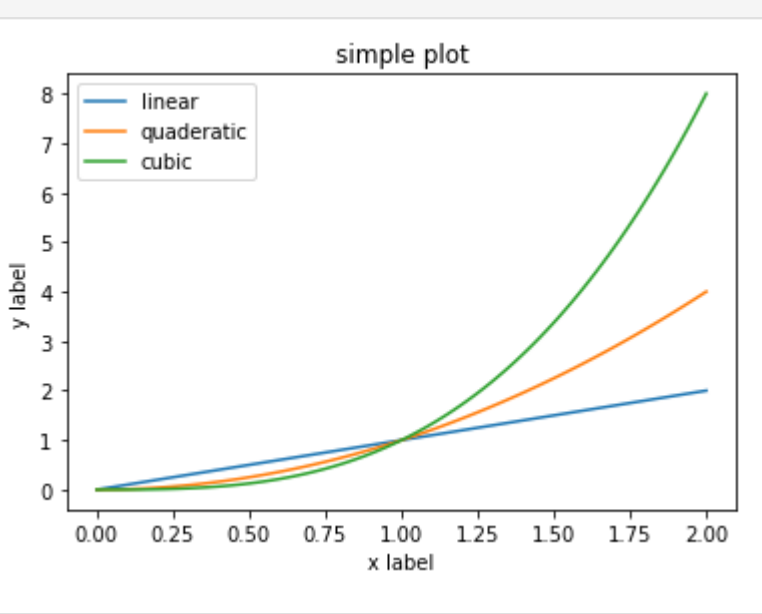
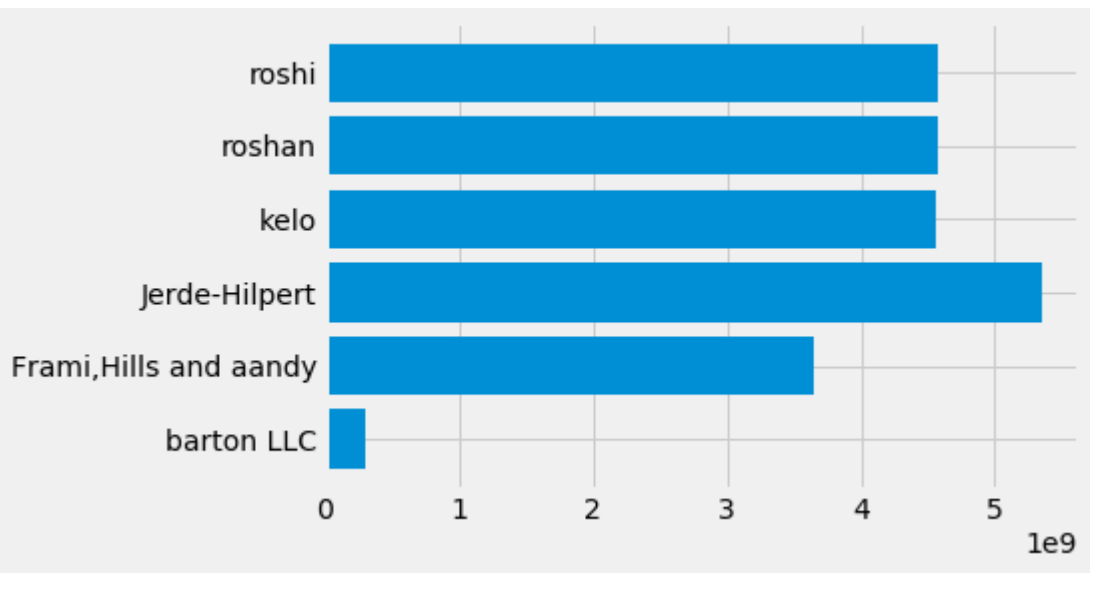
Matlpotlib

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x =np.linspace(0,2,100)
plt.plot(x,x ,label ='linear')
plt.plot(x,x**2, label = 'quaderatic')
plt.plot(x,x**3 , label = 'cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title('simple plot')
plt.legend()
plt.show()
```



In [19]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

data = {  'barton LLC' :  299298367.09,
           'Frami,Hills and aandy': 3645773678.56,
          'Jerde-Hilpert'  : 5345645656.56,
               'kelo'    : 4564787656.56,
                'roshan'  : 4567545678.56,
               'roshi'    : 4567456733.345}

group_data = list(data.values())
group_names =list(data.keys())
group_mean =  np.mean(group_data)
fig,ax = plt.subplots()
ax.barh(group_names,group_data)
```

Out[19]: <BarContainer object of 6 artists>

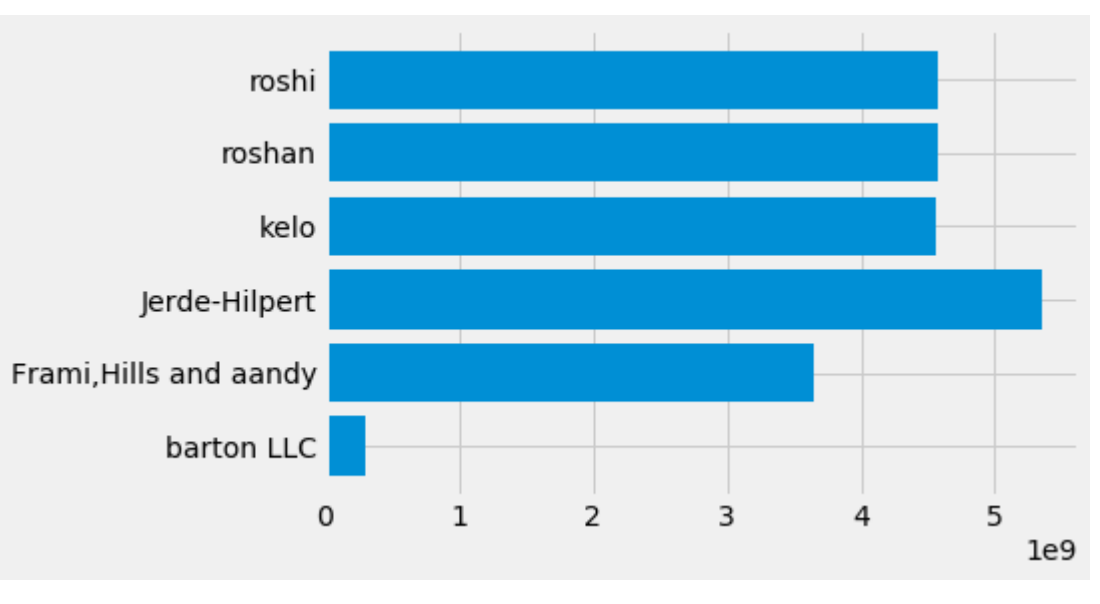


In [14]:
```
print(plt.style.available)
```

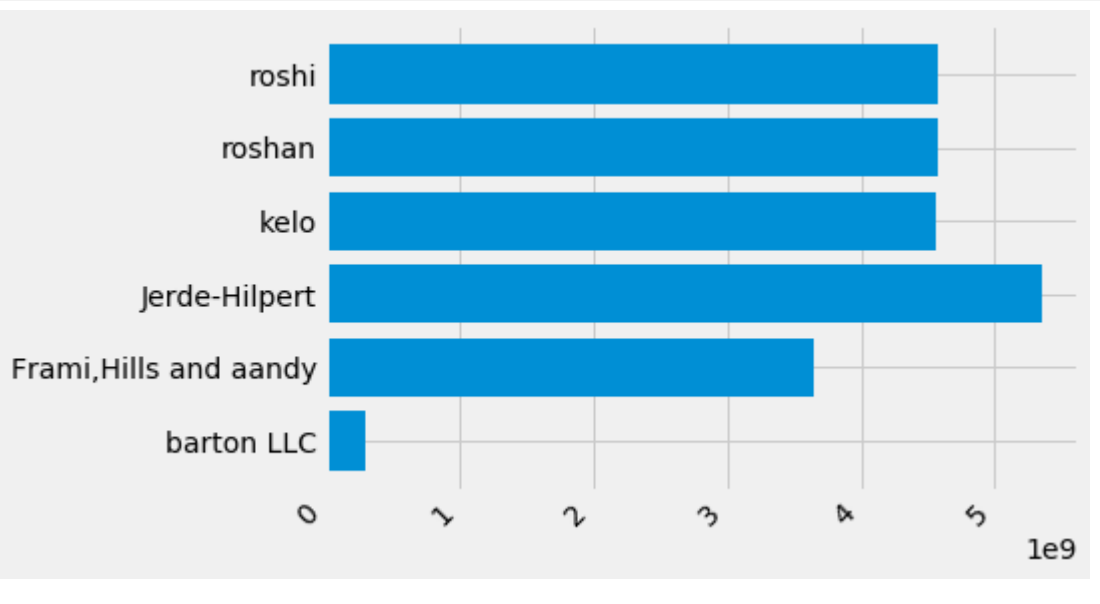
```
['bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-bright', 'seab
orn-colorblind', 'seaborn-dark-palette', 'seaborn-dark', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted',
'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seab
orn-white', 'seaborn-whitegrid', 'seaborn', 'Solarize_Light2', 'tableau-colorblind10', '_classic_test']
```

In [20]:
```
#use of style
plt.style.use('fivethirtyeight')
fig,ax = plt.subplots()
ax.barh(group_names,group_data)
```

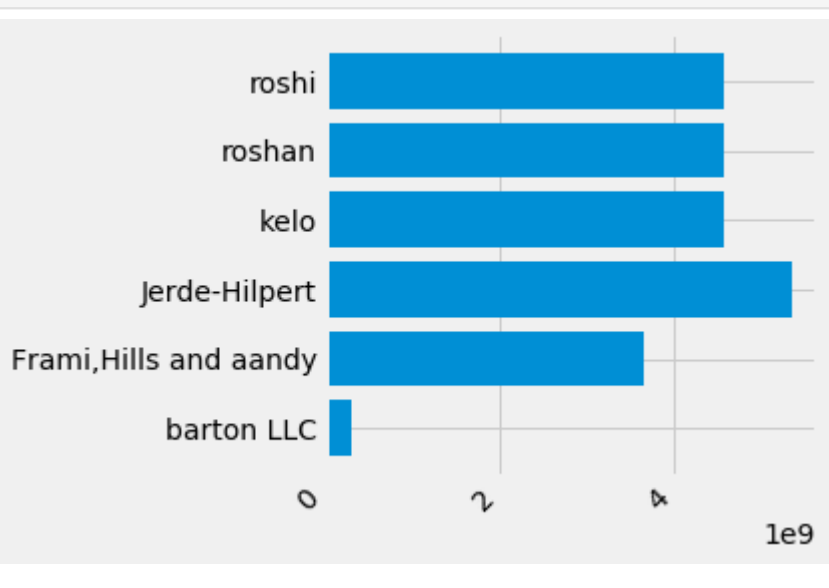
Out[20]: <BarContainer object of 6 artists>



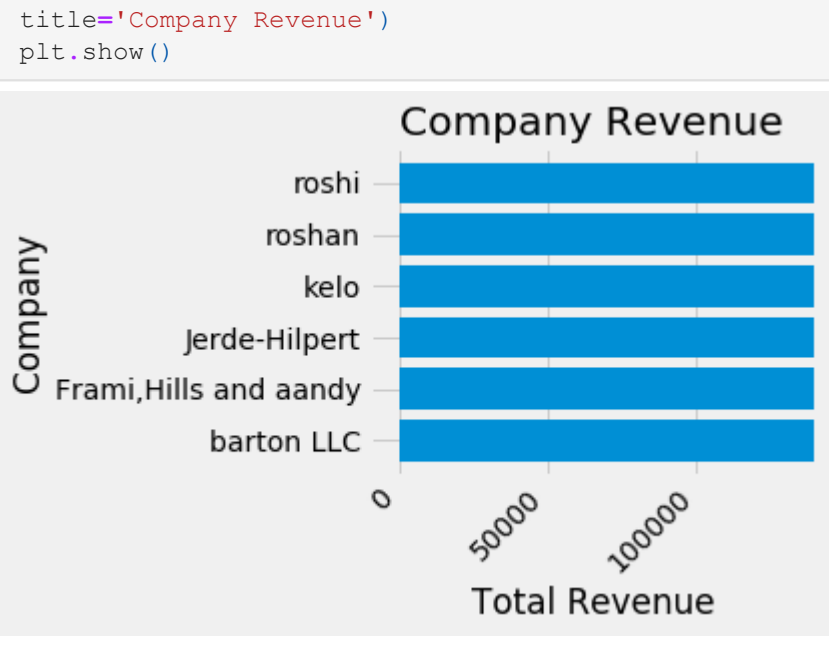
In [25]:
```
#Now we've got a plot with the general look that we want, so let's fine-tune it so that it's ready
#for print. First let's rotate the labels on the x-axis so that they show up more clearly. We can
#gain access to these labels with the axes.Axes.get_xticklabels() method
fig, ax = plt.subplots()
ax.barh(group_names, group_data)
labels = ax.get_xticklabels()
plt.setp(labels, rotation=45, horizontalalignment='right')
plt.show()
```



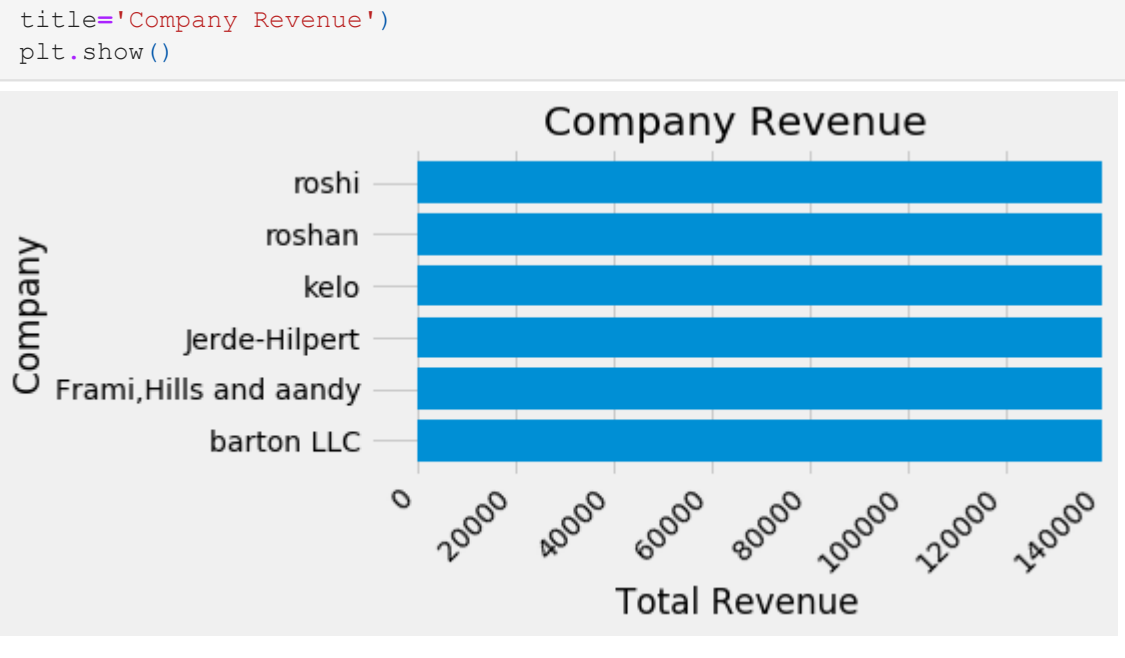
In [27]:
```
#It looks like this cut off some of the labels on the bottom. We can tell Matplotlib to automatically
#make room for elements in the figures that we create. To do this we'll set the autolayout
#value of our rcParams. For more information on controlling the style, layout, and other features
#of plots with rcParams, see Customizing Matplotlib with style sheets and rcParams.
plt.rcParams.update({'figure.autolayout': True})
fig, ax = plt.subplots()
ax.barh(group_names, group_data)
labels = ax.get_xticklabels()
plt.setp(labels, rotation=45, horizontalalignment='right')
plt.show()
```



In [29]:
```
#Next, we'll add labels to the plot. To do this with the OO interface, we can use the axes.Axes.
#set() method to set properties of this Axes object.
fig, ax = plt.subplots()
ax.barh(group_names, group_data)
labels = ax.get_xticklabels()
plt.setp(labels, rotation=45, horizontalalignment='right')
ax.set(xlim=[-10000, 140000], xlabel='Total Revenue', ylabel='Company',
title='Company Revenue')
plt.show()
```

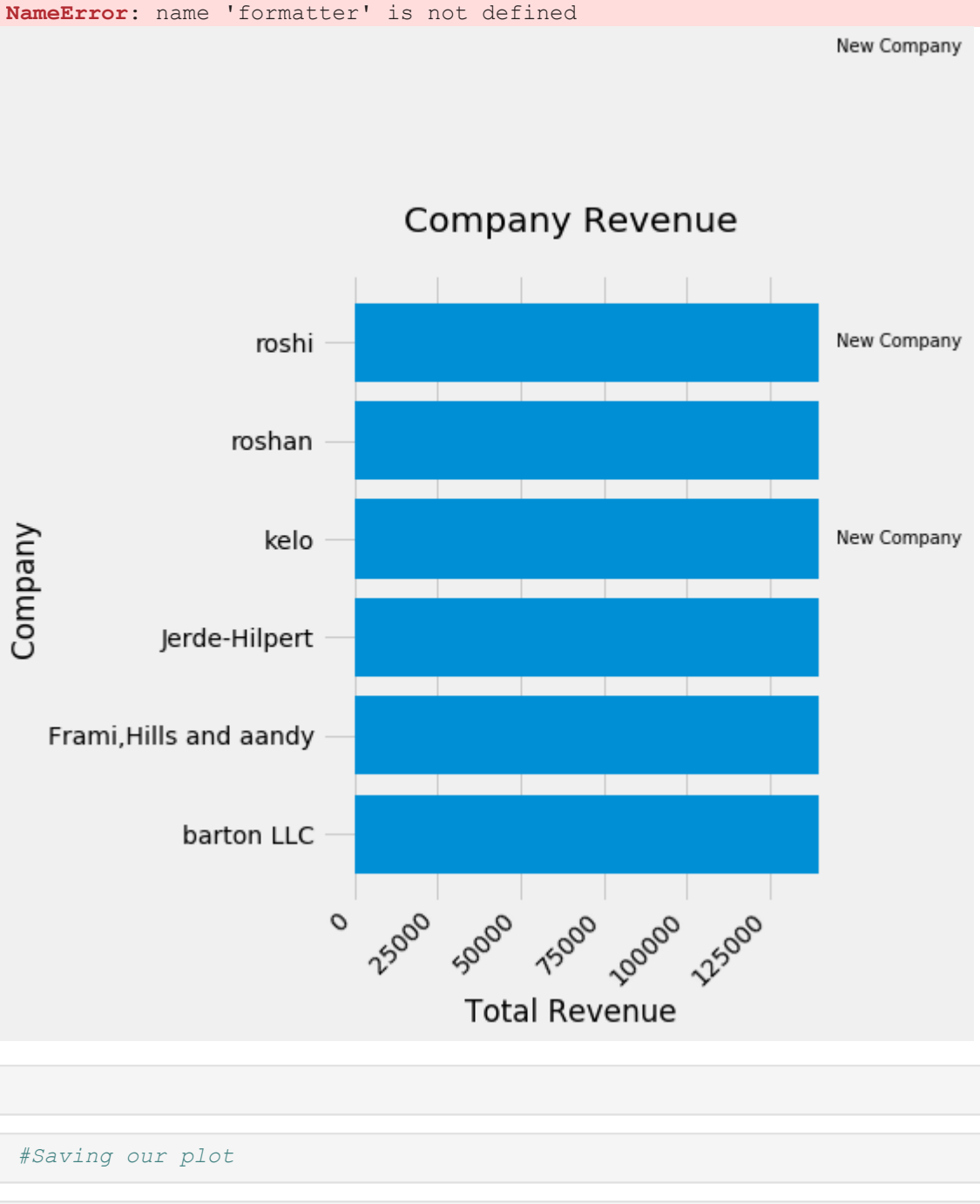


In [31]:
```
#We can also adjust the size of this plot using the pyplot.subplots() function. We can do this
#with the figsize kwarg.
fig, ax = plt.subplots(figsize=(8, 4))
ax.barh(group_names, group_data)
labels = ax.get_xticklabels()
plt.setp(labels, rotation=45, horizontalalignment='right')
ax.set(xlim=[-10000, 140000], xlabel='Total Revenue', ylabel='Company',
title='Company Revenue')
plt.show()
```



In [35]:
```
#We can then apply this formatter to the labels on our plot. To do this, we'll use the xaxis
#attribute of our axis. This lets you perform actions on a specific axis on our plot
fig, ax = plt.subplots(figsize=(8, 8))
ax.barh(group_names, group_data)
labels = ax.get_xticklabels()
plt.setp(labels, rotation=45, horizontalalignment='right')
# Add a vertical line, here we set the style in the function call
ax.axvline(group_mean, ls='--', color='r')
# Annotate new companies
for group in [3, 5, 8]:
    ax.text(145000, group, "New Company", fontsize=10,verticalalignment="center")
# Now we'll move our title up since it's getting a little cramped
ax.title.set(y=1.05)
ax.set(xlim=[-10000, 140000], xlabel='Total Revenue', ylabel='Company',
title='Company Revenue')
ax.xaxis.set_major_formatter(formatter)
ax.set_xticks([0, 25e3, 50e3, 75e3, 100e3, 125e3])
fig.subplots_adjust(right=.1)
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-35-857aa53393ec> in <module>
     14 ax.set(xlim=[-10000, 140000], xlabel='Total Revenue', ylabel='Company',
     15 title='Company Revenue')
---> 16 ax.xaxis.set_major_formatter(formatter)
     17 ax.set_xticks([0, 25e3, 50e3, 75e3, 100e3, 125e3])
     18 fig.subplots_adjust(right=.1)

NameError: name 'formatter' is not defined
```



In [ ]:

In [ ]:
```
#Saving our plot
```

In [ ]:
```
#We can then use the figure.Figure.savefig() in order to save the figure to disk. Note that
#there are several useful flags we'll show below:
```

In [ ]:
```
#• transparent=True makes the background of the saved figure transparent if the format
#supports it.
#• dpi=80 controls the resolution (dots per square inch) of the output.
#• bbox_inches="tight" fits the bounds of the figure to our plot.
# Uncomment this line to save the figure.
# fig.savefig('sales.png', transparent=False, dpi=80, bbox_inches="tight")
```

In [ ]:

Customizing Matplotlib with style sheets and rcParams

In [ ]:
```
#The style package adds support for easy-to-switch plotting "styles" with the same parameters
#as a matplotlib rc file (which is read at startup to configure matplotlib).
```

In [37]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
plt.style.use('ggplot')
data = np.random.randn(50)
```

In [ ]:
```
#To list all available styles, use:
```

In [38]:
```
print(plt.style.available)
```

```
['bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-bright', 'seab
orn-colorblind', 'seaborn-dark-palette', 'seaborn-dark', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted',
'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seab
orn-white', 'seaborn-whitegrid', 'seaborn', 'Solarize_Light2', 'tableau-colorblind10', '_classic_test']
```

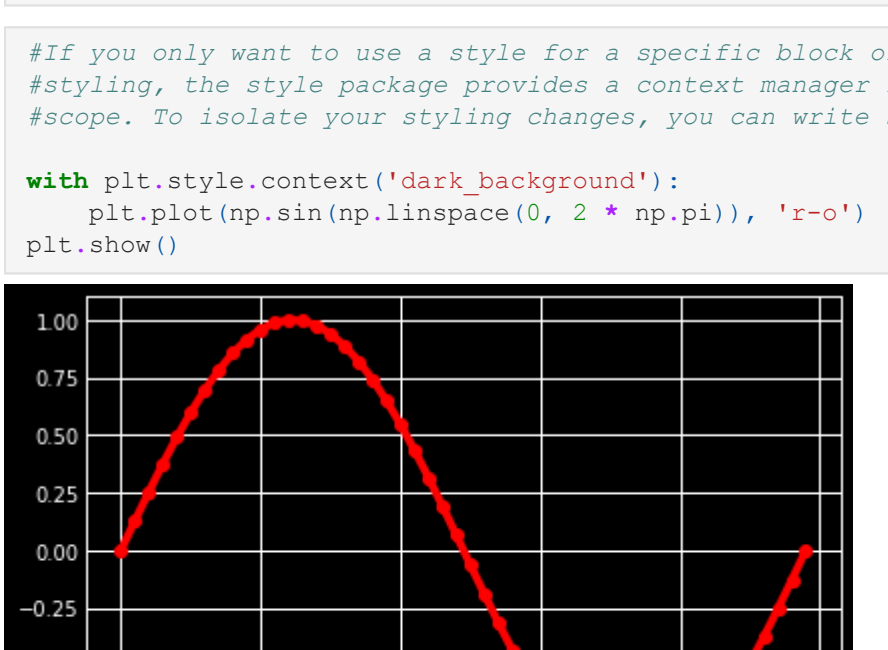
In [ ]:
```
#Defining your own style
#You can create custom styles and use them by calling style.use with the path or URL to
#the style sheet.
#For example, you might want to create mpl_configdir/stylelib/presentation.mplstyle with the
#following:
#axes.titlesize : 24
#axes.labelsize : 20
#lines.linewidth : 3
#lines.markersize : 10
#xtick.labelsize : 16
#ytick.labelsize : 16
#Then, when you want to adapt a plot designed for a paper to one that looks good in a presentation,
#you can just add:
#import matplotlib.pyplot as plt
#plt.style.use('presentation')
```

In [ ]:
```
#Composing styles
```

In [ ]:
```
#Style sheets are designed to be composed together. So you can have a style sheet that customizes
#colors and a separate style sheet that alters element sizes for presentations. These
#styles can easily be combined by passing a list of styles:
#import matplotlib.pyplot as plt
#plt.style.use(['dark_background', 'presentation'])
```

In [ ]:
```
#Temporary styling
```

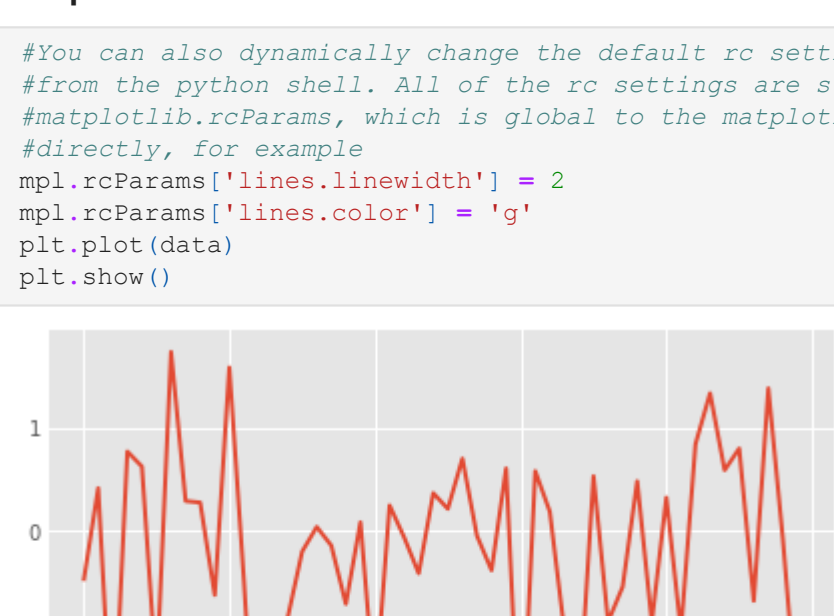
In [44]:
```
#If you only want to use a style for a specific block of code but don't want to change the global
#styling, the style package provides a context manager for limiting your changes to a specific
#scope. To isolate your styling changes, you can write something like the following:

with plt.style.context('dark_background'):
    plt.plot(np.sin(np.linspace(0, 2 * np.pi)), 'r-o')
plt.show()
```



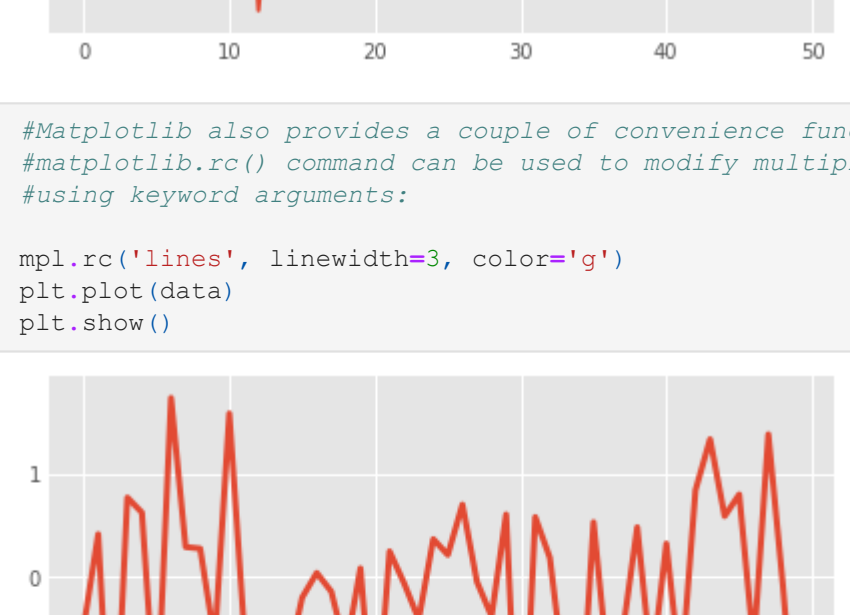
In [ ]:

matplotlib rcParams

In [48]:
```
#You can also dynamically change the default rc settings in a python script or interactively
#from the python shell. All of the rc settings are stored in a dictionary-like variable called
#matplotlib.rcParams, which is global to the matplotlib package. rcParams can be modified
#directly, for example
mpl.rcParams['lines.linewidth'] = 2
mpl.rcParams['lines.color'] = 'g'
plt.plot(data)
plt.show()
```



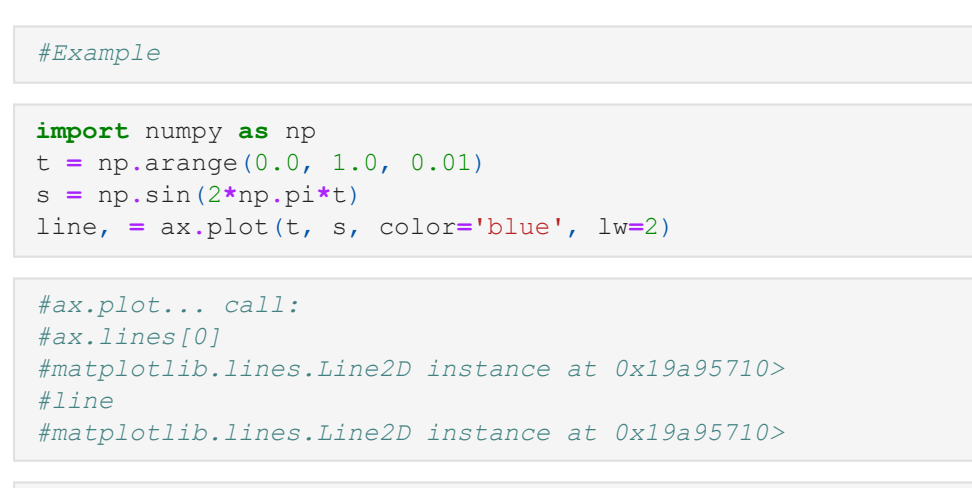
In [53]:
```
#Matplotlib also provides a couple of convenience functions for modifying rc settings. The
#matplotlib.rc() command can be used to modify multiple settings in a single group at once,
#using keyword arguments:

mpl.rc('lines', linewidth=3, color='g')
plt.plot(data)
plt.show()
```



Matplotlib parameter

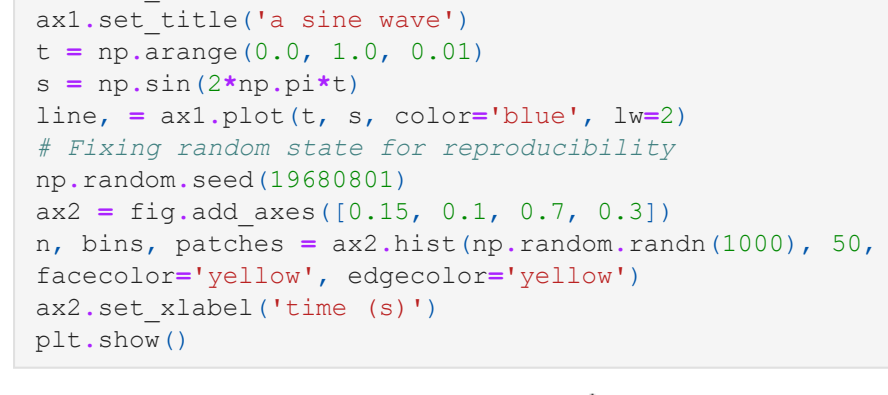
In [ ]:
```
#### LINES

#lines.linewidth : 1.5 ## line width in points
#lines.linestyle : - ## solid line
#lines.color : C0 ## has no affect on plot(); see axes.prop_cycle
#lines.marker : None ## the default marker
#lines.markerfacecolor : auto ## the default markerfacecolor
#lines.markeredgecolor : auto ## the default markeredgecolor
#lines.markeredgewidth : 1.0 ## the line width around the marker symbol
#lines.markersize : 6 ## markersize, in points
#lines.dash_joinstyle : round ## miter|round|bevel
#lines.dash_capstyle : butt ## butt|round|projecting
#lines.solid_joinstyle : round ## miter|round|bevel
#lines.solid_capstyle : projecting ## butt|round|projecting
#lines.antialiased : True ## render lines in antialiased (no jaggies)

#### Boxplot
#boxplot.notch : False
#boxplot.vertical : True
#boxplot.whiskers : 1.5
#boxplot.bootstrap : None
#boxplot.patchartist : False
#boxplot.showmeans : False
#boxplot.showcaps : True
#boxplot.showbox : True
#boxplot.showfliers : True
#boxplot.meanline : False
#boxplot.flierprops.color : black
#boxplot.flierprops.marker : o
#boxplot.flierprops.markerfacecolor : none
#boxplot.flierprops.markeredgecolor : black
#boxplot.flierprops.markeredgewidth : 1.0
#boxplot.flierprops.markersize : 6
#boxplot.flierprops.linestyle : none
#boxplot.flierprops.linewidth : 1.0
#boxplot.boxprops.color : black
#boxplot.boxprops.linewidth : 1.0

#### TICKS
## see http://matplotlib.org/api/axis_api.html#matplotlib.axis.Tick
#xtick.top : False ## draw ticks on the top side
#xtick.bottom : True ## draw ticks on the bottom side
#xtick.labeltop : False ## draw label on the top
#xtick.labelbottom : True ## draw label on the bottom
#xtick.major.size : 3.5 ## major tick size in points
#xtick.minor.size : 2 ## minor tick size in points
#xtick.major.width : 0.8 ## major tick width in points
#xtick.minor.width : 0.6 ## minor tick width in points
#xtick.major.pad : 3.5 ## distance to major tick label in points
#xtick.minor.pad : 3.4 ## distance to the minor tick label in points
#xtick.color : black ## color of the tick labels
#xtick.labelsize : medium ## fontsize of the tick labels
#xtick.direction : out ## direction: in, out, or inout
#xtick.minor.visible : False ## visibility of minor ticks on x-axis
#xtick.major.top : True ## draw x axis top major ticks
#xtick.major.bottom : True ## draw x axis bottom major ticks
#xtick.minor.top : True ## draw x axis top minor ticks
#xtick.minor.bottom : True ## draw x axis bottom minor ticks
#xtick.alignment : center ## alignment of xticks
#ytick.left : True ## draw ticks on the left side
#ytick.right : False ## draw ticks on the right side
#ytick.labelleft : True ## draw tick labels on the left side
#ytick.labelright : False ## draw tick labels on the right side
#ytick.major.size : 3.5 ## major tick size in points
#ytick.minor.size : 2 ## minor tick size in points
#ytick.major.width : 0.8 ## major tick width in points
#ytick.minor.width : 0.6 ## minor tick width in points
#ytick.major.pad : 3.5 ## distance to major tick label in points
#ytick.minor.pad : 3.4 ## distance to the minor tick label in points
#ytick.color : black ## color of the tick labels
#ytick.labelsize : medium ## fontsize of the tick labels
#ytick.direction : out ## direction: in, out, or inout
#ytick.minor.visible : False ## visibility of minor ticks on y-axis
#ytick.major.left : True ## draw y axis left major ticks
#ytick.major.right : True ## draw y axis right major ticks
#ytick.minor.left : True ## draw y axis left minor ticks
#ytick.minor.right : True ## draw y axis right minor ticks
#ytick.alignment : center_baseline ## alignment of yticks

#### Legend
#legend.loc : best
#legend.frameon : True ## if True, draw the legend on a background patch
#legend.framealpha : 0.8 ## legend patch transparency
#legend.facecolor : inherit ## inherit from axes.facecolor; or color spec
#legend.edgecolor : 0.8 ## background patch boundary color
#legend.fancybox : True ## if True, use a rounded box for the
## legend background, else a rectangle
#legend.shadow : False ## if True, give background a shadow effect
#legend.numpoints : 1 ## the number of marker points in the legend line
#legend.scatterpoints : 1 ## number of scatter points
#legend.markerscale : 1.0 ## the relative size of legend markers vs. original
#legend.fontsize : medium
#legend.title_fontsize : None ## None sets to the same as the default axes.
## Dimensions as fraction of fontsize:
#legend.borderpad : 0.4 ## border whitespace
#legend.labelspacing : 0.5 ## the vertical space between the legend entries
#legend.handlelength : 2.0 ## the length of the legend lines
#legend.handleheight : 0.7 ## the height of the legend handle
#legend.handletextpad : 0.8 ## the space between the legend line and legend text
#legend.borderaxespad : 0.5 ## the border between the axes and legend edge
#legend.columnspacing : 2.0 ## column separation

#### FIGURE
## See http://matplotlib.org/api/figure_api.html#matplotlib.figure.Figure
#figure.titlesize : large ## size of the figure title (Figure.suptitle())
#figure.titleweight : normal ## weight of the figure title
#figure.figsize : 6.4, 4.8 ## figure size in inches
#figure.dpi : 100 ## figure dots per inch

#figure.facecolor : white ## figure facecolor
#figure.edgecolor : white ## figure edgecolor
#figure.frameon : True ## enable figure frame
#figure.max_open_warning : 20 ## The maximum number of figures to open through
## the pyplot interface before emitting a warning.
## If less than one this feature is disabled.
## The figure subplot parameters. All dimensions are a fraction of the
#figure.subplot.left : 0.125 ## the left side of the subplots of the figure
#figure.subplot.right : 0.9 ## the right side of the subplots of the figure
#figure.subplot.bottom : 0.11 ## the bottom of the subplots of the figure
#figure.subplot.top : 0.88 ## the top of the subplots of the figure
#figure.subplot.wspace : 0.2 ## the amount of width reserved for space between␣
#,→subplots,
## expressed as a fraction of the average axis width
#figure.subplot.hspace : 0.2 ## the amount of height reserved for space between␣
#,→subplots,
# expressed as a fraction of the average axis height

#### HISTOGRAM PLOTS
#hist.bins : 10

#### SCATTER PLOTS
#scatter.marker : o ## The default marker type for scatter plots.
#scatter.edgecolors : face ## The default edgecolors for scatter plots
```

In [63]:
```
#The standard use is to create a Figure instance, use the Figure to create one or more Axes or Subplot instance
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(2, 1, 1) # two rows, one column, first plot
```



Subplots

In [64]:
```
#The Axes is probably the most important class in the matplotlib API, and the one you will be working with most
#the objects go, and the Axes has many special helper methods (plot(), text(), hist(), imshow()) to create the
#These helper methods will take your data (e.g., numpy arrays and strings) and create primitiveArtist instances
#when requested. Most of you are probably familiar with the Subplot, which is just a specialcase of an Axes tha
#to create an Axes at an arbitrary location, simply use the add_axes() method which takes a list
#of [left, bottom, width, height] values in 0-1 relative figure coordinates:

fig2 = plt.figure()
ax2 = fig2.add_axes([0.15, 0.1, 0.7, 0.3])
```

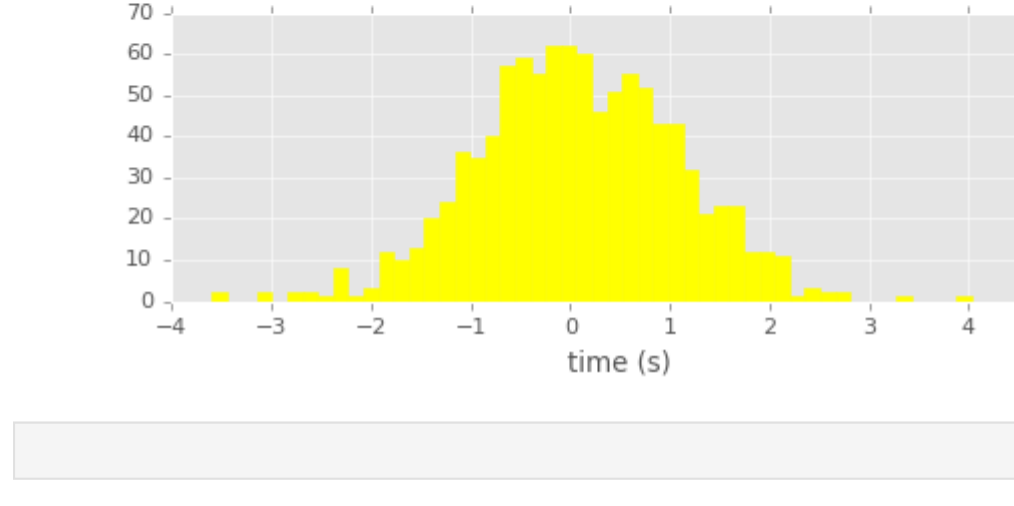


In [ ]:
```
#Example
```

In [65]:
```
import numpy as np
t = np.arange(0.0, 1.0, 0.01)
s = np.sin(2*np.pi*t)
line, = ax.plot(t, s, color='blue', lw=2)
```

In [ ]:
```
#ax.plot... call:
#ax.lines[0]
#matplotlib.lines.Line2D instance at 0x19a95710>
#line
#matplotlib.lines.Line2D instance at 0x19a95710>
```

In [66]:
```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
fig.subplots_adjust(top=0.8)
ax1 = fig.add_subplot(211)
ax1.set_ylabel('volts')
ax1.set_title('a sine wave')
t = np.arange(0.0, 1.0, 0.01)
s = np.sin(2*np.pi*t)
line, = ax1.plot(t, s, color='blue', lw=2)
# Fixing random state for reproducibility
np.random.seed(19680801)
ax2 = fig.add_axes([0.15, 0.1, 0.7, 0.3])
n, bins, patches = ax2.hist(np.random.randn(1000), 50,
facecolor='yellow', edgecolor='yellow')
ax2.set_xlabel('time (s)')
plt.show()
```



In [ ]: