plot( x1 , y2 , color = 'blue' )

plt.show()
```



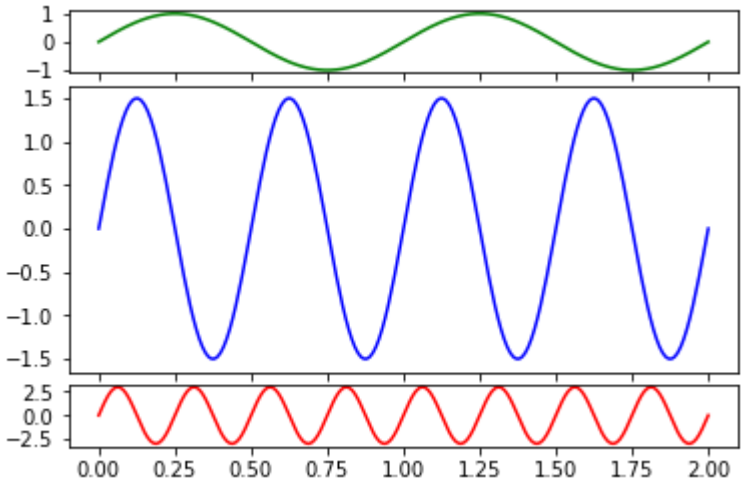
In [67]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (3,1) , (0,0) , rowspan = 1 , colspan = 1 )
ax2 = plt.subplot2grid( (3,1) , (1,0) , rowspan = 1 , colspan = 1 )
ax3 = plt.subplot2grid( (3,1) , (2,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'green' )
ax2.plot( x1 , y2 , color = 'blue' )
ax3.plot( x1 , y3 , color = 'red' )

plt.show()
```



In [68]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (2,2) , (0,0) , rowspan = 1 , colspan = 1 )
ax2 = plt.subplot2grid( (2,2) , (0,1) , rowspan = 1 , colspan = 1 )
ax3 = plt.subplot2grid( (2,2) , (1,0) , rowspan = 1 , colspan = 2 )

ax1.plot( x1 , y1 , color = 'green' )
ax2.plot( x1 , y2 , color = 'blue' )
ax3.plot( x1 , y3 , color = 'red' )

plt.show()
```

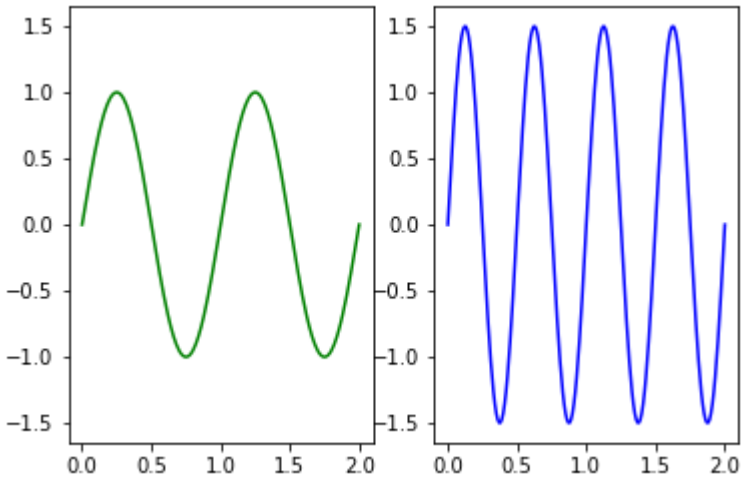


In [69]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (6,1) , (0,0) , rowspan = 1 , colspan = 1 )
ax2 = plt.subplot2grid( (6,1) , (1,0) , rowspan = 4 , colspan = 1 )
ax3 = plt.subplot2grid( (6,1) , (5,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 , color = 'green' )
ax2.plot( x1 , y2 , color = 'blue' )
ax3.plot( x1 , y3 , color = 'red' )

plt.show()
```



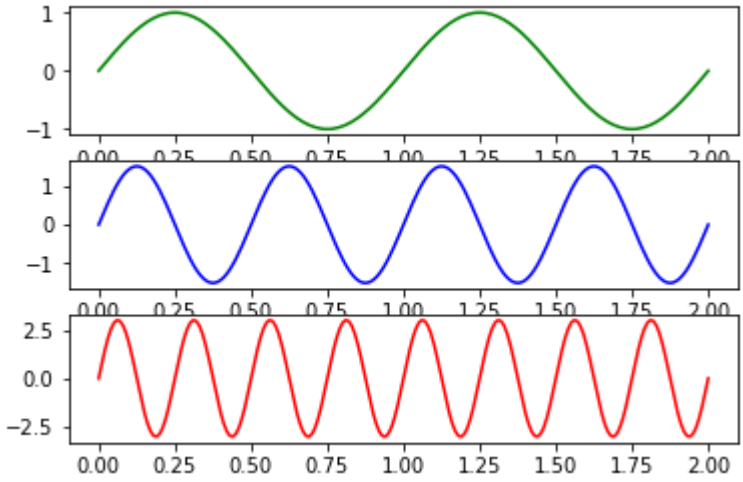
We can, also, create subplots that share the same vertical axis.

In [70]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (1,2) , (0,0) , rowspan = 1 , colspan = 1 )
ax2 = plt.subplot2grid( (1,2) , (0,1) , rowspan = 1 , colspan = 1 , sharey=ax1
 )

ax1.plot( x1 , y1 , color = 'green' )
ax2.plot( x1 , y2 , color = 'blue' )

plt.show()
```



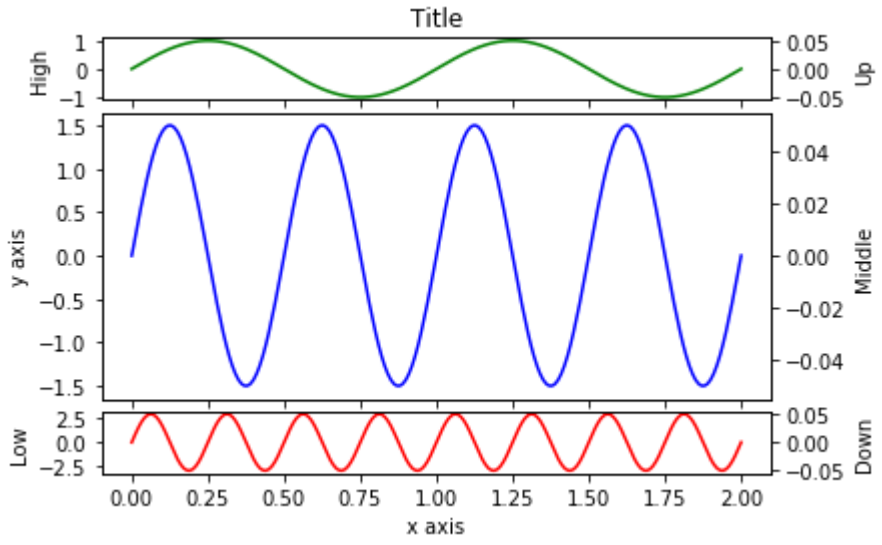
We can, also, create subplots that share the same horizontal axis.

In [71]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (3,1) , (0,0) , rowspan = 1 , colspan = 1 )
ax2 = plt.subplot2grid( (3,1) , (1,0) , rowspan = 1 , colspan = 1 , sharex=ax1
 )
ax3 = plt.subplot2grid( (3,1) , (2,0) , rowspan = 1 , colspan = 1 , sharex=ax1
 )

ax1.plot( x1 , y1 , color = 'green' )
ax2.plot( x1 , y2 , color = 'blue' )
ax3.plot( x1 , y3 , color = 'red' )

plt.show()
```



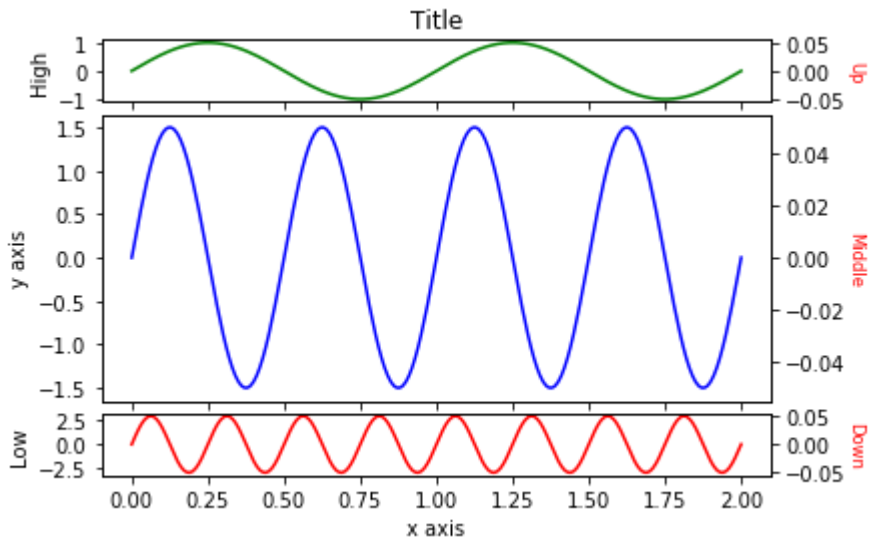
We can, also, add axis labels and titles, but we have to write them in order.

In [72]:
```
fig = plt.figure()
ax1 = plt.subplot2grid( (6,1) , (0,0) , rowspan = 1 , colspan = 1 )
plt.title('Title')
plt.ylabel('High')
ax1v = ax1.twinx()
plt.ylabel('Up')
ax2 = plt.subplot2grid( (6,1) , (1,0) , rowspan = 4 , colspan = 1 )
plt.ylabel('y axis')
ax2v = ax2.twinx()
plt.ylabel('Middle')
ax3 = plt.subplot2grid( (6,1) , (5,0) , rowspan = 1 , colspan = 1 )
plt.xlabel('x axis')
plt.ylabel('Low')
ax3v = ax3.twinx()
plt.ylabel('Down')

ax1.plot( x1 , y1 , color = 'green' )
ax1v.plot( [] , [] )
ax2.plot( x1 , y2 , color = 'blue' )
ax2v.plot( [] , [] )
ax3.plot( x1 , y3 , color = 'red' )
ax3v.plot( [] , [] )

plt.show()
```



We can, also, specify the properties for some of these labels.

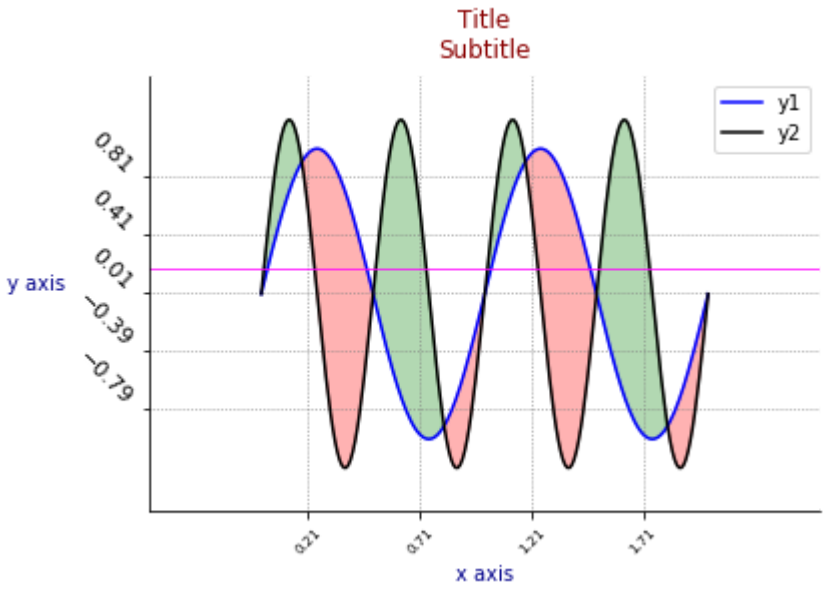
In [73]:
```
label_properties = {
   'fontsize'            : 'small' ,
   'color'               : 'red' ,
   'verticalalignment'   : 'center' ,
   'horizontalalignment' : 'center' ,
   'rotation'            : '-90' }

fig = plt.figure()
ax1 = plt.subplot2grid( (6,1) , (0,0) , rowspan = 1 , colspan = 1 )
plt.title('Title')
plt.ylabel('High')
ax1v = ax1.twinx()
plt.ylabel( 'Up' , fontdict = label_properties )
ax2 = plt.subplot2grid( (6,1) , (1,0) , rowspan = 4 , colspan = 1 )
plt.ylabel('y axis')
ax2v = ax2.twinx()
plt.ylabel( 'Middle' , fontdict = label_properties )
ax3 = plt.subplot2grid( (6,1) , (5,0) , rowspan = 1 , colspan = 1 )
plt.xlabel('x axis')
plt.ylabel('Low')
ax3v = ax3.twinx()
plt.ylabel( 'Down' , fontdict = label_properties )

ax1.plot( x1 , y1 , color = 'green' )
ax1v.plot( [] , [] )
ax2.plot( x1 , y2 , color = 'blue' )
ax2v.plot( [] , [] )
ax3.plot( x1 , y3 , color = 'red' )
ax3v.plot( [] , [] )

plt.show()
```



Summary

Now, we create a plot with some of the most common specifications.

In [74]:
```
# Libraries
import matplotlib.pyplot as plt
import numpy as np

# Data
x1 = np.arange( 0.0 , 2.01 , 0.01 )
y1 = np.sin( 2 * np.pi * x1 )
y2 = 1.2 * np.sin( 4 * np.pi * x1 )

# Figure and Subplots
fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

# Plots
ax1.plot( x1 , y1 , color = 'blue' , label = 'y1' )
ax1.plot( x1 , y2 , color = 'black' , label = 'y2' )
ax1.fill_between( x1 , y1 , y2 , where = y1<y2 , color = 'green' , alpha = 0.3
 )
ax1.fill_between( x1 , y1 , y2 , where = y2<y1 , color = 'red' , alpha = 0.3 )
ax1.axhline( 0.17 , color = 'magenta' , linewidth = 0.71 )
ax1.legend( loc = 1 , ncol = 1 )

# Specifications
ax1.set_xlim( -0.5 , 2.5 )
ax1.set_ylim( -1.5 , 1.5 )
ax1.set_xticks( np.arange( 0.21 , 1.72 , 0.5 ) )
ax1.set_yticks( np.arange( -0.79 , 0.82 , 0.4 ) )
ax1.grid( True , color = 'grey' , linestyle = 'dotted' , linewidth = 0.71 )
ax1.spines['right'].set_visible(False)
ax1.spines['top'].set_visible(False)
for label in ax1.xaxis.get_ticklabels() :
    label.set_rotation(45)
    label.set_fontsize(7)
for label in ax1.yaxis.get_ticklabels() :
    label.set_rotation(-45)
    label.set_fontsize(11)
    label.set_verticalalignment('bottom')

# Title and Labels
plt.xlabel( 'x axis' , fontdict = { 'color' : 'darkblue' ,
                                    'verticalalignment' : 'top' } )
plt.ylabel( 'y axis' , fontdict = { 'color' : 'darkblue' ,
                                    'horizontalalignment' : 'right' ,
                                    'rotation' : 'horizontal' } )
plt.title( 'Title\nSubtitle' , fontdict = { 'fontsize' : 'large' ,
                                            'color' : 'darkred' ,
                                            'verticalalignment' : 'bottom' } )
plt.show()
```



Styles

There are many different styles available to use in our plots.

```
In [75]: print( plt.style.available )
```

```
['seaborn-whitegrid', 'seaborn-bright', 'seaborn-notebook', 'seaborn-ticks',
 'seaborn-muted', '_classic_test', 'fivethirtyeight', 'dark_background', 'sea
born-paper', 'ggplot', 'seaborn', 'grayscale', 'seaborn-pastel', 'seaborn-dar
k-palette', 'seaborn-deep', 'seaborn-dark', 'seaborn-darkgrid', 'seaborn-colo
rblind', 'seaborn-poster', 'classic', 'seaborn-talk', 'bmh', 'seaborn-white']
```

Here is an example with the **ggplot** style.

In [76]:
```
plt.style.use('ggplot')

fig = plt.figure()
ax1 = plt.subplot2grid( (1,1) , (0,0) , rowspan = 1 , colspan = 1 )

ax1.plot( x1 , y1 )

plt.show()
```