```
import numpy as np

import pandas as pd
```

```
import numpy as np
lst = [[1, 2, 3],[4, 5, 6]]
ary1d = np.array(lst)
ary1d
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
float32_ary = ary1d.astype(np.float32)
float32_ary
```

```
array([[1., 2., 3.],
       [4., 5., 6.]], dtype=float32)
```

```
ary2d = np.array([[1, 2, 3],[4, 5, 6]], dtype='int64')
ary2d.itemsize
```

```
8
```

```
ary2d.size
```

```
6
```

```
ary2d.ndim
```

```
2
```

```
ary2d.shape
```

```
(2, 3)
```

```
np.array([1, 2, 3]).shape
```

```
(3,)
```

```
scalar = np.array(5)
scalar
```

```
array(5)
```

```
scalar.ndim
```

```
0
```

```
scalar.shape
```

```
()
```

```
def generator():
    for i in range(10):
        if i % 2:
            yield i

gen = generator()
np.fromiter(gen, dtype=int)
```

```
array([1, 3, 5, 7, 9])
```

```
generator_expression = (i for i in range(10) if i % 2)
np.fromiter(generator_expression, dtype=int)
```

```
array([1, 3, 5, 7, 9])
```

```
np.ones((3, 3))
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

```
np.zeros((3, 3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
np.diag((3, 3, 3))
```

```
array([[3, 0, 0],
       [0, 3, 0],
       [0, 0, 3]])
```

```
np.arange(4., 10.)
```

```
array([4., 5., 6., 7., 8., 9.])
```

```
np.arange(5)
```

```
array([0, 1, 2, 3, 4])
```

```
np.arange(1., 11., 2)
```

```
array([1., 3., 5., 7., 9.])
```

```
np.linspace(0., 1., num=5)
```

```
array([0.  , 0.25, 0.5 , 0.75, 1.  ])
```

```
ary = np.array([1, 2, 3])
ary[0]
```

```
1
```

```
ary[:2] # equivalent to ary[0:2]
```

```
array([1, 2])
```

```
ary = np.array([[1, 2, 3],
                [4, 5, 6]])

ary[0, 0] # upper left
```

```
1
```

```
ary[-1, -1] # lower right
```

```
6
```

```
ary[0, 1] # first row, second column
```

```
2
```

```
ary[0] # entire first row
```

```
array([1, 2, 3])
```

```
ary[:, 0] # entire first column
```

```
array([1, 4])
```

```
ary[:, :2] # first two columns
```

```
array([[1, 2],
       [4, 5]])
```

```
ary[0, 0]
```

```
1
```

```
lst = [[1, 2, 3], [4, 5, 6]]

for row_idx, row_val in enumerate(lst):
    for col_idx, col_val in enumerate(row_val):
        lst[row_idx][col_idx] += 1
lst
```

```
[[2, 3, 4], [5, 6, 7]]
```

```
lst = [[1, 2, 3], [4, 5, 6]]
[[cell + 1 for cell in row] for row in lst]
```

```
[[2, 3, 4], [5, 6, 7]]
```

```
ary = np.array([[1, 2, 3], [4, 5, 6]])
ary = np.add(ary, 1)
ary
```

```
array([[2, 3, 4],
       [5, 6, 7]])
```

```
ary + 1
```

```
array([[3, 4, 5],
       [6, 7, 8]])
```

```
ary**2
```

```
array([[ 4,  9, 16],
       [25, 36, 49]])
```

```
ary = np.array([[1, 2, 3],
                [4, 5, 6]])

np.add.reduce(ary) # column sumns
```

```
array([5, 7, 9])
```

```
np.add.reduce(ary, axis=1) # row sums
```

```
array([ 6, 15])
```

```
ary.sum(axis=0) # column sums
```

```
array([5, 7, 9])
```

```
ary.sum()
```

```
21
```

```
ary1 = np.array([1, 2, 3])
ary2 = np.array([4, 5, 6])

ary1 + ary2
```

```
array([5, 7, 9])
```

```
ary3 = np.array([[4, 5, 6],
                 [7, 8, 9]])

ary3 + ary1 # similarly, ary1 + ary3
```

```
array([[ 5,  7,  9],
       [ 8, 10, 12]])
```

```
try:
    ary3 + np.array([1, 2])
except ValueError as e:
    print('ValueError:', e)
```

```
ValueError: operands could not be broadcast together with shapes (2,3) (2,)
```

```
ary3 + np.array([[1], [2]])
```

```
array([[ 5,  6,  7],
       [ 9, 10, 11]])
```

```
np.array([[1], [2]]) + ary3
```

```
array([[ 5,  6,  7],
       [ 9, 10, 11]])
```

```
ary = np.array([[1, 2, 3],
                [4, 5, 6]])

first_row = ary[0]
first_row += 99
ary
```

```
array([[100, 101, 102],
       [  4,   5,   6]])
```

```
ary = np.array([[1, 2, 3],
                [4, 5, 6]])

first_row = ary[:1]
first_row += 99
ary
```

```
array([[100, 101, 102],
       [  4,   5,   6]])
```

```
ary = np.array([[1, 2, 3],
                [4, 5, 6]])

center_col = ary[:, 1]
center_col += 99
ary
```

```
array([[  1, 101,   3],
       [  4, 104,   6]])
```

```
second_row = ary[1].copy()
second_row += 99
ary
```

```
array([[  1, 101,   3],
       [  4, 104,   6]])
```

```
ary = np.array([[1, 2, 3],
                [4, 5, 6]])

first_row = ary[:1]
first_row += 99
ary
```

```
array([[100, 101, 102],
       [  4,   5,   6]])
```

```
np.may_share_memory(first_row, ary)
```

```
True
```

```
second_row = ary[1].copy()
second_row += 99
ary
```

```
array([[100, 101, 102],
       [  4,   5,   6]])
```

```
np.may_share_memory(second_row, ary)
```

```
False
```

```
ary = np.array([[1, 2, 3],
                [4, 5, 6]])

ary[:, [0, 2]] # first and and last column
```

```
array([[1, 3],
       [4, 6]])
```

```
this_is_a_copy = ary[:, [0, 2]]
this_is_a_copy += 99
ary
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
ary[:, [2, 0]] # first and and last column
```

```
array([[3, 1],
       [6, 4]])
```

```
ary = np.array([[1, 2, 3],
                [4, 5, 6]])

greater3_mask = ary > 3
greater3_mask
```

```
array([[False, False, False],
       [ True,  True,  True]])
```

```
ary[greater3_mask]
```

```
array([4, 5, 6])
```

```
ary[(ary > 3) & (ary % 2 == 0)]
```

```
array([4, 6])
```

```
np.random.seed(123)
np.random.rand(3)
```

```
array([0.69646919, 0.28613933, 0.22685145])
```

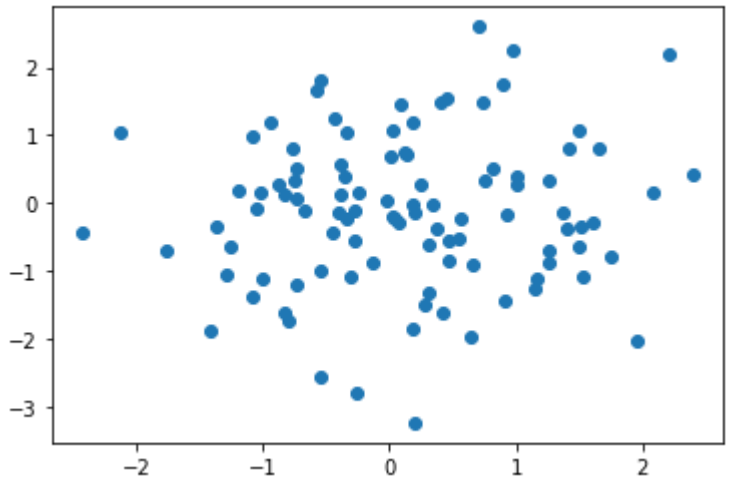
```
rng1 = np.random.RandomState(seed=123)
rng1.rand(3)
```

```
array([0.69646919, 0.28613933, 0.22685145])
```

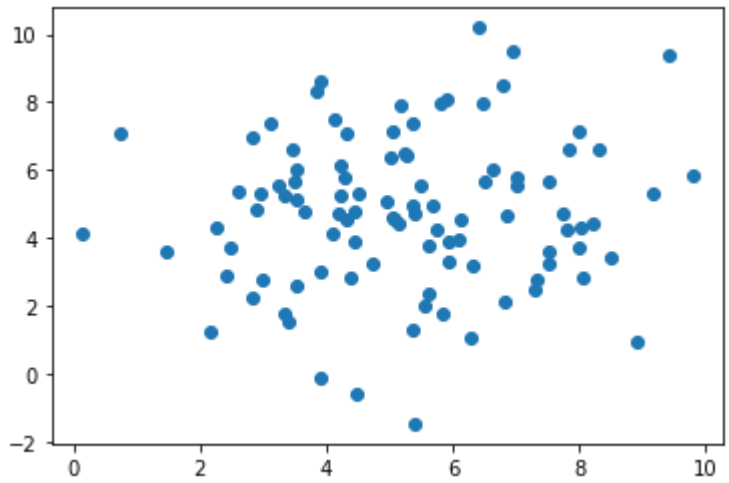
```
rng2 = np.random.RandomState(seed=123)
z_scores = rng2.randn(100, 2)
```

```
%matplotlib inline
import matplotlib.pyplot as plt
```

```
plt.scatter(z_scores[:, 0], z_scores[:, 1])
#plt.savefig('images/numpy-intro/random_1.png', dpi=600)
plt.show()
```




```
rng3 = np.random.RandomState(seed=123)
scores = 2. * rng3.randn(100, 2) + 5.

plt.scatter(scores[:, 0], scores[:, 1])
#plt.savefig('images/numpy-intro/random_2.png', dpi=600)
plt.show()
```




```
ary1d = np.array([1, 2, 3, 4, 5, 6])
ary2d_view = ary1d.reshape(2, 3)
ary2d_view
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
np.may_share_memory(ary2d_view, ary1d)
```

```
True
```

```
ary1d.reshape(2, -1)
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
ary2d = np.array([[1, 2, 3],
                  [4, 5, 6]])

ary2d.reshape(-1)
```

```
array([1, 2, 3, 4, 5, 6])
```

```
ary2d.ravel()
```

```
array([1, 2, 3, 4, 5, 6])
```

```
np.may_share_memory(ary2d.flatten(), ary2d)
```

```
False
```

```
np.may_share_memory(ary2d.ravel(), ary2d)
```

```
True
```

```
ary = np.array([1, 2, 3])

# stack along the first axis
np.concatenate((ary, ary))
```

```
array([1, 2, 3, 1, 2, 3])
```

```
ary = np.array([[1, 2, 3]])

# stack along the first axis (here: rows)
np.concatenate((ary, ary), axis=0)
```

```
array([[1, 2, 3],
       [1, 2, 3]])
```

```
# stack along the second axis (here: column)
np.concatenate((ary, ary), axis=1)
```

```
array([[1, 2, 3, 1, 2, 3]])
```

```
ary = np.array([1, 2, 3, 4])
mask = ary > 2
mask
```

```
array([False, False,  True,  True])
```

```
ary[mask]
```

```
     array([3, 4])
```

```
mask
```

```
     array([False, False,  True,  True])
```

```
mask.sum()
```

```
     2
```

```
mask.nonzero()
```

```
     (array([2, 3]),)
```

```
(ary > 2).nonzero()
```

```
     (array([2, 3]),)
```

```
np.where(ary > 2)
```

```
     (array([2, 3]),)
```

```
np.where(ary > 2, 1, 0)
```

```
     array([0, 0, 1, 1])
```

```
ary = np.array([1, 2, 3, 4])
mask = ary > 2
ary[mask] = 1
ary[~mask] = 0
ary
```

```
     array([0, 0, 1, 1])
```

```
ary = np.array([1, 2, 3, 4])

(ary > 3) | (ary < 2)
```

```
     array([ True, False, False,  True])
```

```
~((ary > 3) | (ary < 2))
```

```
     array([False,  True,  True, False])
```

```
row_vector = np.array([1, 2, 3])
row_vector
```

```
     array([1, 2, 3])
```

```
```

```
column_vector = np.array([[1, 2, 3]]).reshape(-1, 1)
column_vector
```

```
array([[1],
       [2],
       [3]])
```

```
row_vector[:, np.newaxis]
```

```
array([[1],
       [2],
       [3]])
```

```
row_vector[:, None]
```

```
array([[1],
       [2],
       [3]])
```

```
matrix = np.array([[1, 2, 3],
                   [4, 5, 6]])
```

```
np.matmul(matrix, column_vector)
```

```
array([[14],
       [32]])
```

```
np.matmul(matrix, row_vector)
```

```
array([14, 32])
```

```
np.matmul(row_vector, row_vector)
```

```
14
```

```
np.dot(row_vector, row_vector)
```

```
14
```

```
np.dot(matrix, row_vector)
```

```
array([14, 32])
```

```
np.dot(matrix, column_vector)
```

```
array([[14],
       [32]])
```

```
matrix = np.array([[1, 2, 3],
```

```
                  [4, 5, 6]])

matrix.transpose()
```

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

```
np.matmul(matrix, matrix.transpose())
```

```
array([[14, 32],
       [32, 77]])
```

```
matrix.T
```

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

```
ary = np.array([1, 1, 2, 3, 1, 5])
ary_set = np.unique(ary)
ary_set
```

```
array([1, 2, 3, 5])
```

```
ary1 = np.array([1, 2, 3])
ary2 = np.array([3, 4, 5, 6])
np.intersect1d(ary1, ary2, assume_unique=True)
```

```
array([3])
```

```
np.setdiff1d(ary1, ary2, assume_unique=True)
```

```
array([1, 2])
```

```
np.union1d(ary1, ary2) # does not have assume_unique
```

```
array([1, 2, 3, 4, 5, 6])
```

```
np.union1d(np.setdiff1d(ary1, ary2,
                        assume_unique=True),
           np.setdiff1d(ary2, ary1,
                        assume_unique=True))
```

```
array([1, 2, 4, 5, 6])
```

```
ary1 = np.array([1, 2, 3])

np.save('ary-data.npy', ary1)
np.load('ary-data.npy')
```

```
array([1, 2, 3])
```

```
np.savez('ary-data.npz', ary1, ary2)
```

```
d = np.load('ary-data.npz')
d.keys()
```

```
KeysView(<numpy.lib.npyio.NpzFile object at 0x7f72e7c40d10>)
```

```
d['arr_0']
```

```
array([1, 2, 3])
```

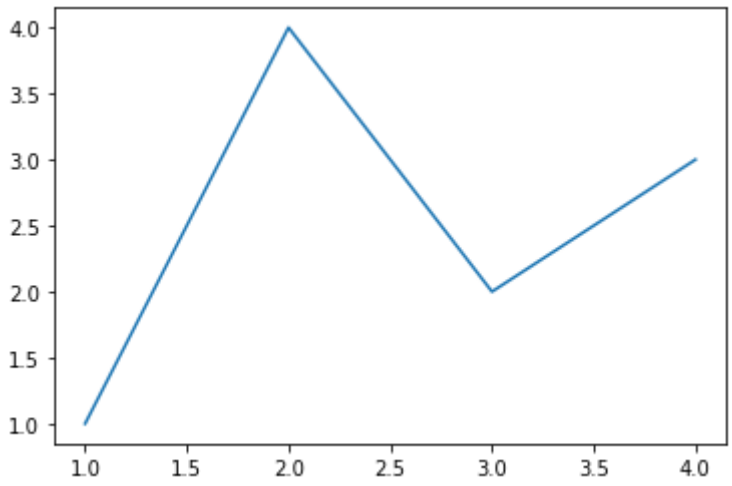
```
kwargs = {'ary1':ary1, 'ary2':ary2}

np.savez('ary-data.npz',
         **kwargs)
```

```
np.load('ary-data.npz')
d = np.load('ary-data.npz')
d['ary1']
```

```
array([1, 2, 3])
```

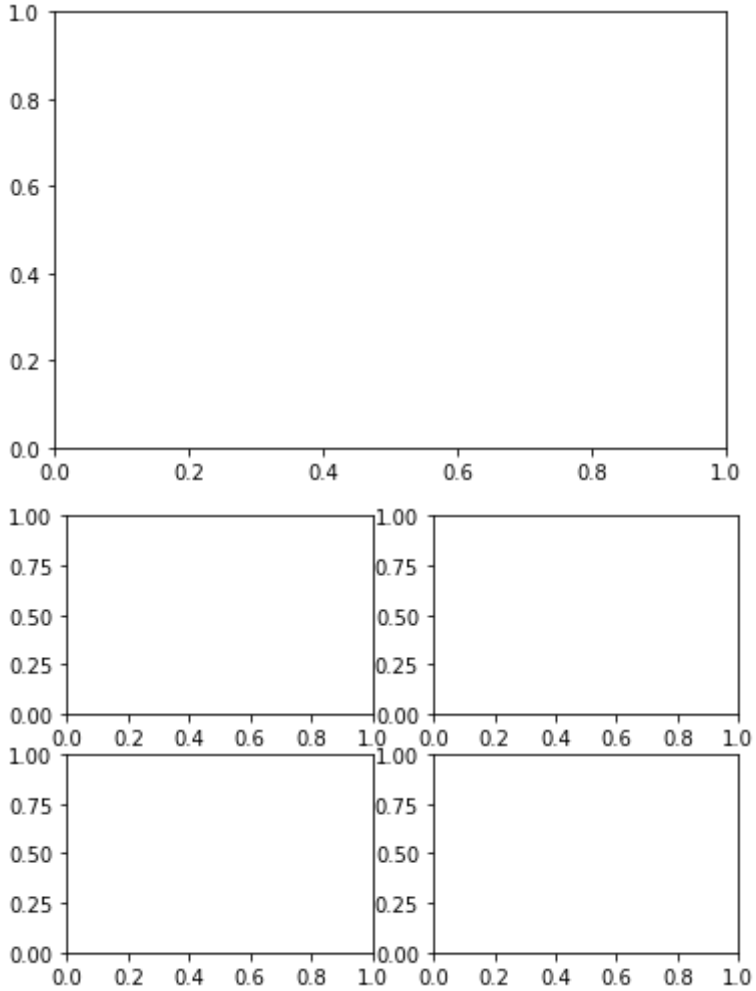
```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
```

```
fig, ax = plt.subplots()  # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]);  # Plot some data on the axes.
```




```
fig = plt.figure()  # an empty figure with no Axes
fig, ax = plt.subplots()  # a figure with a single Axes
fig, axs = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
```

```
<Figure size 432x288 with 0 Axes>
```

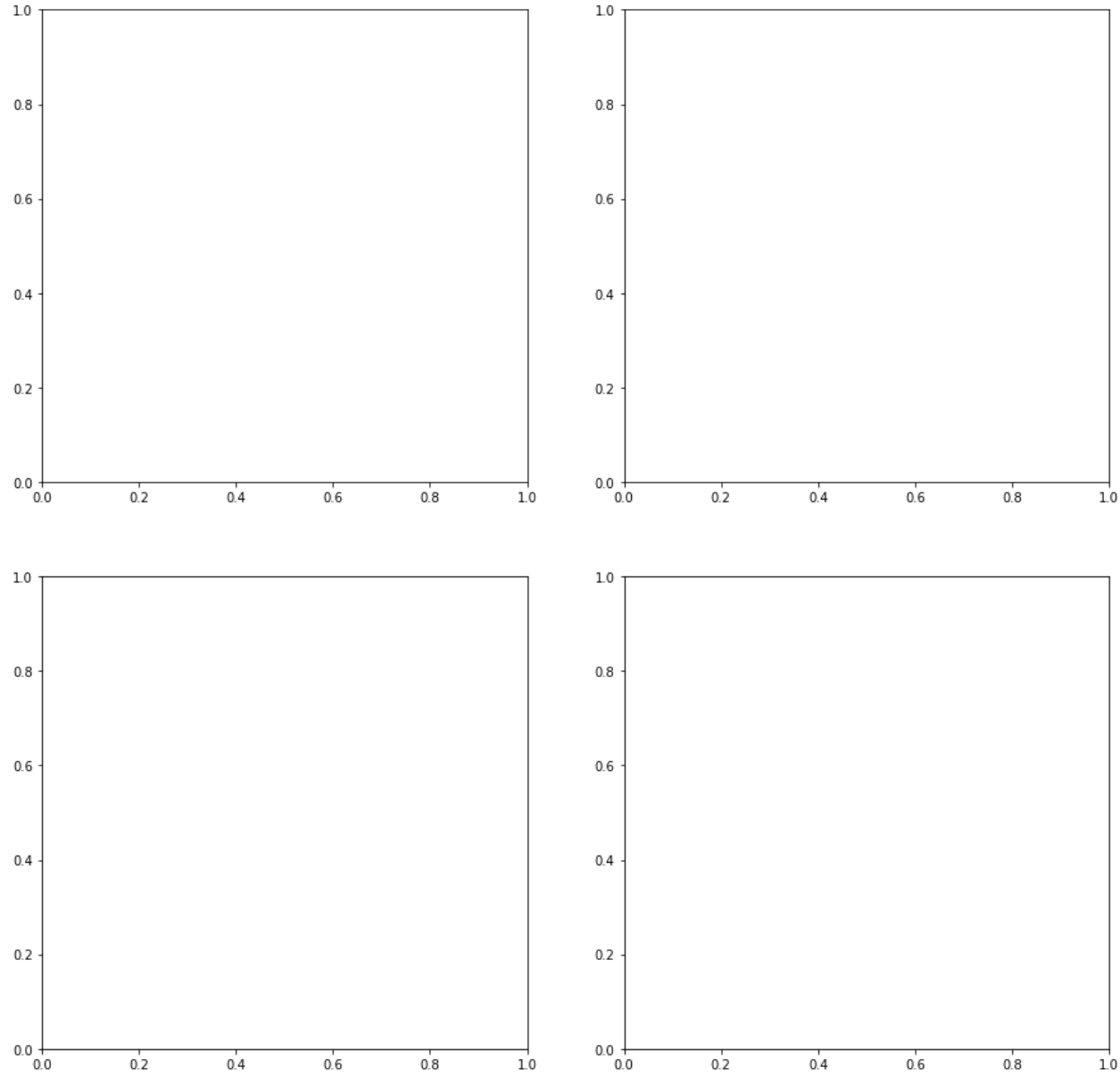


```
b = np.matrix([[1, 2], [3, 4]])
b_asarray = np.asarray(b)
```

```
np.random.seed(19680801)  # seed the random number generator.
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

f, axs = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.scatter('a', 'b', c='c', s='d', data=data)
ax.set_xlabel('entry a')
ax.set_ylabel('entry b');
```

```
x = np.linspace(0, 2, 100)  # Sample data.

# Note that even in the OO-style, we use `.pyplot.figure` to create the Figure.
f, axs = plt.subplots(2,2,figsize=(15,15))
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-label to the axes.
```

```
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend();  # Add a legend.
```

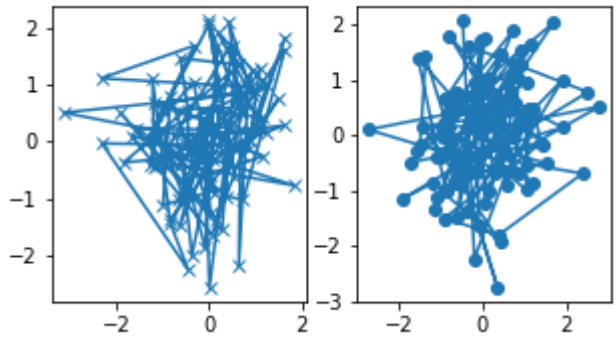



```
x = np.linspace(0, 2, 100)  # Sample data.

plt.figure(figsize=(5, 2.7), layout='constrained')
plt.plot(x, x, label='linear')  # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')
```

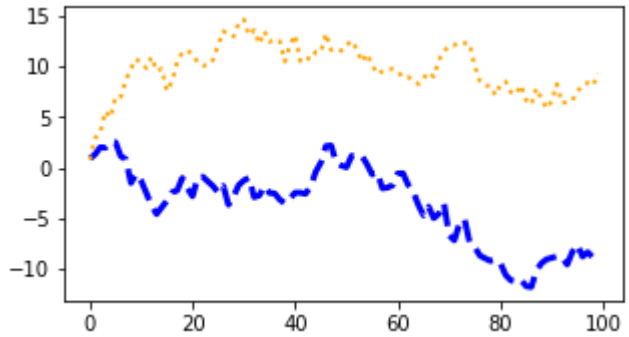
```
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend();
```

```
def my_plotter(ax, data1, data2, param_dict):
    """
    A helper function to make a graph.
    """
    out = ax.plot(data1, data2, **param_dict)
    return out
```

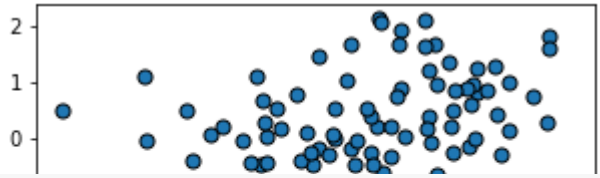
```
data1, data2, data3, data4 = np.random.randn(4, 100)  # make 4 random data sets
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(5, 2.7))
my_plotter(ax1, data1, data2, {'marker': 'x'})
my_plotter(ax2, data3, data4, {'marker': 'o'});
```



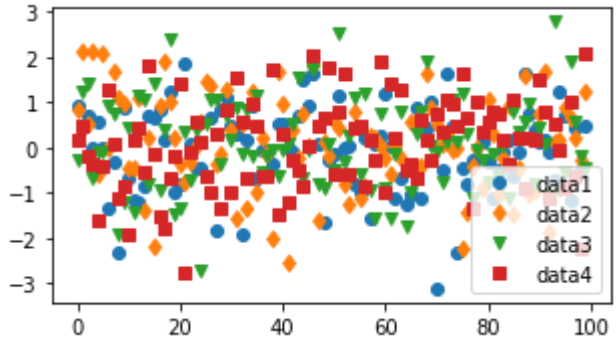

```
fig, ax = plt.subplots(figsize=(5, 2.7))
x = np.arange(len(data1))
ax.plot(x, np.cumsum(data1), color='blue', linewidth=3, linestyle='--')
l, = ax.plot(x, np.cumsum(data2), color='orange', linewidth=2)
l.set_linestyle(':');
```




```
fig, ax = plt.subplots(figsize=(5, 2.7))
ax.scatter(data1, data2, s=50, facecolor='C0', edgecolor='k');
```



```
fig, ax = plt.subplots(figsize=(5, 2.7))
ax.plot(data1, 'o', label='data1')
ax.plot(data2, 'd', label='data2')
ax.plot(data3, 'v', label='data3')
ax.plot(data4, 's', label='data4')
ax.legend();
```

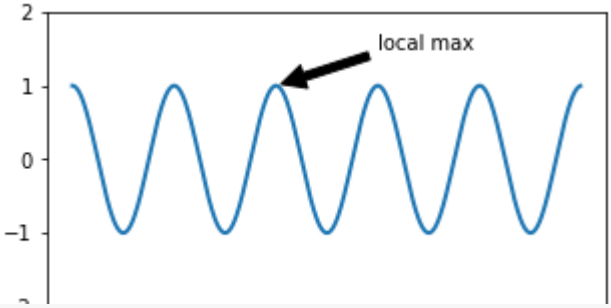



```
mu, sigma = 115, 15
x = mu + sigma * np.random.randn(10000)
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
# the histogram of the data
n, bins, patches = ax.hist(x, 50, density=1, facecolor='C0', alpha=0.75)

ax.set_xlabel('Length [cm]')
ax.set_ylabel('Probability')
ax.set_title('Aardvark lengths\n (not really)')
ax.text(75, .025, r'$\mu=115,\ \sigma=15$')
ax.axis([55, 175, 0, 0.03])
ax.grid(True);
```

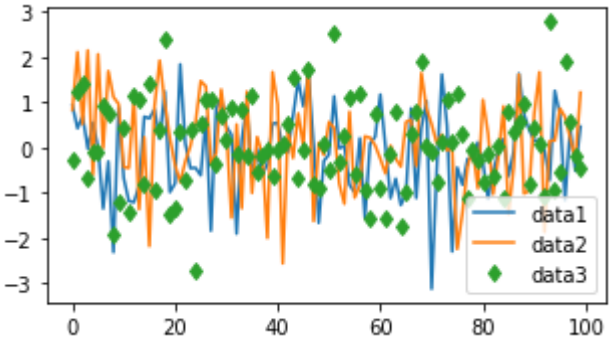
```
ax.set_title(r'$\sigma_i=15$')
```

```
Text(0.5, 1.0, '$\\sigma_i=15$')
```

```
fig, ax = plt.subplots(figsize=(5, 2.7))

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2 * np.pi * t)
line, = ax.plot(t, s, lw=2)

ax.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
            arrowprops=dict(facecolor='black', shrink=0.05))

ax.set_ylim(-2, 2);
```




```
fig, ax = plt.subplots(figsize=(5, 2.7))
ax.plot(np.arange(len(data1)), data1, label='data1')
ax.plot(np.arange(len(data2)), data2, label='data2')
ax.plot(np.arange(len(data3)), data3, 'd', label='data3')
ax.legend();
```




```
xdata = np.arange(len(data1))  # make an ordinal for this
data = 10**data1
axs[0].plot(xdata, data)

axs[1].set_yscale('log')
axs[1].plot(xdata, data);
```

```
plt.subplots(constrained_layout=True)
axs[0].plot(xdata, data1)
axs[0].set_title('Automatic ticks')

axs[1].plot(xdata, data1)
axs[1].set_xticks(np.arange(0, 100, 30), ['zero', '30', 'sixty', '90'])
axs[1].set_yticks([-1.5, 0, 1.5])  # note that we don't need to specify labels
axs[1].set_title('Manual ticks');
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-24-25a3d75c576f> in <module>()
      1
      2 plt.subplots(constrained_layout=True)
----> 3 axs[0].plot(xdata, data1)
      4 axs[0].set_title('Automatic ticks')
      5

AttributeError: 'numpy.ndarray' object has no attribute 'plot'
```

SEARCH STACK OVERFLOW


```
fig, (ax1, ax3) = plt.subplots(1, 2, figsize=(7, 2.7), layout='constrained')
l1, = ax1.plot(t, s)
ax2 = ax1.twinx()
l2, = ax2.plot(t, range(len(t)), 'C1')
ax2.legend([l1, l2], ['Sine (left)', 'Straight (right)'])

ax3.plot(t, s)
ax3.set_xlabel('Angle [rad]')
ax4 = ax3.secondary_xaxis('top', functions=(np.rad2deg, np.deg2rad))
ax4.set_xlabel('Angle [°]')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-25-3e861179f4e6> in <module>()
----> 1 fig, (ax1, ax3) = plt.subplots(1, 2, figsize=(7, 2.7),
layout='constrained')
      2 l1, = ax1.plot(t, s)
      3 ax2 = ax1.twinx()
      4 l2, = ax2.plot(t, range(len(t)), 'C1')
      5 ax2.legend([l1, l2], ['Sine (left)', 'Straight (right)'])
```

2 frames

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backend_bases.py in
new_figure_manager(cls, num, *args, **kwargs)
   3355         from matplotlib.figure import Figure
   3356         fig_cls = kwargs.pop('FigureClass', Figure)
-> 3357         fig = fig_cls(*args, **kwargs)
   3358         return cls.new_figure_manager_given_figure(num, fig)
   3359

TypeError: __init__() got an unexpected keyword argument 'layout'
```

```
X, Y = np.meshgrid(np.linspace(-3, 3, 128), np.linspace(-3, 3, 128))
Z = (1 - X/2 + X**5 + Y**3) * np.exp(-X**2 - Y**2)

fig, axs = plt.subplots(2, 2, layout='constrained')
pc = axs[0, 0].pcolormesh(X, Y, Z, vmin=-1, vmax=1, cmap='RdBu_r')
```

```
fig.colorbar(pc, ax=axs[0, 0])
axs[0, 0].set_title('pcolormesh()')

co = axs[0, 1].contourf(X, Y, Z, levels=np.linspace(-1.25, 1.25, 11))
fig.colorbar(co, ax=axs[0, 1])
axs[0, 1].set_title('contourf()')

pc = axs[1, 0].imshow(Z**2 * 100, cmap='plasma',
                      norm=mpl.colors.LogNorm(vmin=0.01, vmax=100))
fig.colorbar(pc, ax=axs[1, 0], extend='both')
axs[1, 0].set_title('imshow() with LogNorm()')

pc = axs[1, 1].scatter(data1, data2, c=data3, cmap='RdBu_r')
fig.colorbar(pc, ax=axs[1, 1], extend='both')
axs[1, 1].set_title('scatter()')
```

```
fig, axd = plt.subplot_mosaic([['upleft', 'right'],['lowleft', 'right']], layout='constrai
axd['upleft'].set_title('upleft')
axd['lowleft'].set_title('lowleft')
axd['right'].set_title('right');
```

```
import numpy as np

import pandas as pd
```

```
s = pd.Series([1, 3, 5, np.nan, 6, 8])

s
```

```
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

```
dates = pd.date_range("20130101", periods=6)

dates
```

```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')
```

```
df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list("ABCD"))

df
```

	A	B	C	D
2013-01-01	-0.145952	-0.330309	0.394935	-2.594903
2013-01-02	1.498796	0.634593	1.336970	1.650071
2013-01-03	0.553553	-0.626539	0.837869	0.308600
2013-01-04	-0.458280	0.210197	0.580140	-0.780947

```
df2 = pd.DataFrame(
    {
        "A": 1.0,
        "B": pd.Timestamp("20130102"),
        "C": pd.Series(1, index=list(range(4)), dtype="float32"),
        "D": np.array([3] * 4, dtype="int32"),
        "E": pd.Categorical(["test", "train", "test", "train"]),
        "F": "foo",
    }
)


df2
```

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

```
df2.dtypes
```

```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

```
df.head()
```

```
df.tail(3)
```

	A	B	C	D
2013-01-04	-0.458280	0.210197	0.580140	-0.780947
2013-01-05	1.467531	1.307422	0.157655	-0.336390
2013-01-06	0.561616	-1.249349	0.801693	-0.270997

```
df.index
```

```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')
```

```
df.columns
```

```
Index(['A', 'B', 'C', 'D'], dtype='object')
```

```
df.to_numpy()
```

```
array([[-0.14595228, -0.33030869,  0.39493543, -2.59490335],
       [ 1.49879631,  0.63459282,  1.33697012,  1.65007119],
       [ 0.55355335, -0.62653932,  0.8378685 ,  0.30859976],
       [-0.45828045,  0.2101973 ,  0.58014011, -0.78094743],
       [ 1.46753131,  1.30742212,  0.15765547, -0.33639023],
       [ 0.56161632, -1.249349  ,  0.80169328, -0.27099652]])
```

```
df2.to_numpy()
```

```
array([[1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'train', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'train', 'foo']],
      dtype=object)
```

```
df.describe()
```

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000

```
df.T
```

	2013-01-01	2013-01-02	2013-01-03	2013-01-04	2013-01-05	2013-01-06
A	-0.145952	1.498796	0.553553	-0.458280	1.467531	0.561616
B	-0.330309	0.634593	-0.626539	0.210197	1.307422	-1.249349
C	0.394935	1.336970	0.837869	0.580140	0.157655	0.801693
D	-2.594903	1.650071	0.308600	-0.780947	-0.336390	-0.270997

```
df.sort_index(axis=1, ascending=False)
```

	D	C	B	A
2013-01-01	-2.594903	0.394935	-0.330309	-0.145952
2013-01-02	1.650071	1.336970	0.634593	1.498796
2013-01-03	0.308600	0.837869	-0.626539	0.553553
2013-01-04	-0.780947	0.580140	0.210197	-0.458280
2013-01-05	-0.336390	0.157655	1.307422	1.467531
2013-01-06	-0.270997	0.801693	-1.249349	0.561616

```
df.sort_values(by="B")
```

	A	B	C	D
2013-01-06	0.561616	-1.249349	0.801693	-0.270997
2013-01-03	0.553553	-0.626539	0.837869	0.308600
2013-01-01	-0.145952	-0.330309	0.394935	-2.594903
2013-01-04	-0.458280	0.210197	0.580140	-0.780947
2013-01-02	1.498796	0.634593	1.336970	1.650071
2013-01-05	1.467531	1.307422	0.157655	-0.336390

```
df["A"]
```

```
2013-01-01   -0.145952
2013-01-02    1.498796
2013-01-03    0.553553
2013-01-04   -0.458280
2013-01-05    1.467531
```

```
2013-01-06    0.561616
Freq: D, Name: A, dtype: float64
```

```
df[0:3]
```

	A	B	C	D
2013-01-01	-0.145952	-0.330309	0.394935	-2.594903
2013-01-02	1.498796	0.634593	1.336970	1.650071
2013-01-03	0.553553	-0.626539	0.837869	0.308600

```
df["20130102":"20130104"]
```

	A	B	C	D
2013-01-02	1.498796	0.634593	1.336970	1.650071
2013-01-03	0.553553	-0.626539	0.837869	0.308600
2013-01-04	-0.458280	0.210197	0.580140	-0.780947

```
df.loc[dates[0]]
```

```
A   -0.145952
B   -0.330309
C    0.394935
D   -2.594903
Name: 2013-01-01 00:00:00, dtype: float64
```

```
df.loc[:, ["A", "B"]]
```

	A	B
2013-01-01	-0.145952	-0.330309
2013-01-02	1.498796	0.634593
2013-01-03	0.553553	-0.626539
2013-01-04	-0.458280	0.210197
2013-01-05	1.467531	1.307422
2013-01-06	0.561616	-1.249349

```
df.loc["20130102":"20130104", ["A", "B"]]
```

	A	B
2013-01-02	1.498796	0.634593
2013-01-03	0.553553	-0.626539
2013-01-04	-0.458280	0.210197

```
df.loc["20130102", ["A", "B"]]
df.loc[dates[0], "A"]
```

```
-0.1459522825276902
```

```
df.at[dates[0], "A"]
```

```
-0.1459522825276902
```

```
df.iloc[3]
```

```
A   -0.458280
B    0.210197
C    0.580140
D   -0.780947
Name: 2013-01-04 00:00:00, dtype: float64
```

```
df.iloc[3:5, 0:2]
```

	A	B
2013-01-04	-0.458280	0.210197
2013-01-05	1.467531	1.307422

```
df.iloc[[1, 2, 4], [0, 2]]
```

	A	C
2013-01-02	1.498796	1.336970
2013-01-03	0.553553	0.837869
2013-01-05	1.467531	0.157655

```
df.iloc[1:3, :]
```

	A	B	C	D
2013-01-02	1.498796	0.634593	1.336970	1.650071
2013-01-03	0.553553	-0.626539	0.837869	0.308600

```
df.iloc[:, 1:3]
```

	B	C
2013-01-01	-0.330309	0.394935
2013-01-02	0.634593	1.336970
2013-01-03	-0.626539	0.837869

```
df.iloc[1, 1]
```

```
0.6345928202697303
```

2013-01-06	-1.249349	0.801693

```
df[df["A"] > 0]
```

	A	B	C	D
2013-01-02	1.498796	0.634593	1.336970	1.650071
2013-01-03	0.553553	-0.626539	0.837869	0.308600
2013-01-05	1.467531	1.307422	0.157655	-0.336390
2013-01-06	0.561616	-1.249349	0.801693	-0.270997

```
df[df > 0]
```

	A	B	C	D
2013-01-01	NaN	NaN	0.394935	NaN
2013-01-02	1.498796	0.634593	1.336970	1.650071
2013-01-03	0.553553	NaN	0.837869	0.308600
2013-01-04	NaN	0.210197	0.580140	NaN
2013-01-05	1.467531	1.307422	0.157655	NaN
2013-01-06	0.561616	NaN	0.801693	NaN

```
df2 = df.copy()
```

```
df2["E"] = ["one", "one", "two", "three", "four", "three"]
```

```
df2
```

	A	B	C	D	E
2013-01-01	-0.145952	-0.330309	0.394935	-2.594903	one

```
s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date_range("20130102", periods=6))

s1
```

```
2013-01-02    1
2013-01-03    2
2013-01-04    3
2013-01-05    4
2013-01-06    5
2013-01-07    6
Freq: D, dtype: int64
```

```
df.at[dates[0], "A"] = 0
```

```
df.iat[0, 1] = 0
```

```
df.loc[:, "D"] = np.array([5] * len(df))
```

```
df
```

	A	B	C	D
2013-01-01	0.000000	0.000000	0.394935	5
2013-01-02	1.498796	0.634593	1.336970	5
2013-01-03	0.553553	-0.626539	0.837869	5
2013-01-04	-0.458280	0.210197	0.580140	5
2013-01-05	1.467531	1.307422	0.157655	5
2013-01-06	0.561616	-1.249349	0.801693	5

```
df2 = df.copy()

df2[df2 > 0] = -df2

df2
```

	A	B	C	D
2013-01-01	0.000000	0.000000	-0.394935	-5

```
df1 = df.reindex(index=dates[0:4], columns=list(df.columns) + ["E"])

df1.loc[dates[0] : dates[1], "E"] = 1

df1
```

	A	B	C	D	E
2013-01-01	0.000000	0.000000	0.394935	5	1.0
2013-01-02	1.498796	0.634593	1.336970	5	1.0
2013-01-03	0.553553	-0.626539	0.837869	5	NaN
2013-01-04	-0.458280	0.210197	0.580140	5	NaN

```
df1.dropna(how="any")
```

	A	B	C	D	E
2013-01-01	0.000000	0.000000	0.394935	5	1.0
2013-01-02	1.498796	0.634593	1.336970	5	1.0

```
df1.fillna(value=5)
```

	A	B	C	D	E
2013-01-01	0.000000	0.000000	0.394935	5	1.0
2013-01-02	1.498796	0.634593	1.336970	5	1.0
2013-01-03	0.553553	-0.626539	0.837869	5	5.0
2013-01-04	-0.458280	0.210197	0.580140	5	5.0

```
pd.isna(df1)
```

	A	B	C	D	E
2013-01-01	False	False	False	False	False
2013-01-02	False	False	False	False	False
2013-01-03	False	False	False	False	True
2013-01-04	False	False	False	False	True

```
df.mean()
```

```
A    0.603869
B    0.046054
C    0.684877
D    5.000000
dtype: float64
```

```
df.mean(1)
```

```
2013-01-01    1.348734
2013-01-02    2.117590
2013-01-03    1.441221
2013-01-04    1.333014
2013-01-05    1.983152
2013-01-06    1.278490
Freq: D, dtype: float64
```

```
s = pd.Series([1, 3, 5, np.nan, 6, 8], index=dates).shift(2)

s
```

```
2013-01-01    NaN
2013-01-02    NaN
2013-01-03    1.0
2013-01-04    3.0
2013-01-05    5.0
2013-01-06    NaN
Freq: D, dtype: float64
```

```
df.sub(s, axis="index")
```

	A	B	C	D
2013-01-01	NaN	NaN	NaN	NaN
2013-01-02	NaN	NaN	NaN	NaN
2013-01-03	-0.446447	-1.626539	-0.162131	4.0
2013-01-04	-3.458280	-2.789803	-2.419860	2.0
2013-01-05	-3.532469	-3.692578	-4.842345	0.0
2013-01-06	NaN	NaN	NaN	NaN

```
df.apply(np.cumsum)
```

	A	B	C	D
2013-01-01	0.000000	0.000000	0.394935	5

```
df.apply(lambda x: x.max() - x.min())
```

```
A    1.957077
B    2.556771
C    1.179315
D    0.000000
dtype: float64
```

2013-01-06	3.623217	0.276324	4.109263	30

```
s = pd.Series(np.random.randint(0, 7, size=10))

s
```

```
0    0
1    4
2    3
3    1
4    6
5    2
6    2
7    3
8    3
9    5
dtype: int64
```

```
s.value_counts()
```

```
3    3
2    2
0    1
4    1
1    1
6    1
5    1
dtype: int64
```

```
s = pd.Series(["A", "B", "C", "Aaba", "Baca", np.nan, "CABA", "dog", "cat"])

s.str.lower()
```

```
0       a
1       b
2       c
3    aaba
4    baca
5     NaN
6    caba
7     dog
8     cat
dtype: object
```

```
df = pd.DataFrame(np.random.randn(10, 4))

df
```

	0	1	2	3
0	0.090984	-1.318892	-0.006259	0.307443
1	0.031692	0.480549	-1.526520	0.211143
2	-0.183941	0.863865	-1.175773	0.635172
3	-0.005282	-0.376967	-0.147516	-0.511776
4	0.178680	0.416004	-2.264985	-1.470742
5	0.503691	0.296149	0.007445	0.713537
6	-1.492790	0.058591	0.352129	0.551890
7	-2.443975	-0.058262	-0.124915	0.041671
8	-0.819072	-0.144144	-1.467030	2.483946
9	-0.174613	1.347673	0.122115	-0.925616

```
pieces = [df[:3], df[3:7], df[7:]]

pd.concat(pieces)
```

	0	1	2	3
0	0.090984	-1.318892	-0.006259	0.307443
1	0.031692	0.480549	-1.526520	0.211143
2	-0.183941	0.863865	-1.175773	0.635172
3	-0.005282	-0.376967	-0.147516	-0.511776
4	0.178680	0.416004	-2.264985	-1.470742
5	0.503691	0.296149	0.007445	0.713537
6	-1.492790	0.058591	0.352129	0.551890
7	-2.443975	-0.058262	-0.124915	0.041671
8	-0.819072	-0.144144	-1.467030	2.483946
9	-0.174613	1.347673	0.122115	-0.925616

```
left = pd.DataFrame({"key": ["foo", "foo"], "lval": [1, 2]})

right = pd.DataFrame({"key": ["foo", "foo"], "rval": [4, 5]})

left
```

	key	lval
0	foo	1

```
right
```

	key	rval
0	foo	4
1	foo	5

```
pd.merge(left, right, on="key")
```

	key	lval	rval
0	foo	1	4
1	foo	1	5
2	foo	2	4
3	foo	2	5

```
left = pd.DataFrame({"key": ["foo", "bar"], "lval": [1, 2]})

right = pd.DataFrame({"key": ["foo", "bar"], "rval": [4, 5]})

left
```

	key	lval
0	foo	1
1	bar	2

```
right
```

	key	rval
0	foo	4
1	bar	5

```
pd.merge(left, right, on="key")
```

key lval rval

```
df = pd.DataFrame(
    {
        "A": ["foo", "bar", "foo", "bar", "foo", "bar", "foo", "foo"],
        "B": ["one", "one", "two", "three", "two", "two", "one", "three"],
        "C": np.random.randn(8),
        "D": np.random.randn(8),
    }
)


df
```

	A	B	C	D
0	foo	one	1.909345	-2.128007
1	bar	one	0.151867	-1.347114
2	foo	two	-1.145397	2.040462
3	bar	three	0.966964	0.102040
4	foo	two	-1.209227	-0.806181
5	bar	two	-1.397139	-0.567552
6	foo	one	-0.924547	1.414512
7	foo	three	0.322822	-0.300061

```
df.groupby("A").sum()
```

	C	D
A		
bar	-0.278308	-1.812626
foo	-1.047004	0.220725

```
df.groupby(["A", "B"]).sum()
```

		C	D

```
tuples = list(
    zip(
        *[
            ["bar", "bar", "baz", "baz", "foo", "foo", "qux", "qux"],
            ["one", "two", "one", "two", "one", "two", "one", "two"],
        ]
    )
)


index = pd.MultiIndex.from_tuples(tuples, names=["first", "second"])

df = pd.DataFrame(np.random.randn(8, 2), index=index, columns=["A", "B"])

df2 = df[:4]

df2
```

		A	B
first	**second**		
bar	**one**	-0.541802	0.044443
	two	-0.086991	0.116709
baz	**one**	0.046315	-1.336985
	two	0.130910	-1.540559

```
stacked = df2.stack()

stacked
```

```
first  second
bar    one     A   -0.541802
               B    0.044443
       two     A   -0.086991
               B    0.116709
baz    one     A    0.046315
               B   -1.336985
       two     A    0.130910
               B   -1.540559
dtype: float64
```

```
stacked.unstack()
```

		A	B
first	second		
bar	one	-0.541802	0.044443

```
stacked.unstack(1)
```

	second	one	two
first			
bar	A	-0.541802	-0.086991
	B	0.044443	0.116709
baz	A	0.046315	0.130910
	B	-1.336985	-1.540559

```
stacked.unstack(0)
```

	first	bar	baz
second			
one	A	-0.541802	0.046315
	B	0.044443	-1.336985
two	A	-0.086991	0.130910
	B	0.116709	-1.540559

```
df = pd.DataFrame(
    {
        "A": ["one", "one", "two", "three"] * 3,
        "B": ["A", "B", "C"] * 4,
        "C": ["foo", "foo", "foo", "bar", "bar", "bar"] * 2,
        "D": np.random.randn(12),
        "E": np.random.randn(12),
    }
)


df
```

	A	B	C	D	E
0	one	A	foo	0.954564	-0.702732
1	one	B	foo	0.113960	-0.849759
2	two	C	foo	0.541079	0.274659
3	three	A	bar	0.985804	0.746256
4	one	B	bar	0.765961	0.466429
5	one	C	bar	0.776110	-0.305332
6	two	A	foo	-2.604128	-1.448413
7	three	B	foo	0.545701	0.677449

```
pd.pivot_table(df, values="D", index=["A", "B"], columns=["C"])
```

	C	bar	foo
A	**B**		
one	**A**	0.143559	0.954564
	B	0.765961	0.113960
	C	0.776110	1.418905
three	**A**	0.985804	NaN
	B	NaN	0.545701
	C	0.773642	NaN
two	**A**	NaN	-2.604128
	B	0.496993	NaN
	C	NaN	0.541079

```
rng = pd.date_range("1/1/2012", periods=100, freq="S")

ts = pd.Series(np.random.randint(0, 500, len(rng)), index=rng)

ts.resample("5Min").sum()
```

```
2012-01-01    24162
Freq: 5T, dtype: int64
```

```
rng = pd.date_range("3/6/2012 00:00", periods=5, freq="D")

ts = pd.Series(np.random.randn(len(rng)), rng)

ts
```

```
2012-03-06   -2.689825
2012-03-07   -0.087022
```

```
2012-03-08    0.194207
2012-03-09   -1.152934
2012-03-10    1.091162
Freq: D, dtype: float64
```

```
ts_utc = ts.tz_localize("UTC")

ts_utc
```

```
2012-03-06 00:00:00+00:00   -2.689825
2012-03-07 00:00:00+00:00   -0.087022
2012-03-08 00:00:00+00:00    0.194207
2012-03-09 00:00:00+00:00   -1.152934
2012-03-10 00:00:00+00:00    1.091162
Freq: D, dtype: float64
```

```
ts_utc.tz_convert("US/Eastern")
```

```
2012-03-05 19:00:00-05:00   -2.689825
2012-03-06 19:00:00-05:00   -0.087022
2012-03-07 19:00:00-05:00    0.194207
2012-03-08 19:00:00-05:00   -1.152934
2012-03-09 19:00:00-05:00    1.091162
Freq: D, dtype: float64
```

```
rng = pd.date_range("1/1/2012", periods=5, freq="M")

ts = pd.Series(np.random.randn(len(rng)), index=rng)

ts
```

```
2012-01-31   -0.124430
2012-02-29   -0.700134
2012-03-31    0.086870
2012-04-30    0.648851
2012-05-31    1.506559
Freq: M, dtype: float64
```

```
ps = ts.to_period()

ps
```

```
2012-01   -0.124430
2012-02   -0.700134
2012-03    0.086870
2012-04    0.648851
2012-05    1.506559
Freq: M, dtype: float64
```

```
ps.to_timestamp()
```

```
2012-01-01   -0.124430
2012-02-01   -0.700134
```

```
2012-03-01    0.086870
2012-04-01    0.648851
2012-05-01    1.506559
Freq: MS, dtype: float64
```

```
prng = pd.period_range("1990Q1", "2000Q4", freq="Q-NOV")

ts = pd.Series(np.random.randn(len(prng)), prng)

ts.index = (prng.asfreq("M", "e") + 1).asfreq("H", "s") + 9

ts.head()
```

```
1990-03-01 09:00   -1.309451
1990-06-01 09:00    1.170297
1990-09-01 09:00    1.025901
1990-12-01 09:00   -0.071088
1991-03-01 09:00    0.739350
Freq: H, dtype: float64
```

```
df = pd.DataFrame(
    {"id": [1, 2, 3, 4, 5, 6], "raw_grade": ["a", "b", "b", "a", "a", "e"]}
)
```

```
df["grade"] = df["raw_grade"].astype("category")

df["grade"]
```

```
0    a
1    b
2    b
3    a
4    a
5    e
Name: grade, dtype: category
Categories (3, object): ['a', 'b', 'e']
```

```
df["grade"].cat.categories = ["very good", "good", "very bad"]
```

```
df["grade"] = df["grade"].cat.set_categories(
    ["very bad", "bad", "medium", "good", "very good"]
)


df["grade"]
```

```
0    very good
1         good
2         good
3    very good
4    very good
5     very bad
```

```
Name: grade, dtype: category
Categories (5, object): ['very bad', 'bad', 'medium', 'good', 'very good']
```

```
df.sort_values(by="grade")
```

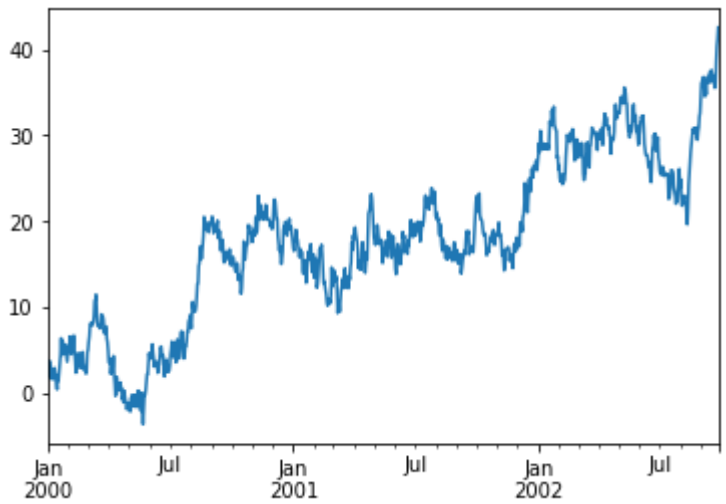
	id	raw_grade	grade
5	6	e	very bad
1	2	b	good
2	3	b	good
0	1	a	very good
3	4	a	very good
4	5	a	very good

```
df.groupby("grade").size()
```

```
grade
very bad     1
bad          0
medium       0
good         2
very good    3
dtype: int64
```

```
import matplotlib.pyplot as plt

plt.close("all")
```

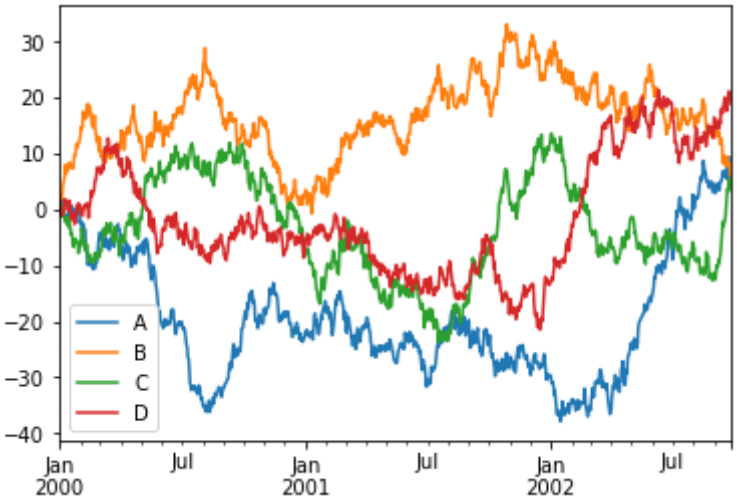
```
ts = pd.Series(np.random.randn(1000), index=pd.date_range("1/1/2000", periods=1000))

ts = ts.cumsum()

ts.plot();
```



```
plt.show();
```

```
df = pd.DataFrame(
    np.random.randn(1000, 4), index=ts.index, columns=["A", "B", "C", "D"]
)


df = df.cumsum()

plt.figure();

df.plot();

plt.legend(loc='best');
```

```
<Figure size 432x288 with 0 Axes>
```




```
df.to_csv("foo.csv")
```

```
pd.read_csv("foo.csv")
```

	Unnamed: 0	A	B	C	D
0	2000-01-01	2.212552	2.253718	0.131432	0.180574
1	2000-01-02	2.594616	2.571183	0.086387	0.032442

```
df.to_hdf("foo.h5", "df")
```

3	2000-01-04	1.724739	1.752178	0.817960	-1.139637

```
pd.read_hdf("foo.h5", "df")
```

	A	B	C	D
2000-01-01	2.212552	2.253718	0.131432	0.180574
2000-01-02	2.594616	2.571183	0.086387	0.032442
2000-01-03	1.696580	1.961364	0.244322	-1.116857
2000-01-04	1.724739	1.752178	0.817960	-1.139637
2000-01-05	-0.926906	2.612831	-1.026550	-1.002175
...	...	...	...	...
2002-09-22	7.350385	7.872723	3.198046	19.874160
2002-09-23	5.582936	8.248891	4.608959	21.104611
2002-09-24	6.168999	7.544143	5.499245	19.967234
2002-09-25	6.702700	6.011274	3.709312	19.919386
2002-09-26	9.323200	7.532263	4.073594	18.817277

1000 rows × 4 columns

```
df.to_excel("foo.xlsx", sheet_name="Sheet1")
```

```
pd.read_excel("foo.xlsx", "Sheet1", index_col=None, na_values=["NA"])
```

	Unnamed: 0	A	B	C	D
0	2000-01-01	2.212552	2.253718	0.131432	0.180574
1	2000-01-02	2.594616	2.571183	0.086387	0.032442
2	2000-01-03	1.696580	1.961364	0.244322	-1.116857
3	2000-01-04	1.724739	1.752178	0.817960	-1.139637
4	2000-01-05	-0.926906	2.612831	-1.026550	-1.002175
...	...	...	...	...	...
995	2002-09-22	7.350385	7.872723	3.198046	19.874160
996	2002-09-23	5.582936	8.248891	4.608959	21.104611
997	2002-09-24	6.168999	7.544143	5.499245	19.967234
998	2002-09-25	6.702700	6.011274	3.709312	19.919386
999	2002-09-26	9.323200	7.532263	4.073594	18.817277

1000 rows × 5 columns