```
Hansen@Hansens-MacBook-Pro:~ $ " clear

Hansen@Hansens-MacBook-Pro:~ $ " ipython
Python 2.7.12 |Anaconda custom (x86_64)| (default, Jul  2 2016, 17:43:17)
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import numpy as np

In [2]: data = [1, 2, 3]

In [3]: data.type
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-3-296913ab6e35> in <module>()
----> 1 data.type

AttributeError: 'list' object has no attribute 'type'

In [4]: data.type()
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-4-f5d544e46bb2> in <module>()
----> 1 data.type()

AttributeError: 'list' object has no attribute 'type'

In [5]: data.
data.append    data.extend    data.insert    data.remove    data.sort
data.count     data.index     data.pop       data.reverse

In [5]: data1 = [6 ,7.5 ,8, 0 ,1]

In [6]: ls
Applications/         Library/              Public/               curl
Creative Cloud Files/ MDIGB0_ideal.xbgf     PycharmProjects/      experiments/
Desktop/              Mathematica/          Untitled.ipynb        nohup.out
Developer/            Movies/               amber_test/           requirement.txt
Documents/            Music/                anaconda/             seaborn-data/
Downloads/            Parallels/            android/              solarized/
Dropbox/              Pictures/             bin/                  texput.log

In [7]: arr1 = np.array(data1)

In [8]: arr1.
arr1.T              arr1.copy           arr1.imag           arr1.ravel          arr1.sum
arr1.all            arr1.ctypes         arr1.item           arr1.real           arr1.swapaxes
arr1.any            arr1.cumprod        arr1.itemset        arr1.repeat         arr1.take
arr1.argmax         arr1.cumsum         arr1.itemsize       arr1.reshape        arr1.tobytes
arr1.argmin         arr1.data           arr1.max            arr1.resize         arr1.tofile
arr1.argpartition   arr1.diagonal       arr1.mean           arr1.round          arr1.tolist
arr1.argsort        arr1.dot            arr1.min            arr1.searchsorted   arr1.tostring
arr1.astype         arr1.dtype          arr1.nbytes         arr1.setfield       arr1.trace
arr1.base           arr1.dump           arr1.ndim           arr1.setflags       arr1.transpose
arr1.byteswap       arr1.dumps          arr1.newbyteorder   arr1.shape          arr1.var
arr1.choose         arr1.fill           arr1.nonzero        arr1.size           arr1.view
arr1.clip           arr1.flags          arr1.partition      arr1.sort
arr1.compress       arr1.flat           arr1.prod           arr1.squeeze
arr1.conj           arr1.flatten        arr1.ptp            arr1.std
arr1.conjugate      arr1.getfield       arr1.put            arr1.strides

In [8]: arr1.dtype
Out[8]: dtype('float64')

In [9]: arr1.shape
Out[9]: (5,)

In [10]: arr1.size
Out[10]: 5

In [11]: arr.ndim
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-ff9c433e9bba> in <module>()
----> 1 arr.ndim

NameError: name 'arr' is not defined

In [12]: a
%alias        %autoindent   all           and           apply         assert
```

```
%alias_magic  %automagic    amber_test/   android/      arr1
%autocall     abs           anaconda/     any           as

In [12]: arr1.ndim
Out[12]: 1

In [13]: str1 = 'kkk'

In [14]: arr2 = np.array(str1)

In [15]: arr2
Out[15]:
array('kkk',
      dtype='|S3')

In [16]: print arr2
kkk

In [17]: calibers = np.array([.22, .270, .357, .380, .44, .50], dtype = np.float64)

In [18]: calibers
Out[18]: array([ 0.22 ,  0.27 ,  0.357,  0.38 ,  0.44 ,  0.5  ])

In [19]: caliber.dtype
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-19-6e1e056b5924> in <module>()
----> 1 caliber.dtype

NameError: name 'caliber' is not defined

In [20]: calibers.dtype
Out[20]: dtype('float64')

In [21]: int_array.astype(calibers.dtype)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-bbc3393af524> in <module>()
----> 1 int_array.astype(calibers.dtype)

NameError: name 'int_array' is not defined

In [22]: np.empty?
Docstring:
empty(shape, dtype=float, order='C')

Return a new array of given shape and type, without initializing entries.

Parameters
----------
shape : int or tuple of int
    Shape of the empty array
dtype : data-type, optional
    Desired output data-type.
order : {'C', 'F'}, optional
    Whether to store multi-dimensional data in row-major
    (C-style) or column-major (Fortran-style) order in
    memory.

Returns
-------
out : ndarray
    Array of uninitialized (arbitrary) data of the given shape, dtype, and
    order.  Object arrays will be initialized to None.

See Also
--------
empty_like, zeros, ones
```

```
Notes
-----
`empty`, unlike `zeros`, does not set the array values to zero,
and may therefore be marginally faster.  On the other hand, it requires
the user to manually set all the values in the array, and should be
used with caution.

Examples
--------
>>> np.empty([2, 2])
array([[ -9.74499359e+001,   6.69583040e-309],
       [  2.13182611e-314,   3.06959433e-309]])         #random

>>> np.empty([2, 2], dtype=int)
array([[-1073741821, -1067949133],
       [  496041986,    19249760]])                     #random
Type:      builtin_function_or_method

In [23]: np.empty??
Docstring:
empty(shape, dtype=float, order='C')

Return a new array of given shape and type, without initializing entries.

Parameters
----------
shape : int or tuple of int
    Shape of the empty array
dtype : data-type, optional
    Desired output data-type.
order : {'C', 'F'}, optional
    Whether to store multi-dimensional data in row-major
    (C-style) or column-major (Fortran-style) order in
    memory.

Returns
-------
out : ndarray
    Array of uninitialized (arbitrary) data of the given shape, dtype, and
    order.  Object arrays will be initialized to None.

See Also
--------
empty_like, zeros, ones

Notes
-----
`empty`, unlike `zeros`, does not set the array values to zero,
and may therefore be marginally faster.  On the other hand, it requires
the user to manually set all the values in the array, and should be
used with caution.

Examples
--------
>>> np.empty([2, 2])
array([[ -9.74499359e+001,   6.69583040e-309],
       [  2.13182611e-314,   3.06959433e-309]])         #random

>>> np.empty([2, 2], dtype=int)
array([[-1073741821, -1067949133],
       [  496041986,    19249760]])                     #random
Type:      builtin_function_or_method

In [24]: np.empty(8, dtype = 'u4')
Out[24]:
array([      0,       0,       0,       0, 4194308,       1, 4233920,
            1], dtype=uint32)
```

```
In [25]: np.empty(8, dtype = '4')
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-25-0c729741e9ef> in <module>()
----> 1 np.empty(8, dtype = '4')

TypeError: data type "" not understood

In [26]: np.empty(8, dtype = 'i4')
Out[26]: array([0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)

In [27]: np.empty(8, dtype = 'i4')
Out[27]: array([0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)

In [28]: np.empty(8, dtype = 'f4')
Out[28]: array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.], dtype=float32)

In [29]: np.empty(8, dtype = 'int')
Out[29]: array([0, 0, 0, 0, 0, 0, 0, 0])

In [30]: np.empty([2,2,2], dtype = 'int')
Out[30]:
array([[[0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0]]])

In [31]: np.empty([2,2,2], dtype = int)
Out[31]:
array([[[0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0]]])

In [32]: np.empty([2,2], dtype = int)
Out[32]:
array([[0, 0],
       [0, 0]])

In [33]: arr3d = np.array([[[1,2,3],[4,5,6]],[[7,8,9], [10,11,12]]])

In [34]: arr3d
Out[34]:
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])

In [35]: names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])

In [36]: data = randn(7,4)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-36-cb7c4b32d8ce> in <module>()
----> 1 data = randn(7,4)

NameError: name 'randn' is not defined

In [37]: randn
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-37-398a9a52c608> in <module>()
----> 1 randn
```

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

In [38]: data = np.randn(7, 4)
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-38-999554d4bcda> in <module>()
----> 1 data = np.randn(7, 4)

AttributeError: 'module' object has no attribute 'randn'

In [39]: data = np.random.randn(7, 4)

In [40]: data
Out[40]:
array([[-1.52433546, -0.32703791, -0.84868919,  1.09946958],
       [-0.88259789, -0.06316187, -0.15245984,  0.04303572],
       [ 1.3625452 , -1.12694417,  0.31368561,  0.56077444],
       [-0.53090207, -0.99261563, -0.79623224,  0.0475795 ],
       [ 1.0203938 , -1.28077757, -0.5116905 , -0.35825162],
       [ 0.9362092 ,  1.0289195 ,  1.26334767,  0.98402902],
       [-1.32131556, -0.91838682, -0.4337269 ,  0.15345908]])

In [41]: names
Out[41]:
array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'],
      dtype='|S4')

In [42]: data[name == 'Bob', 3]
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-42-1fc590ae27e9> in <module>()
----> 1 data[name == 'Bob', 3]

NameError: name 'name' is not defined

In [43]: data[names == 'Bob', 3]
Out[43]: array([ 1.09946958,  0.0475795 ])

In [44]: data[(0, 3), 3]
Out[44]: array([ 1.09946958,  0.0475795 ])

In [45]: np.
Display all 583 possibilities? (y or n)
np.ALLOW_THREADS              np.empty_like               np.nested_iters
np.BUFSIZE                    np.equal                    np.newaxis
np.CLIP                       np.errstate                 np.newbuffer
np.ComplexWarning             np.euler_gamma              np.nextafter
np.DataSource                 np.exp                      np.nonzero
np.ERR_CALL                   np.exp2                     np.not_equal
np.ERR_DEFAULT                np.expand_dims              np.nper
np.ERR_IGNORE                 np.expm1                    np.npv
np.ERR_LOG                    np.extract                  np.numarray
np.ERR_PRINT                  np.eye                      np.number
np.ERR_RAISE                  np.fabs                     np.obj2sctype
np.ERR_WARN                   np.fastCopyAndTranspose     np.object
np.FLOATING_POINT_SUPPORT     np.fft                      np.object0
np.FPE_DIVIDEBYZERO           np.fill_diagonal            np.object_
np.FPE_INVALID                np.find_common_type         np.ogrid
np.FPE_OVERFLOW               np.finfo                    np.oldnumeric
np.FPE_UNDERFLOW              np.fix                      np.ones
np.False_                     np.flatiter                 np.ones_like
np.Inf                        np.flatnonzero              np.outer
np.Infinity                   np.flexible                 np.packbits
np.MAXDIMS                    np.fliplr                   np.pad
np.MAY_SHARE_BOUNDS           np.flipud                   np.partition
np.MAY_SHARE_EXACT            np.float                    np.percentile
np.MachAr                     np.float128                 np.pi
np.ModuleDeprecationWarning   np.float16                  np.piecewise
```

```
np.NAN
np.NINF
np.NZERO
np.NaN
np.PINF
np.PZERO
np.PackageLoader
np.RAISE
np.RankWarning
np.SHIFT_DIVIDEBYZERO
np.SHIFT_INVALID
np.SHIFT_OVERFLOW
np.SHIFT_UNDERFLOW
np.ScalarType
np.Tester
np.TooHardError
np.True_
np.UFUNC_BUFSIZE_DEFAULT
np.UFUNC_PYVALS_NAME
np.VisibleDeprecationWarning
np.WRAP
np.abs
np.absolute
np.absolute_import
np.add
np.add_docstring
np.add_newdoc
np.add_newdoc_ufunc
np.add_newdocs
np.alen
np.all
np.allclose
np.alltrue
np.alterdot
np.amax
np.amin
np.angle
np.any
np.append
np.apply_along_axis
np.apply_over_axes
np.arange
np.arccos
np.arccosh
np.arcsin
np.arcsinh
np.arctan
np.arctan2
np.arctanh
np.argmax
np.argmin
np.argpartition
np.argsort
np.argwhere
np.around
np.array
np.array2string
np.array_equal
np.array_equiv
np.array_repr
np.array_split
np.array_str
np.asanyarray
np.asarray
np.asarray_chkfinite
np.ascontiguousarray
np.asfarray
np.asfortranarray
np.float32
np.float64
np.float_
np.floating
np.floor
np.floor_divide
np.fmax
np.fmin
np.fmod
np.format_parser
np.frexp
np.frombuffer
np.fromfile
np.fromfunction
np.fromiter
np.frompyfunc
np.fromregex
np.fromstring
np.full
np.full_like
np.fv
np.generic
np.genfromtxt
np.get_array_wrap
np.get_include
np.get_printoptions
np.getbuffer
np.getbufsize
np.geterr
np.geterrcall
np.geterrobj
np.gradient
np.greater
np.greater_equal
np.half
np.hamming
np.hanning
np.histogram
np.histogram2d
np.histogramdd
np.hsplit
np.hstack
np.hypot
np.i0
np.identity
np.iinfo
np.imag
np.in1d
np.index_exp
np.indices
np.inexact
np.inf
np.info
np.infty
np.inner
np.insert
np.int
np.int0
np.int16
np.int32
np.int64
np.int8
np.int_
np.int_asbuffer
np.intc
np.integer
np.interp
np.intersect1d
np.pkgload
np.place
np.pmt
np.poly
np.poly1d
np.polyadd
np.polyder
np.polydiv
np.polyfit
np.polyint
np.polymul
np.polynomial
np.polysub
np.polyval
np.power
np.ppmt
np.print_function
np.prod
np.product
np.promote_types
np.ptp
np.put
np.putmask
np.pv
np.r_
np.rad2deg
np.radians
np.random
np.rank
np.rate
np.ravel
np.ravel_multi_index
np.real
np.real_if_close
np.rec
np.recarray
np.recfromcsv
np.recfromtxt
np.reciprocal
np.record
np.remainder
np.repeat
np.require
np.reshape
np.resize
np.restoredot
np.result_type
np.right_shift
np.rint
np.roll
np.rollaxis
np.roots
np.rot90
np.round
np.round_
np.row_stack
np.s_
np.safe_eval
np.save
np.savetxt
np.savez
np.savez_compressed
np.sctype2char
np.sctypeDict
np.sctypeNA
np.sctypes
np.searchsorted
np.select
```

```
np.asmatrix
np.asscalar
np.atleast_1d
np.atleast_2d
np.atleast_3d
np.average
np.bartlett
np.base_repr
np.bench
np.binary_repr
np.bincount
np.bitwise_and
np.bitwise_not
np.bitwise_or
np.bitwise_xor
np.blackman
np.bmat
np.bool
np.bool8
np.bool_
np.broadcast
np.broadcast_arrays
np.broadcast_to
np.busday_count
np.busday_offset
np.busdaycalendar
np.byte
np.byte_bounds
np.bytes_
np.c_
np.can_cast
np.cast
np.cbrt
np.cdouble
np.ceil
np.cfloat
np.char
np.character
np.chararray
np.choose
np.clip
np.clongdouble
np.clongfloat
np.column_stack
np.common_type
np.compare_chararrays
np.compat
np.complex
np.complex128
np.complex256
np.complex64
np.complex_
np.complexfloating
np.compress
np.concatenate
np.conj
np.conjugate
np.convolve
np.copy
np.copysign
np.copyto
np.core
np.corrcoef
np.correlate
np.cos
np.cosh
np.count_nonzero
np.cov
np.intp
np.invert
np.ipmt
np.irr
np.is_busday
np.isclose
np.iscomplex
np.iscomplexobj
np.isfinite
np.isfortran
np.isinf
np.isnan
np.isneginf
np.isposinf
np.isreal
np.isrealobj
np.isscalar
np.issctype
np.issubclass_
np.issubdtype
np.issubsctype
np.iterable
np.ix_
np.kaiser
np.kron
np.ldexp
np.left_shift
np.less
np.less_equal
np.lexsort
np.lib
np.linalg
np.linspace
np.little_endian
np.load
np.loads
np.loadtxt
np.log
np.log10
np.log1p
np.log2
np.logaddexp
np.logaddexp2
np.logical_and
np.logical_not
np.logical_or
np.logical_xor
np.logspace
np.long
np.longcomplex
np.longdouble
np.longfloat
np.longlong
np.lookfor
np.ma
np.mafromtxt
np.mask_indices
np.mat
np.math
np.matmul
np.matrix
np.matrixlib
np.max
np.maximum
np.maximum_sctype
np.may_share_memory
np.mean
np.median
np.set_numeric_ops
np.set_printoptions
np.set_string_function
np.setbufsize
np.setdiff1d
np.seterr
np.seterrcall
np.seterrobj
np.setxor1d
np.shape
np.shares_memory
np.short
np.show_config
np.sign
np.signbit
np.signedinteger
np.sin
np.sinc
np.single
np.singlecomplex
np.sinh
np.size
np.sometrue
np.sort
np.sort_complex
np.source
np.spacing
np.split
np.sqrt
np.square
np.squeeze
np.stack
np.std
np.str
np.str_
np.string0
np.string_
np.subtract
np.sum
np.swapaxes
np.take
np.tan
np.tanh
np.tensordot
np.test
np.testing
np.tile
np.timedelta64
np.trace
np.transpose
np.trapz
np.tri
np.tril
np.tril_indices
np.tril_indices_from
np.trim_zeros
np.triu
np.triu_indices
np.triu_indices_from
np.true_divide
np.trunc
np.typeDict
np.typeNA
np.typecodes
np.typename
np.ubyte
np.ufunc
np.uint
```

```
np.cross                    np.memmap                   np.uint0
np.csingle                  np.meshgrid                 np.uint16
np.ctypeslib                np.mgrid                    np.uint32
np.cumprod                  np.min                      np.uint64
np.cumproduct               np.min_scalar_type          np.uint8
np.cumsum                   np.minimum                  np.uintc
np.datetime64               np.mintypecode              np.uintp
np.datetime_as_string       np.mirr                     np.ulonglong
np.datetime_data            np.mod                      np.unicode
np.deg2rad                  np.modf                     np.unicode0
np.degrees                  np.moveaxis                 np.unicode_
np.delete                   np.msort                    np.union1d
np.deprecate                np.multiply                 np.unique
np.deprecate_with_doc       np.nan                      np.unpackbits
np.diag                     np.nan_to_num               np.unravel_index
np.diag_indices             np.nanargmax                np.unsignedinteger
np.diag_indices_from        np.nanargmin                np.unwrap
np.diagflat                 np.nanmax                   np.ushort
np.diagonal                 np.nanmean                  np.vander
np.diff                     np.nanmedian                np.var
np.digitize                 np.nanmin                   np.vdot
np.disp                     np.nanpercentile            np.vectorize
np.divide                   np.nanprod                  np.version
np.division                 np.nanstd                   np.void
np.dot                      np.nansum                   np.void0
np.double                   np.nanvar                   np.vsplit
np.dsplit                   np.nbytes                   np.vstack
np.dstack                   np.ndarray                  np.warnings
np.dtype                    np.ndenumerate              np.where
np.e                        np.ndfromtxt                np.who
np.ediff1d                  np.ndim                     np.zeros
np.einsum                   np.ndindex                  np.zeros_like
np.emath                    np.nditer
np.empty                    np.negative
```

```
In [45]: arr = np.random.randn(7)

In [46]: arr
Out[46]:
array([ 2.70499407, -0.34452092, -0.88641991, -1.25973759, -0.48000149,
        0.78200997, -1.63438477])

In [47]: arr *= 5

In [48]: arr *= 5

In [49]: arr \= 5
  File "<ipython-input-49-c5df5af48126>", line 1
    arr \= 5
        ^
SyntaxError: unexpected character after line continuation character


In [50]: arr /= 5

In [51]: arr
Out[51]:
array([ 13.52497037,  -1.72260462,  -4.43209957,  -6.29868793,
        -2.40000743,   3.91004984,  -8.17192383])

In [52]: np.modf(arr)
Out[52]:
(array([ 0.52497037, -0.72260462, -0.43209957, -0.29868793, -0.40000743,
         0.91004984, -0.17192383]),
 array([ 13.,  -1.,  -4.,  -6.,  -2.,   3.,  -8.]))

In [53]: np.meshgrid?
```

```
In [54]:

In [54]: points = np.arange(-5, 5, 0.01)

In [55]: po
%popd   points  pow

In [55]: points
Out[55]:
array([ -5.00000000e+00,  -4.99000000e+00,  -4.98000000e+00,
        -4.97000000e+00,  -4.96000000e+00,  -4.95000000e+00,
        -4.94000000e+00,  -4.93000000e+00,  -4.92000000e+00,
        -4.91000000e+00,  -4.90000000e+00,  -4.89000000e+00,
        -4.88000000e+00,  -4.87000000e+00,  -4.86000000e+00,
        -4.85000000e+00,  -4.84000000e+00,  -4.83000000e+00,
        -4.82000000e+00,  -4.81000000e+00,  -4.80000000e+00,
        -4.79000000e+00,  -4.78000000e+00,  -4.77000000e+00,
        -4.76000000e+00,  -4.75000000e+00,  -4.74000000e+00,
        -4.73000000e+00,  -4.72000000e+00,  -4.71000000e+00,
        -4.70000000e+00,  -4.69000000e+00,  -4.68000000e+00,
        -4.67000000e+00,  -4.66000000e+00,  -4.65000000e+00,
        -4.64000000e+00,  -4.63000000e+00,  -4.62000000e+00,
        -4.61000000e+00,  -4.60000000e+00,  -4.59000000e+00,
        -4.58000000e+00,  -4.57000000e+00,  -4.56000000e+00,
        -4.55000000e+00,  -4.54000000e+00,  -4.53000000e+00,
        -4.52000000e+00,  -4.51000000e+00,  -4.50000000e+00,
        -4.49000000e+00,  -4.48000000e+00,  -4.47000000e+00,
        -4.46000000e+00,  -4.45000000e+00,  -4.44000000e+00,
        -4.43000000e+00,  -4.42000000e+00,  -4.41000000e+00,
        -4.40000000e+00,  -4.39000000e+00,  -4.38000000e+00,
        -4.37000000e+00,  -4.36000000e+00,  -4.35000000e+00,
        -4.34000000e+00,  -4.33000000e+00,  -4.32000000e+00,
        -4.31000000e+00,  -4.30000000e+00,  -4.29000000e+00,
        -4.28000000e+00,  -4.27000000e+00,  -4.26000000e+00,
        -4.25000000e+00,  -4.24000000e+00,  -4.23000000e+00,
        -4.22000000e+00,  -4.21000000e+00,  -4.20000000e+00,
        -4.19000000e+00,  -4.18000000e+00,  -4.17000000e+00,
        -4.16000000e+00,  -4.15000000e+00,  -4.14000000e+00,
        -4.13000000e+00,  -4.12000000e+00,  -4.11000000e+00,
        -4.10000000e+00,  -4.09000000e+00,  -4.08000000e+00,
        -4.07000000e+00,  -4.06000000e+00,  -4.05000000e+00,
        -4.04000000e+00,  -4.03000000e+00,  -4.02000000e+00,
        -4.01000000e+00,  -4.00000000e+00,  -3.99000000e+00,
        -3.98000000e+00,  -3.97000000e+00,  -3.96000000e+00,
        -3.95000000e+00,  -3.94000000e+00,  -3.93000000e+00,
        -3.92000000e+00,  -3.91000000e+00,  -3.90000000e+00,
        -3.89000000e+00,  -3.88000000e+00,  -3.87000000e+00,
        -3.86000000e+00,  -3.85000000e+00,  -3.84000000e+00,
        -3.83000000e+00,  -3.82000000e+00,  -3.81000000e+00,
        -3.80000000e+00,  -3.79000000e+00,  -3.78000000e+00,
        -3.77000000e+00,  -3.76000000e+00,  -3.75000000e+00,
        -3.74000000e+00,  -3.73000000e+00,  -3.72000000e+00,
        -3.71000000e+00,  -3.70000000e+00,  -3.69000000e+00,
        -3.68000000e+00,  -3.67000000e+00,  -3.66000000e+00,
        -3.65000000e+00,  -3.64000000e+00,  -3.63000000e+00,
        -3.62000000e+00,  -3.61000000e+00,  -3.60000000e+00,
        -3.59000000e+00,  -3.58000000e+00,  -3.57000000e+00,
        -3.56000000e+00,  -3.55000000e+00,  -3.54000000e+00,
        -3.53000000e+00,  -3.52000000e+00,  -3.51000000e+00,
        -3.50000000e+00,  -3.49000000e+00,  -3.48000000e+00,
        -3.47000000e+00,  -3.46000000e+00,  -3.45000000e+00,
        -3.44000000e+00,  -3.43000000e+00,  -3.42000000e+00,
        -3.41000000e+00,  -3.40000000e+00,  -3.39000000e+00,
        -3.38000000e+00,  -3.37000000e+00,  -3.36000000e+00,
        -3.35000000e+00,  -3.34000000e+00,  -3.33000000e+00,
        -3.32000000e+00,  -3.31000000e+00,  -3.30000000e+00,
        -3.29000000e+00,  -3.28000000e+00,  -3.27000000e+00,
        -3.26000000e+00,  -3.25000000e+00,  -3.24000000e+00,
```

-3.23000000e+00,  -3.22000000e+00,  -3.21000000e+00,
-3.20000000e+00,  -3.19000000e+00,  -3.18000000e+00,
-3.17000000e+00,  -3.16000000e+00,  -3.15000000e+00,
-3.14000000e+00,  -3.13000000e+00,  -3.12000000e+00,
-3.11000000e+00,  -3.10000000e+00,  -3.09000000e+00,
-3.08000000e+00,  -3.07000000e+00,  -3.06000000e+00,
-3.05000000e+00,  -3.04000000e+00,  -3.03000000e+00,
-3.02000000e+00,  -3.01000000e+00,  -3.00000000e+00,
-2.99000000e+00,  -2.98000000e+00,  -2.97000000e+00,
-2.96000000e+00,  -2.95000000e+00,  -2.94000000e+00,
-2.93000000e+00,  -2.92000000e+00,  -2.91000000e+00,
-2.90000000e+00,  -2.89000000e+00,  -2.88000000e+00,
-2.87000000e+00,  -2.86000000e+00,  -2.85000000e+00,
-2.84000000e+00,  -2.83000000e+00,  -2.82000000e+00,
-2.81000000e+00,  -2.80000000e+00,  -2.79000000e+00,
-2.78000000e+00,  -2.77000000e+00,  -2.76000000e+00,
-2.75000000e+00,  -2.74000000e+00,  -2.73000000e+00,
-2.72000000e+00,  -2.71000000e+00,  -2.70000000e+00,
-2.69000000e+00,  -2.68000000e+00,  -2.67000000e+00,
-2.66000000e+00,  -2.65000000e+00,  -2.64000000e+00,
-2.63000000e+00,  -2.62000000e+00,  -2.61000000e+00,
-2.60000000e+00,  -2.59000000e+00,  -2.58000000e+00,
-2.57000000e+00,  -2.56000000e+00,  -2.55000000e+00,
-2.54000000e+00,  -2.53000000e+00,  -2.52000000e+00,
-2.51000000e+00,  -2.50000000e+00,  -2.49000000e+00,
-2.48000000e+00,  -2.47000000e+00,  -2.46000000e+00,
-2.45000000e+00,  -2.44000000e+00,  -2.43000000e+00,
-2.42000000e+00,  -2.41000000e+00,  -2.40000000e+00,
-2.39000000e+00,  -2.38000000e+00,  -2.37000000e+00,
-2.36000000e+00,  -2.35000000e+00,  -2.34000000e+00,
-2.33000000e+00,  -2.32000000e+00,  -2.31000000e+00,
-2.30000000e+00,  -2.29000000e+00,  -2.28000000e+00,
-2.27000000e+00,  -2.26000000e+00,  -2.25000000e+00,
-2.24000000e+00,  -2.23000000e+00,  -2.22000000e+00,
-2.21000000e+00,  -2.20000000e+00,  -2.19000000e+00,
-2.18000000e+00,  -2.17000000e+00,  -2.16000000e+00,
-2.15000000e+00,  -2.14000000e+00,  -2.13000000e+00,
-2.12000000e+00,  -2.11000000e+00,  -2.10000000e+00,
-2.09000000e+00,  -2.08000000e+00,  -2.07000000e+00,
-2.06000000e+00,  -2.05000000e+00,  -2.04000000e+00,
-2.03000000e+00,  -2.02000000e+00,  -2.01000000e+00,
-2.00000000e+00,  -1.99000000e+00,  -1.98000000e+00,
-1.97000000e+00,  -1.96000000e+00,  -1.95000000e+00,
-1.94000000e+00,  -1.93000000e+00,  -1.92000000e+00,
-1.91000000e+00,  -1.90000000e+00,  -1.89000000e+00,
-1.88000000e+00,  -1.87000000e+00,  -1.86000000e+00,
-1.85000000e+00,  -1.84000000e+00,  -1.83000000e+00,
-1.82000000e+00,  -1.81000000e+00,  -1.80000000e+00,
-1.79000000e+00,  -1.78000000e+00,  -1.77000000e+00,
-1.76000000e+00,  -1.75000000e+00,  -1.74000000e+00,
-1.73000000e+00,  -1.72000000e+00,  -1.71000000e+00,
-1.70000000e+00,  -1.69000000e+00,  -1.68000000e+00,
-1.67000000e+00,  -1.66000000e+00,  -1.65000000e+00,
-1.64000000e+00,  -1.63000000e+00,  -1.62000000e+00,
-1.61000000e+00,  -1.60000000e+00,  -1.59000000e+00,
-1.58000000e+00,  -1.57000000e+00,  -1.56000000e+00,
-1.55000000e+00,  -1.54000000e+00,  -1.53000000e+00,
-1.52000000e+00,  -1.51000000e+00,  -1.50000000e+00,
-1.49000000e+00,  -1.48000000e+00,  -1.47000000e+00,
-1.46000000e+00,  -1.45000000e+00,  -1.44000000e+00,
-1.43000000e+00,  -1.42000000e+00,  -1.41000000e+00,
-1.40000000e+00,  -1.39000000e+00,  -1.38000000e+00,
-1.37000000e+00,  -1.36000000e+00,  -1.35000000e+00,
-1.34000000e+00,  -1.33000000e+00,  -1.32000000e+00,
-1.31000000e+00,  -1.30000000e+00,  -1.29000000e+00,
-1.28000000e+00,  -1.27000000e+00,  -1.26000000e+00,
-1.25000000e+00,  -1.24000000e+00,  -1.23000000e+00,
-1.22000000e+00,  -1.21000000e+00,  -1.20000000e+00,

```
-1.19000000e+00,  -1.18000000e+00,  -1.17000000e+00,
-1.16000000e+00,  -1.15000000e+00,  -1.14000000e+00,
-1.13000000e+00,  -1.12000000e+00,  -1.11000000e+00,
-1.10000000e+00,  -1.09000000e+00,  -1.08000000e+00,
-1.07000000e+00,  -1.06000000e+00,  -1.05000000e+00,
-1.04000000e+00,  -1.03000000e+00,  -1.02000000e+00,
-1.01000000e+00,  -1.00000000e+00,  -9.90000000e-01,
-9.80000000e-01,  -9.70000000e-01,  -9.60000000e-01,
-9.50000000e-01,  -9.40000000e-01,  -9.30000000e-01,
-9.20000000e-01,  -9.10000000e-01,  -9.00000000e-01,
-8.90000000e-01,  -8.80000000e-01,  -8.70000000e-01,
-8.60000000e-01,  -8.50000000e-01,  -8.40000000e-01,
-8.30000000e-01,  -8.20000000e-01,  -8.10000000e-01,
-8.00000000e-01,  -7.90000000e-01,  -7.80000000e-01,
-7.70000000e-01,  -7.60000000e-01,  -7.50000000e-01,
-7.40000000e-01,  -7.30000000e-01,  -7.20000000e-01,
-7.10000000e-01,  -7.00000000e-01,  -6.90000000e-01,
-6.80000000e-01,  -6.70000000e-01,  -6.60000000e-01,
-6.50000000e-01,  -6.40000000e-01,  -6.30000000e-01,
-6.20000000e-01,  -6.10000000e-01,  -6.00000000e-01,
-5.90000000e-01,  -5.80000000e-01,  -5.70000000e-01,
-5.60000000e-01,  -5.50000000e-01,  -5.40000000e-01,
-5.30000000e-01,  -5.20000000e-01,  -5.10000000e-01,
-5.00000000e-01,  -4.90000000e-01,  -4.80000000e-01,
-4.70000000e-01,  -4.60000000e-01,  -4.50000000e-01,
-4.40000000e-01,  -4.30000000e-01,  -4.20000000e-01,
-4.10000000e-01,  -4.00000000e-01,  -3.90000000e-01,
-3.80000000e-01,  -3.70000000e-01,  -3.60000000e-01,
-3.50000000e-01,  -3.40000000e-01,  -3.30000000e-01,
-3.20000000e-01,  -3.10000000e-01,  -3.00000000e-01,
-2.90000000e-01,  -2.80000000e-01,  -2.70000000e-01,
-2.60000000e-01,  -2.50000000e-01,  -2.40000000e-01,
-2.30000000e-01,  -2.20000000e-01,  -2.10000000e-01,
-2.00000000e-01,  -1.90000000e-01,  -1.80000000e-01,
-1.70000000e-01,  -1.60000000e-01,  -1.50000000e-01,
-1.40000000e-01,  -1.30000000e-01,  -1.20000000e-01,
-1.10000000e-01,  -1.00000000e-01,  -9.00000000e-02,
-8.00000000e-02,  -7.00000000e-02,  -6.00000000e-02,
-5.00000000e-02,  -4.00000000e-02,  -3.00000000e-02,
-2.00000000e-02,  -1.00000000e-02,  -1.06581410e-13,
 1.00000000e-02,   2.00000000e-02,   3.00000000e-02,
 4.00000000e-02,   5.00000000e-02,   6.00000000e-02,
 7.00000000e-02,   8.00000000e-02,   9.00000000e-02,
 1.00000000e-01,   1.10000000e-01,   1.20000000e-01,
 1.30000000e-01,   1.40000000e-01,   1.50000000e-01,
 1.60000000e-01,   1.70000000e-01,   1.80000000e-01,
 1.90000000e-01,   2.00000000e-01,   2.10000000e-01,
 2.20000000e-01,   2.30000000e-01,   2.40000000e-01,
 2.50000000e-01,   2.60000000e-01,   2.70000000e-01,
 2.80000000e-01,   2.90000000e-01,   3.00000000e-01,
 3.10000000e-01,   3.20000000e-01,   3.30000000e-01,
 3.40000000e-01,   3.50000000e-01,   3.60000000e-01,
 3.70000000e-01,   3.80000000e-01,   3.90000000e-01,
 4.00000000e-01,   4.10000000e-01,   4.20000000e-01,
 4.30000000e-01,   4.40000000e-01,   4.50000000e-01,
 4.60000000e-01,   4.70000000e-01,   4.80000000e-01,
 4.90000000e-01,   5.00000000e-01,   5.10000000e-01,
 5.20000000e-01,   5.30000000e-01,   5.40000000e-01,
 5.50000000e-01,   5.60000000e-01,   5.70000000e-01,
 5.80000000e-01,   5.90000000e-01,   6.00000000e-01,
 6.10000000e-01,   6.20000000e-01,   6.30000000e-01,
 6.40000000e-01,   6.50000000e-01,   6.60000000e-01,
 6.70000000e-01,   6.80000000e-01,   6.90000000e-01,
 7.00000000e-01,   7.10000000e-01,   7.20000000e-01,
 7.30000000e-01,   7.40000000e-01,   7.50000000e-01,
 7.60000000e-01,   7.70000000e-01,   7.80000000e-01,
 7.90000000e-01,   8.00000000e-01,   8.10000000e-01,
 8.20000000e-01,   8.30000000e-01,   8.40000000e-01,
```

```
8.50000000e-01,   8.60000000e-01,   8.70000000e-01,
8.80000000e-01,   8.90000000e-01,   9.00000000e-01,
9.10000000e-01,   9.20000000e-01,   9.30000000e-01,
9.40000000e-01,   9.50000000e-01,   9.60000000e-01,
9.70000000e-01,   9.80000000e-01,   9.90000000e-01,
1.00000000e+00,   1.01000000e+00,   1.02000000e+00,
1.03000000e+00,   1.04000000e+00,   1.05000000e+00,
1.06000000e+00,   1.07000000e+00,   1.08000000e+00,
1.09000000e+00,   1.10000000e+00,   1.11000000e+00,
1.12000000e+00,   1.13000000e+00,   1.14000000e+00,
1.15000000e+00,   1.16000000e+00,   1.17000000e+00,
1.18000000e+00,   1.19000000e+00,   1.20000000e+00,
1.21000000e+00,   1.22000000e+00,   1.23000000e+00,
1.24000000e+00,   1.25000000e+00,   1.26000000e+00,
1.27000000e+00,   1.28000000e+00,   1.29000000e+00,
1.30000000e+00,   1.31000000e+00,   1.32000000e+00,
1.33000000e+00,   1.34000000e+00,   1.35000000e+00,
1.36000000e+00,   1.37000000e+00,   1.38000000e+00,
1.39000000e+00,   1.40000000e+00,   1.41000000e+00,
1.42000000e+00,   1.43000000e+00,   1.44000000e+00,
1.45000000e+00,   1.46000000e+00,   1.47000000e+00,
1.48000000e+00,   1.49000000e+00,   1.50000000e+00,
1.51000000e+00,   1.52000000e+00,   1.53000000e+00,
1.54000000e+00,   1.55000000e+00,   1.56000000e+00,
1.57000000e+00,   1.58000000e+00,   1.59000000e+00,
1.60000000e+00,   1.61000000e+00,   1.62000000e+00,
1.63000000e+00,   1.64000000e+00,   1.65000000e+00,
1.66000000e+00,   1.67000000e+00,   1.68000000e+00,
1.69000000e+00,   1.70000000e+00,   1.71000000e+00,
1.72000000e+00,   1.73000000e+00,   1.74000000e+00,
1.75000000e+00,   1.76000000e+00,   1.77000000e+00,
1.78000000e+00,   1.79000000e+00,   1.80000000e+00,
1.81000000e+00,   1.82000000e+00,   1.83000000e+00,
1.84000000e+00,   1.85000000e+00,   1.86000000e+00,
1.87000000e+00,   1.88000000e+00,   1.89000000e+00,
1.90000000e+00,   1.91000000e+00,   1.92000000e+00,
1.93000000e+00,   1.94000000e+00,   1.95000000e+00,
1.96000000e+00,   1.97000000e+00,   1.98000000e+00,
1.99000000e+00,   2.00000000e+00,   2.01000000e+00,
2.02000000e+00,   2.03000000e+00,   2.04000000e+00,
2.05000000e+00,   2.06000000e+00,   2.07000000e+00,
2.08000000e+00,   2.09000000e+00,   2.10000000e+00,
2.11000000e+00,   2.12000000e+00,   2.13000000e+00,
2.14000000e+00,   2.15000000e+00,   2.16000000e+00,
2.17000000e+00,   2.18000000e+00,   2.19000000e+00,
2.20000000e+00,   2.21000000e+00,   2.22000000e+00,
2.23000000e+00,   2.24000000e+00,   2.25000000e+00,
2.26000000e+00,   2.27000000e+00,   2.28000000e+00,
2.29000000e+00,   2.30000000e+00,   2.31000000e+00,
2.32000000e+00,   2.33000000e+00,   2.34000000e+00,
2.35000000e+00,   2.36000000e+00,   2.37000000e+00,
2.38000000e+00,   2.39000000e+00,   2.40000000e+00,
2.41000000e+00,   2.42000000e+00,   2.43000000e+00,
2.44000000e+00,   2.45000000e+00,   2.46000000e+00,
2.47000000e+00,   2.48000000e+00,   2.49000000e+00,
2.50000000e+00,   2.51000000e+00,   2.52000000e+00,
2.53000000e+00,   2.54000000e+00,   2.55000000e+00,
2.56000000e+00,   2.57000000e+00,   2.58000000e+00,
2.59000000e+00,   2.60000000e+00,   2.61000000e+00,
2.62000000e+00,   2.63000000e+00,   2.64000000e+00,
2.65000000e+00,   2.66000000e+00,   2.67000000e+00,
2.68000000e+00,   2.69000000e+00,   2.70000000e+00,
2.71000000e+00,   2.72000000e+00,   2.73000000e+00,
2.74000000e+00,   2.75000000e+00,   2.76000000e+00,
2.77000000e+00,   2.78000000e+00,   2.79000000e+00,
2.80000000e+00,   2.81000000e+00,   2.82000000e+00,
2.83000000e+00,   2.84000000e+00,   2.85000000e+00,
2.86000000e+00,   2.87000000e+00,   2.88000000e+00,
```

2.89000000e+00,  2.90000000e+00,  2.91000000e+00,
2.92000000e+00,  2.93000000e+00,  2.94000000e+00,
2.95000000e+00,  2.96000000e+00,  2.97000000e+00,
2.98000000e+00,  2.99000000e+00,  3.00000000e+00,
3.01000000e+00,  3.02000000e+00,  3.03000000e+00,
3.04000000e+00,  3.05000000e+00,  3.06000000e+00,
3.07000000e+00,  3.08000000e+00,  3.09000000e+00,
3.10000000e+00,  3.11000000e+00,  3.12000000e+00,
3.13000000e+00,  3.14000000e+00,  3.15000000e+00,
3.16000000e+00,  3.17000000e+00,  3.18000000e+00,
3.19000000e+00,  3.20000000e+00,  3.21000000e+00,
3.22000000e+00,  3.23000000e+00,  3.24000000e+00,
3.25000000e+00,  3.26000000e+00,  3.27000000e+00,
3.28000000e+00,  3.29000000e+00,  3.30000000e+00,
3.31000000e+00,  3.32000000e+00,  3.33000000e+00,
3.34000000e+00,  3.35000000e+00,  3.36000000e+00,
3.37000000e+00,  3.38000000e+00,  3.39000000e+00,
3.40000000e+00,  3.41000000e+00,  3.42000000e+00,
3.43000000e+00,  3.44000000e+00,  3.45000000e+00,
3.46000000e+00,  3.47000000e+00,  3.48000000e+00,
3.49000000e+00,  3.50000000e+00,  3.51000000e+00,
3.52000000e+00,  3.53000000e+00,  3.54000000e+00,
3.55000000e+00,  3.56000000e+00,  3.57000000e+00,
3.58000000e+00,  3.59000000e+00,  3.60000000e+00,
3.61000000e+00,  3.62000000e+00,  3.63000000e+00,
3.64000000e+00,  3.65000000e+00,  3.66000000e+00,
3.67000000e+00,  3.68000000e+00,  3.69000000e+00,
3.70000000e+00,  3.71000000e+00,  3.72000000e+00,
3.73000000e+00,  3.74000000e+00,  3.75000000e+00,
3.76000000e+00,  3.77000000e+00,  3.78000000e+00,
3.79000000e+00,  3.80000000e+00,  3.81000000e+00,
3.82000000e+00,  3.83000000e+00,  3.84000000e+00,
3.85000000e+00,  3.86000000e+00,  3.87000000e+00,
3.88000000e+00,  3.89000000e+00,  3.90000000e+00,
3.91000000e+00,  3.92000000e+00,  3.93000000e+00,
3.94000000e+00,  3.95000000e+00,  3.96000000e+00,
3.97000000e+00,  3.98000000e+00,  3.99000000e+00,
4.00000000e+00,  4.01000000e+00,  4.02000000e+00,
4.03000000e+00,  4.04000000e+00,  4.05000000e+00,
4.06000000e+00,  4.07000000e+00,  4.08000000e+00,
4.09000000e+00,  4.10000000e+00,  4.11000000e+00,
4.12000000e+00,  4.13000000e+00,  4.14000000e+00,
4.15000000e+00,  4.16000000e+00,  4.17000000e+00,
4.18000000e+00,  4.19000000e+00,  4.20000000e+00,
4.21000000e+00,  4.22000000e+00,  4.23000000e+00,
4.24000000e+00,  4.25000000e+00,  4.26000000e+00,
4.27000000e+00,  4.28000000e+00,  4.29000000e+00,
4.30000000e+00,  4.31000000e+00,  4.32000000e+00,
4.33000000e+00,  4.34000000e+00,  4.35000000e+00,
4.36000000e+00,  4.37000000e+00,  4.38000000e+00,
4.39000000e+00,  4.40000000e+00,  4.41000000e+00,
4.42000000e+00,  4.43000000e+00,  4.44000000e+00,
4.45000000e+00,  4.46000000e+00,  4.47000000e+00,
4.48000000e+00,  4.49000000e+00,  4.50000000e+00,
4.51000000e+00,  4.52000000e+00,  4.53000000e+00,
4.54000000e+00,  4.55000000e+00,  4.56000000e+00,
4.57000000e+00,  4.58000000e+00,  4.59000000e+00,
4.60000000e+00,  4.61000000e+00,  4.62000000e+00,
4.63000000e+00,  4.64000000e+00,  4.65000000e+00,
4.66000000e+00,  4.67000000e+00,  4.68000000e+00,
4.69000000e+00,  4.70000000e+00,  4.71000000e+00,
4.72000000e+00,  4.73000000e+00,  4.74000000e+00,
4.75000000e+00,  4.76000000e+00,  4.77000000e+00,
4.78000000e+00,  4.79000000e+00,  4.80000000e+00,
4.81000000e+00,  4.82000000e+00,  4.83000000e+00,
4.84000000e+00,  4.85000000e+00,  4.86000000e+00,
4.87000000e+00,  4.88000000e+00,  4.89000000e+00,
4.90000000e+00,  4.91000000e+00,  4.92000000e+00,

```
         4.93000000e+00,   4.94000000e+00,   4.95000000e+00,
         4.96000000e+00,   4.97000000e+00,   4.98000000e+00,
         4.99000000e+00])

In [56]: xs, ys = np.meshgrid(points, points)

In [57]: xs
Out[57]:
array([[-5.  , -4.99, -4.98, ...,  4.97,  4.98,  4.99],
       [-5.  , -4.99, -4.98, ...,  4.97,  4.98,  4.99],
       [-5.  , -4.99, -4.98, ...,  4.97,  4.98,  4.99],
       ...,
       [-5.  , -4.99, -4.98, ...,  4.97,  4.98,  4.99],
       [-5.  , -4.99, -4.98, ...,  4.97,  4.98,  4.99],
       [-5.  , -4.99, -4.98, ...,  4.97,  4.98,  4.99]])

In [58]: ys
Out[58]:
array([[-5.  , -5.  , -5.  , ..., -5.  , -5.  , -5.  ],
       [-4.99, -4.99, -4.99, ..., -4.99, -4.99, -4.99],
       [-4.98, -4.98, -4.98, ..., -4.98, -4.98, -4.98],
       ...,
       [ 4.97,  4.97,  4.97, ...,  4.97,  4.97,  4.97],
       [ 4.98,  4.98,  4.98, ...,  4.98,  4.98,  4.98],
       [ 4.99,  4.99,  4.99, ...,  4.99,  4.99,  4.99]])

In [59]: import matplotlib.pyplot as plt

In [60]: %matplotlib
Using matplotlib backend: MacOSX

In [61]: z = np.sqrt(xs**2 + ys**2)

In [62]: z
Out[62]:
array([[ 7.07106781,  7.06400028,  7.05693985, ...,  7.04988652,
         7.05693985,  7.06400028],
       [ 7.06400028,  7.05692568,  7.04985815, ...,  7.04279774,
         7.04985815,  7.05692568],
       [ 7.05693985,  7.04985815,  7.04278354, ...,  7.03571603,
         7.04278354,  7.04985815],
       ...,
       [ 7.04988652,  7.04279774,  7.03571603, ...,  7.0286414 ,
         7.03571603,  7.04279774],
       [ 7.05693985,  7.04985815,  7.04278354, ...,  7.03571603,
         7.04278354,  7.04985815],
       [ 7.06400028,  7.05692568,  7.04985815, ...,  7.04279774,
         7.04985815,  7.05692568]])

In [63]: plt.imshow(z, cmap = plt.cm.grey); ply.colorbar()
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-63-5a86dbc5197e> in <module>()
----> 1 plt.imshow(z, cmap = plt.cm.grey); ply.colorbar()

AttributeError: 'module' object has no attribute 'grey'

In [64]: plt.imshow(z, cmap = plt.cm.gray); ply.colorbar()
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-64-ebdb61bf9200> in <module>()
----> 1 plt.imshow(z, cmap = plt.cm.gray); ply.colorbar()

NameError: name 'ply' is not defined

In [65]: plt.imshow(z, cmap = plt.cm.gray); plt.colorbar()
Out[65]: <matplotlib.colorbar.Colorbar at 0x114424a10>
```

```
In [66]: arr = np.random.randn(4, 4)

In [67]: arr
Out[67]:
array([[ 0.99715976, -0.99285941, -1.29983152,  0.26311053],
       [ 1.44502703,  0.76717446,  0.45337043,  0.19472739],
       [ 0.48680544,  0.51659521, -0.17987949, -0.50011625],
       [ 1.21627682,  1.46086937,  0.4304918 , -0.28471634]])

In [68]: arr
Out[68]:
array([[ 0.99715976, -0.99285941, -1.29983152,  0.26311053],
       [ 1.44502703,  0.76717446,  0.45337043,  0.19472739],
       [ 0.48680544,  0.51659521, -0.17987949, -0.50011625],
       [ 1.21627682,  1.46086937,  0.4304918 , -0.28471634]])

In [69]: arr = np.array([[0,1,2], [3,4,5], [6,7,8]])

In [70]: arr
Out[70]:
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])

In [71]: arr.cumsum(0)
Out[71]:
array([[ 0,  1,  2],
       [ 3,  5,  7],
       [ 9, 12, 15]])

In [72]: arr.cumprod(0)
Out[72]:
array([[ 0,  1,  2],
       [ 0,  4, 10],
       [ 0, 28, 80]])

In [73]: arr.cumprod(1)
Out[73]:
array([[  0,   0,   0],
       [  3,  12,  60],
       [  6,  42, 336]])

In [74]: arr.cumprod(2)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-74-db4582eaf51a> in <module>()
----> 1 arr.cumprod(2)

ValueError: axis(=2) out of bounds

In [75]: arr.cumprod()
Out[75]: array([0, 0, 0, 0, 0, 0, 0, 0, 0])

In [76]: arr.cumsum()
Out[76]: array([ 0,  1,  3,  6, 10, 15, 21, 28, 36])

In [77]: arr = np.random.randn(8)

In [78]: arr.sort(-1)

In [79]: arr
Out[79]:
array([-1.67419884, -1.0020941 , -0.64800766,  0.5882544 ,  0.68288943,
        1.01370844,  1.41173385,  1.69355966])

In [80]: arr.sort()

In [81]: arr
```

```
Out[81]:
array([-1.67419884, -1.0020941 , -0.64800766,  0.5882544 ,  0.68288943,
        1.01370844,  1.41173385,  1.69355966])

In [82]: arr.sort(1)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-82-e68aa76b5f43> in <module>()
----> 1 arr.sort(1)

ValueError: axis(=1) out of bounds

In [83]: arr.sort(0)

In [84]: arr
Out[84]:
array([-1.67419884, -1.0020941 , -0.64800766,  0.5882544 ,  0.68288943,
        1.01370844,  1.41173385,  1.69355966])

In [85]: arr.sort?
Docstring:
a.sort(axis=-1, kind='quicksort', order=None)

Sort an array, in-place.

Parameters
----------
axis : int, optional
    Axis along which to sort. Default is -1, which means sort along the
    last axis.
kind : {'quicksort', 'mergesort', 'heapsort'}, optional
    Sorting algorithm. Default is 'quicksort'.
order : str or list of str, optional
    When `a` is an array with fields defined, this argument specifies
    which fields to compare first, second, etc.  A single field can
    be specified as a string, and not all fields need be specified,
    but unspecified fields will still be used, in the order in which
    they come up in the dtype, to break ties.

See Also
--------
numpy.sort : Return a sorted copy of an array.
argsort : Indirect sort.
lexsort : Indirect stable sort on multiple keys.
searchsorted : Find elements in sorted array.
partition: Partial sort.

Notes
-----
See ``sort`` for notes on the different sorting algorithms.

Examples
--------
>>> a = np.array([[1,4], [3,1]])
>>> a.sort(axis=1)
>>> a
array([[1, 4],
       [1, 3]])
>>> a.sort(axis=0)
>>> a
array([[1, 3],
       [1, 4]])

Use the `order` keyword to specify a field to use when sorting a
structured array:

>>> a = np.array([('a', 2), ('c', 1)], dtype=[('x', 'S1'), ('y', int)])
>>> a.sort(order='y')
```

```
>>> a
array([('c', 1), ('a', 2)],
      dtype=[('x', '|S1'), ('y', '<i4')])
Type:       builtin_function_or_method

In [86]: np.random.randn?
Docstring:
randn(d0, d1, ..., dn)

Return a sample (or samples) from the "standard normal" distribution.

If positive, int_like or int-convertible arguments are provided,
`randn` generates an array of shape ``(d0, d1, ..., dn)``, filled
with random floats sampled from a univariate "normal" (Gaussian)
distribution of mean 0 and variance 1 (if any of the :math:`d_i` are
floats, they are first converted to integers by truncation). A single
float randomly sampled from the distribution is returned if no
argument is provided.

This is a convenience function.  If you want an interface that takes a
tuple as the first argument, use `numpy.random.standard_normal` instead.

Parameters
----------
d0, d1, ..., dn : int, optional
    The dimensions of the returned array, should be all positive.
    If no argument is given a single Python float is returned.

Returns
-------
Z : ndarray or float
    A ``(d0, d1, ..., dn)``-shaped array of floating-point samples from
    the standard normal distribution, or a single such float if
    no parameters were supplied.

See Also
--------
random.standard_normal : Similar, but takes a tuple as its argument.

Notes
-----
For random samples from :math:`N(\mu, \sigma^2)`, use:

``sigma * np.random.randn(...) + mu``

Examples
--------
>>> np.random.randn()
2.1923875335537315 #random

Two-by-four array of samples from N(3, 6.25):

>>> 2.5 * np.random.randn(2, 4) + 3
array([[-4.49401501,  4.00950034, -1.81814867,  7.29718677],  #random
       [ 0.39924804,  4.68456316,  4.99394529,  4.84057254]]) #random
Type:       builtin_function_or_method

In [87]: np.ones(3)
Out[87]: array([ 1.,  1.,  1.])

In [88]: np.ones(3,3)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-88-370a00780af9> in <module>()
----> 1 np.ones(3,3)

/Users/Hansen/anaconda/lib/python2.7/site-packages/numpy/core/numeric.pyc in ones(shape, dtype, order
)
```

```
    188
    189     """
--> 190     a = empty(shape, dtype, order)
    191     multiarray.copyto(a, 1, casting='unsafe')
    192     return a

TypeError: data type not understood

In [89]: np.ones([3,3])
Out[89]:
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])

In [90]: %timeit np.random.normal(size = 8)
The slowest run took 180.23 times longer than the fastest. This could mean that an intermediate resul
t is being cached.
100000 loops, best of 3: 2.06 µs per loop

In [91]: %timeit np.random.normal(size = 1000000)
10 loops, best of 3: 38.9 ms per loop

In [92]: nwalks = 5000

In [93]: nsteps = 1000

In [94]: draws = np.random.randint(0, 2, size = (nwalks, nsteps))

In [95]: steps = np.where(de)
%%debug  %debug   def      del      delattr

In [95]: steps = np.where(dr)
draws    dreload

In [95]: steps = np.where(draws > 0, 1, -1)

In [96]: walks = steps.cumsum(1)

In [97]: walks
Out[97]:
array([[ -1,   0,   1, ...,  38,  37,  38],
       [ -1,   0,  -1, ...,   2,   3,   4],
       [ -1,  -2,  -3, ..., -56, -55, -54],
       ...,
       [  1,   0,  -1, ...,  38,  37,  36],
       [  1,   0,  -1, ..., -20, -19, -20],
       [  1,   2,   3, ...,  34,  35,  34]])

In [98]: walks.max(nwalks -1 )
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-98-63532e358457> in <module>()
----> 1 walks.max(nwalks -1 )

/Users/Hansen/anaconda/lib/python2.7/site-packages/numpy/core/_methods.pyc in _amax(a, axis, out, kee
pdims)
     24 # small reductions
     25 def _amax(a, axis=None, out=None, keepdims=False):
---> 26     return umr_maximum(a, axis, None, out, keepdims)
     27
     28 def _amin(a, axis=None, out=None, keepdims=False):

ValueError: 'axis' entry is out of bounds

In [99]: walks.max(nteps -1 )
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-99-511f61c365aa> in <module>()
```

```
----> 1 walks.max(nteps -1 )

NameError: name 'nteps' is not defined

In [100]: walks.max(nsteps -1 )
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-100-1dc7e8d8c620> in <module>()
----> 1 walks.max(nsteps -1 )

/Users/Hansen/anaconda/lib/python2.7/site-packages/numpy/core/_methods.pyc in _amax(a, axis, out, kee
pdims)
     24 # small reductions
     25 def _amax(a, axis=None, out=None, keepdims=False):
---> 26     return umr_maximum(a, axis, None, out, keepdims)
     27
     28 def _amin(a, axis=None, out=None, keepdims=False):

ValueError: 'axis' entry is out of bounds

In [101]: walks.size()
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-101-1b6c024318ca> in <module>()
----> 1 walks.size()

TypeError: 'int' object is not callable

In [102]: walks
Out[102]:
array([[ -1,   0,   1, ...,  38,  37,  38],
       [ -1,   0,  -1, ...,   2,   3,   4],
       [ -1,  -2,  -3, ..., -56, -55, -54],
       ...,
       [  1,   0,  -1, ...,  38,  37,  36],
       [  1,   0,  -1, ..., -20, -19, -20],
       [  1,   2,   3, ...,  34,  35,  34]])

In [103]:
```