Chapter 2 Lab

```
from notebookfuncs import *
```

```
print("Fit a model with ", 11, " variables")
```

```
Fit a model with  11  variables
```

```
print?
```

```
Signature: print(*args, sep=' ', end='\n', file=None, flush=False)
Docstring:
Prints the values to a stream, or to sys.stdout by default.

sep
  string inserted between values, default a space.
end
  string appended after the last value, default a newline.
file
  a file-like object (stream); defaults to the current sys.stdout.
flush
  whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

```
3 + 5
```

```
8
```

```
"hello, " + " " + "world!"
```

```
'hello,  world!'
```

```
x = [3, 4, 5]
x
```

```
[3, 4, 5]
```

```
y = [4, 9, 7]
x + y
```

```
[3, 4, 5, 4, 9, 7]
```

```
import numpy as np
```

```
x = np.array([3, 4, 5])
y = np.array([4, 9, 7])
x + y
```

```
array([ 7, 13, 12])
```

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

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

```
x.ndim
```

```
2
```

```
x.dtype
```

```
dtype('int64')
```

```
x = np.array([[1, 2], [3.0, 4]])
x.dtype
```

```
dtype('float64')
```

```
np.array([[1, 2], [3, 4]], float).dtype
```

```
dtype('float64')
```

```
x.shape
```

```
(2, 2)
```

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

```
10
```

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

```
10
```

```
x = np.array([1, 2, 3, 4, 5, 6])
print("Beginning x:\n ", x)
x_reshape = x.reshape(2, 3)
print("reshaped x:\n", x_reshape)
```

```
Beginning x:
  [1 2 3 4 5 6]
reshaped x:
 [[1 2 3]
 [4 5 6]]
```

```
x_reshape[0, 0]
```

```
1
```

```
x_reshape[1, 2]
```

```
6
```

```
print("x before we modify x_reshape:\n", x)
print("x_reshape before we modify x_reshape:\n", x_reshape)
x_reshape[0, 0] = 5
print("x_reshape after we modify its top left element:\n", x_reshape)
print("x after we modify top left element of x_reshape:\n", x)
```

```
x before we modify x_reshape:
 [1 2 3 4 5 6]
x_reshape before we modify x_reshape:
 [[1 2 3]
 [4 5 6]]
x_reshape after we modify its top left element:
 [[5 2 3]
 [4 5 6]]
x after we modify top left element of x_reshape:
 [5 2 3 4 5 6]
```

```
my_tuple = (1, 2, 3)
# type error
# my_tuple[0] = 10
```

```
(1, 2, 3)
```

```
x_reshape.shape, x_reshape.ndim, x_reshape.T
```

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

```
np.sqrt(x)
```

```
array([2.23606798, 1.41421356, 1.73205081, 2.        , 2.23606798,
       2.44948974])
```

```
x**2
```

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

```
x**0.5
```

```
array([2.23606798, 1.41421356, 1.73205081, 2.        , 2.23606798,
       2.44948974])
```

```
np.random.normal?
```

```
Docstring:
normal(loc=0.0, scale=1.0, size=None)

Draw random samples from a normal (Gaussian) distribution.

The probability density function of the normal distribution, first
derived by De Moivre and 200 years later by both Gauss and Laplace
independently [2]_, is often called the bell curve because of
its characteristic shape (see the example below).

The normal distributions occurs often in nature.  For example, it
describes the commonly occurring distribution of samples influenced
by a large number of tiny, random disturbances, each with its own
unique distribution [2]_.

.. note::
    New code should use the `~numpy.random.Generator.normal`
    method of a `~numpy.random.Generator` instance instead;
    please see the :ref:`random-quick-start`.

Parameters
----------
loc : float or array_like of floats
    Mean ("centre") of the distribution.
scale : float or array_like of floats
    Standard deviation (spread or "width") of the distribution. Must be
    non-negative.
size : int or tuple of ints, optional
    Output shape.  If the given shape is, e.g., ``(m, n, k)``, then
    ``m * n * k`` samples are drawn.  If size is ``None`` (default),
    a single value is returned if ``loc`` and ``scale`` are both scalars.
    Otherwise, ``np.broadcast(loc, scale).size`` samples are drawn.

Returns
```

```
-------
out : ndarray or scalar
    Drawn samples from the parameterized normal distribution.

See Also
--------
scipy.stats.norm : probability density function, distribution or
    cumulative density function, etc.
random.Generator.normal: which should be used for new code.

Notes
-----
The probability density for the Gaussian distribution is

.. math:: p(x) = \frac{1}{\sqrt{ 2 \pi \sigma^2 }}
                 e^{ - \frac{ (x - \mu)^2 } {2 \sigma^2} },

where :math:`\mu` is the mean and :math:`\sigma` the standard
deviation. The square of the standard deviation, :math:`\sigma^2`,
is called the variance.

The function has its peak at the mean, and its "spread" increases with
the standard deviation (the function reaches 0.607 times its maximum at
:math:`x + \sigma` and :math:`x - \sigma` [2]_).  This implies that
normal is more likely to return samples lying close to the mean, rather
than those far away.

References
----------
.. [1] Wikipedia, "Normal distribution",
       https://en.wikipedia.org/wiki/Normal_distribution
.. [2] P. R. Peebles Jr., "Central Limit Theorem" in "Probability,
       Random Variables and Random Signal Principles", 4th ed., 2001,
       pp. 51, 51, 125.

Examples
--------
Draw samples from the distribution:

>>> mu, sigma = 0, 0.1 # mean and standard deviation
>>> s = np.random.normal(mu, sigma, 1000)

Verify the mean and the variance:
```

```
>>> abs(mu - np.mean(s))
0.0  # may vary

>>> abs(sigma - np.std(s, ddof=1))
0.1  # may vary

Display the histogram of the samples, along with
the probability density function:

>>> import matplotlib.pyplot as plt
>>> count, bins, ignored = plt.hist(s, 30, density=True)
>>> plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *
...                np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
...          linewidth=2, color='r')
>>> plt.show()

Two-by-four array of samples from the normal distribution with
mean 3 and standard deviation 2.5:

>>> np.random.normal(3, 2.5, size=(2, 4))
array([[-4.49401501,  4.00950034, -1.81814867,  7.29718677],   # random
       [ 0.39924804,  4.68456316,  4.99394529,  4.84057254]])  # random
Type:      builtin_function_or_method
```

```
x = np.random.normal(size=50)
x
```

```
array([ 0.08041773, -0.03026667, -0.75024594,  0.55309515, -0.05462602,
       -0.45598797, -1.31841838,  1.7935072 , -1.51596114, -0.50126252,
       -1.30425185, -0.03340125, -1.35537998,  1.69153596,  1.42804145,
        0.07790256,  0.1058116 ,  1.10016254,  1.40118536, -0.17548991,
       -0.87869758,  0.69268256,  0.74976422, -1.1463767 ,  0.13973715,
        0.71530199, -1.27774347,  0.0999863 , -0.9482597 ,  0.60169475,
       -0.11348222, -0.68666708, -1.29890979,  0.30813716,  1.75730809,
        0.43512618, -1.52411143,  0.71597162, -1.1332048 , -1.3348259 ,
        0.6642063 , -0.12842332, -0.08288676, -0.04559009, -1.76561078,
        0.19567015, -0.46276504,  1.44365583, -0.62148203,  0.46484459])
```

```
y = x + np.random.normal(loc=50, scale=1, size=50)
```

```
array([50.0979533 , 49.26404285, 49.71799704, 49.41562921, 49.15535843,
```

```
       50.0941145 , 49.22353957, 51.91211385, 46.78992371, 49.35636789,
       47.89630029, 47.81683737, 50.34305891, 51.92732736, 52.42157283,
       49.59542562, 49.37032237, 51.34191185, 52.21067084, 49.4501618 ,
       49.25157637, 52.41226159, 49.73448335, 52.02641758, 48.32782489,
       50.86437636, 48.34873085, 49.79934532, 51.17431798, 50.12031479,
       50.35817915, 50.00245766, 46.69029529, 50.42398331, 50.94783997,
       50.09013429, 47.71924593, 51.71021843, 49.15006387, 47.22155613,
       49.64516452, 48.90729793, 50.82616726, 50.31144179, 46.30683944,
       50.2992735 , 50.07729159, 52.25433213, 49.15060364, 49.82369515])
```

```
np.corrcoef(x, y)
```

```
array([[1.        , 0.70414113],
       [0.70414113, 1.        ]])
```

```
print(np.random.normal(scale=5, size=2))
print(np.random.normal(scale=5, size=2))
```

```
[8.32626704 0.60313317]
[ 2.64990968 -1.09541093]
```

```
rng = np.random.default_rng(1303)
print(rng.normal(scale=5, size=2))
rng = np.random.default_rng(1303)
print(rng.normal(scale=5, size=2))
```

```
[ 4.09482632 -1.07485605]
[ 4.09482632 -1.07485605]
```

```
rng = np.random.default_rng(3)
y = rng.standard_normal(10)
np.mean(y), y.mean()
```

```
(-0.1126795190952861, -0.1126795190952861)
```

```
np.var(y), y.var(), np.mean((y - y.mean()) ** 2)
```

```
(2.7243406406465125, 2.7243406406465125, 2.7243406406465125)
```

```
np.sqrt(np.var(y)), np.std(y)
```

```
(1.6505576756498128, 1.6505576756498128)
```

```
np.var?
```

```
Signature:
np.var(
    a,
    axis=None,
    dtype=None,
    out=None,
    ddof=0,
    keepdims=<no value>,
    *,
    where=<no value>,
)
Call signature:  np.var(*args, **kwargs)
Type:            _ArrayFunctionDispatcher
String form:     <function var at 0x7eba602e3ec0>
File:            ~/ISLP/islpenv/lib/python3.12/site-packages/numpy/core/fromnumeric.py
Docstring:
Compute the variance along the specified axis.

Returns the variance of the array elements, a measure of the spread of a
distribution.  The variance is computed for the flattened array by
default, otherwise over the specified axis.

Parameters
----------
a : array_like
    Array containing numbers whose variance is desired.  If `a` is not an
    array, a conversion is attempted.
axis : None or int or tuple of ints, optional
    Axis or axes along which the variance is computed.  The default is to
    compute the variance of the flattened array.

    .. versionadded:: 1.7.0

    If this is a tuple of ints, a variance is performed over multiple axes,
    instead of a single axis or all the axes as before.
```

```
dtype : data-type, optional
    Type to use in computing the variance.  For arrays of integer type
    the default is `float64`; for arrays of float types it is the same as
    the array type.
out : ndarray, optional
    Alternate output array in which to place the result.  It must have
    the same shape as the expected output, but the type is cast if
    necessary.
ddof : int, optional
    "Delta Degrees of Freedom": the divisor used in the calculation is
    ``N - ddof``, where ``N`` represents the number of elements. By
    default `ddof` is zero.
keepdims : bool, optional
    If this is set to True, the axes which are reduced are left
    in the result as dimensions with size one. With this option,
    the result will broadcast correctly against the input array.

    If the default value is passed, then `keepdims` will not be
    passed through to the `var` method of sub-classes of
    `ndarray`, however any non-default value will be.  If the
    sub-class' method does not implement `keepdims` any
    exceptions will be raised.

where : array_like of bool, optional
    Elements to include in the variance. See `~numpy.ufunc.reduce` for
    details.

    .. versionadded:: 1.20.0

Returns
-------
variance : ndarray, see dtype parameter above
    If ``out=None``, returns a new array containing the variance;
    otherwise, a reference to the output array is returned.

See Also
--------
std, mean, nanmean, nanstd, nanvar
:ref:`ufuncs-output-type`

Notes
-----
The variance is the average of the squared deviations from the mean,
```

```
i.e.,  ``var = mean(x)``, where ``x = abs(a - a.mean())**2``.

The mean is typically calculated as ``x.sum() / N``, where ``N = len(x)``.
If, however, `ddof` is specified, the divisor ``N - ddof`` is used
instead.  In standard statistical practice, ``ddof=1`` provides an
unbiased estimator of the variance of a hypothetical infinite population.
``ddof=0`` provides a maximum likelihood estimate of the variance for
normally distributed variables.

Note that for complex numbers, the absolute value is taken before
squaring, so that the result is always real and nonnegative.

For floating-point input, the variance is computed using the same
precision the input has.  Depending on the input data, this can cause
the results to be inaccurate, especially for `float32` (see example
below).  Specifying a higher-accuracy accumulator using the ``dtype``
keyword can alleviate this issue.

Examples
--------
>>> a = np.array([[1, 2], [3, 4]])
>>> np.var(a)
1.25
>>> np.var(a, axis=0)
array([1.,  1.])
>>> np.var(a, axis=1)
array([0.25,  0.25])

In single precision, var() can be inaccurate:

>>> a = np.zeros((2, 512*512), dtype=np.float32)
>>> a[0, :] = 1.0
>>> a[1, :] = 0.1
>>> np.var(a)
0.20250003

Computing the variance in float64 is more accurate:

>>> np.var(a, dtype=np.float64)
0.20249999932944759 # may vary
>>> ((1-0.55)**2 + (0.1-0.55)**2)/2
0.2025
```

```
Specifying a where argument:

>>> a = np.array([[14, 8, 11, 10], [7, 9, 10, 11], [10, 15, 5, 10]])
>>> np.var(a)
6.833333333333333 # may vary
>>> np.var(a, where=[[True], [True], [False]])
4.0
Class docstring:
Class to wrap functions with checks for __array_function__ overrides.

All arguments are required, and can only be passed by position.

Parameters
----------
dispatcher : function or None
    The dispatcher function that returns a single sequence-like object
    of all arguments relevant.  It must have the same signature (except
    the default values) as the actual implementation.
    If ``None``, this is a ``like=`` dispatcher and the
    ``_ArrayFunctionDispatcher`` must be called with ``like`` as the
    first (additional and positional) argument.
implementation : function
    Function that implements the operation on NumPy arrays without
    overrides.  Arguments passed calling the ``_ArrayFunctionDispatcher``
    will be forwarded to this (and the ``dispatcher``) as if using
    ``*args, **kwargs``.

Attributes
----------
_implementation : function
    The original implementation passed in.
```

```
X = rng.standard_normal((10, 3))
X
```

```
array([[ 0.22578661, -0.35263079, -0.28128742],
       [-0.66804635, -1.05515055, -0.39080098],
       [ 0.48194539, -0.23855361,  0.9577587 ],
       [-0.19980213,  0.02425957,  1.54582085],
       [ 0.54510552, -0.50522874, -0.18283897],
       [ 0.54052513,  1.93508803, -0.26962033],
       [-0.24355868,  1.0023136 , -0.88645994],
```

```
       [-0.29172023,  0.88253897,  0.58035002],
       [ 0.0915167 ,  0.67010435, -2.82816231],
       [ 1.02130682, -0.95964476, -1.66861984]])
```

```
X.mean(axis=0)
```

```
array([ 0.15030588,  0.14030961, -0.34238602])
```

```
X.mean(axis=0)
```

```
array([ 0.15030588,  0.14030961, -0.34238602])
```

```
X.mean(0)
```

```
array([ 0.15030588,  0.14030961, -0.34238602])
```

```
X.mean(1)
```

```
array([-0.13604387, -0.70466596,  0.40038349,  0.45675943, -0.04765406,
        0.73533095, -0.04256834,  0.39038958, -0.68884708, -0.53565259])
```

```
X.mean()
```

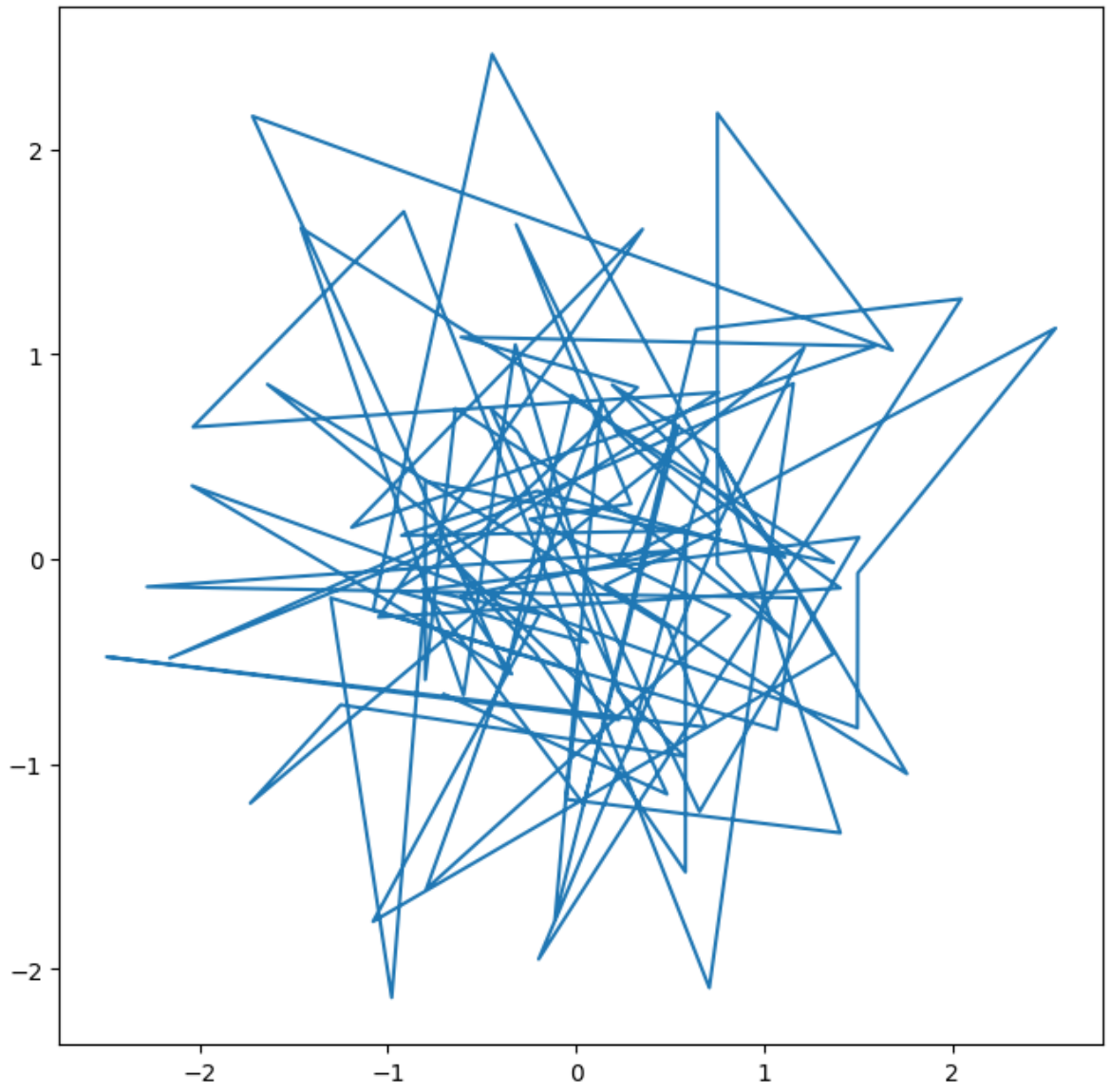
```
-0.017256845138782704
```

```
ax.plot?
```

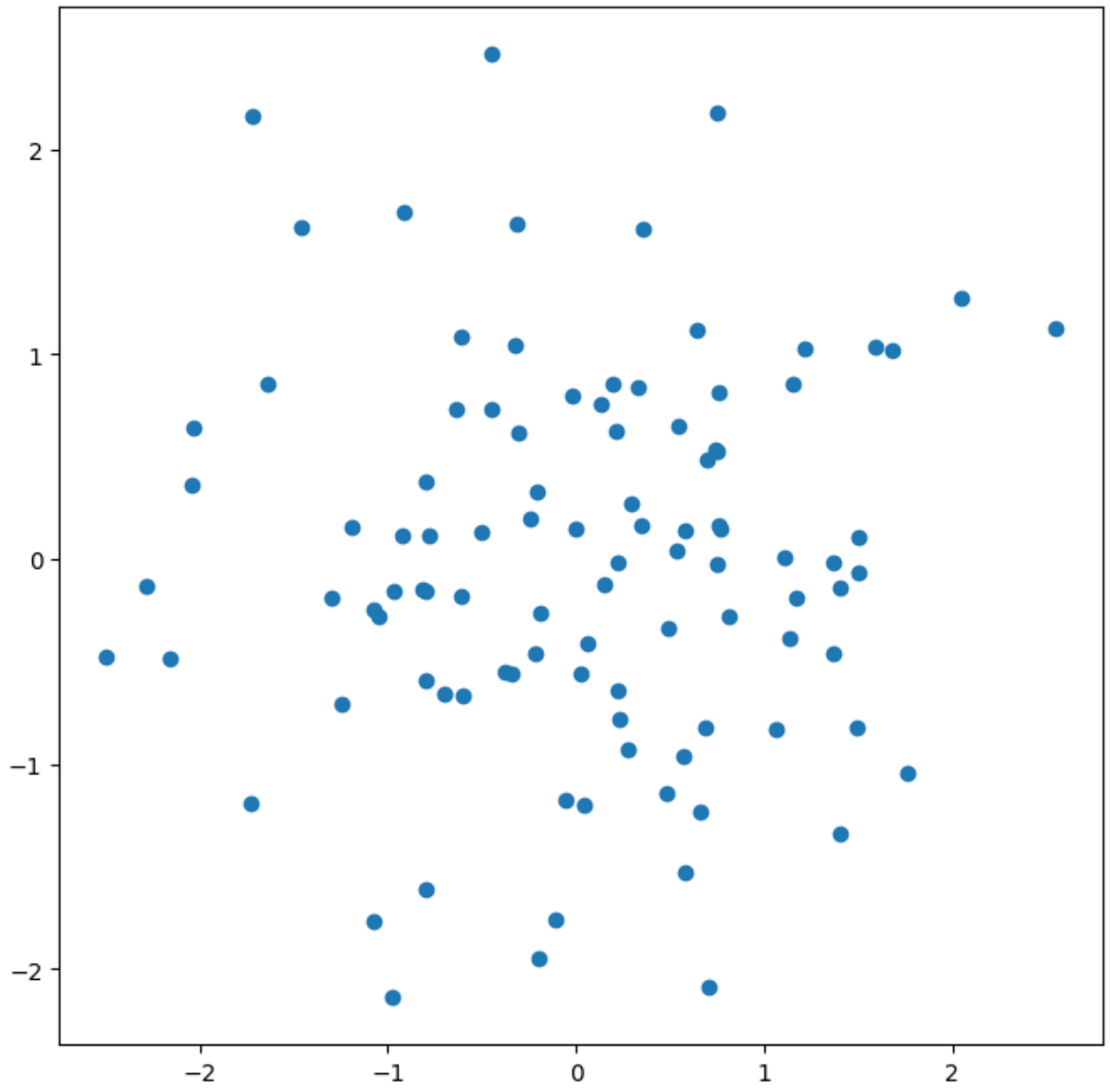
```
Object `ax.plot` not found.
```

```
from matplotlib.pyplot import subplots
```

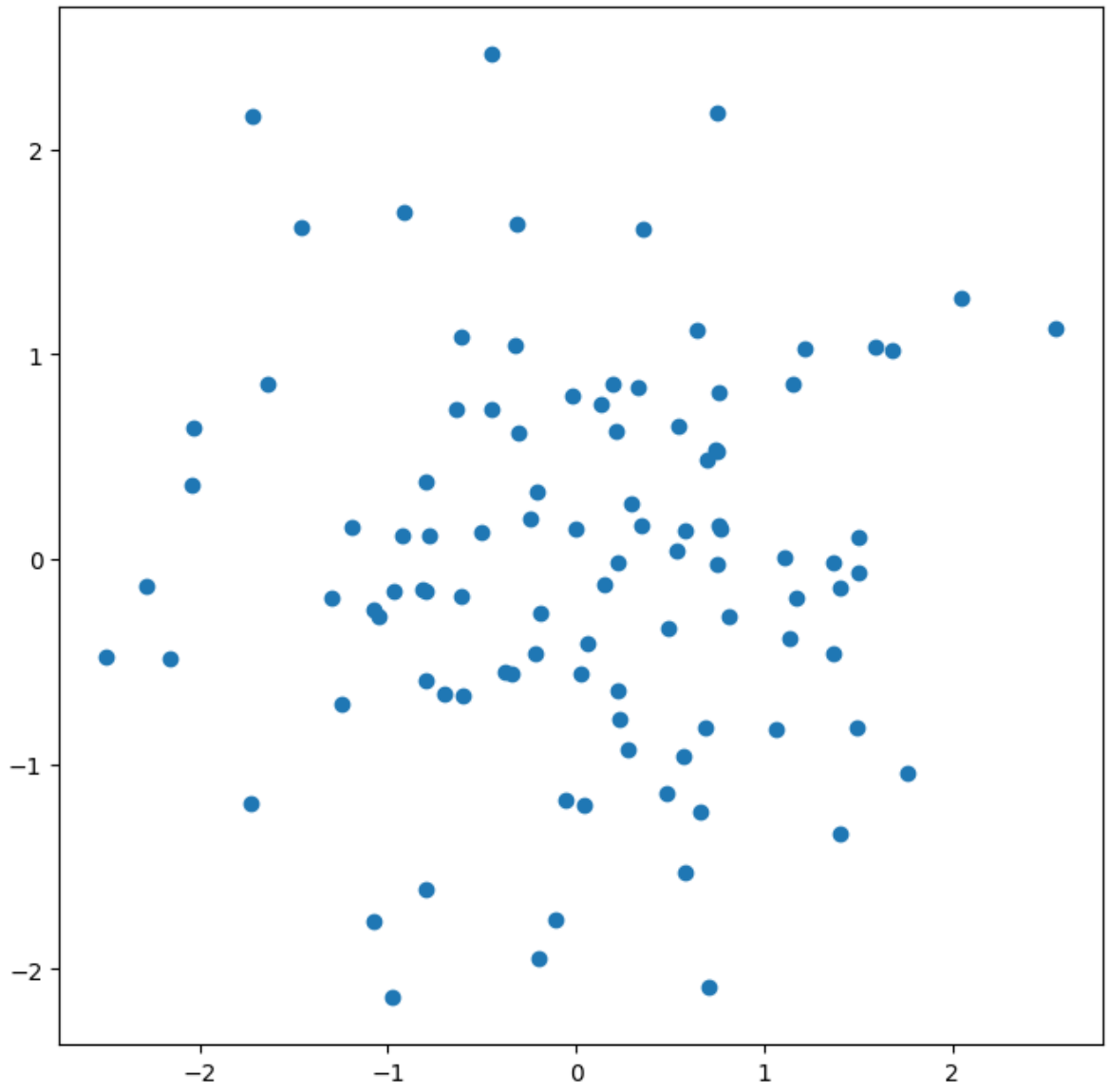
```
fig, ax = subplots(figsize=(8, 8))
x = rng.standard_normal(100)
y = rng.standard_normal(100)
ax.plot(x, y);
```




```
fig, ax = subplots(figsize=(8, 8))
ax.plot(x, y, "o");
```

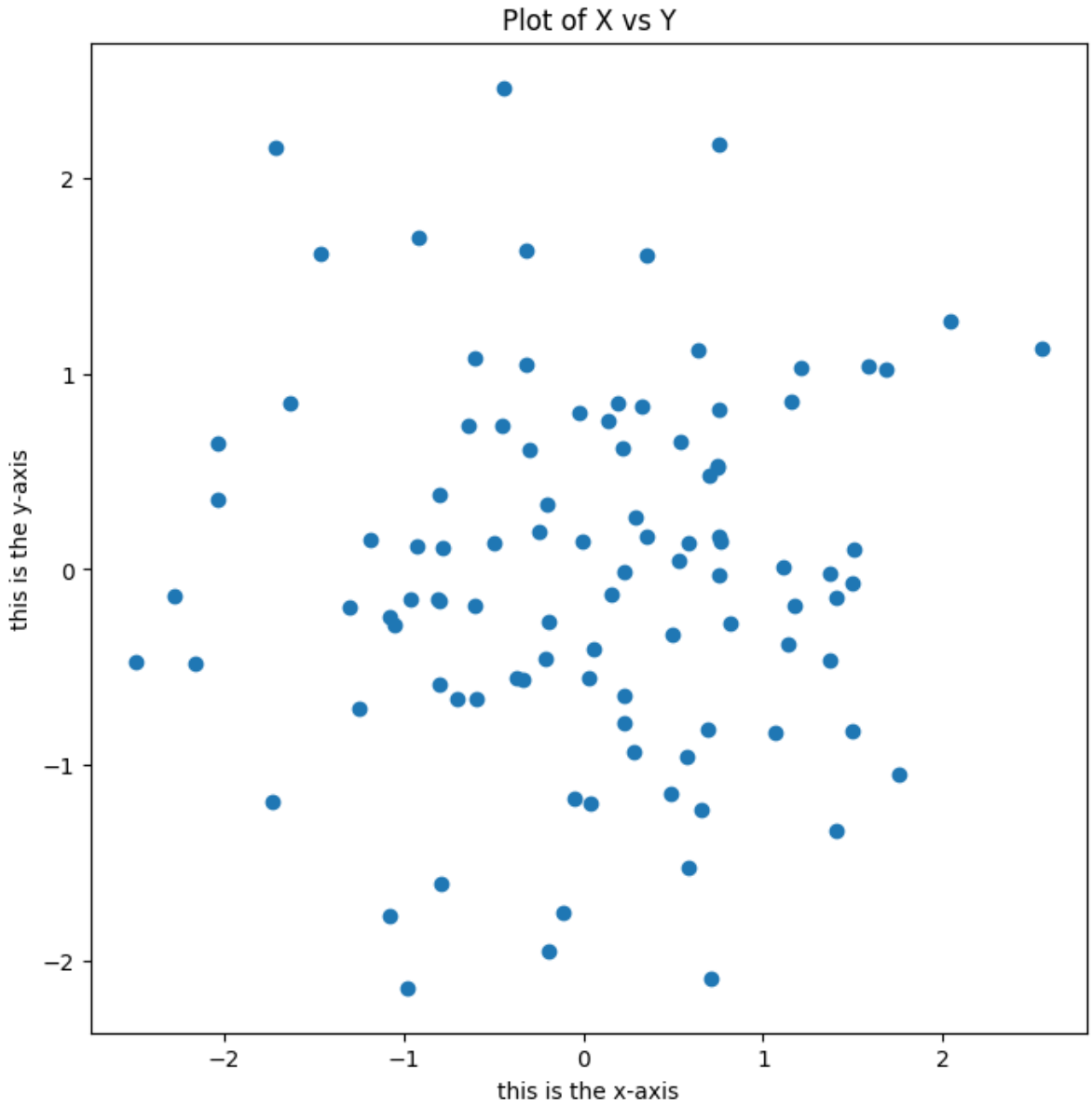



```
fig, ax = subplots(figsize=(8, 8))
ax.scatter(x, y, marker="o");
```

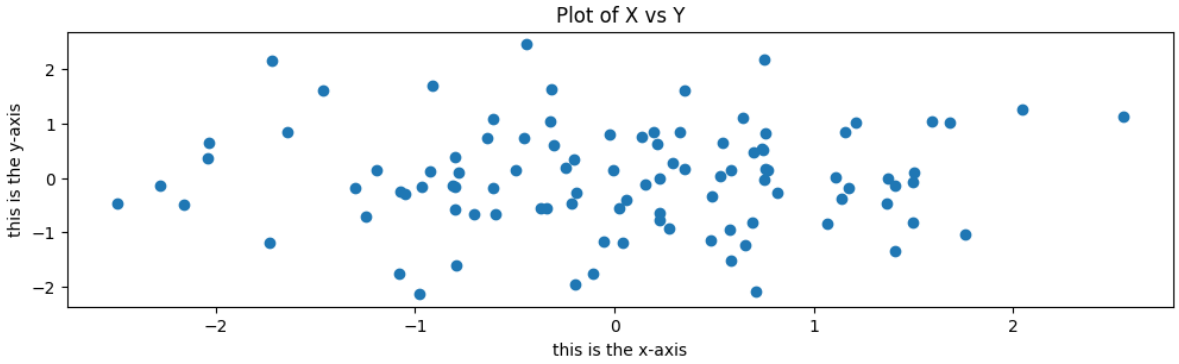



```
fig, ax = subplots(figsize=(8, 8))

ax.scatter(x, y, marker="o")
ax.set_xlabel("this is the x-axis")
ax.set_ylabel("this is the y-axis")
ax.set_title("Plot of X vs Y")
```

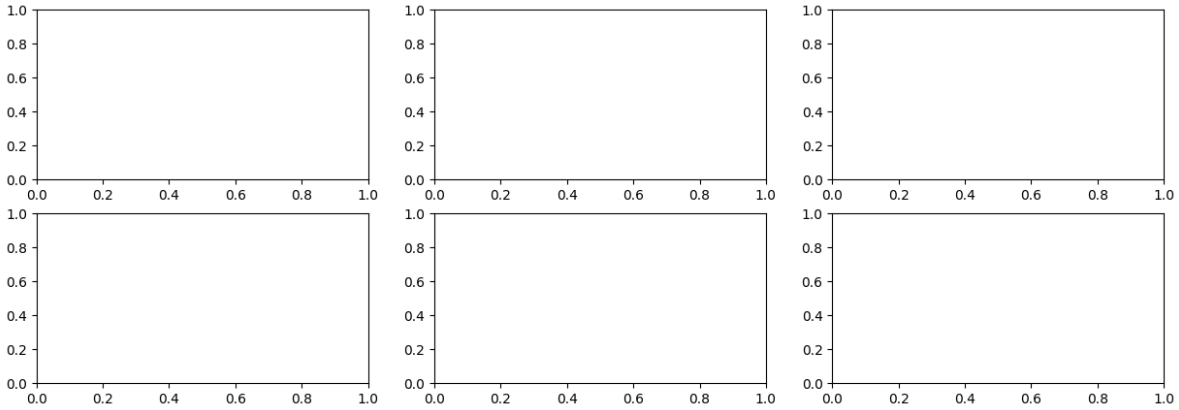
```
Text(0.5, 1.0, 'Plot of X vs Y')
```



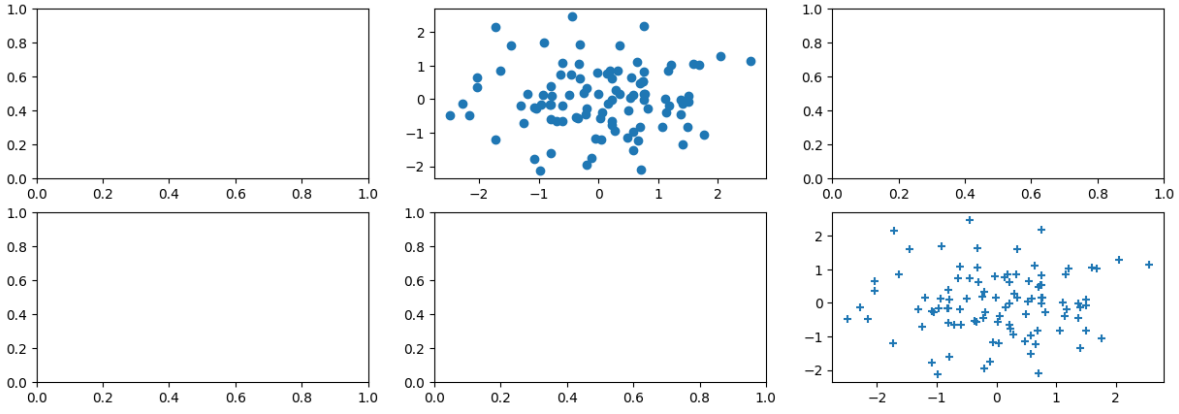

```
fig.set_size_inches(12, 3)
fig
```




```
fig, axes = subplots(nrows=2, ncols=3, figsize=(15, 5))
```

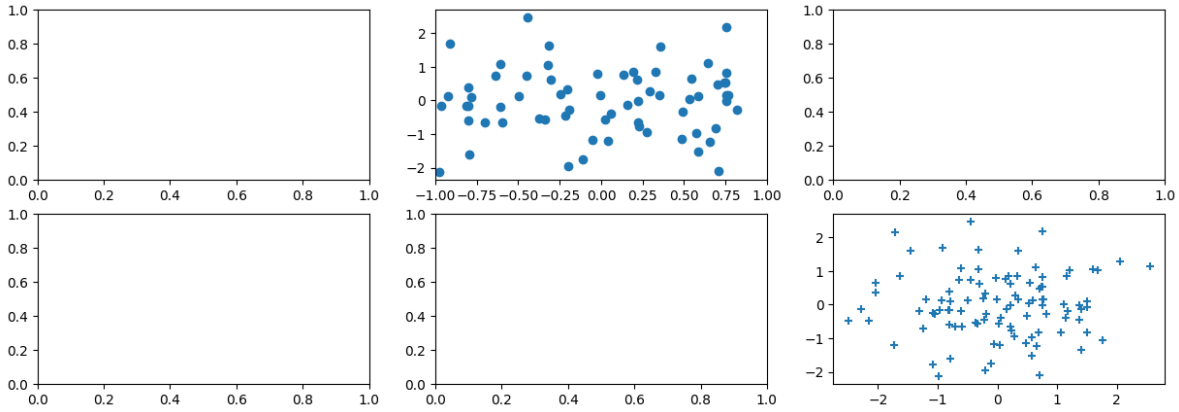


```
axes[0, 1].plot(x, y, "o")
axes[1, 2].scatter(x, y, marker="+")
fig
```



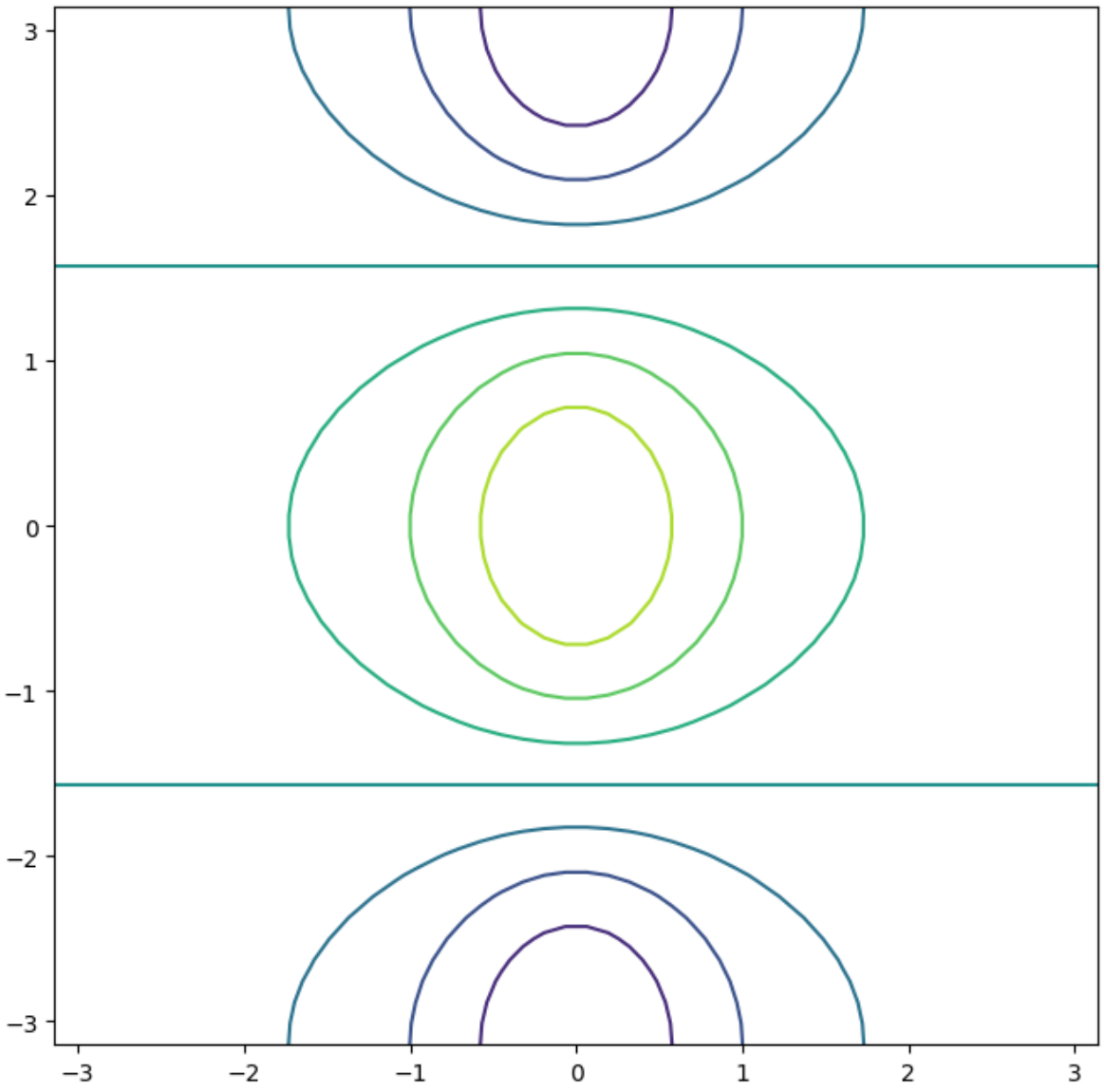
```
fig.savefig("Figure.png", dpi=400)
fig.savefig("Figure.pdf", dpi=200)
```

```
axes[0, 1].set_xlim([-1, 1])
fig.savefig("Figure_updated.jpg")
fig
```

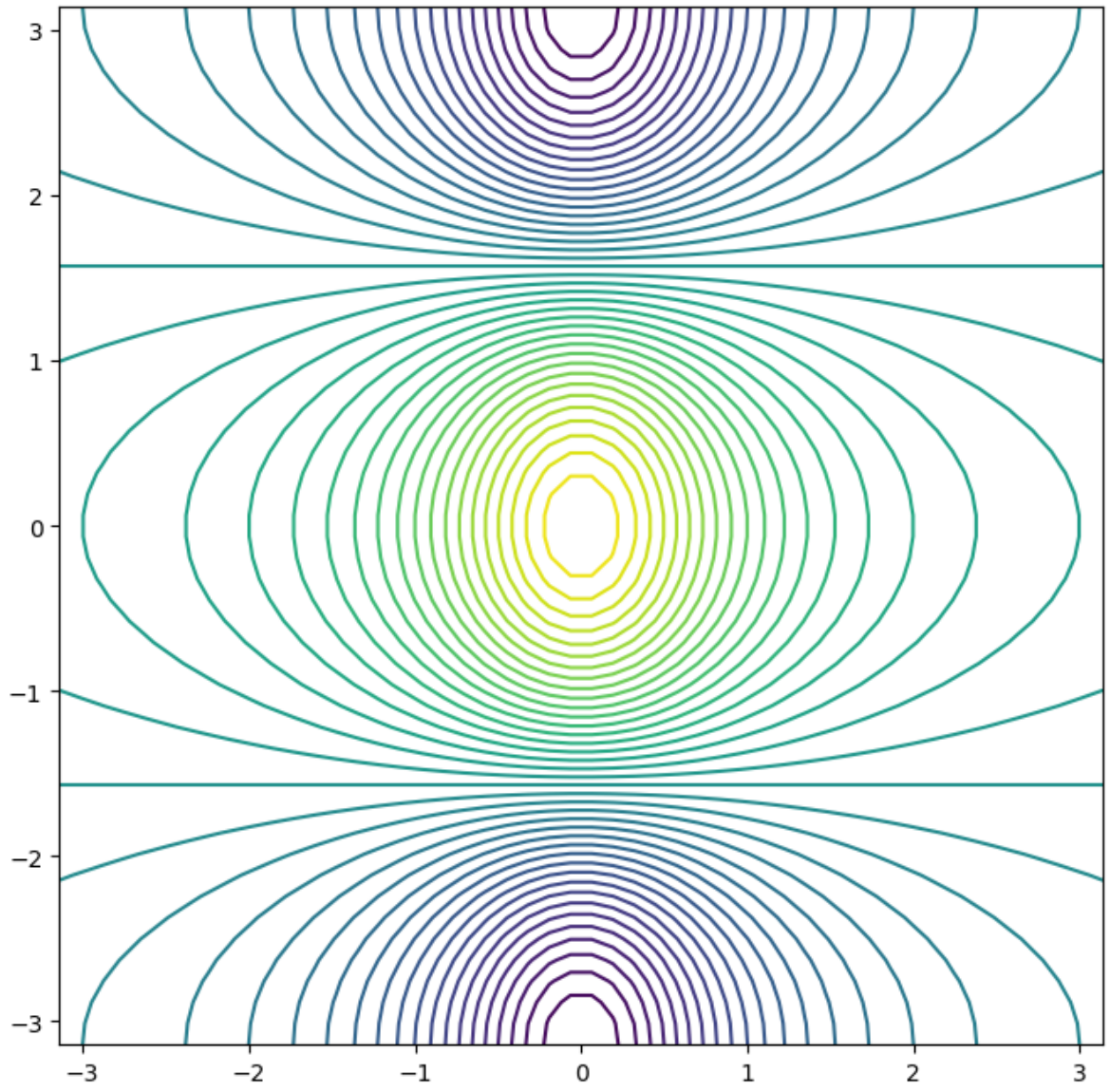


```
fig, ax = subplots(figsize=(8, 8))
x = np.linspace(-np.pi, np.pi, 50)
print(x)
y = x
f = np.multiply.outer(np.cos(y), 1 / (1 + x**2))
ax.contour(x, y, f);
```

```
[-3.14159265 -3.01336438 -2.88513611 -2.75690784 -2.62867957 -2.5004513
 -2.37222302 -2.24399475 -2.11576648 -1.98753821 -1.85930994 -1.73108167
 -1.60285339 -1.47462512 -1.34639685 -1.21816858 -1.08994031 -0.96171204
 -0.83348377 -0.70525549 -0.57702722 -0.44879895 -0.32057068 -0.19234241
 -0.06411414  0.06411414  0.19234241  0.32057068  0.44879895  0.57702722
  0.70525549  0.83348377  0.96171204  1.08994031  1.21816858  1.34639685
  1.47462512  1.60285339  1.73108167  1.85930994  1.98753821  2.11576648
  2.24399475  2.37222302  2.5004513   2.62867957  2.75690784  2.88513611
  3.01336438  3.14159265]
```

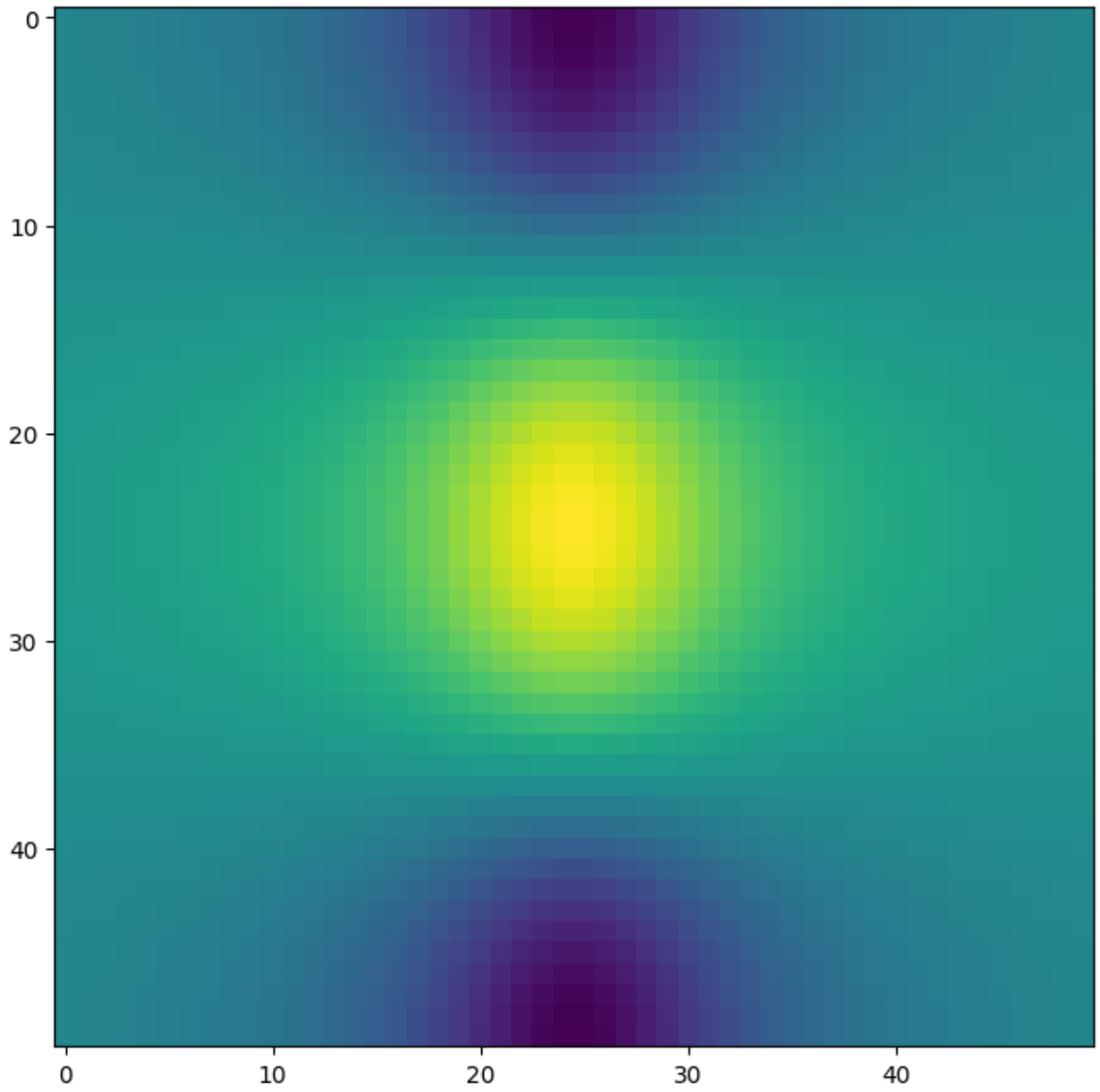



```
fig, ax = subplots(figsize=(8, 8))
ax.contour(x, y, f, levels=45);
```




```
fig, ax = subplots(figsize=(8, 8))
ax.imshow(f);
```

```
Executing <Handle BaseAsyncIOLoop._handle_events(28, 1) created at /usr/lib/python3.12/asyncio
Executing <Handle IOLoop._run_callback(functools.par...7eba43ea1d00>)) created at /home/linus
```




```
seq1 = np.linspace(0, 10, 11)
seq1
```

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

```
seq2 = np.arange(0, 10)
seq2
```

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

```
"Hello, world!"[3:6]
```

```
'lo,'
```

```
"Hello, world!"[slice(3, 6)]
```

```
'lo,'
```

```
A = np.array(np.arange(16)).reshape((4, 4))
print(A)
A[1, 2]
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

6
```

```
A[[1, 3]]
```

```
array([[ 4,  5,  6,  7],
       [12, 13, 14, 15]])
```

```
A[:, [0, 2]]
```

```
array([[ 0,  2],
       [ 4,  6],
       [ 8, 10],
       [12, 14]])
```

```
A[[1, 3], [0, 2]]
```

```
array([ 4, 14])
```

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

```
array([ 4, 14])
```

```
A[[1, 3]][:, [0, 2]]
```

```
array([[ 4,  6],
       [12, 14]])
```

```
idx = np.ix_([1, 3], [0, 2, 3])
A[idx]
```

```
array([[ 4,  6,  7],
       [12, 14, 15]])
```

```
A[1:4:2, 0:3:2]
```

```
array([[ 4,  6],
       [12, 14]])
```

```
keep_rows = np.zeros(A.shape[0], bool)
keep_rows
```

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

```
keep_rows[[1, 3]] = True
keep_rows
```

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

```
np.all(keep_rows == np.array([0, 1, 0, 1]))
```

```
True
```

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

```
array([[0, 1, 2, 3],
       [4, 5, 6, 7],
       [0, 1, 2, 3],
       [4, 5, 6, 7]])
```

```
A[keep_rows]
```

```
array([[ 4,  5,  6,  7],
       [12, 13, 14, 15]])
```

```
keep_cols = np.zeros(A.shape[1], bool)
keep_cols
```

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

```
keep_cols[[0, 2, 3]] = True
keep_cols
```

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

```
idx_bool = np.ix_(keep_rows, keep_cols)
idx_bool
```

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

```
A[idx_bool]
```

```
array([[ 4,  6,  7],
       [12, 14, 15]])
```

```
idx_mixed = np.ix_([1, 3], keep_cols)
idx_mixed
```

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

```
A[idx_mixed]
```

```
array([[ 4,  6,  7],
       [12, 14, 15]])
```

Reading in a data set

```
import pandas as pd

Auto = pd.read_csv("Auto.csv")
Auto
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle m
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
392	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
393	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
394	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
395	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
396	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

```
Auto = pd.read_csv("Auto.data", sep="\s+")
Auto
```

```
<>:1: SyntaxWarning: invalid escape sequence '\s'
<>:1: SyntaxWarning: invalid escape sequence '\s'
/tmp/ipykernel_14647/4041205627.py:1: SyntaxWarning: invalid escape sequence '\s'
  Auto = pd.read_csv("Auto.data", sep="\s+")
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle m
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
392	27.0	4	140.0	86.00	2790.0	15.6	82	1	ford mustang gl
393	44.0	4	97.0	52.00	2130.0	24.6	82	2	vw pickup
394	32.0	4	135.0	84.00	2295.0	11.6	82	1	dodge rampage
395	28.0	4	120.0	79.00	2625.0	18.6	82	1	ford ranger
396	31.0	4	119.0	82.00	2720.0	19.4	82	1	chevy s-10

```
Auto["horsepower"]
```

```
0      130.0
1      165.0
2      150.0
3      150.0
4      140.0
       ...
392    86.00
393    52.00
394    84.00
395    79.00
396    82.00
Name: horsepower, Length: 397, dtype: object
```

```
np.unique(Auto["horsepower"])
```

```
array(['100.0', '102.0', '103.0', '105.0', '107.0', '108.0', '110.0',
       '112.0', '113.0', '115.0', '116.0', '120.0', '122.0', '125.0',
       '129.0', '130.0', '132.0', '133.0', '135.0', '137.0', '138.0',
       '139.0', '140.0', '142.0', '145.0', '148.0', '149.0', '150.0',
       '152.0', '153.0', '155.0', '158.0', '160.0', '165.0', '167.0',
       '170.0', '175.0', '180.0', '190.0', '193.0', '198.0', '200.0',
       '208.0', '210.0', '215.0', '220.0', '225.0', '230.0', '46.00',
       '48.00', '49.00', '52.00', '53.00', '54.00', '58.00', '60.00',
       '61.00', '62.00', '63.00', '64.00', '65.00', '66.00', '67.00',
       '68.00', '69.00', '70.00', '71.00', '72.00', '74.00', '75.00',
```

```
       '76.00', '77.00', '78.00', '79.00', '80.00', '81.00', '82.00',
       '83.00', '84.00', '85.00', '86.00', '87.00', '88.00', '89.00',
       '90.00', '91.00', '92.00', '93.00', '94.00', '95.00', '96.00',
       '97.00', '98.00', '?'], dtype=object)
```

```
Auto = pd.read_csv("Auto.data", na_values=["?"], sep="\s+")
Auto["horsepower"].sum()
```

```
<>:1: SyntaxWarning: invalid escape sequence '\s'
<>:1: SyntaxWarning: invalid escape sequence '\s'
/tmp/ipykernel_14647/3247743814.py:1: SyntaxWarning: invalid escape sequence '\s'
  Auto = pd.read_csv("Auto.data", na_values=["?"], sep="\s+")
```

```
40952.0
```

```
Auto.shape
```

```
(397, 9)
```

```
Auto_new = Auto.dropna()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle m
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
392	27.0	4	140.0	86.0	2790.0	15.6	82	1	ford mustang gl
393	44.0	4	97.0	52.0	2130.0	24.6	82	2	vw pickup
394	32.0	4	135.0	84.0	2295.0	11.6	82	1	dodge rampage
395	28.0	4	120.0	79.0	2625.0	18.6	82	1	ford ranger
396	31.0	4	119.0	82.0	2720.0	19.4	82	1	chevy s-10

```
Auto_new.shape
```

```
(392, 9)
```

```
Auto = Auto_new
Auto.columns
```

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'year', 'origin', 'name'],
      dtype='object')
```

```
Auto[:3]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle mal
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite

```
idx_80 = Auto["year"] > 80
Auto[idx_80]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
338	27.2	4	135.0	84.0	2490.0	15.7	81	1	plymouth reliant
339	26.6	4	151.0	84.0	2635.0	16.4	81	1	buick skylark
340	25.8	4	156.0	92.0	2620.0	14.4	81	1	dodge aries wagon
341	23.5	6	173.0	110.0	2725.0	12.6	81	1	chevrolet citation
342	30.0	4	135.0	84.0	2385.0	12.9	81	1	plymouth reliant
343	39.1	4	79.0	58.0	1755.0	16.9	81	3	toyota starlet
344	39.0	4	86.0	64.0	1875.0	16.4	81	1	plymouth champ
345	35.1	4	81.0	60.0	1760.0	16.1	81	3	honda civic 1300
346	32.3	4	97.0	67.0	2065.0	17.8	81	3	subaru
347	37.0	4	85.0	65.0	1975.0	19.4	81	3	datsun 210 mpg
348	37.7	4	89.0	62.0	2050.0	17.3	81	3	toyota tercel
349	34.1	4	91.0	68.0	1985.0	16.0	81	3	mazda glc 4
350	34.7	4	105.0	63.0	2215.0	14.9	81	1	plymouth horizon 4
351	34.4	4	98.0	65.0	2045.0	16.2	81	1	ford escort 4w
352	29.9	4	98.0	65.0	2380.0	20.7	81	1	ford escort 2h
353	33.0	4	105.0	74.0	2190.0	14.2	81	2	volkswagen jetta
355	33.7	4	107.0	75.0	2210.0	14.4	81	3	honda prelude
356	32.4	4	108.0	75.0	2350.0	16.8	81	3	toyota corolla
357	32.9	4	119.0	100.0	2615.0	14.8	81	3	datsun 200sx
358	31.6	4	120.0	74.0	2635.0	18.3	81	3	mazda 626

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
359	28.1	4	141.0	80.0	3230.0	20.4	81	2	peugeot 505s turbo
360	30.7	6	145.0	76.0	3160.0	19.6	81	2	volvo diesel
361	25.4	6	168.0	116.0	2900.0	12.6	81	3	toyota cressida
362	24.2	6	146.0	120.0	2930.0	13.8	81	3	datsun 810 maxima
363	22.4	6	231.0	110.0	3415.0	15.8	81	1	buick century
364	26.6	8	350.0	105.0	3725.0	19.0	81	1	oldsmobile cutlass l
365	20.2	6	200.0	88.0	3060.0	17.1	81	1	ford granada gl
366	17.6	6	225.0	85.0	3465.0	16.6	81	1	chrysler lebaron sal
367	28.0	4	112.0	88.0	2605.0	19.6	82	1	chevrolet cavalier
368	27.0	4	112.0	88.0	2640.0	18.6	82	1	chevrolet cavalier w
369	34.0	4	112.0	88.0	2395.0	18.0	82	1	chevrolet cavalier 2-
370	31.0	4	112.0	85.0	2575.0	16.2	82	1	pontiac j2000 se ha
371	29.0	4	135.0	84.0	2525.0	16.0	82	1	dodge aries se
372	27.0	4	151.0	90.0	2735.0	18.0	82	1	pontiac phoenix
373	24.0	4	140.0	92.0	2865.0	16.4	82	1	ford fairmont futura
374	36.0	4	105.0	74.0	1980.0	15.3	82	2	volkswagen rabbit l
375	37.0	4	91.0	68.0	2025.0	18.2	82	3	mazda glc custom l
376	31.0	4	91.0	68.0	1970.0	17.6	82	3	mazda glc custom
377	38.0	4	105.0	63.0	2125.0	14.7	82	1	plymouth horizon m
378	36.0	4	98.0	70.0	2125.0	17.3	82	1	mercury lynx l
379	36.0	4	120.0	88.0	2160.0	14.5	82	3	nissan stanza xe
380	36.0	4	107.0	75.0	2205.0	14.5	82	3	honda accord
381	34.0	4	108.0	70.0	2245.0	16.9	82	3	toyota corolla
382	38.0	4	91.0	67.0	1965.0	15.0	82	3	honda civic
383	32.0	4	91.0	67.0	1965.0	15.7	82	3	honda civic (auto)
384	38.0	4	91.0	67.0	1995.0	16.2	82	3	datsun 310 gx
385	25.0	6	181.0	110.0	2945.0	16.4	82	1	buick century limite
386	38.0	6	262.0	85.0	3015.0	17.0	82	1	oldsmobile cutlass c
387	26.0	4	156.0	92.0	2585.0	14.5	82	1	chrysler lebaron me
388	22.0	6	232.0	112.0	2835.0	14.7	82	1	ford granada l
389	32.0	4	144.0	96.0	2665.0	13.9	82	3	toyota celica gt
390	36.0	4	135.0	84.0	2370.0	13.0	82	1	dodge charger 2.2
391	27.0	4	151.0	90.0	2950.0	17.3	82	1	chevrolet camaro
392	27.0	4	140.0	86.0	2790.0	15.6	82	1	ford mustang gl
393	44.0	4	97.0	52.0	2130.0	24.6	82	2	vw pickup
394	32.0	4	135.0	84.0	2295.0	11.6	82	1	dodge rampage
395	28.0	4	120.0	79.0	2625.0	18.6	82	1	ford ranger
396	31.0	4	119.0	82.0	2720.0	19.4	82	1	chevy s-10

```
Auto[["mpg", "horsepower"]]
```

	mpg	horsepower
0	18.0	130.0
1	15.0	165.0
2	18.0	150.0
3	16.0	150.0
4	17.0	140.0
...	...	...
392	27.0	86.0
393	44.0	52.0
394	32.0	84.0
395	28.0	79.0
396	31.0	82.0

```
Auto.index
```

```
Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
       ...
       387, 388, 389, 390, 391, 392, 393, 394, 395, 396],
      dtype='int64', length=392)
```

```
Auto_re = Auto.set_index("name")
Auto_re
```

name	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
chevrolet chevelle malibu	18.0	8	307.0	130.0	3504.0	12.0	70	1
buick skylark 320	15.0	8	350.0	165.0	3693.0	11.5	70	1
plymouth satellite	18.0	8	318.0	150.0	3436.0	11.0	70	1
amc rebel sst	16.0	8	304.0	150.0	3433.0	12.0	70	1
ford torino	17.0	8	302.0	140.0	3449.0	10.5	70	1
...	...	...	...	...	...	...	...	...
ford mustang gl	27.0	4	140.0	86.0	2790.0	15.6	82	1
vw pickup	44.0	4	97.0	52.0	2130.0	24.6	82	2
dodge rampage	32.0	4	135.0	84.0	2295.0	11.6	82	1
ford ranger	28.0	4	120.0	79.0	2625.0	18.6	82	1
chevy s-10	31.0	4	119.0	82.0	2720.0	19.4	82	1

```
Auto_re.columns
```

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'year', 'origin'],
      dtype='object')
```

```
Auto_re.shape
```

```
(392, 8)
```

```
rows = ["amc rebel sst", "ford torino"]
Auto_re.loc[rows]
```

name	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
amc rebel sst	16.0	8	304.0	150.0	3433.0	12.0	70	1
ford torino	17.0	8	302.0	140.0	3449.0	10.5	70	1

```
Auto_re.iloc[[3, 4]]
```

name	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
amc rebel sst	16.0	8	304.0	150.0	3433.0	12.0	70	1
ford torino	17.0	8	302.0	140.0	3449.0	10.5	70	1

```
Auto_re.iloc[:, [0, 2, 3]]
```

name	mpg	displacement	horsepower
chevrolet chevelle malibu	18.0	307.0	130.0
buick skylark 320	15.0	350.0	165.0
plymouth satellite	18.0	318.0	150.0
amc rebel sst	16.0	304.0	150.0
ford torino	17.0	302.0	140.0
...	...	...	...

name	mpg	displacement	horsepower
ford mustang gl	27.0	140.0	86.0
vw pickup	44.0	97.0	52.0
dodge rampage	32.0	135.0	84.0
ford ranger	28.0	120.0	79.0
chevy s-10	31.0	119.0	82.0

```
Auto_re.iloc[[3, 4], [0, 2, 3]]
```

name	mpg	displacement	horsepower
amc rebel sst	16.0	304.0	150.0
ford torino	17.0	302.0	140.0

```
Auto_re.loc["ford galaxie 500", ["mpg", "origin"]]
```

name	mpg	origin
ford galaxie 500	15.0	1
ford galaxie 500	14.0	1
ford galaxie 500	14.0	1

```
idx_80 = Auto_re["year"] > 80
Auto_re.loc[idx_80]
```

name	mpg	cylinders	displacement	horsepower	weight	acceleration	year
plymouth reliant	27.2	4	135.0	84.0	2490.0	15.7	81
buick skylark	26.6	4	151.0	84.0	2635.0	16.4	81
dodge aries wagon (sw)	25.8	4	156.0	92.0	2620.0	14.4	81
chevrolet citation	23.5	6	173.0	110.0	2725.0	12.6	81
plymouth reliant	30.0	4	135.0	84.0	2385.0	12.9	81
toyota starlet	39.1	4	79.0	58.0	1755.0	16.9	81
plymouth champ	39.0	4	86.0	64.0	1875.0	16.4	81
honda civic 1300	35.1	4	81.0	60.0	1760.0	16.1	81

name	mpg	cylinders	displacement	horsepower	weight	acceleration	year
subaru	32.3	4	97.0	67.0	2065.0	17.8	81
datsun 210 mpg	37.0	4	85.0	65.0	1975.0	19.4	81
toyota tercel	37.7	4	89.0	62.0	2050.0	17.3	81
mazda glc 4	34.1	4	91.0	68.0	1985.0	16.0	81
plymouth horizon 4	34.7	4	105.0	63.0	2215.0	14.9	81
ford escort 4w	34.4	4	98.0	65.0	2045.0	16.2	81
ford escort 2h	29.9	4	98.0	65.0	2380.0	20.7	81
volkswagen jetta	33.0	4	105.0	74.0	2190.0	14.2	81
honda prelude	33.7	4	107.0	75.0	2210.0	14.4	81
toyota corolla	32.4	4	108.0	75.0	2350.0	16.8	81
datsun 200sx	32.9	4	119.0	100.0	2615.0	14.8	81
mazda 626	31.6	4	120.0	74.0	2635.0	18.3	81
peugeot 505s turbo diesel	28.1	4	141.0	80.0	3230.0	20.4	81
volvo diesel	30.7	6	145.0	76.0	3160.0	19.6	81
toyota cressida	25.4	6	168.0	116.0	2900.0	12.6	81
datsun 810 maxima	24.2	6	146.0	120.0	2930.0	13.8	81
buick century	22.4	6	231.0	110.0	3415.0	15.8	81
oldsmobile cutlass ls	26.6	8	350.0	105.0	3725.0	19.0	81
ford granada gl	20.2	6	200.0	88.0	3060.0	17.1	81
chrysler lebaron salon	17.6	6	225.0	85.0	3465.0	16.6	81
chevrolet cavalier	28.0	4	112.0	88.0	2605.0	19.6	82
chevrolet cavalier wagon	27.0	4	112.0	88.0	2640.0	18.6	82
chevrolet cavalier 2-door	34.0	4	112.0	88.0	2395.0	18.0	82
pontiac j2000 se hatchback	31.0	4	112.0	85.0	2575.0	16.2	82
dodge aries se	29.0	4	135.0	84.0	2525.0	16.0	82
pontiac phoenix	27.0	4	151.0	90.0	2735.0	18.0	82
ford fairmont futura	24.0	4	140.0	92.0	2865.0	16.4	82
volkswagen rabbit l	36.0	4	105.0	74.0	1980.0	15.3	82
mazda glc custom l	37.0	4	91.0	68.0	2025.0	18.2	82
mazda glc custom	31.0	4	91.0	68.0	1970.0	17.6	82
plymouth horizon miser	38.0	4	105.0	63.0	2125.0	14.7	82
mercury lynx l	36.0	4	98.0	70.0	2125.0	17.3	82
nissan stanza xe	36.0	4	120.0	88.0	2160.0	14.5	82
honda accord	36.0	4	107.0	75.0	2205.0	14.5	82
toyota corolla	34.0	4	108.0	70.0	2245.0	16.9	82
honda civic	38.0	4	91.0	67.0	1965.0	15.0	82
honda civic (auto)	32.0	4	91.0	67.0	1965.0	15.7	82
datsun 310 gx	38.0	4	91.0	67.0	1995.0	16.2	82
buick century limited	25.0	6	181.0	110.0	2945.0	16.4	82
oldsmobile cutlass ciera (diesel)	38.0	6	262.0	85.0	3015.0	17.0	82

name	mpg	cylinders	displacement	horsepower	weight	acceleration	year
chrysler lebaron medallion	26.0	4	156.0	92.0	2585.0	14.5	82
ford granada l	22.0	6	232.0	112.0	2835.0	14.7	82
toyota celica gt	32.0	4	144.0	96.0	2665.0	13.9	82
dodge charger 2.2	36.0	4	135.0	84.0	2370.0	13.0	82
chevrolet camaro	27.0	4	151.0	90.0	2950.0	17.3	82
ford mustang gl	27.0	4	140.0	86.0	2790.0	15.6	82
vw pickup	44.0	4	97.0	52.0	2130.0	24.6	82
dodge rampage	32.0	4	135.0	84.0	2295.0	11.6	82
ford ranger	28.0	4	120.0	79.0	2625.0	18.6	82
chevy s-10	31.0	4	119.0	82.0	2720.0	19.4	82

```
Auto_re.loc[idx_80, ["weight", "origin"]]
```

name	weight	origin
plymouth reliant	2490.0	1
buick skylark	2635.0	1
dodge aries wagon (sw)	2620.0	1
chevrolet citation	2725.0	1
plymouth reliant	2385.0	1
toyota starlet	1755.0	3
plymouth champ	1875.0	1
honda civic 1300	1760.0	3
subaru	2065.0	3
datsun 210 mpg	1975.0	3
toyota tercel	2050.0	3
mazda glc 4	1985.0	3
plymouth horizon 4	2215.0	1
ford escort 4w	2045.0	1
ford escort 2h	2380.0	1
volkswagen jetta	2190.0	2
honda prelude	2210.0	3
toyota corolla	2350.0	3
datsun 200sx	2615.0	3
mazda 626	2635.0	3
peugeot 505s turbo diesel	3230.0	2
volvo diesel	3160.0	2
toyota cressida	2900.0	3

name	weight	origin
datsun 810 maxima	2930.0	3
buick century	3415.0	1
oldsmobile cutlass ls	3725.0	1
ford granada gl	3060.0	1
chrysler lebaron salon	3465.0	1
chevrolet cavalier	2605.0	1
chevrolet cavalier wagon	2640.0	1
chevrolet cavalier 2-door	2395.0	1
pontiac j2000 se hatchback	2575.0	1
dodge aries se	2525.0	1
pontiac phoenix	2735.0	1
ford fairmont futura	2865.0	1
volkswagen rabbit l	1980.0	2
mazda glc custom l	2025.0	3
mazda glc custom	1970.0	3
plymouth horizon miser	2125.0	1
mercury lynx l	2125.0	1
nissan stanza xe	2160.0	3
honda accord	2205.0	3
toyota corolla	2245.0	3
honda civic	1965.0	3
honda civic (auto)	1965.0	3
datsun 310 gx	1995.0	3
buick century limited	2945.0	1
oldsmobile cutlass ciera (diesel)	3015.0	1
chrysler lebaron medallion	2585.0	1
ford granada l	2835.0	1
toyota celica gt	2665.0	3
dodge charger 2.2	2370.0	1
chevrolet camaro	2950.0	1
ford mustang gl	2790.0	1
vw pickup	2130.0	2
dodge rampage	2295.0	1
ford ranger	2625.0	1
chevy s-10	2720.0	1

```
Auto_re.loc[lambda df: df["year"] > 80, ["weight", "origin"]]
```

name	weight	origin
plymouth reliant	2490.0	1
buick skylark	2635.0	1
dodge aries wagon (sw)	2620.0	1
chevrolet citation	2725.0	1
plymouth reliant	2385.0	1
toyota starlet	1755.0	3
plymouth champ	1875.0	1
honda civic 1300	1760.0	3
subaru	2065.0	3
datsun 210 mpg	1975.0	3
toyota tercel	2050.0	3
mazda glc 4	1985.0	3
plymouth horizon 4	2215.0	1
ford escort 4w	2045.0	1
ford escort 2h	2380.0	1
volkswagen jetta	2190.0	2
honda prelude	2210.0	3
toyota corolla	2350.0	3
datsun 200sx	2615.0	3
mazda 626	2635.0	3
peugeot 505s turbo diesel	3230.0	2
volvo diesel	3160.0	2
toyota cressida	2900.0	3
datsun 810 maxima	2930.0	3
buick century	3415.0	1
oldsmobile cutlass ls	3725.0	1
ford granada gl	3060.0	1
chrysler lebaron salon	3465.0	1
chevrolet cavalier	2605.0	1
chevrolet cavalier wagon	2640.0	1
chevrolet cavalier 2-door	2395.0	1
pontiac j2000 se hatchback	2575.0	1
dodge aries se	2525.0	1
pontiac phoenix	2735.0	1
ford fairmont futura	2865.0	1
volkswagen rabbit l	1980.0	2
mazda glc custom l	2025.0	3
mazda glc custom	1970.0	3
plymouth horizon miser	2125.0	1
mercury lynx l	2125.0	1

name	weight	origin
nissan stanza xe	2160.0	3
honda accord	2205.0	3
toyota corolla	2245.0	3
honda civic	1965.0	3
honda civic (auto)	1965.0	3
datsun 310 gx	1995.0	3
buick century limited	2945.0	1
oldsmobile cutlass ciera (diesel)	3015.0	1
chrysler lebaron medallion	2585.0	1
ford granada l	2835.0	1
toyota celica gt	2665.0	3
dodge charger 2.2	2370.0	1
chevrolet camaro	2950.0	1
ford mustang gl	2790.0	1
vw pickup	2130.0	2
dodge rampage	2295.0	1
ford ranger	2625.0	1
chevy s-10	2720.0	1

```
Auto_re.loc[lambda df: (df["year"] > 80) & (df["mpg"] > 30), ["weight", "origin"]]
```

name	weight	origin
toyota starlet	1755.0	3
plymouth champ	1875.0	1
honda civic 1300	1760.0	3
subaru	2065.0	3
datsun 210 mpg	1975.0	3
toyota tercel	2050.0	3
mazda glc 4	1985.0	3
plymouth horizon 4	2215.0	1
ford escort 4w	2045.0	1
volkswagen jetta	2190.0	2
honda prelude	2210.0	3
toyota corolla	2350.0	3
datsun 200sx	2615.0	3
mazda 626	2635.0	3
volvo diesel	3160.0	2

name	weight	origin
chevrolet cavalier 2-door	2395.0	1
pontiac j2000 se hatchback	2575.0	1
volkswagen rabbit l	1980.0	2
mazda glc custom l	2025.0	3
mazda glc custom	1970.0	3
plymouth horizon miser	2125.0	1
mercury lynx l	2125.0	1
nissan stanza xe	2160.0	3
honda accord	2205.0	3
toyota corolla	2245.0	3
honda civic	1965.0	3
honda civic (auto)	1965.0	3
datsun 310 gx	1995.0	3
oldsmobile cutlass ciera (diesel)	3015.0	1
toyota celica gt	2665.0	3
dodge charger 2.2	2370.0	1
vw pickup	2130.0	2
dodge rampage	2295.0	1
chevy s-10	2720.0	1

```
Auto_re.loc[
    lambda df: (df["displacement"] < 300)
    & (df.index.str.contains("ford") | df.index.str.contains("datsun")),
    ["weight", "origin"],
]
```

name	weight	origin
ford maverick	2587.0	1
datsun pl510	2130.0	3
datsun pl510	2130.0	3
ford torino 500	3302.0	1
ford mustang	3139.0	1
datsun 1200	1613.0	3
ford pinto runabout	2226.0	1
ford pinto (sw)	2395.0	1
datsun 510 (sw)	2288.0	3
ford maverick	3021.0	1

name	weight	origin
datsun 610	2379.0	3
ford pinto	2310.0	1
datsun b210	1950.0	3
ford pinto	2451.0	1
datsun 710	2003.0	3
ford maverick	3158.0	1
ford pinto	2639.0	1
datsun 710	2545.0	3
ford pinto	2984.0	1
ford maverick	3012.0	1
ford granada ghia	3574.0	1
datsun b-210	1990.0	3
ford pinto	2565.0	1
datsun f-10 hatchback	1945.0	3
ford granada	3525.0	1
ford mustang ii 2+2	2755.0	1
datsun 810	2815.0	3
ford fiesta	1800.0	1
datsun b210 gx	2070.0	3
ford fairmont (auto)	2965.0	1
ford fairmont (man)	2720.0	1
datsun 510	2300.0	3
datsun 200-sx	2405.0	3
ford fairmont 4	2890.0	1
datsun 210	2020.0	3
datsun 310	2019.0	3
ford fairmont	2870.0	1
datsun 510 hatchback	2434.0	3
datsun 210	2110.0	3
datsun 280-zx	2910.0	3
datsun 210 mpg	1975.0	3
ford escort 4w	2045.0	1
ford escort 2h	2380.0	1
datsun 200sx	2615.0	3
datsun 810 maxima	2930.0	3
ford granada gl	3060.0	1
ford fairmont futura	2865.0	1
datsun 310 gx	1995.0	3
ford granada l	2835.0	1
ford mustang gl	2790.0	1

name	weight	origin
ford ranger	2625.0	1

for loops

```
total = 0
for value in [3, 2, 9]:
    total += value
print("total is: {0}".format(total))
```

```
total is: 14
```

```
total = 0
for value in [3, 2, 9]:
    for weight in [3, 2, 1]:
        total += weight * value
print("total is: {0}".format(total))
```

```
total is: 84
```

```
total = 0
for value, weight in zip([3, 2, 9], [0.2, 0.3, 0.5]):
    total += weight * value
print("weighted average is: {0}".format(total))
```

```
weighted average is: 5.7
```

```
rng = np.random.default_rng(1)
A = rng.standard_normal((127, 5))
A
```

```
array([[ 3.45584192e-01,  8.21618144e-01,  3.30437076e-01,
        -1.30315723e+00,  9.05355867e-01],
       [ 4.46374572e-01, -5.36953235e-01,  5.81118104e-01,
         3.64572396e-01,  2.94132497e-01],
       [ 2.84222413e-02,  5.46712987e-01, -7.36454087e-01,
```

```
 -1.62909948e-01, -4.82119313e-01],
[ 5.98846213e-01,  3.97221075e-02, -2.92456751e-01,
 -7.81908462e-01, -2.57192241e-01],
[ 8.14218052e-03, -2.75602905e-01,  1.29406381e+00,
  1.00672432e+00, -2.71116248e+00],
[-1.88901325e+00, -1.74772092e-01, -4.22190412e-01,
  2.13642997e-01,  2.17321931e-01],
[ 2.11783876e+00, -1.11202076e+00, -3.77605007e-01,
  2.04277161e+00,  6.46702996e-01],
[ 6.63063372e-01, -5.14006372e-01, -1.64807517e+00,
  1.67464744e-01,  1.09014088e-01],
[-1.22735205e+00, -6.83226662e-01, -7.20436797e-02,
 -9.44751623e-01, -9.82699679e-02],
[ 9.54830275e-02,  3.55862371e-02, -5.06291658e-01,
  5.93748072e-01,  8.91166954e-01],
[ 3.20848305e-01, -8.18230227e-01,  7.31652284e-01,
 -5.01440018e-01,  8.79160618e-01],
[-1.07178742e+00,  9.14467203e-01, -2.00634546e-02,
 -1.24874889e+00, -3.13899472e-01],
[ 5.41022788e-02,  2.72791339e-01, -9.82188125e-01,
 -1.10737305e+00,  1.99584533e-01],
[-4.66749617e-01,  2.35505612e-01,  7.59519522e-01,
 -1.64878737e+00,  2.54388117e-01],
[ 1.22464697e+00, -2.97526844e-01, -8.10814583e-01,
  7.52243827e-01,  2.53446516e-01],
[ 8.95883071e-01, -3.45215710e-01, -1.48181827e+00,
 -1.10010765e-01, -4.45828153e-01],
[ 7.75323822e-01,  1.93632848e-01, -1.63084923e+00,
 -1.19516308e+00,  8.83789037e-01],
[ 6.79765017e-01, -6.40243366e-01, -1.04879657e-03,
  4.45573554e-01,  4.68404336e-01],
[ 8.76242196e-01,  2.56485627e-01, -9.48283390e-02,
 -2.58848065e-01,  1.05574280e+00],
[-2.25085428e+00, -1.38655325e-01,  3.30001040e-02,
 -1.42534896e+00,  3.32813613e-01],
[-6.51281012e-01,  8.62444796e-01, -1.25592084e-01,
  6.69153241e-01,  1.21884361e+00],
[ 3.82929583e-01, -8.75721143e-01, -1.51431863e+00,
  1.75338412e+00, -1.11292193e-01],
[-6.88564948e-01,  1.44257088e-01, -1.91411330e-01,
  8.52142264e-01,  3.39281824e-02],
[ 1.37495836e-02, -7.14579721e-01,  4.69568099e-01,
 -1.03386672e+00,  6.65889440e-01],
```

```
[ 1.52393751e+00, -1.52468604e+00, -2.46622923e+00,
  6.16878755e-01,  2.54789782e+00],
[-1.00092485e+00, -1.25069576e+00,  5.88968934e-01,
 -8.40721590e-01, -5.06025484e-01],
[-3.48117467e-01,  5.32002086e-01, -4.05302361e-01,
  2.77882840e-01, -1.76533259e-01],
[-8.44671104e-01, -3.19826258e-01, -9.50399665e-01,
  6.51498587e-03, -1.12386623e+00],
[-1.09289437e+00,  1.45696182e+00, -5.31842203e-02,
 -5.39020255e-02,  5.11536420e-01],
[-4.20857003e-01, -2.28535367e-01,  4.25148735e-01,
  2.82415842e-01, -1.15929673e+00],
[ 8.33342597e-01, -5.90434943e-01, -1.05607895e+00,
 -9.00475070e-01, -3.90545344e-01],
[ 1.62730025e+00, -1.17553590e+00,  1.60075893e-01,
 -2.13782436e+00, -1.56693315e-03],
[ 8.99566417e-01, -2.36663322e-01, -6.29354924e-01,
  2.31511064e-01,  7.00151751e-01],
[ 6.63657571e-01,  1.97247385e+00,  2.09167472e-01,
 -5.92410100e-01, -1.25979190e-01],
[-7.24985456e-02,  1.08737383e-01, -3.00278132e-02,
  1.73965939e-01, -1.67085006e+00],
[ 8.29628956e-01, -5.74739269e-01, -1.17315870e+00,
  6.37751160e-01,  1.31732601e+00],
[ 4.93028149e-01,  1.61159314e-01, -9.32220305e-01,
  2.87156734e+00,  8.80258621e-01],
[-1.13929467e+00, -7.79637916e-01,  8.69792486e-02,
 -1.55473113e+00,  1.68630407e-01],
[-4.59071556e-01,  1.22627060e+00,  9.62154664e-01,
 -2.71128544e+00,  4.17025860e-02],
[-1.61746750e+00,  1.10963800e+00,  1.68105869e-01,
  5.48405452e-01, -1.06512473e+00],
[ 1.82843024e+00,  2.02007337e+00, -1.06477104e+00,
  3.72815122e-01, -6.73302428e-01],
[-2.35699369e-02, -1.26563698e+00,  1.86714551e+00,
 -9.69179511e-01, -2.96083815e-01],
[ 5.01482931e-01, -6.47560678e-01, -2.39312430e-01,
 -5.63639846e-01, -1.33460755e-01],
[-1.17054264e+00, -4.37988076e-01, -2.06892925e-01,
 -3.33726004e-01,  5.66899549e-02],
[-2.93102219e-01,  7.53211408e-01, -3.23195926e-01,
 -1.36649596e-01, -6.64781336e-01],
[-5.26514840e-01, -1.26449279e+00,  5.18784921e-01,
```

```
 -1.14251802e+00, -7.45856398e-01],
[ 3.59244652e-01,  4.02573366e-01, -4.00114751e-01,
 -2.01926581e+00,  4.20513287e-01],
[ 2.59563459e-01, -1.41238122e+00,  7.70322083e-01,
 -7.01099800e-01, -1.12618812e+00],
[ 9.57307110e-02, -1.78470431e-01,  2.02624001e-01,
 -1.60574806e+00,  1.81223012e+00],
[-6.02658615e-01, -1.53965931e+00,  6.18842189e-01,
 -3.54804130e-01,  3.24858486e-01],
[-3.39608431e-01, -5.97403605e-02,  2.45772844e-01,
 -7.46652884e-01,  6.78739596e-01],
[-4.69900099e-01, -8.69687144e-01,  7.70324218e-02,
  4.45041278e-01, -2.29079342e-01],
[-8.62519787e-01,  6.19785566e-01, -1.76032879e+00,
 -1.03086414e+00,  3.95228905e-02],
[-1.36105940e+00,  2.79942642e-02, -5.48631180e-02,
  8.98739789e-01, -9.14790352e-01],
[-6.25906524e-01,  3.33181685e-01, -2.45756359e+00,
  3.10004230e+00, -6.98650730e-01],
[-7.29835053e-01,  8.61127511e-01, -3.98318414e-02,
 -1.77942862e+00,  6.26927380e-01],
[ 8.55377834e-01, -4.49946273e-01, -2.81600358e-01,
  4.85984597e-01, -9.08780250e-01],
[ 4.38388564e-01,  1.99298510e-01, -6.74932616e-01,
 -1.39210187e+00, -2.25605831e-01],
[-8.75422258e-01,  1.00141023e+00,  1.44085368e-01,
  7.82084523e-01,  1.34621935e-01],
[ 2.62901117e-01, -7.82998917e-01,  6.68047427e-01,
  1.78469827e+00, -3.09687556e-01],
[-5.92774528e-01, -1.57836702e-01, -4.81280284e-01,
 -7.01479299e-01,  1.38193644e-01],
[-2.90917533e-01,  1.43887359e+00,  2.01642955e-04,
  3.23911978e-01,  9.52021868e-01],
[-3.00755853e-01,  1.43673654e+00, -6.32694213e-01,
 -8.08327702e-01, -3.66267019e-01],
[-1.14717392e-01, -1.40131820e+00, -3.50947841e-02,
 -1.66748659e+00,  1.39213840e+00],
[-8.09969514e-02, -6.41957725e-01, -9.08338118e-01,
 -3.84430474e-01, -2.23079468e-01],
[-1.04451097e+00, -9.19795094e-01, -1.87175005e-01,
 -5.20783748e-01,  9.39213104e-01],
[ 1.13787037e+00,  1.60212036e-02,  4.73599572e-01,
 -1.33518837e+00,  6.37417709e-01],
```

```
[-3.05834695e-02,  4.84668316e-01,  1.60035613e+00,
 -2.28085776e+00,  2.60948179e-01],
[-1.09911906e+00,  5.92196691e-01, -1.31331344e+00,
 -4.95399567e-01,  2.02733756e-01],
[ 6.13509386e-01,  7.47940893e-02, -7.92830939e-01,
 -5.53532218e-01,  8.84502891e-01],
[-5.49961569e-03, -1.68381440e+00,  8.43662708e-01,
  4.16248480e-01,  8.73415545e-01],
[-3.36627466e-01,  8.28156518e-01, -1.06106492e+00,
  5.69999811e-01, -4.90380310e-01],
[ 6.74351705e-01,  1.00564412e+00, -7.35990152e-01,
 -5.12294795e-02,  3.89546265e-02],
[ 1.18966482e+00,  7.10558091e-01, -1.21927754e+00,
  4.57608261e-01,  7.45089209e-01],
[ 2.12380590e+00, -1.67914910e+00, -5.36352300e-01,
  1.33337119e+00, -1.35507030e+00],
[-1.19946058e+00,  5.17082195e-01,  1.01840866e+00,
 -6.68680487e-01,  5.40127172e-01],
[ 1.16955269e-01,  1.51874903e+00, -1.51843971e-03,
  9.90247312e-01, -9.03117859e-01],
[-1.84878849e-01, -9.67044538e-02,  1.13910795e+00,
  5.79613040e-01, -7.51753131e-01],
[ 6.81967764e-01,  7.70631276e-01, -1.11645722e-01,
 -2.57662315e-01, -1.93803245e-01],
[-1.69499241e+00,  1.88743008e-01,  2.34577929e-01,
 -8.65528524e-01,  7.42446250e-01],
[-1.37296694e+00, -5.51812821e-01, -4.75219069e-01,
  1.98750983e+00, -1.59918406e+00],
[ 5.62630255e-01,  9.42123278e-01,  3.77725564e-01,
  1.18180794e+00, -1.00273769e+00],
[-2.28025860e+00,  7.66817889e-01, -1.19582816e+00,
 -3.26399877e-01, -1.17884600e+00],
[ 1.05234186e+00,  8.65150235e-01, -7.20063662e-01,
  9.03491845e-01,  1.21277259e-01],
[-1.39388831e-01,  5.74166620e-02, -2.02749360e-01,
  6.15005264e-01,  3.10347570e-01],
[-3.49467146e-01,  1.00625575e+00, -6.11475615e-01,
  2.86933643e-01,  4.22155371e-01],
[ 1.47880706e+00, -5.05441363e-01,  1.74308245e+00,
  1.76164952e-01, -1.93976911e-01],
[-6.72472751e-01,  5.86293419e-01,  4.89352444e-02,
 -1.10360927e+00, -1.12788718e+00],
[-5.55519208e-01, -6.76006004e-01,  1.09030239e+00,
```

```
  1.37553830e+00,  8.68559624e-01],
[ 3.57868890e-01, -4.28238208e-01,  5.44832109e-02,
  8.84707006e-01,  2.13097223e+00],
[ 9.12899205e-01, -2.80412144e-01,  3.80250758e-02,
 -4.82544901e-01, -7.82398576e-01],
[-1.84242002e-01,  1.95614692e-01,  1.83590159e+00,
  5.20324700e-02,  1.35735807e+00],
[ 1.76748908e+00,  7.88078703e-02,  1.60779889e+00,
  7.14935405e-01, -4.17535295e-01],
[ 2.65180025e-01,  2.28778180e-02, -2.32339030e-01,
 -1.98781233e-01,  1.44209303e-01],
[ 4.34181140e-01, -8.84881875e-01, -9.38766200e-03,
 -1.52235502e+00,  2.60795499e-01],
[ 6.23599476e-01,  1.63954838e-01,  2.85853152e-01,
  5.89943104e-01, -6.65100844e-01],
[-2.39048859e-01,  5.10922044e-01,  1.00188573e+00,
  3.94921401e-01,  2.55285942e+00],
[-9.13605722e-02,  9.99660429e-01,  1.26672768e+00,
 -1.33769846e-01, -8.20512882e-01],
[-1.18260293e+00,  1.62277000e-01,  1.11029195e+00,
  2.71668399e-01,  1.72856075e-01],
[-3.79213883e-01,  5.61029733e-01, -2.13583105e+00,
  2.32373251e-01,  2.81263096e-02],
[-1.37034025e+00,  2.17559792e+00, -1.38741323e+00,
 -1.07752050e+00, -1.20086311e+00],
[ 1.11036780e+00, -8.88084861e-01,  6.68656413e-01,
  5.87510153e-01,  2.59670411e-01],
[-1.30757891e+00, -6.12106378e-01,  1.67311497e+00,
 -1.29075440e+00, -8.31654960e-01],
[-1.62246523e-01,  8.08990072e-01,  2.51639191e-01,
  7.42044159e-01, -1.06729423e+00],
[ 9.44776594e-01,  5.69028817e-01, -1.59456456e+00,
  1.53991621e+00,  2.29239786e+00],
[-7.68577357e-01,  5.56528166e-02,  1.39742258e+00,
 -1.48124023e+00, -1.98986032e+00],
[-1.29694412e+00, -5.67066786e-01, -5.78442212e-01,
  6.07347174e-01,  2.68536372e-01],
[-1.24607726e+00,  5.67379970e-01,  1.87408134e+00,
  1.19705428e+00,  9.91107448e-01],
[ 2.01779285e-02,  9.83631395e-01, -9.66132820e-01,
  7.51014637e-01, -8.72484195e-02],
[ 1.13090570e+00,  4.66010551e-01, -1.08987979e+00,
  1.27992953e-01,  1.21822147e+00],
```

```
       [-1.12923389e+00, -5.59139878e-01, -7.68673008e-01,
        -1.49564594e+00,  9.61268275e-01],
       [ 1.31099730e+00,  8.00043207e-01,  2.42570315e-01,
        -5.20313843e-02,  2.39380301e-01],
       [-7.31555917e-01,  8.93845600e-01,  1.04549232e+00,
         9.32543534e-01, -5.30948218e-01],
       [ 7.84043419e-02, -1.65780819e-01,  1.78857479e+00,
         1.79696202e-01, -1.88207089e+00],
       [ 3.98505651e-01,  1.88226625e+00,  6.85175716e-01,
         8.79479710e-01,  3.60612668e-02],
       [-1.97138949e+00, -1.81025775e+00, -1.24594777e+00,
        -1.26832037e-01,  3.09998496e-01],
       [ 6.89414152e-01, -3.40458043e-01,  9.56929258e-01,
        -2.79748758e-01, -7.04489515e-01],
       [ 8.51641715e-01, -9.14444983e-01, -2.72908493e+00,
        -1.05995676e+00,  9.41595455e-02],
       [-3.08017776e+00, -3.57138046e-01, -3.32090359e-01,
        -1.42645597e+00, -1.48250346e+00],
       [-4.62461948e-01, -5.47100170e-01,  1.25991851e+00,
         3.74863045e-01, -1.58085545e+00],
       [-8.54651470e-01,  7.05141527e-01,  1.89597677e+00,
         3.86499494e-01,  3.10712545e-01],
       [ 1.86075246e+00, -2.26714440e-02, -3.09100958e-01,
        -1.41324878e+00, -5.13016082e-01],
       [ 2.17871274e+00, -1.42174139e+00,  1.11917068e-02,
        -1.40832743e+00,  1.28475748e-01],
       [ 8.87647203e-01, -2.43649320e-01,  7.30347270e-01,
         7.20843826e-01,  4.47024816e-01],
       [ 1.71624800e+00,  7.78837554e-01, -3.04959566e-01,
        -6.80823548e-01, -8.45306348e-01],
       [ 4.75867438e-01, -3.23721814e-01,  2.72916704e+00,
         1.84244761e+00, -2.14524000e-01],
       [-3.29131254e-01,  1.69017655e+00, -1.88335978e+00,
        -4.51551059e-01,  9.50756951e-01]])
```

```
A.shape
```

```
(127, 5)
```

```
M = rng.choice([0, np.nan], p=[0.8, 0.2], size=A.shape)
M
```

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

```
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0., nan, nan,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[nan,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0., nan, nan, nan, nan],
[nan,  0.,  0.,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[ 0.,  0.,  0., nan,  0.],
[ 0., nan,  0., nan,  0.],
[ 0.,  0.,  0., nan,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0., nan,  0.,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[nan, nan, nan, nan,  0.],
[ 0.,  0.,  0., nan,  0.],
[ 0., nan,  0.,  0.,  0.],
[ 0., nan,  0.,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0., nan],
[ 0.,  0., nan,  0.,  0.],
[nan, nan,  0.,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0., nan,  0., nan, nan],
[ 0., nan,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0., nan,  0.,  0.,  0.],
[ 0.,  0.,  0., nan,  0.],
[ 0., nan,  0.,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[nan,  0., nan,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[ 0.,  0., nan,  0., nan],
[ 0., nan,  0.,  0.,  0.],
[nan, nan, nan,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
```

```
[ 0.,  0.,  0.,  0.,  0.],
[nan, nan,  0.,  0.,  0.],
[nan,  0.,  0.,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0., nan,  0.],
[ 0., nan,  0.,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[ 0., nan, nan, nan,  0.],
[ 0.,  0., nan,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0., nan, nan,  0.,  0.],
[ 0., nan,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0., nan,  0.,  0.],
[ 0.,  0.,  0.,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0., nan, nan, nan,  0.],
[ 0., nan,  0.,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0., nan, nan],
[ 0.,  0., nan,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0., nan,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[nan,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[nan,  0., nan, nan, nan],
[ 0.,  0.,  0.,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[ 0., nan,  0.,  0., nan],
[ 0.,  0.,  0.,  0.,  0.],
[nan,  0.,  0.,  0.,  0.],
[nan,  0.,  0., nan,  0.],
[ 0.,  0.,  0.,  0., nan],
[ 0.,  0., nan,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.]])
```

```
A += M
A
```

```
array([[ 3.45584192e-01,  8.21618144e-01,  3.30437076e-01,
        -1.30315723e+00,             nan],
       [            nan, -5.36953235e-01,  5.81118104e-01,
         3.64572396e-01,  2.94132497e-01],
       [            nan,  5.46712987e-01,             nan,
        -1.62909948e-01, -4.82119313e-01],
       [ 5.98846213e-01,             nan,             nan,
        -7.81908462e-01,             nan],
       [ 8.14218052e-03, -2.75602905e-01,  1.29406381e+00,
                    nan, -2.71116248e+00],
       [-1.88901325e+00, -1.74772092e-01, -4.22190412e-01,
         2.13642997e-01,  2.17321931e-01],
       [ 2.11783876e+00, -1.11202076e+00, -3.77605007e-01,
         2.04277161e+00,             nan],
       [            nan,             nan, -1.64807517e+00,
         1.67464744e-01,  1.09014088e-01],
       [-1.22735205e+00, -6.83226662e-01,             nan,
                    nan, -9.82699679e-02],
       [ 9.54830275e-02,  3.55862371e-02, -5.06291658e-01,
                    nan,  8.91166954e-01],
       [            nan, -8.18230227e-01,  7.31652284e-01,
        -5.01440018e-01,             nan],
       [            nan,  9.14467203e-01, -2.00634546e-02,
        -1.24874889e+00, -3.13899472e-01],
       [ 5.41022788e-02,  2.72791339e-01,             nan,
        -1.10737305e+00,  1.99584533e-01],
       [-4.66749617e-01,  2.35505612e-01,  7.59519522e-01,
        -1.64878737e+00,  2.54388117e-01],
       [ 1.22464697e+00, -2.97526844e-01, -8.10814583e-01,
                    nan,             nan],
       [ 8.95883071e-01, -3.45215710e-01,             nan,
        -1.10010765e-01, -4.45828153e-01],
       [ 7.75323822e-01,  1.93632848e-01, -1.63084923e+00,
        -1.19516308e+00,  8.83789037e-01],
       [ 6.79765017e-01,             nan,             nan,
         4.45573554e-01,  4.68404336e-01],
       [ 8.76242196e-01,  2.56485627e-01, -9.48283390e-02,
        -2.58848065e-01,  1.05574280e+00],
       [-2.25085428e+00, -1.38655325e-01,  3.30001040e-02,
```

```
 -1.42534896e+00,  3.32813613e-01],
[-6.51281012e-01,             nan, -1.25592084e-01,
             nan,             nan],
[ 3.82929583e-01,             nan, -1.51431863e+00,
  1.75338412e+00,             nan],
[-6.88564948e-01,             nan, -1.91411330e-01,
             nan,  3.39281824e-02],
[            nan, -7.14579721e-01,  4.69568099e-01,
             nan,  6.65889440e-01],
[ 1.52393751e+00,             nan,             nan,
  6.16878755e-01,             nan],
[-1.00092485e+00,             nan,             nan,
 -8.40721590e-01, -5.06025484e-01],
[-3.48117467e-01,             nan, -4.05302361e-01,
  2.77882840e-01,             nan],
[-8.44671104e-01, -3.19826258e-01, -9.50399665e-01,
             nan,             nan],
[-1.09289437e+00,  1.45696182e+00,             nan,
 -5.39020255e-02,  5.11536420e-01],
[-4.20857003e-01, -2.28535367e-01,  4.25148735e-01,
  2.82415842e-01,             nan],
[ 8.33342597e-01, -5.90434943e-01,             nan,
             nan,             nan],
[ 1.62730025e+00, -1.17553590e+00,  1.60075893e-01,
             nan, -1.56693315e-03],
[ 8.99566417e-01,             nan,             nan,
  2.31511064e-01,  7.00151751e-01],
[            nan,  1.97247385e+00,             nan,
 -5.92410100e-01, -1.25979190e-01],
[-7.24985456e-02,  1.08737383e-01, -3.00278132e-02,
  1.73965939e-01, -1.67085006e+00],
[ 8.29628956e-01,             nan, -1.17315870e+00,
  6.37751160e-01,  1.31732601e+00],
[ 4.93028149e-01,  1.61159314e-01, -9.32220305e-01,
  2.87156734e+00,  8.80258621e-01],
[-1.13929467e+00, -7.79637916e-01,  8.69792486e-02,
             nan,  1.68630407e-01],
[            nan,  1.22627060e+00,  9.62154664e-01,
 -2.71128544e+00,  4.17025860e-02],
[-1.61746750e+00,  1.10963800e+00,  1.68105869e-01,
  5.48405452e-01, -1.06512473e+00],
[ 1.82843024e+00,             nan,             nan,
  3.72815122e-01, -6.73302428e-01],
```

```
[           nan, -1.26563698e+00,  1.86714551e+00,
 -9.69179511e-01, -2.96083815e-01],
[ 5.01482931e-01, -6.47560678e-01, -2.39312430e-01,
 -5.63639846e-01, -1.33460755e-01],
[-1.17054264e+00, -4.37988076e-01, -2.06892925e-01,
 -3.33726004e-01,  5.66899549e-02],
[-2.93102219e-01,  7.53211408e-01,            nan,
            nan, -6.64781336e-01],
[-5.26514840e-01, -1.26449279e+00,  5.18784921e-01,
 -1.14251802e+00, -7.45856398e-01],
[           nan,  4.02573366e-01, -4.00114751e-01,
 -2.01926581e+00,  4.20513287e-01],
[ 2.59563459e-01, -1.41238122e+00,  7.70322083e-01,
 -7.01099800e-01, -1.12618812e+00],
[ 9.57307110e-02,            nan,            nan,
            nan,            nan],
[           nan, -1.53965931e+00,  6.18842189e-01,
 -3.54804130e-01,  3.24858486e-01],
[-3.39608431e-01, -5.97403605e-02,            nan,
 -7.46652884e-01,  6.78739596e-01],
[-4.69900099e-01, -8.69687144e-01,  7.70324218e-02,
            nan, -2.29079342e-01],
[-8.62519787e-01,            nan, -1.76032879e+00,
            nan,  3.95228905e-02],
[-1.36105940e+00,  2.79942642e-02, -5.48631180e-02,
            nan, -9.14790352e-01],
[-6.25906524e-01,  3.33181685e-01, -2.45756359e+00,
  3.10004230e+00, -6.98650730e-01],
[-7.29835053e-01,            nan, -3.98318414e-02,
 -1.77942862e+00,  6.26927380e-01],
[ 8.55377834e-01, -4.49946273e-01,            nan,
  4.85984597e-01, -9.08780250e-01],
[           nan,            nan,            nan,
            nan, -2.25605831e-01],
[-8.75422258e-01,  1.00141023e+00,  1.44085368e-01,
            nan,  1.34621935e-01],
[ 2.62901117e-01,            nan,  6.68047427e-01,
  1.78469827e+00, -3.09687556e-01],
[-5.92774528e-01,            nan, -4.81280284e-01,
 -7.01479299e-01,  1.38193644e-01],
[-2.90917533e-01,  1.43887359e+00,            nan,
  3.23911978e-01,  9.52021868e-01],
[-3.00755853e-01,  1.43673654e+00, -6.32694213e-01,
```

```
 -8.08327702e-01, -3.66267019e-01],
[-1.14717392e-01, -1.40131820e+00, -3.50947841e-02,
 -1.66748659e+00,             nan],
[-8.09969514e-02, -6.41957725e-01,             nan,
 -3.84430474e-01, -2.23079468e-01],
[            nan,             nan, -1.87175005e-01,
 -5.20783748e-01,  9.39213104e-01],
[ 1.13787037e+00,  1.60212036e-02,             nan,
 -1.33518837e+00,  6.37417709e-01],
[-3.05834695e-02,  4.84668316e-01,             nan,
 -2.28085776e+00,  2.60948179e-01],
[-1.09911906e+00,  5.92196691e-01, -1.31331344e+00,
 -4.95399567e-01,  2.02733756e-01],
[ 6.13509386e-01,  7.47940893e-02, -7.92830939e-01,
 -5.53532218e-01,  8.84502891e-01],
[-5.49961569e-03,             nan,  8.43662708e-01,
             nan,             nan],
[-3.36627466e-01,             nan, -1.06106492e+00,
  5.69999811e-01, -4.90380310e-01],
[ 6.74351705e-01,  1.00564412e+00, -7.35990152e-01,
 -5.12294795e-02,  3.89546265e-02],
[ 1.18966482e+00,             nan, -1.21927754e+00,
  4.57608261e-01,  7.45089209e-01],
[ 2.12380590e+00, -1.67914910e+00, -5.36352300e-01,
             nan, -1.35507030e+00],
[-1.19946058e+00,             nan,  1.01840866e+00,
 -6.68680487e-01,  5.40127172e-01],
[ 1.16955269e-01,  1.51874903e+00,             nan,
  9.90247312e-01, -9.03117859e-01],
[            nan, -9.67044538e-02,             nan,
  5.79613040e-01, -7.51753131e-01],
[ 6.81967764e-01,  7.70631276e-01,             nan,
 -2.57662315e-01, -1.93803245e-01],
[-1.69499241e+00,  1.88743008e-01,             nan,
 -8.65528524e-01,             nan],
[-1.37296694e+00,             nan, -4.75219069e-01,
  1.98750983e+00, -1.59918406e+00],
[            nan,             nan,             nan,
  1.18180794e+00,             nan],
[-2.28025860e+00,  7.66817889e-01, -1.19582816e+00,
 -3.26399877e-01, -1.17884600e+00],
[ 1.05234186e+00,  8.65150235e-01,             nan,
  9.03491845e-01,  1.21277259e-01],
```

```
[-1.39388831e-01,  5.74166620e-02, -2.02749360e-01,
  6.15005264e-01,  3.10347570e-01],
[-3.49467146e-01,  1.00625575e+00, -6.11475615e-01,
  2.86933643e-01,  4.22155371e-01],
[ 1.47880706e+00, -5.05441363e-01,  1.74308245e+00,
  1.76164952e-01, -1.93976911e-01],
[            nan,             nan,  4.89352444e-02,
 -1.10360927e+00, -1.12788718e+00],
[            nan, -6.76006004e-01,  1.09030239e+00,
  1.37553830e+00,             nan],
[ 3.57868890e-01, -4.28238208e-01,  5.44832109e-02,
  8.84707006e-01,  2.13097223e+00],
[ 9.12899205e-01, -2.80412144e-01,  3.80250758e-02,
             nan, -7.82398576e-01],
[-1.84242002e-01,             nan,  1.83590159e+00,
  5.20324700e-02,  1.35735807e+00],
[ 1.76748908e+00,  7.88078703e-02,             nan,
  7.14935405e-01, -4.17535295e-01],
[ 2.65180025e-01,             nan,             nan,
             nan,  1.44209303e-01],
[ 4.34181140e-01, -8.84881875e-01,             nan,
 -1.52235502e+00,             nan],
[ 6.23599476e-01,  1.63954838e-01,  2.85853152e-01,
  5.89943104e-01, -6.65100844e-01],
[-2.39048859e-01,             nan,             nan,
  3.94921401e-01,  2.55285942e+00],
[-9.13605722e-02,             nan,  1.26672768e+00,
 -1.33769846e-01, -8.20512882e-01],
[-1.18260293e+00,  1.62277000e-01,  1.11029195e+00,
  2.71668399e-01,             nan],
[-3.79213883e-01,  5.61029733e-01, -2.13583105e+00,
  2.32373251e-01,  2.81263096e-02],
[-1.37034025e+00,  2.17559792e+00, -1.38741323e+00,
 -1.07752050e+00,             nan],
[ 1.11036780e+00, -8.88084861e-01,  6.68656413e-01,
  5.87510153e-01,  2.59670411e-01],
[-1.30757891e+00, -6.12106378e-01,  1.67311497e+00,
 -1.29075440e+00, -8.31654960e-01],
[-1.62246523e-01,  8.08990072e-01,             nan,
  7.42044159e-01, -1.06729423e+00],
[ 9.44776594e-01,  5.69028817e-01, -1.59456456e+00,
  1.53991621e+00,             nan],
[-7.68577357e-01,  5.56528166e-02,  1.39742258e+00,
```

```
 -1.48124023e+00, -1.98986032e+00],
[-1.29694412e+00,             nan,             nan,
            nan,  2.68536372e-01],
[-1.24607726e+00,             nan,  1.87408134e+00,
  1.19705428e+00,             nan],
[ 2.01779285e-02,  9.83631395e-01, -9.66132820e-01,
  7.51014637e-01, -8.72484195e-02],
[ 1.13090570e+00,  4.66010551e-01, -1.08987979e+00,
            nan,             nan],
[-1.12923389e+00, -5.59139878e-01,             nan,
 -1.49564594e+00,             nan],
[ 1.31099730e+00,  8.00043207e-01,  2.42570315e-01,
 -5.20313843e-02,  2.39380301e-01],
[-7.31555917e-01,  8.93845600e-01,  1.04549232e+00,
            nan, -5.30948218e-01],
[ 7.84043419e-02, -1.65780819e-01,  1.78857479e+00,
  1.79696202e-01, -1.88207089e+00],
[            nan,  1.88226625e+00,  6.85175716e-01,
  8.79479710e-01,  3.60612668e-02],
[-1.97138949e+00, -1.81025775e+00, -1.24594777e+00,
 -1.26832037e-01,  3.09998496e-01],
[            nan, -3.40458043e-01,             nan,
            nan,             nan],
[ 8.51641715e-01, -9.14444983e-01, -2.72908493e+00,
 -1.05995676e+00,             nan],
[-3.08017776e+00, -3.57138046e-01, -3.32090359e-01,
 -1.42645597e+00, -1.48250346e+00],
[-4.62461948e-01,             nan,  1.25991851e+00,
  3.74863045e-01,             nan],
[-8.54651470e-01,  7.05141527e-01,  1.89597677e+00,
  3.86499494e-01,  3.10712545e-01],
[            nan, -2.26714440e-02, -3.09100958e-01,
 -1.41324878e+00, -5.13016082e-01],
[            nan, -1.42174139e+00,  1.11917068e-02,
            nan,  1.28475748e-01],
[ 8.87647203e-01, -2.43649320e-01,  7.30347270e-01,
  7.20843826e-01,             nan],
[ 1.71624800e+00,  7.78837554e-01,             nan,
 -6.80823548e-01, -8.45306348e-01],
[ 4.75867438e-01, -3.23721814e-01,  2.72916704e+00,
  1.84244761e+00, -2.14524000e-01],
[-3.29131254e-01,  1.69017655e+00, -1.88335978e+00,
 -4.51551059e-01,  9.50756951e-01]])
```

```
D = pd.DataFrame(A, columns=["food", "bar", "pickle", "snack", "popcorn"])
D
```

	food	bar	pickle	snack	popcorn
0	0.345584	0.821618	0.330437	-1.303157	NaN
1	NaN	-0.536953	0.581118	0.364572	0.294132
2	NaN	0.546713	NaN	-0.162910	-0.482119
3	0.598846	NaN	NaN	-0.781908	NaN
4	0.008142	-0.275603	1.294064	NaN	-2.711162
...	...	...	...	...	...
122	NaN	-1.421741	0.011192	NaN	0.128476
123	0.887647	-0.243649	0.730347	0.720844	NaN
124	1.716248	0.778838	NaN	-0.680824	-0.845306
125	0.475867	-0.323722	2.729167	1.842448	-0.214524
126	-0.329131	1.690177	-1.883360	-0.451551	0.950757

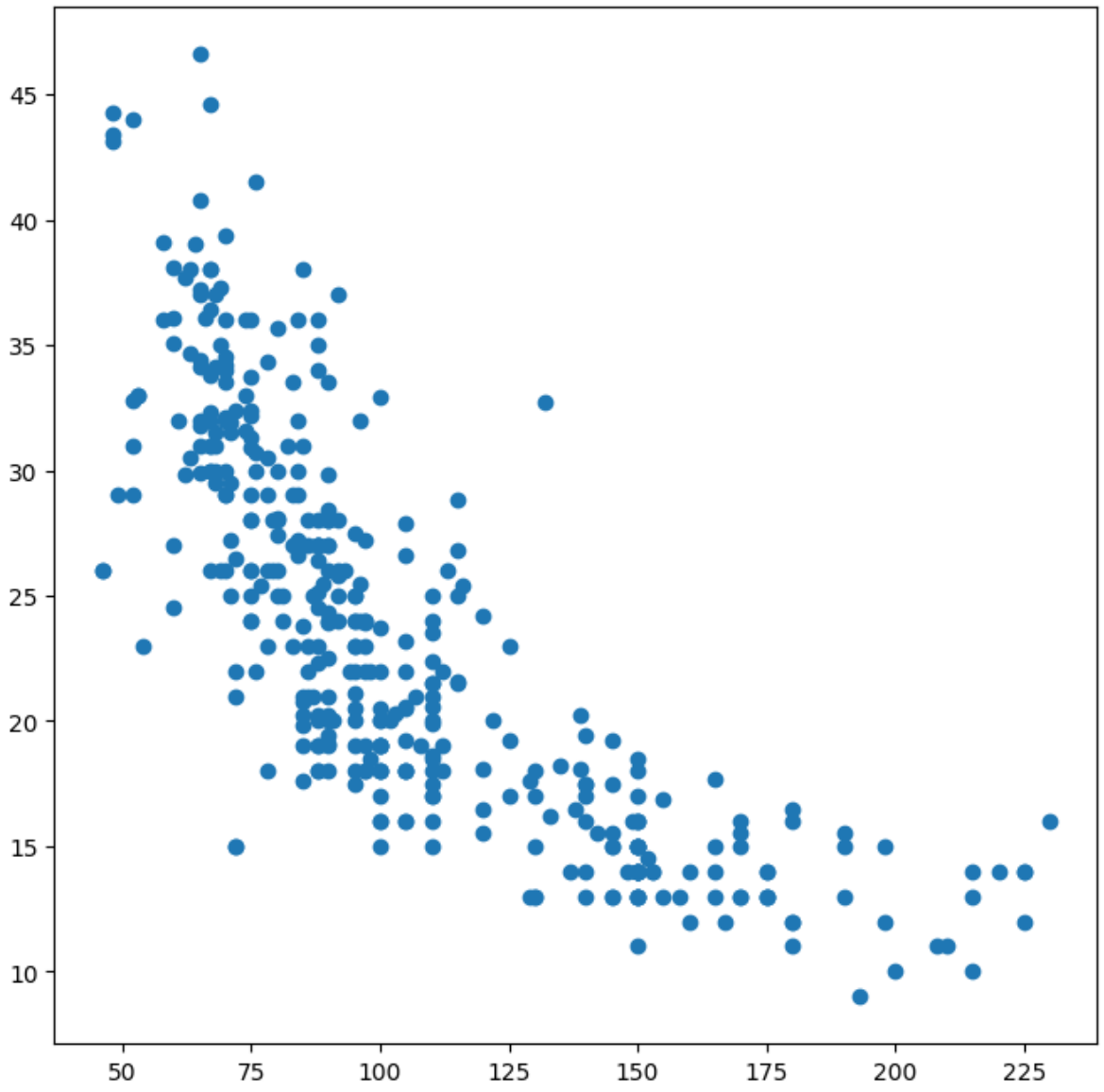
```
D[:3]
```

	food	bar	pickle	snack	popcorn
0	0.345584	0.821618	0.330437	-1.303157	NaN
1	NaN	-0.536953	0.581118	0.364572	0.294132
2	NaN	0.546713	NaN	-0.162910	-0.482119

```
for col in D.columns:
    template = "Column {0} has {1: .2%} missing values"
    print(template.format(col, np.isnan(D[col]).mean()))
```

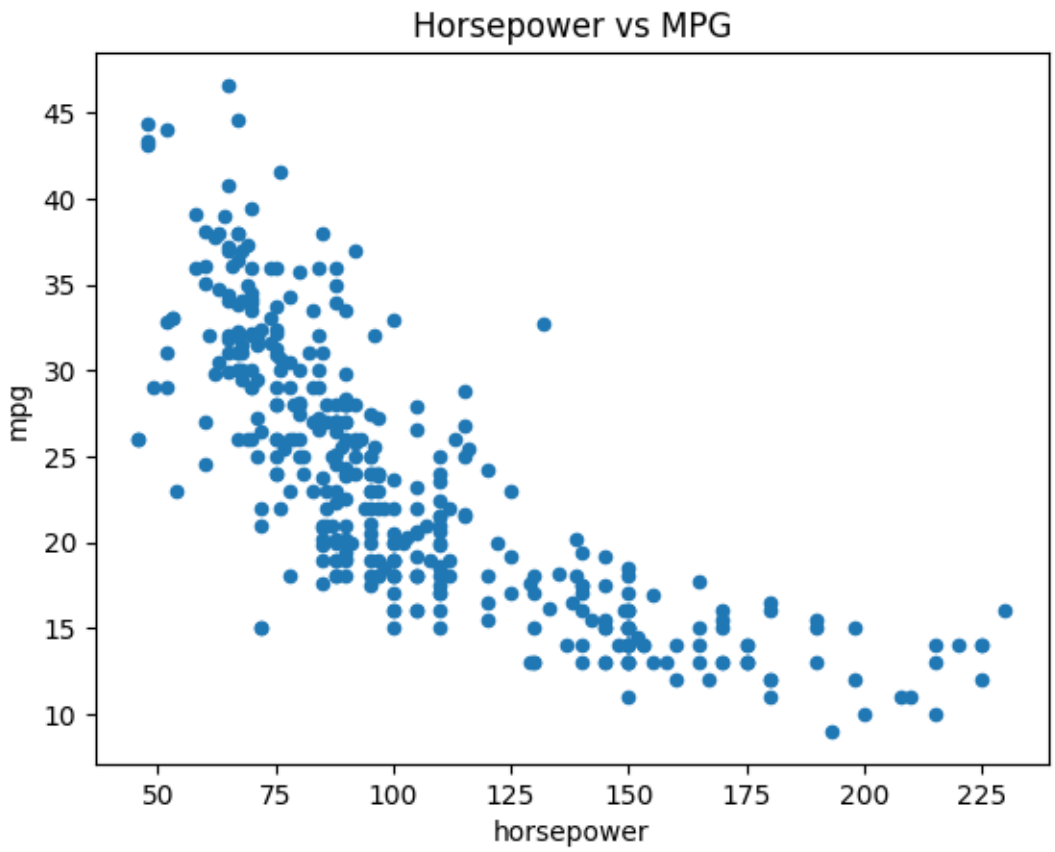
```
Column food has  16.54% missing values
Column bar has  25.98% missing values
Column pickle has  29.13% missing values
Column snack has  21.26% missing values
Column popcorn has  22.83% missing values
```

```
fig, ax = subplots(figsize=(8, 8))
ax.plot(Auto["horsepower"].values, Auto["mpg"].values, "o");
```



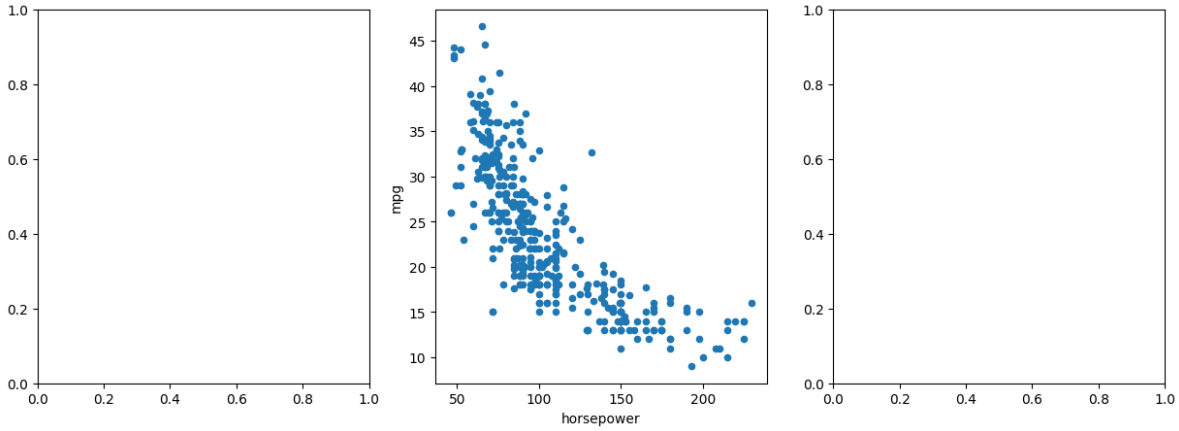

```
ax = Auto.plot.scatter("horsepower", "mpg")
ax.set_title("Horsepower vs MPG")
```

```
Text(0.5, 1.0, 'Horsepower vs MPG')
```




```
fig = ax.figure
fig.savefig("hp_mpg.png")
```

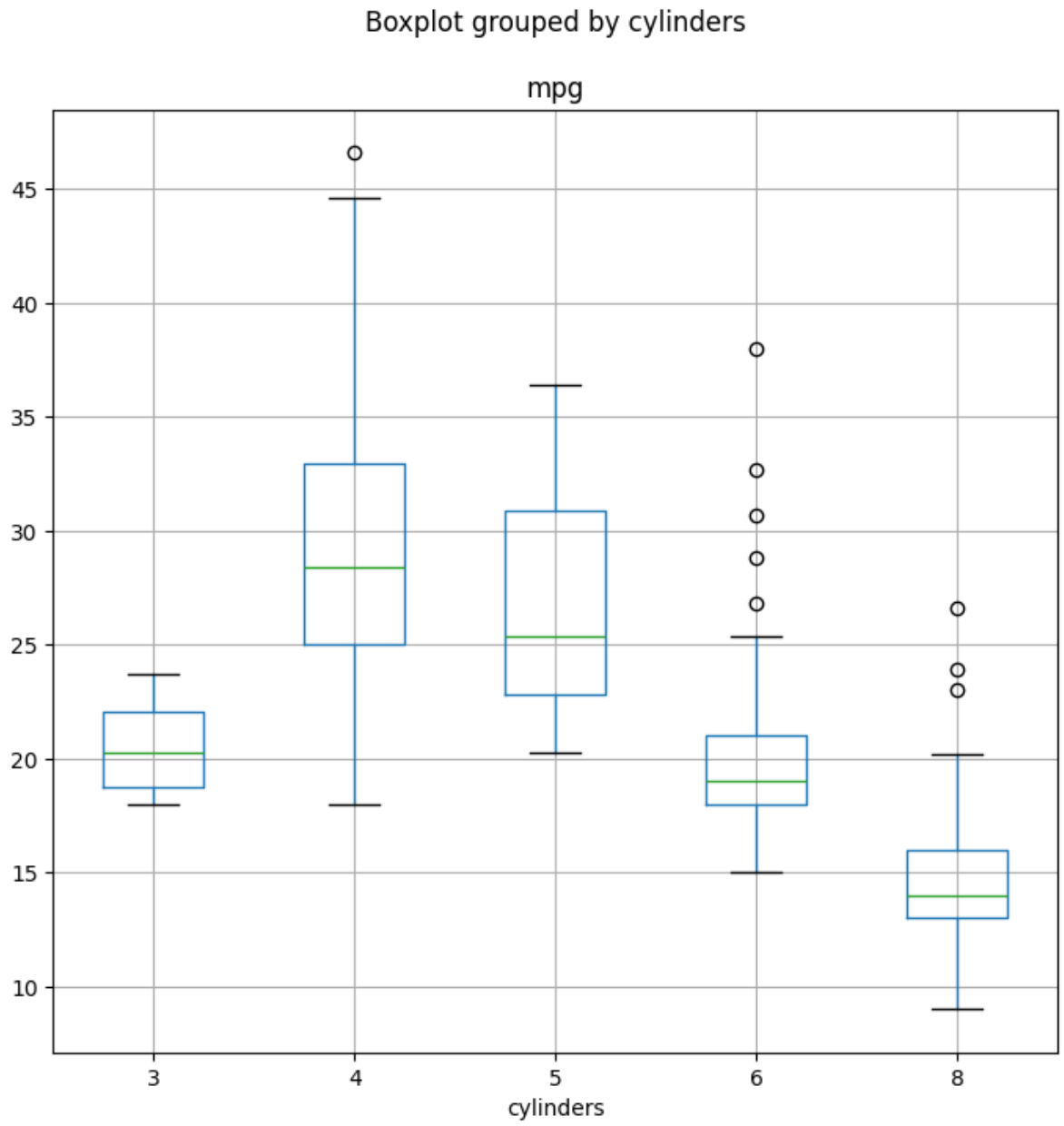
```
fig, axes = subplots(ncols=3, figsize=(15, 5))
Auto.plot.scatter("horsepower", "mpg", ax=axes[1])
```



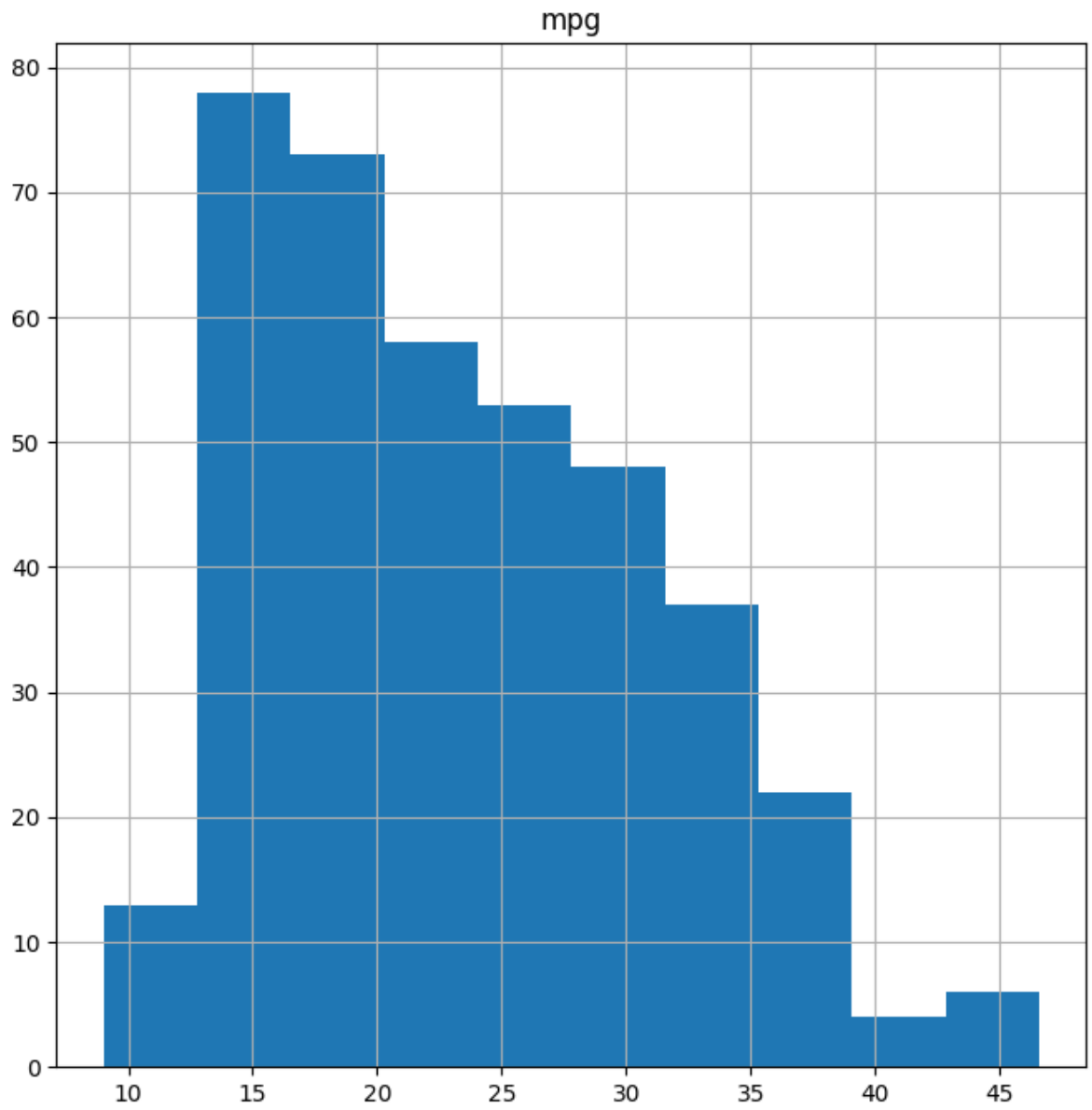

```
Auto.cylinders = pd.Series(Auto.cylinders, dtype="category")
Auto.cylinders.dtype
```

```
CategoricalDtype(categories=[3, 4, 5, 6, 8], ordered=False, categories_dtype=int64)
```

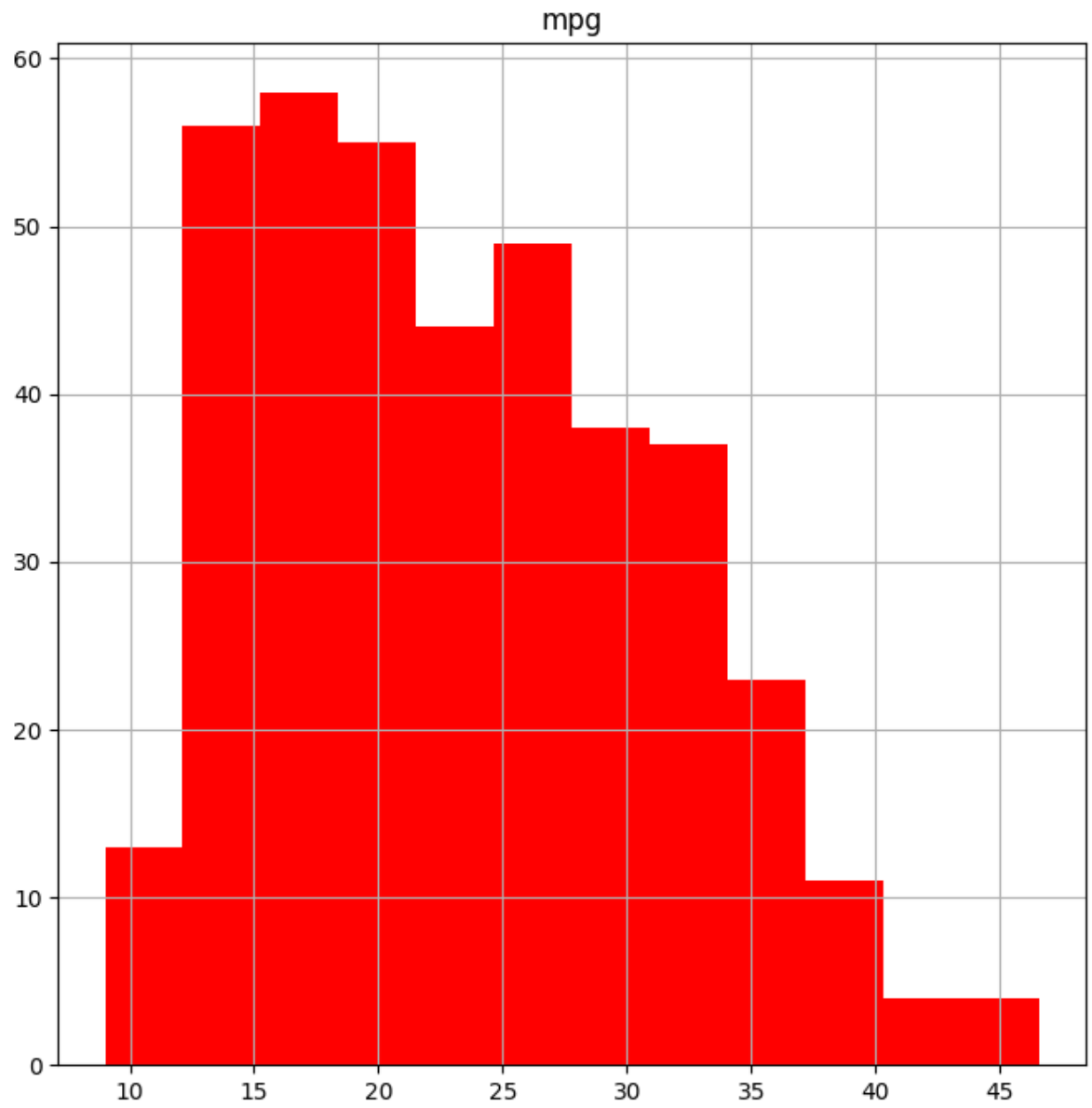
```
fig, ax = subplots(figsize=(8, 8))
Auto.boxplot("mpg", by="cylinders", ax=ax);
```




```
fig, ax = subplots(figsize=(8, 8));
Auto.hist("mpg", ax=ax);
```




```
fig, ax = subplots(figsize=(8, 8))
Auto.hist("mpg", color="red", bins=12, ax=ax);
```




```
Auto.hist?
```

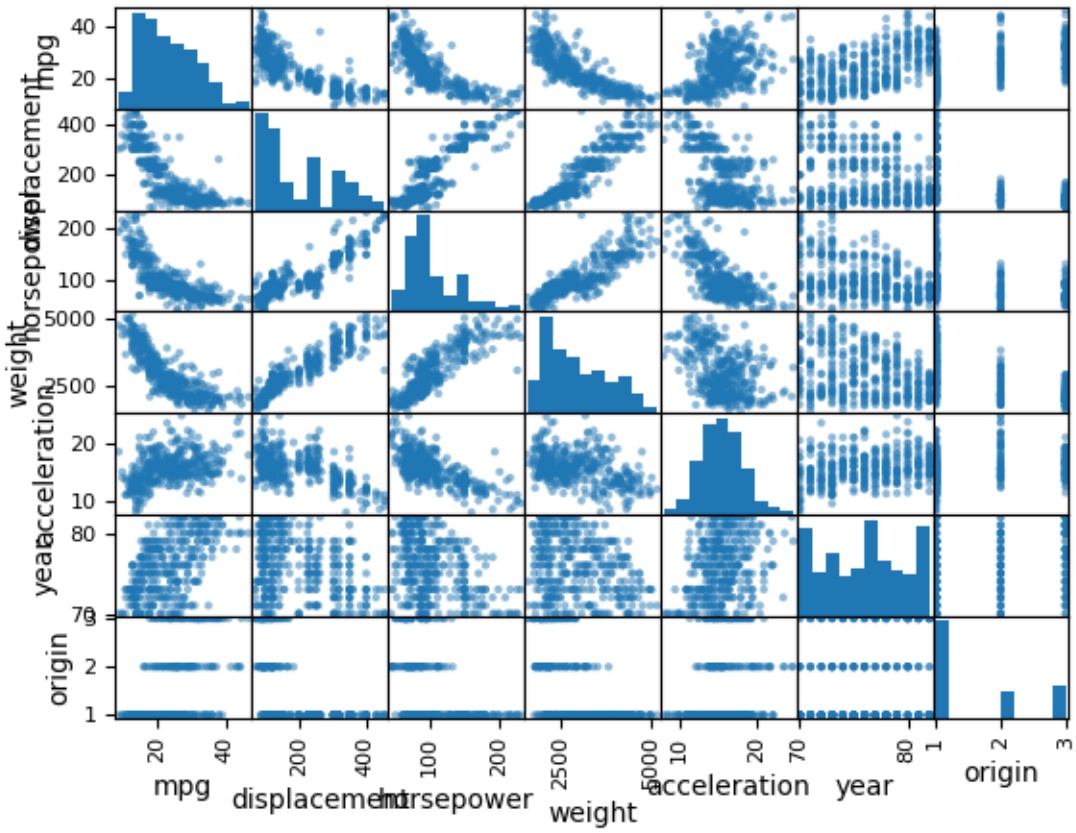
```
Signature:
Auto.hist(
    column: 'IndexLabel | None' = None,
    by=None,
    grid: 'bool' = True,
    xlabelsize: 'int | None' = None,
```

```
    xrot: 'float | None' = None,
    ylabelsize: 'int | None' = None,
    yrot: 'float | None' = None,
    ax=None,
    sharex: 'bool' = False,
    sharey: 'bool' = False,
    figsize: 'tuple[int, int] | None' = None,
    layout: 'tuple[int, int] | None' = None,
    bins: 'int | Sequence[int]' = 10,
    backend: 'str | None' = None,
    legend: 'bool' = False,
    **kwargs,
)
Docstring:
Make a histogram of the DataFrame's columns.

A `histogram`_ is a representation of the distribution of data.
This function calls :meth:`matplotlib.pyplot.hist`, on each series in
the DataFrame, resulting in one histogram per column.

.. _histogram: https://en.wikipedia.org/wiki/Histogram

Parameters
----------
data : DataFrame
    The pandas object holding the data.
column : str or sequence, optional
    If passed, will be used to limit data to a subset of columns.
by : object, optional
    If passed, then used to form histograms for separate groups.
grid : bool, default True
    Whether to show axis grid lines.
xlabelsize : int, default None
    If specified changes the x-axis label size.
xrot : float, default None
    Rotation of x axis labels. For example, a value of 90 displays the
    x labels rotated 90 degrees clockwise.
ylabelsize : int, default None
    If specified changes the y-axis label size.
yrot : float, default None
    Rotation of y axis labels. For example, a value of 90 displays the
    y labels rotated 90 degrees clockwise.
ax : Matplotlib axes object, default None
```

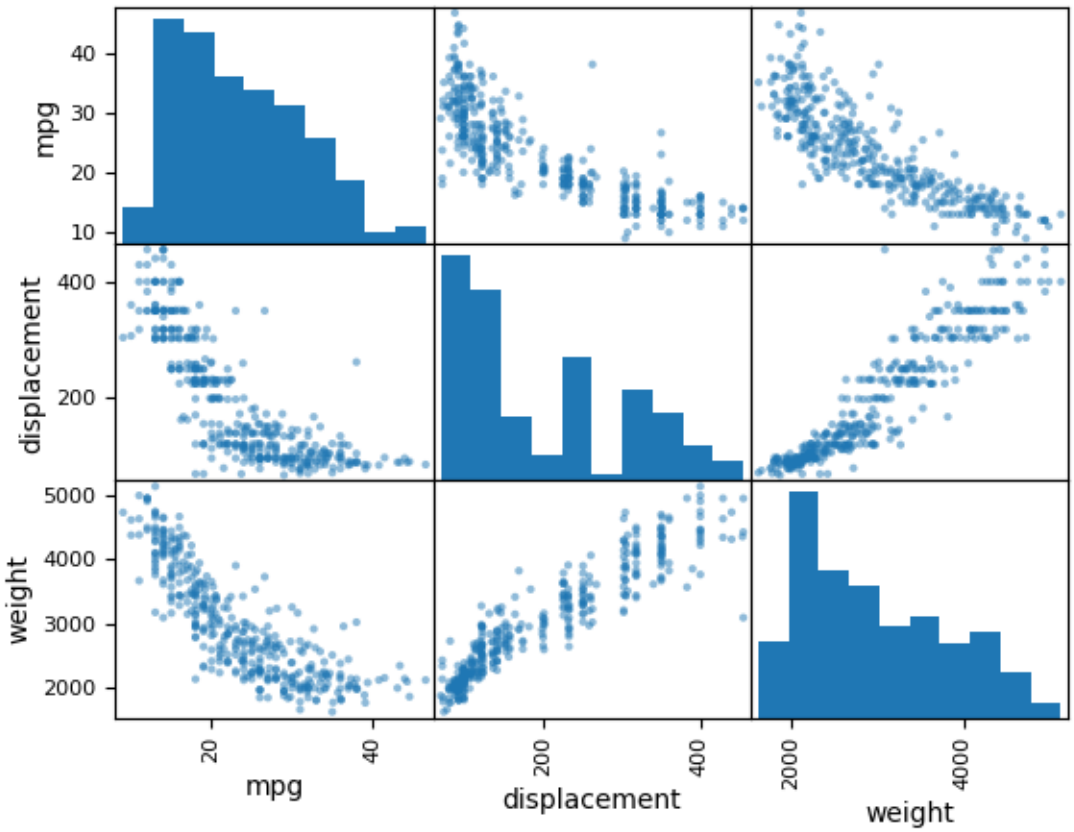
```
    The axes to plot the histogram on.
sharex : bool, default True if ax is None else False
    In case subplots=True, share x axis and set some x axis labels to
    invisible; defaults to True if ax is None otherwise False if an ax
    is passed in.
    Note that passing in both an ax and sharex=True will alter all x axis
    labels for all subplots in a figure.
sharey : bool, default False
    In case subplots=True, share y axis and set some y axis labels to
    invisible.
figsize : tuple, optional
    The size in inches of the figure to create. Uses the value in
    `matplotlib.rcParams` by default.
layout : tuple, optional
    Tuple of (rows, columns) for the layout of the histograms.
bins : int or sequence, default 10
    Number of histogram bins to be used. If an integer is given, bins + 1
    bin edges are calculated and returned. If bins is a sequence, gives
    bin edges, including left edge of first bin and right edge of last
    bin. In this case, bins is returned unmodified.

backend : str, default None
    Backend to use instead of the backend specified in the option
    ``plotting.backend``. For instance, 'matplotlib'. Alternatively, to
    specify the ``plotting.backend`` for the whole session, set
    ``pd.options.plotting.backend``.

legend : bool, default False
    Whether to show the legend.

**kwargs
    All other plotting keyword arguments to be passed to
    :meth:`matplotlib.pyplot.hist`.

Returns
-------
matplotlib.AxesSubplot or numpy.ndarray of them

See Also
--------
matplotlib.pyplot.hist : Plot a histogram using matplotlib.

Examples
```

```
--------
This example draws a histogram based on the length and width of
some animals, displayed in three bins

.. plot::
    :context: close-figs

    >>> data = {'length': [1.5, 0.5, 1.2, 0.9, 3],
    ...         'width': [0.7, 0.2, 0.15, 0.2, 1.1]}
    >>> index = ['pig', 'rabbit', 'duck', 'chicken', 'horse']
    >>> df = pd.DataFrame(data, index=index)
    >>> hist = df.hist(bins=3)
File:      ~/ISLP/islpenv/lib/python3.12/site-packages/pandas/plotting/_core.py
Type:      method
```

```
pd.plotting.scatter_matrix(Auto);
```



```
pd.plotting.scatter_matrix(Auto[["mpg", "displacement", "weight"]]);
```




```
Auto[["mpg", "weight"]].describe()
```

	mpg	weight
count	392.000000	392.000000
mean	23.445918	2977.584184
std	7.805007	849.402560
min	9.000000	1613.000000
25%	17.000000	2225.250000
50%	22.750000	2803.500000
75%	29.000000	3614.750000
max	46.600000	5140.000000

```
Auto[["cylinders"]].describe()
```

	cylinders
count	392
unique	5
top	4
freq	199

```
Auto[["mpg"]].describe()
```

	mpg
count	392.000000
mean	23.445918
std	7.805007
min	9.000000
25%	17.000000
50%	22.750000
75%	29.000000
max	46.600000

```
allDone()
```

```
<IPython.lib.display.Audio object>
```