Matplotlib

- Update to latest version
- python3 -m pip install --upgrade matplotlib

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

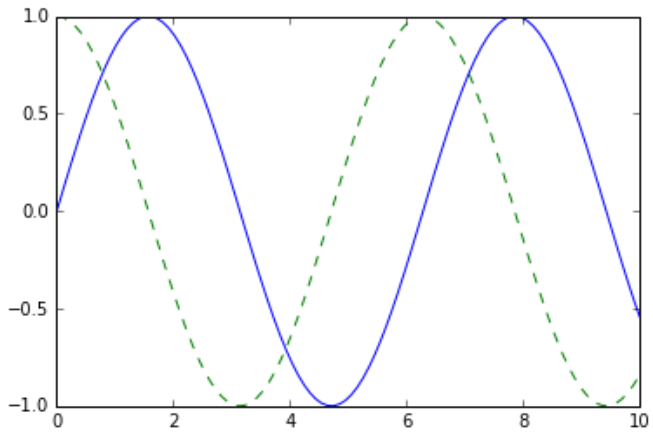
In [2]:
```
import matplotlib
matplotlib.__version__
```

Out[2]:
```
'3.1.1'
```

In [3]:
```
plt.style.use('classic')
```

In [4]:
```
%matplotlib inline
```

In [5]:
```
x = np.linspace(0, 10, 100)

fig = plt.figure()

plt.plot(x, np.sin(x), '-')
plt.plot(x, np.cos(x), '--');
```



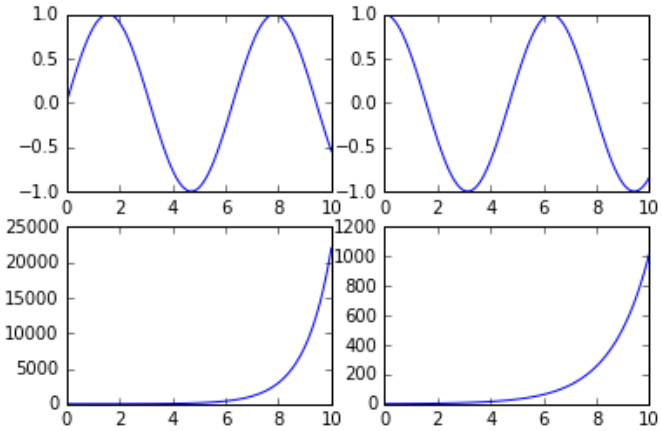
In [6]:
```
fig.savefig('plot1.png')
```

In [7]:
```
!ls -l *.png
```

```
-rw-r--r--  1 skalathur  staff  26238 Oct 14 11:39 plot1.png
```

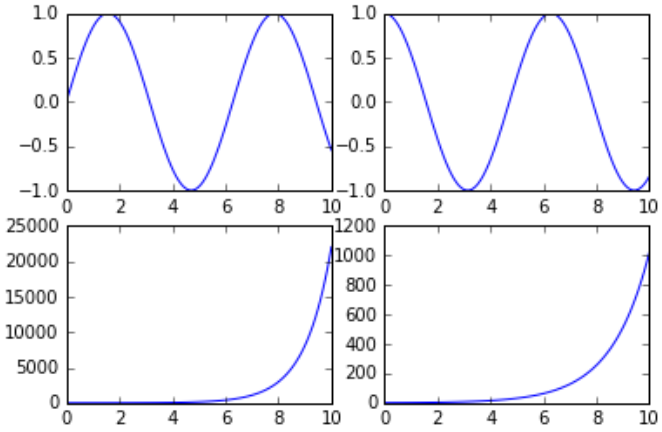
MATLAB-style interface

In [8]:
```
x = np.linspace(0, 10, 100)

plt.figure()  # create a plot figure

# create the first of four panels and set current axis
plt.subplot(2, 2, 1) # (rows, columns, panel number)
plt.plot(x, np.sin(x))

# create the second panel and set current axis
plt.subplot(2, 2, 2)
plt.plot(x, np.cos(x))

# create the third panel and set current axis
plt.subplot(2, 2, 3)
plt.plot(x, np.exp(x))

# create the fouth panel and set current axis
plt.subplot(2, 2, 4)
plt.plot(x, 2 ** x);
```



Object-oriented interface

In [9]:
```
x = np.linspace(0, 10, 100)

# First create a grid of plots
# ax will be an array of 2 x 2 Axes objects

fig, ax = plt.subplots(2,2)

# Call plot() method on the appropriate object
ax[0,0].plot(x, np.sin(x))
ax[0,1].plot(x, np.cos(x))
ax[1,0].plot(x, np.exp(x))
ax[1,1].plot(x, 2 ** x);
```

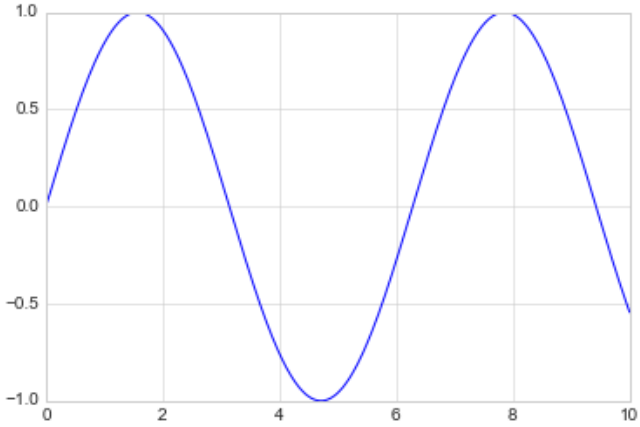


Line plots

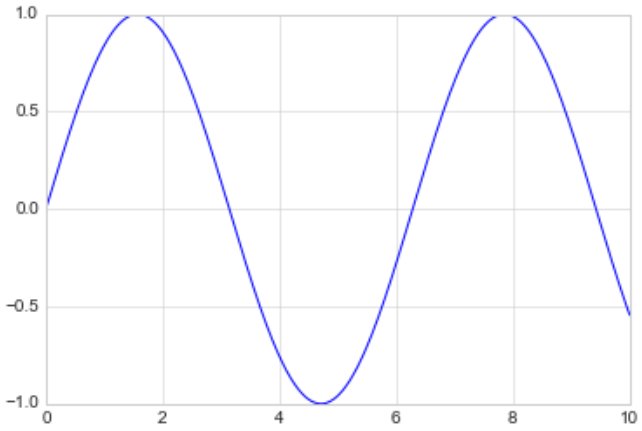
- y = f(x)

In [10]:
```
plt.style.use('seaborn-whitegrid')
```

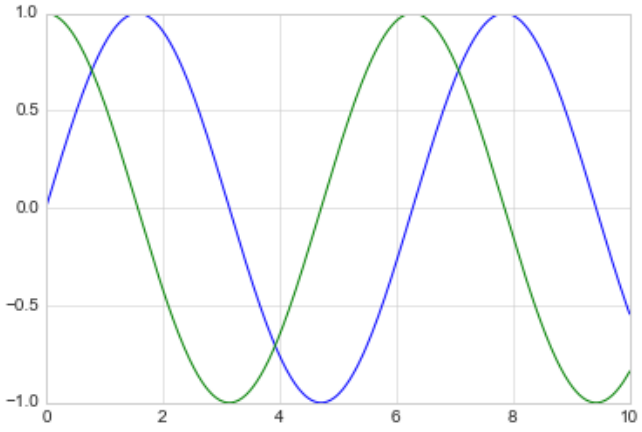
In [11]:
```
fig = plt.figure()
ax = plt.axes()

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x));
```



In [12]:
```
# Alternatively

x = np.linspace(0, 10, 1000)
plt.plot(x, np.sin(x));
```

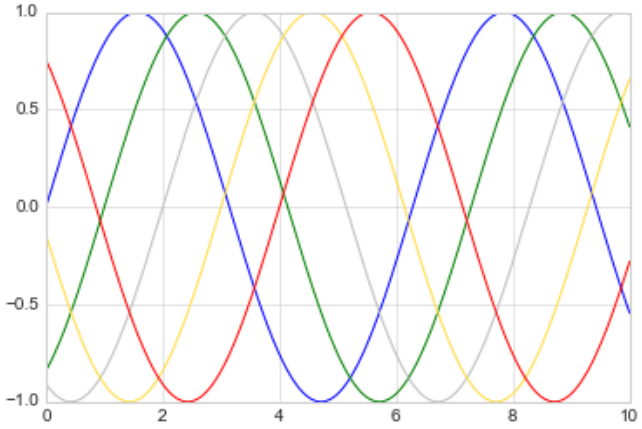


In [13]:
```
# Single figure with multiple lines

x = np.linspace(0, 10, 1000)

plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x));
```

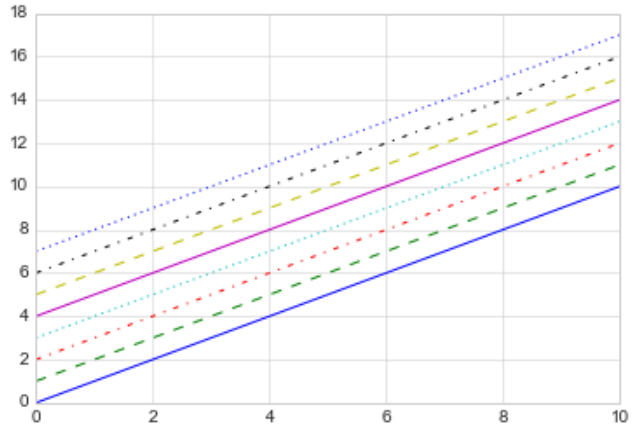


Line Colors and Styles

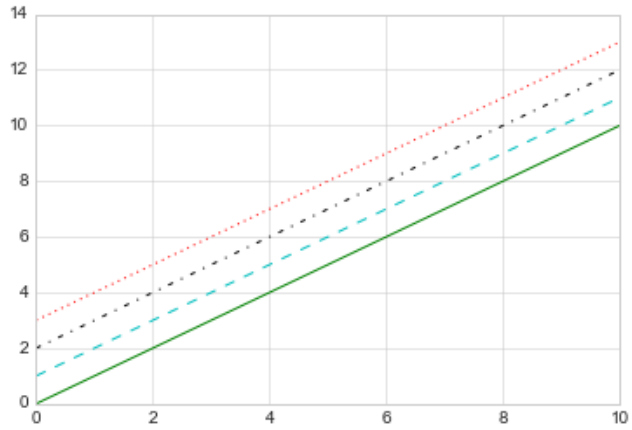
In [14]:
```
x = np.linspace(0, 10, 1000)

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=(1.0,0.0,0.0)); # RGB tuple, values 0 to 1
```



In [15]:
```
x = np.linspace(0, 10, 1000)

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
```

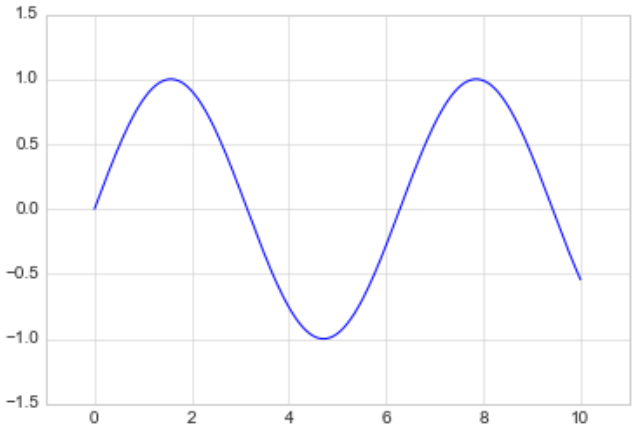


In [16]:
```
x = np.linspace(0, 10, 1000)

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
```

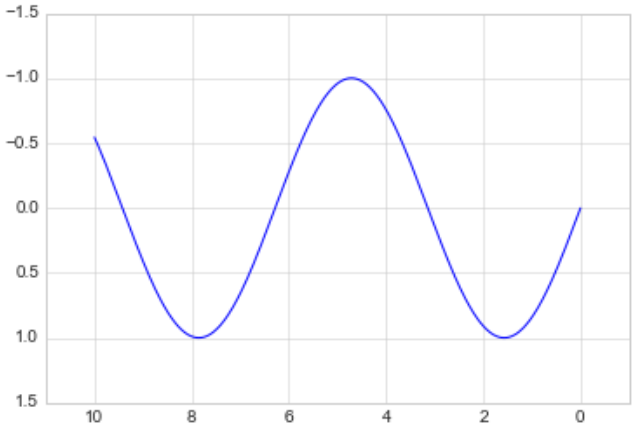


Axes Limits

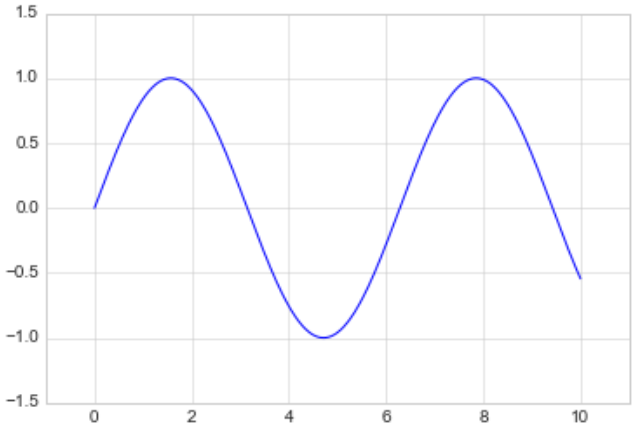
In [17]:
```
x = np.linspace(0, 10, 1000)

plt.plot(x, np.sin(x))

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



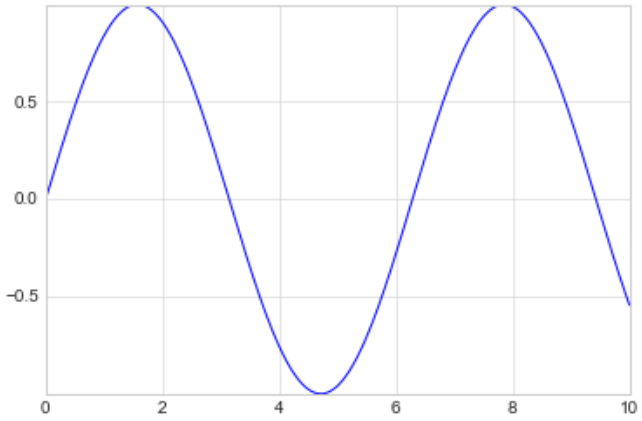
In [18]:
```
# axes in reverse

x = np.linspace(0, 10, 1000)

plt.plot(x, np.sin(x))

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



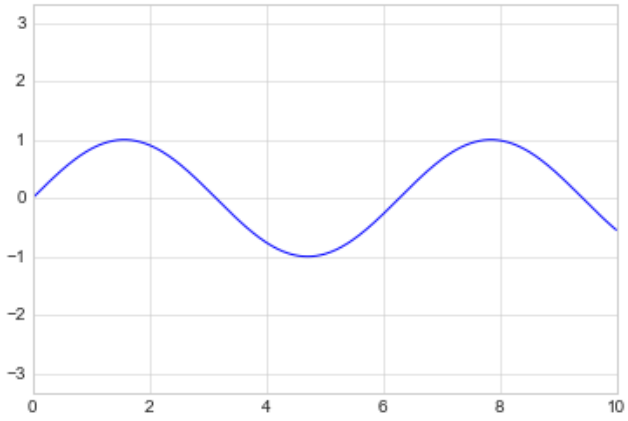
In [19]:
```
# single call axis limits

x = np.linspace(0, 10, 1000)

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



In [20]:
```
x = np.linspace(0, 10, 1000)

plt.plot(x, np.sin(x))
plt.axis('tight');
```

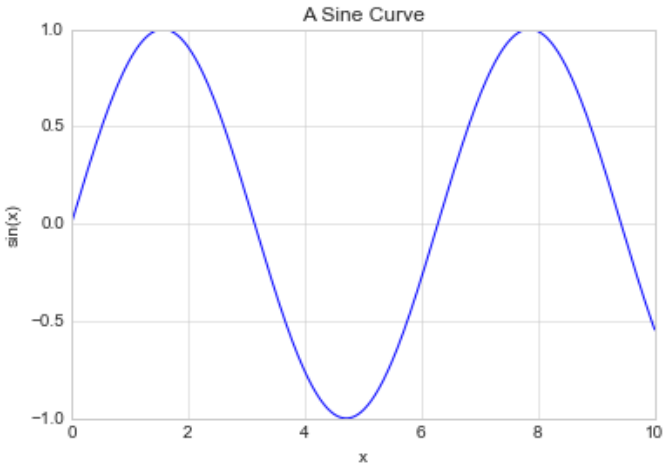


In [21]:
```
x = np.linspace(0, 10, 1000)

plt.plot(x, np.sin(x))
plt.axis('equal');
```

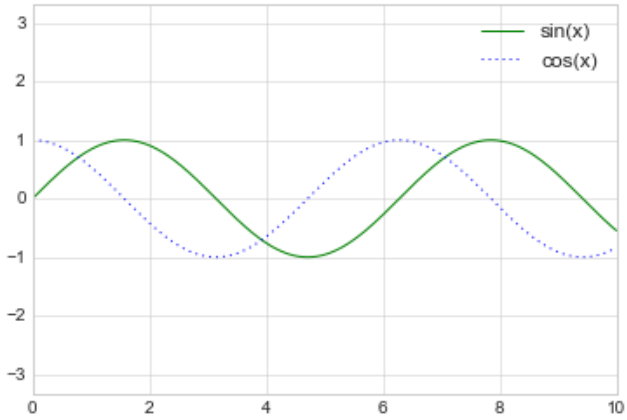


Labeling Plots

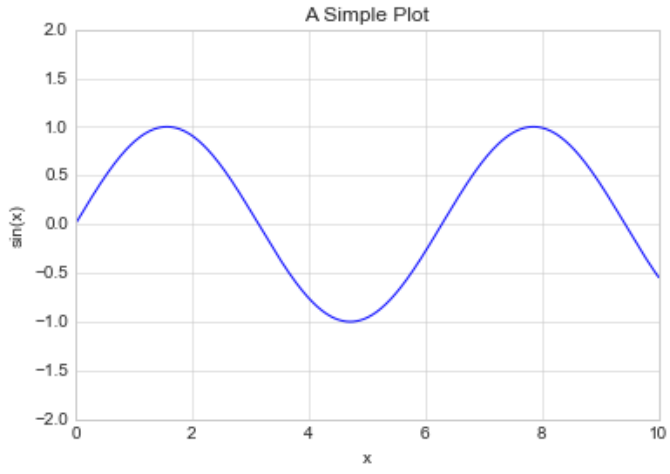
In [22]:
```
x = np.linspace(0, 10, 1000)

plt.plot(x, np.sin(x))

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



In [23]:
```
# legend

x = np.linspace(0, 10, 1000)

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 [24]:
```
## Object-oriented interface

x = np.linspace(0, 10, 1000)

ax = plt.axes()
ax.plot(x, np.sin(x))
ax.set(xlim=(0, 10), ylim=(-2, 2),
       xlabel='x', ylabel='sin(x)',
       title='A Simple Plot');
```

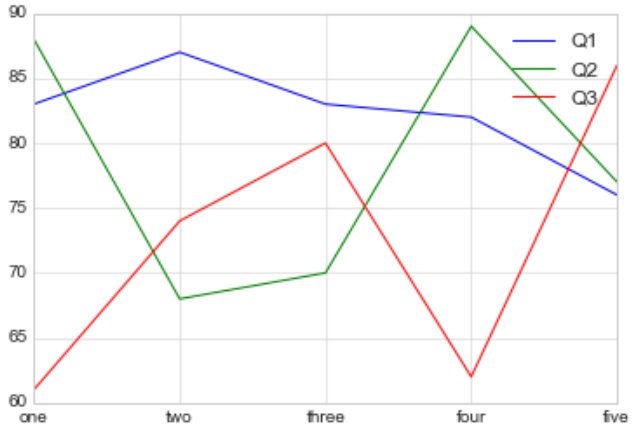


In [25]:
```
# pandas

np.random.seed(813)
df = pd.DataFrame(np.random.randint(60, 90, (5, 3)),
                  columns=['Q1', 'Q2', 'Q3'],
                  index=['one', 'two', 'three', 'four', 'five'])
df
```

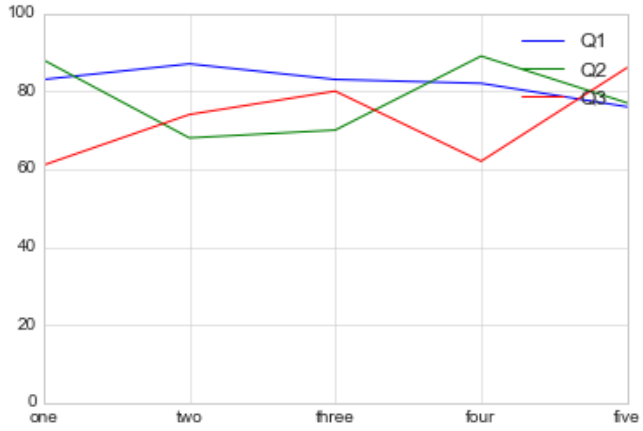
Out[25]:

	Q1	Q2	Q3
one	83	88	61
two	87	68	74
three	83	70	80
four	82	89	62
five	76	77	86

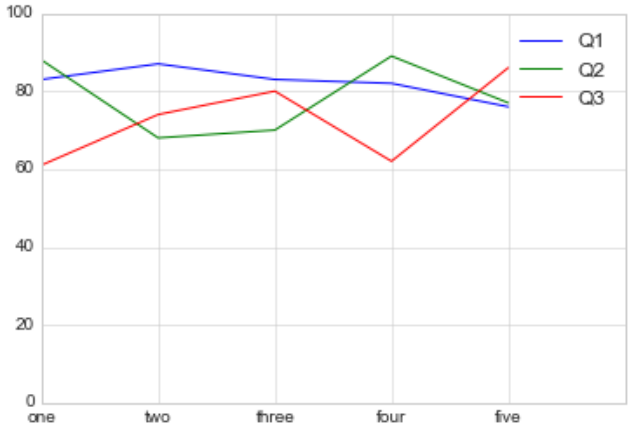
In [26]:
```
plt.plot(df)
plt.legend(df);
```



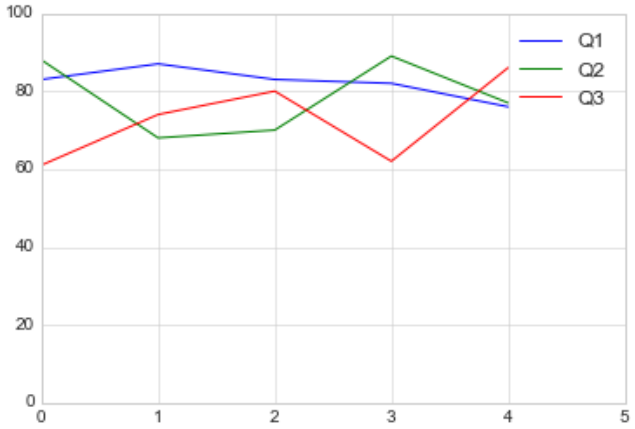
In [27]:
```
plt.ylim(0, 100);
plt.plot(df)
plt.legend(df);
```



In [28]:
```
plt.axis([0, 5, 0, 100])
plt.plot(df)
plt.legend(df);
```



In [29]:
```
plt.axis([0, 5, 0, 100])
plt.plot(np.arange(5), df)
plt.legend(df);
```

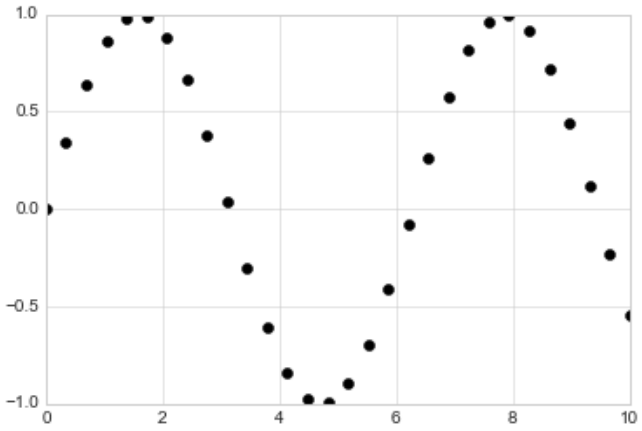


In [ ]:
```

```

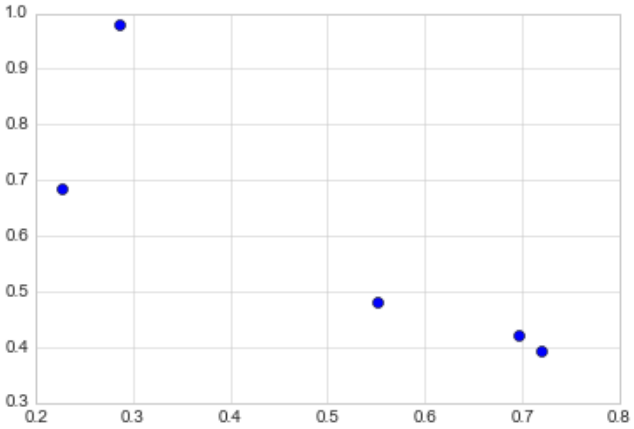
Scatter Plots

- points represented individually with a dot, circle, or other shape

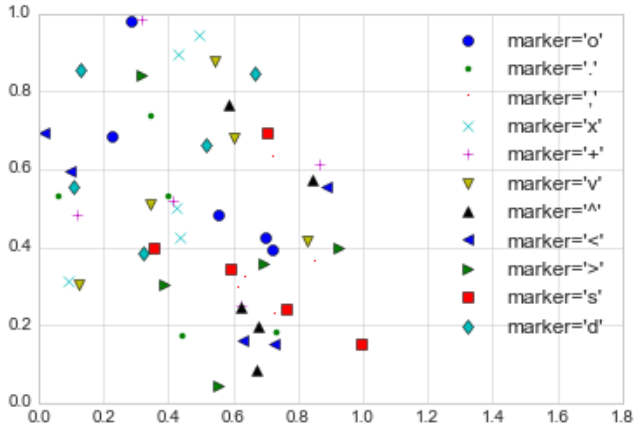
In [30]:
```
x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x, y, 'o', color='black');
```



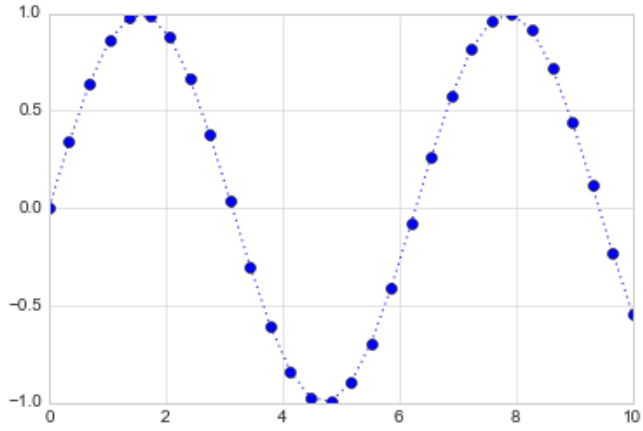
In [31]:
```
np.random.seed(123)
plt.plot(np.random.rand(5), np.random.rand(5), 'o');
```



In [32]:
```
# common marker symbols

np.random.seed(123)

for marker in ['o', '.', ',', 'x', '+', 'v', '^', '<', '>', 's', 'd']:
    plt.plot(np.random.rand(5), np.random.rand(5), marker,
             label="marker='{0}'".format(marker))

    plt.legend()
plt.xlim(0, 1.8);
```



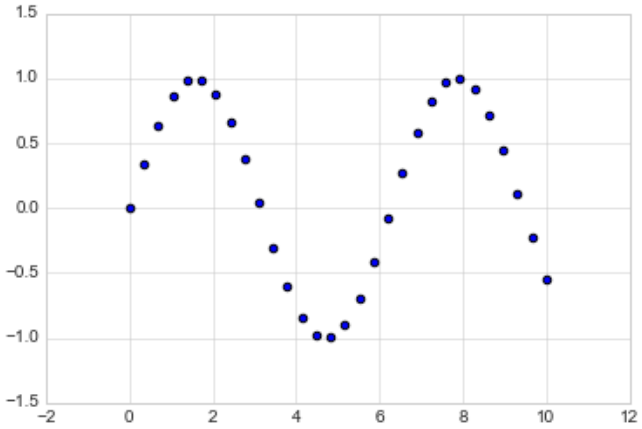
In [33]:
```
# use marker together with line and color codes

plt.plot(x, y, ':ob');
```



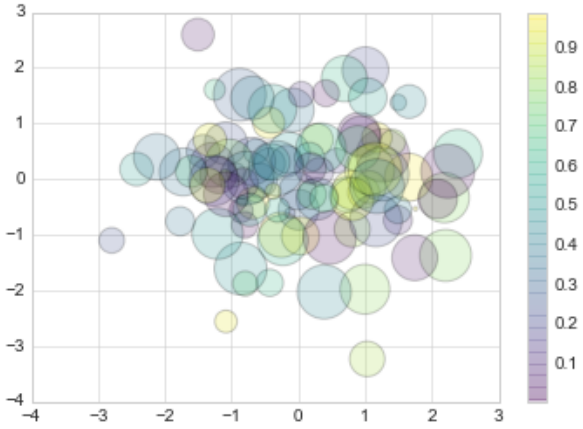
plt.scatter

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html
(https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html)

In [34]:
```
x = np.linspace(0, 10, 30)
y = np.sin(x)
plt.scatter(x, y, marker='o');
```



In [35]:
```
# each point can be configured

np.random.seed(123)
x = np.random.randn(100)
y = np.random.randn(100)
colors = np.random.rand(100)
sizes = 1000 * np.random.rand(100)

plt.scatter(x, y, c=colors, s=sizes, alpha=0.2,
            cmap='viridis')
plt.colorbar();  # show color scale
```

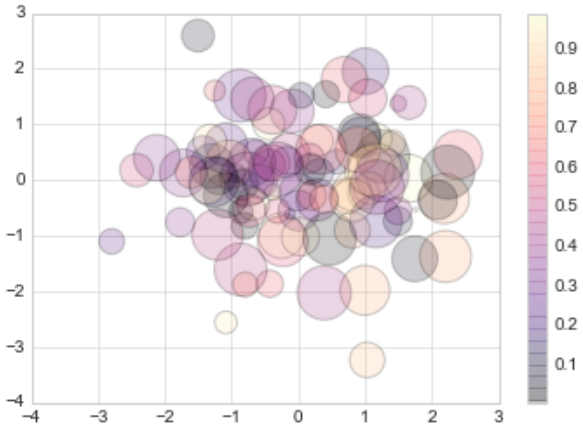


- if all points have the same properties, prefer plt.plot over plt.scatter for better performance

Colormaps

- https://matplotlib.org/3.1.1/gallery/color/colormap_reference.html (https://matplotlib.org/3.1.1/gallery/color/colormap_reference.html)

In [36]:
```
plt.scatter(x, y, c=colors, s=sizes, alpha=0.2,
            cmap='magma')
plt.colorbar();  # show color scale
```



Case Study - Population vs. Area

In [37]:
```
cities = pd.read_csv('http://people.bu.edu/kalathur/datasets/california_cities.csv')
cities.head()
```

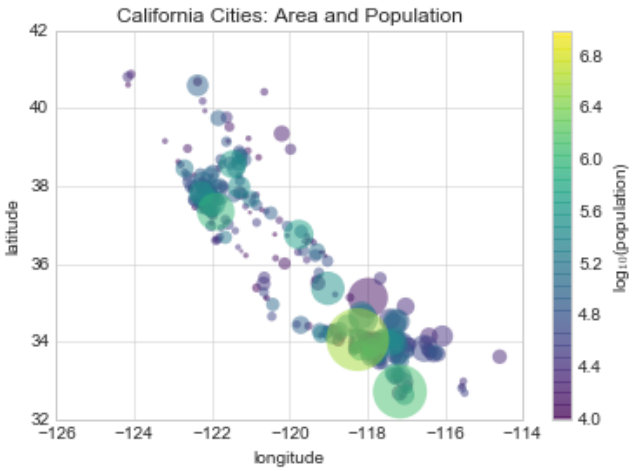
Out[37]:

	city	latd	longd	elevation_m	elevation_ft	population_total	area_total_sq_mi	area_land_sq_mi	a
0	Emeryville	37.831389	-122.285278	7.0	23.0	10080	2.010	1.246	
1	ShastaLake	40.678056	-122.370000	246.0	810.0	10164	10.929	10.921	
2	Newman	37.315000	-121.022500	27.0	89.0	10224	2.102	2.102	
3	MorroBay	35.379167	-120.853333	19.0	62.0	10234	10.322	5.303	
4	Exeter	36.294167	-119.142778	119.0	390.0	10334	2.463	2.463	

In [38]:
```
lat, lon = cities['latd'], cities['longd']

population, area = cities['population_total'], cities['area_total_km2']

population.min(), population.max(), area.min(), area.max()
```

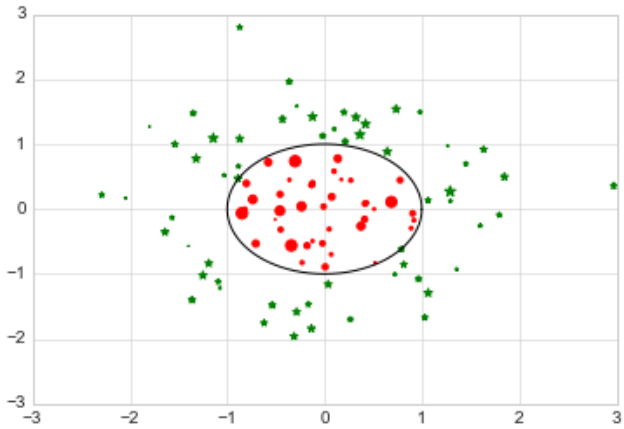
Out[38]:
```
(10080, 3884307, 2.477, 1302.0)
```

In [39]:
```
# Scatter the points, using size and color but no label
plt.scatter(lon, lat, label=None,
            c=np.log10(population), cmap='viridis',
            s=area, linewidth=0, alpha=0.5)
plt.axis(aspect='equal')
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.colorbar(label='log$_{10}$(population)')
plt.clim(4, 7)

plt.title('California Cities: Area and Population');
```



Case Study - Masked Scatter Plot

In [40]:
```
np.random.seed(321)
N = 100
r0 = 1.0 # boundary radius

x = np.random.randn(N)
y = np.random.randn(N)
r = np.sqrt(x * x + y * y)

# for size parameter
area = np.pi * (10 * np.random.randn(N))**2
area = np.sqrt(area)

# mask into two regions
area1 = np.ma.masked_where(r < r0, area)
area2 = np.ma.masked_where(r >= r0, area)

plt.axis([-3, 3, -3, 3]);

plt.scatter(x, y, s=area1, marker='*', color='g')
plt.scatter(x, y, s=area2, marker='o', color='r')

# Show the boundary between the regions:
theta = np.arange(0, 2* np.pi , .01)
plt.plot(r0 * np.cos(theta), r0 * np.sin(theta), 'k')

plt.show()
```

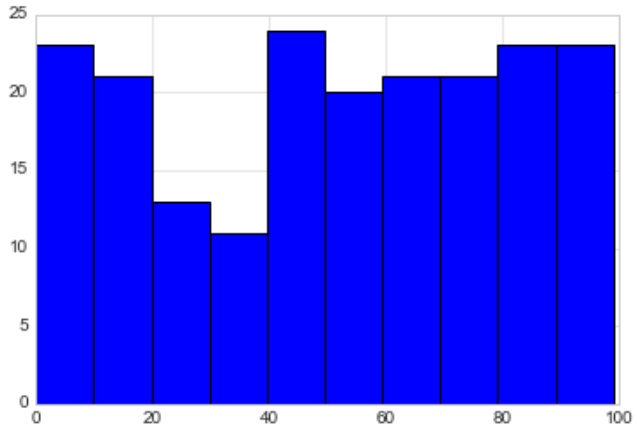


Histogram

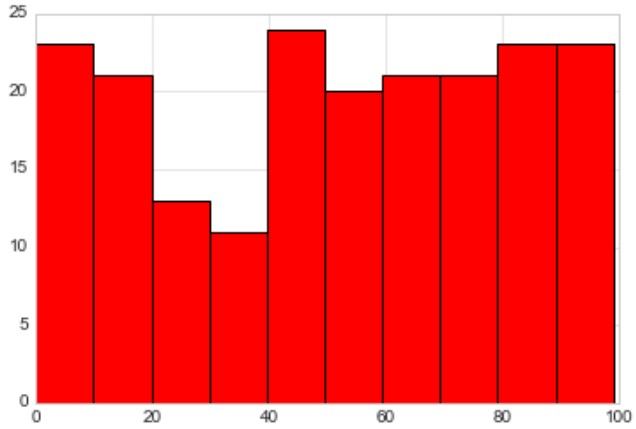
https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hist.html (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hist.html)

In [41]:
```
np.random.seed(123)

data = np.random.randint(0,100,200)
plt.hist(data)
```

Out[41]:
```
(array([23., 21., 13., 11., 24., 20., 21., 21., 23., 23.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 <a list of 10 Patch objects>)
```

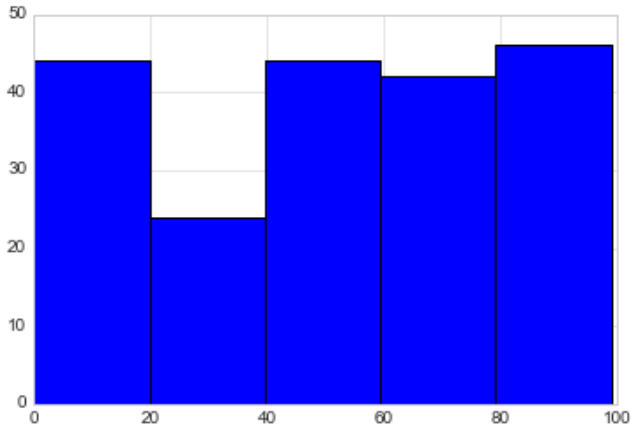


In [42]:
```
plt.hist(data, color='r');
```

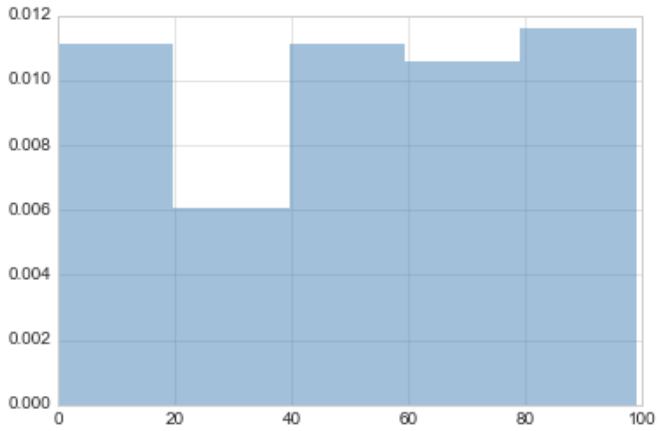


In [43]:
```
plt.hist(data, bins=5)
```

Out[43]:
```
(array([44., 24., 44., 42., 46.]),
 array([ 0. , 19.8, 39.6, 59.4, 79.2, 99. ]),
 <a list of 5 Patch objects>)
```

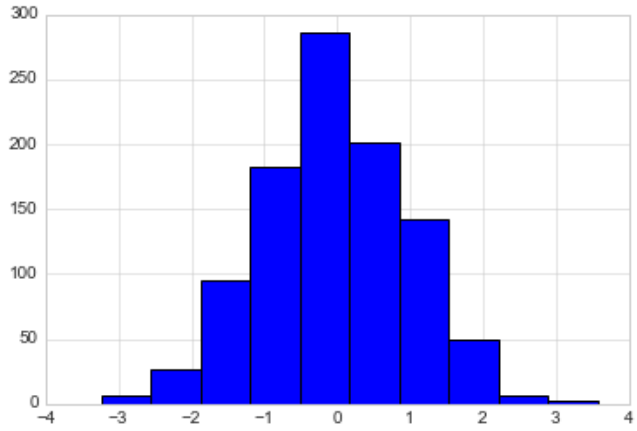


In [44]:
```
plt.hist(data, bins=5, density=True, alpha=0.5,
         color='steelblue', edgecolor='none');
```

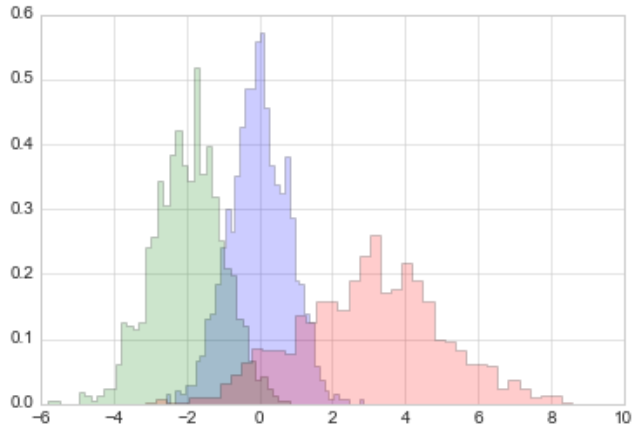


In [45]:
```
np.random.seed(123)

data = np.random.randn(1000)
plt.hist(data)
```

Out[45]:
```
(array([  7.,  27.,  95., 183., 286., 202., 142.,  49.,   7.,   2.]),
 array([-3.23105501, -2.55079159, -1.87052816, -1.19026474, -0.51000132,
         0.17026211,  0.85052553,  1.53078895,  2.21105237,  2.8913158 ,
         3.57157922]),
 <a list of 10 Patch objects>)
```



In [46]:
```
np.random.seed(123)

x1 = np.random.normal(0, 0.8, 1000)  # mean, sd, number of values
x2 = np.random.normal(-2, 1, 1000)
x3 = np.random.normal(3, 2, 1000)

kwargs = dict(histtype='stepfilled', alpha=0.2, density=True, bins=40)

plt.hist(x1, **kwargs)
plt.hist(x2, **kwargs)
plt.hist(x3, **kwargs);
```



Bar Chart

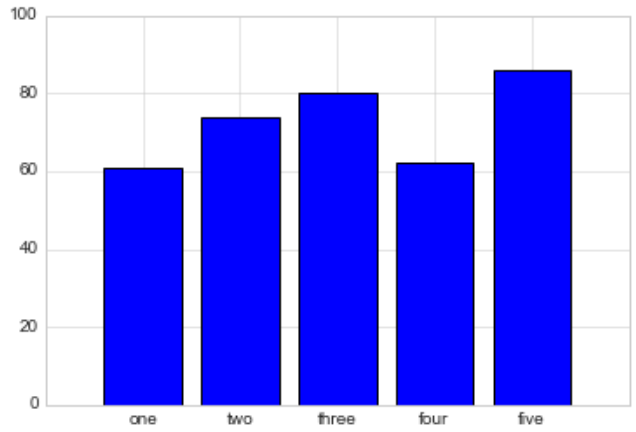
https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.bar.html
(https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.bar.html)

In [47]:
```
df
```

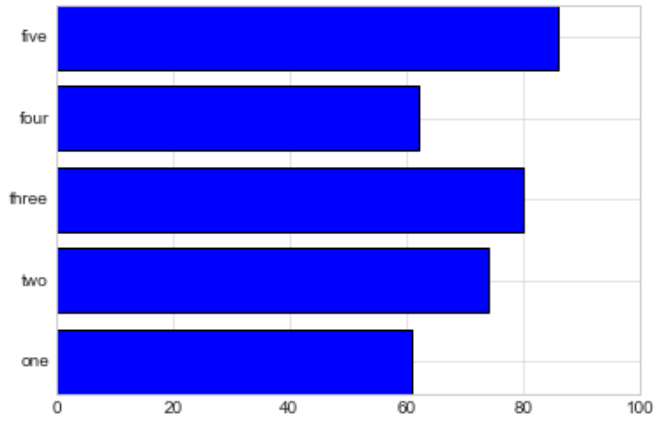
Out[47]:

	Q1	Q2	Q3
one	83	88	61
two	87	68	74
three	83	70	80
four	82	89	62
five	76	77	86

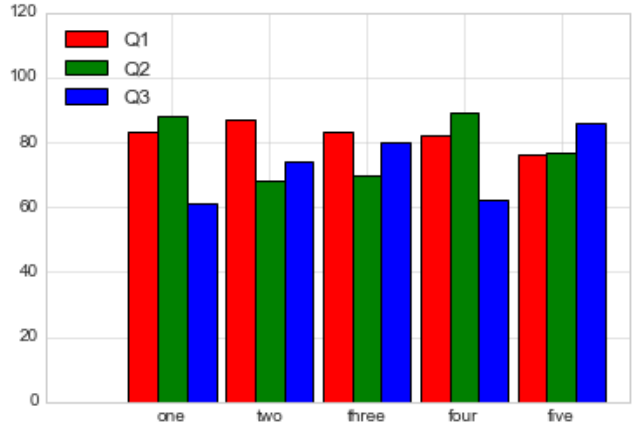
In [48]:
```
plt.bar(df.index, df['Q3'])
plt.axis([-1,5, 0,100]);
```



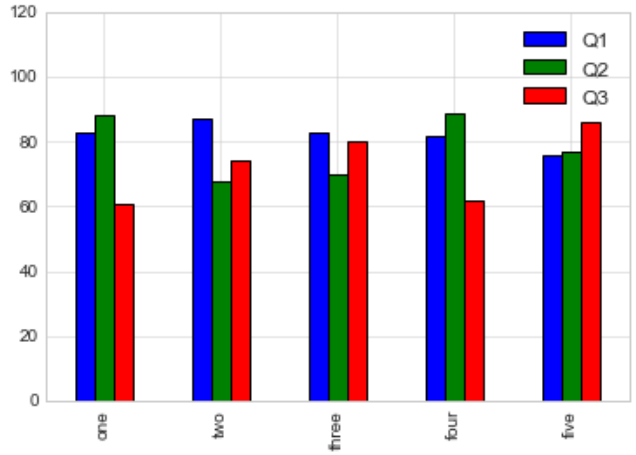
In [49]:
```
plt.barh(df.index, df['Q3'])
plt.xlim(0, 100);
```



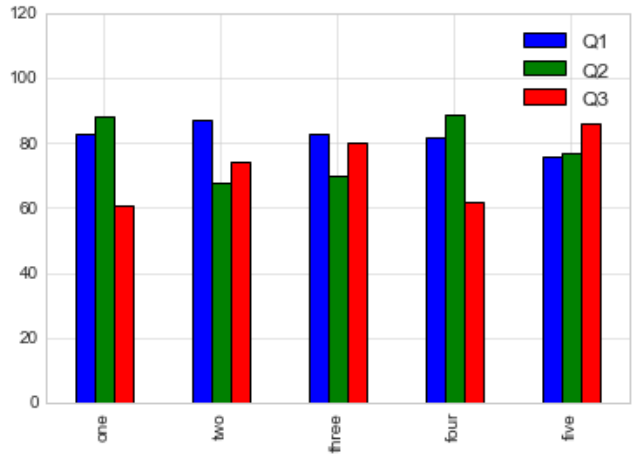
In [50]:
```
index = np.arange(len(df))
barWidth = 0.3

plt.ylim(0, 120)
plt.bar(index,              df['Q1'], barWidth, color='r', label='Q1')
plt.bar(index + barWidth,   df['Q2'], barWidth, color='g', label = 'Q2')
plt.bar(index + 2*barWidth, df['Q3'], barWidth, color='b', label = 'Q3')

plt.legend(loc = 2)
plt.xticks(index+barWidth, df.index);
```



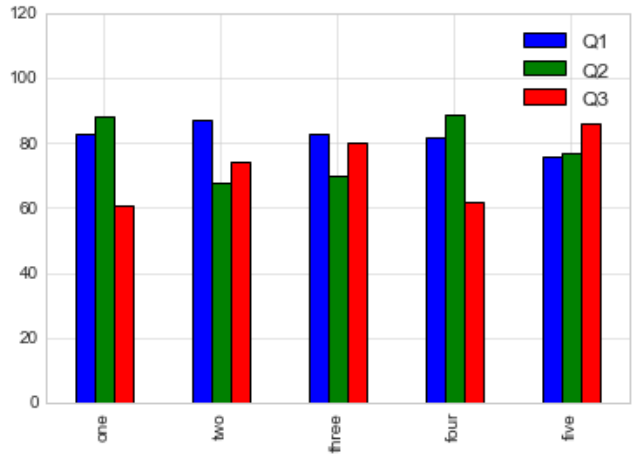
In [51]:
```
# Using pandas

fig = plt.figure()
ax = plt.axes()
ax.set(ylim=(0, 120))

df.plot(kind = 'bar', ax=ax);
```



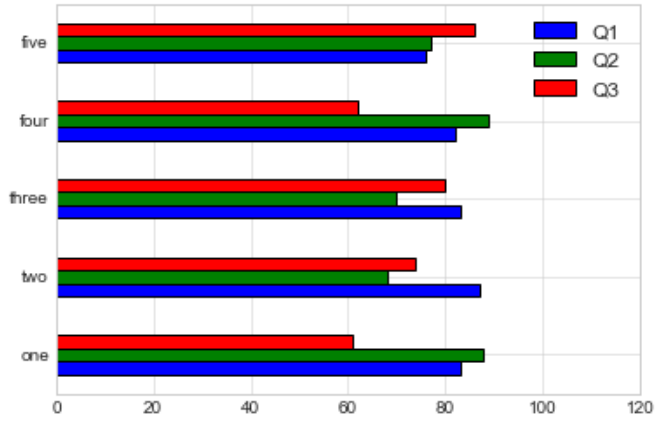
In [52]:
```
fig, ax = plt.subplots()

ax.set(ylim=(0, 120))

df.plot(kind = 'bar', ax=ax);
```



In [53]:
```
df.plot(kind = 'bar', ylim=(0, 120));
```



In [54]:
```
df.plot(kind = 'barh', xlim=(0, 120));
```



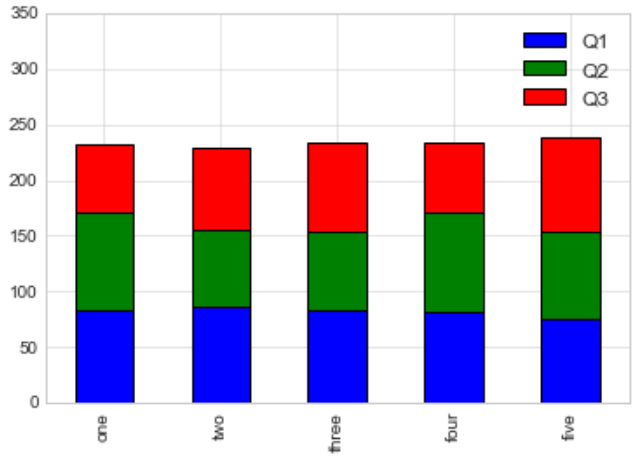
Stacked Bar Chart

In [55]:
```
df
```

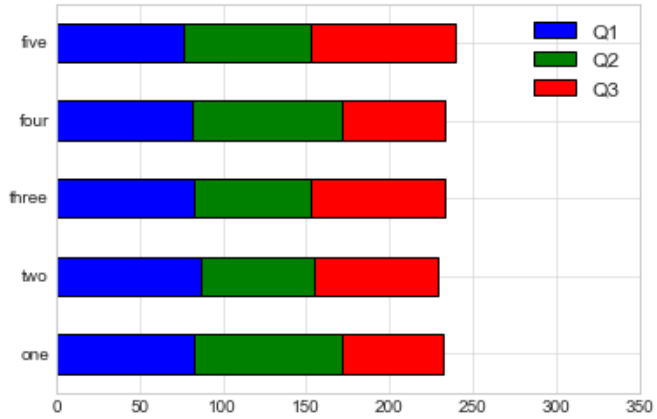
Out[55]:

	Q1	Q2	Q3
one	83	88	61
two	87	68	74
three	83	70	80
four	82	89	62
five	76	77	86

In [56]:
```
df.plot(kind = 'bar', stacked = True, ylim=(0, 350));
```



In [57]:
```
df.plot(kind = 'barh', stacked = True, xlim=(0, 350));
```

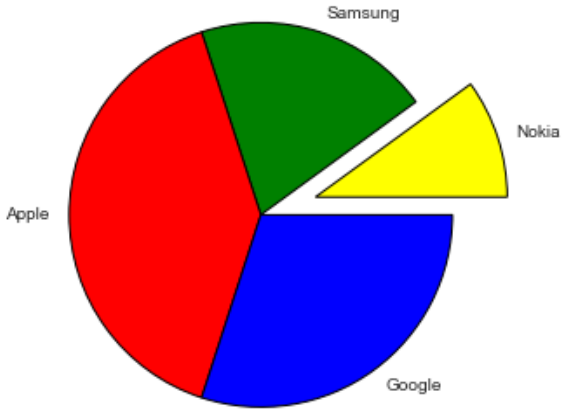


Pie Charts

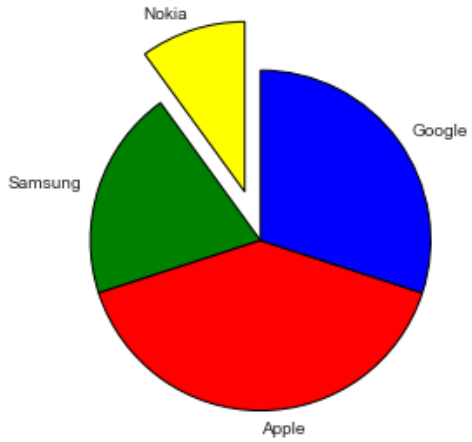
https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.pie.html
(https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.pie.html)

In [58]:
```
labels = ['Nokia','Samsung','Apple','Google']
values = [100,200,400, 300]
colors = ['yellow','green','red','blue']

plt.pie(values, labels=labels, colors=colors)
plt.axis('equal');
```

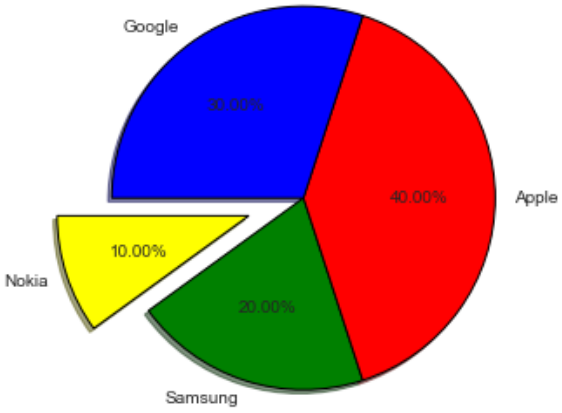
In [59]:
```
labels = ['Nokia','Samsung','Apple','Google']
values = [10,20,40, 30]
colors = ['yellow','green','red','blue']
explode = [0.3,0,0,0]

plt.pie(values, labels=labels, colors=colors,
        explode=explode)
plt.axis('equal');
```



In [60]:
```
plt.pie(values, labels=labels, colors=colors,
        explode=explode, startangle=90)
plt.axis('equal');
```



In [61]:
```
plt.pie(values,labels=labels,colors=colors,
        explode=explode,shadow=True,
        autopct='%.2f%%',startangle=180)
plt.axis('equal');
```

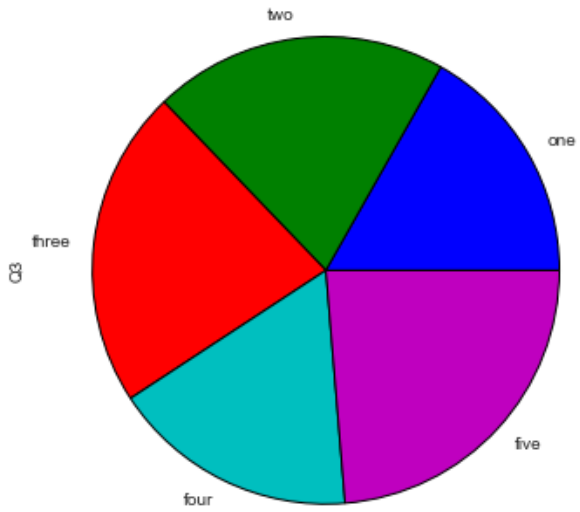


In [62]:
```
# Pandas

df
```

Out[62]:

	Q1	Q2	Q3
one	83	88	61
two	87	68	74
three	83	70	80
four	82	89	62
five	76	77	86

In [63]:
```
df['Q3'].plot(kind='pie', figsize=(6,6));
# 6 inch x 6 inch
```



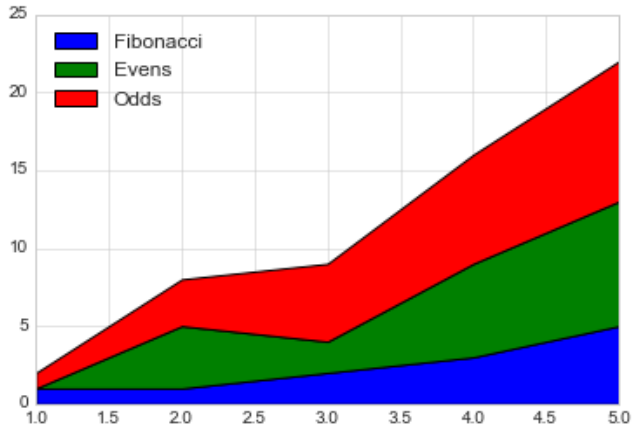
In [64]:
```
df.plot(kind='pie', subplots=True,
        figsize=(12,8), legend=False);
```

Stacked Area Plot

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.stackplot.html
(https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.stackplot.html)

In [65]:
```
x = [1, 2, 3, 4, 5]

y1 = [1, 1, 2, 3, 5]
y2 = [0, 4, 2, 6, 8]
y3 = [1, 3, 5, 7, 9]
```

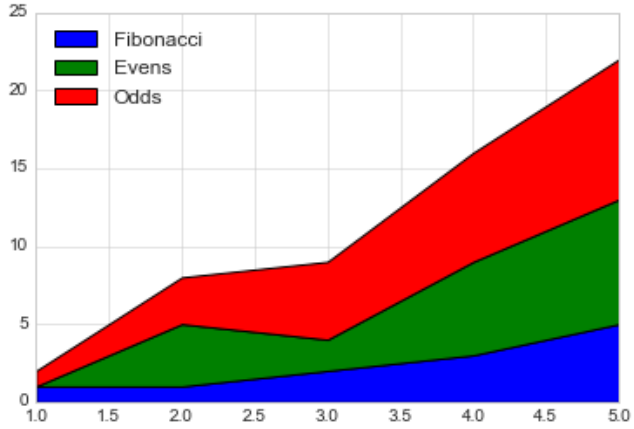
In [66]:
```
labels = ["Fibonacci ", "Evens", "Odds"]

fig, ax = plt.subplots()
ax.stackplot(x, y1, y2, y3, labels=labels)
ax.legend(loc='upper left')
plt.show()
```



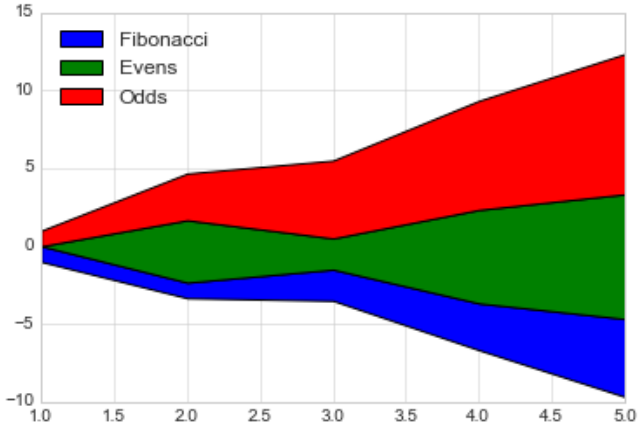
In [67]:
```
y = np.vstack([y1, y2, y3])
y
```

Out[67]:
```
array([[1, 1, 2, 3, 5],
       [0, 4, 2, 6, 8],
       [1, 3, 5, 7, 9]])
```

In [68]:
```
fig, ax = plt.subplots()
ax.stackplot(x, y, labels=labels)
ax.legend(loc='upper left')
plt.show()
```



In [69]:
```
fig, ax = plt.subplots()
ax.stackplot(x, y, labels=labels, baseline="wiggle")
ax.legend(loc='upper left')
plt.show()
```



In [ ]: