100 numpy exercises

1. Import the numpy package under the name `np` (★☆☆)

In [1]:
```
import numpy as np
```

2. Print the numpy version and the configuration (★☆☆)

In [7]:
```
print(np.__version__)
print(np.show_config())
```

```
1.21.5
blas_mkl_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/Users/roatny/opt/anaconda3/lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/Users/roatny/opt/anaconda3/include']
blas_opt_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/Users/roatny/opt/anaconda3/lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/Users/roatny/opt/anaconda3/include']
lapack_mkl_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/Users/roatny/opt/anaconda3/lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/Users/roatny/opt/anaconda3/include']
lapack_opt_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/Users/roatny/opt/anaconda3/lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/Users/roatny/opt/anaconda3/include']
Supported SIMD extensions in this NumPy install:
    baseline = SSE,SSE2,SSE3
    found = SSSE3,SSE41,POPCNT,SSE42
    not found = AVX,F16C,FMA3,AVX2,AVX512F,AVX512CD,AVX512_KNL,AVX
512_SKX,AVX512_CLX,AVX512_CNL,AVX512_ICL
None
```

3. Create a null vector of size 10 (★☆☆)

In [8]:

```
null=np.zeros(10)
print(null)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

4. How to find the memory size of any array (★☆☆)

In [9]:

```
x = np.array([100, 20, 34])

print("Memory size of the array:",x.size)
```

```
Memory size of the array: 3
```

5. How to get the documentation of the numpy add function from the command line? (★☆☆)

In [10]:

```
np.info(np.add) #Why?
```

```
add(x1, x2, /, out=None, *, where=True, casting='same_kind', order
='K', dtype=None, subok=True[, signature, extobj])

Add arguments element-wise.

Parameters
----------
x1, x2 : array_like
    The arrays to be added.
    If ``x1.shape != x2.shape``, they must be broadcastable to a c
ommon
    shape (which becomes the shape of the output).
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it mu
st have
    a shape that the inputs broadcast to. If not provided or None,
    a freshly-allocated array is returned. A tuple (possible only
as a
    keyword argument) must have length equal to the number of outp
uts.
where : array_like, optional
    This condition is broadcast over the input. At locations where
the
    condition is True, the `out` array will be set to the ufunc re
sult.
    Elsewhere, the `out` array will retain its original value.
    Note that if an uninitialized `out` array is created via the d
```

```
efault
    ``out=None``, locations within it where the condition is False
will
    remain uninitialized.
**kwargs
    For other keyword-only arguments, see the
    :ref:`ufunc docs <ufuncs.kwargs>`.

Returns
-------
add : ndarray or scalar
    The sum of `x1` and `x2`, element-wise.
    This is a scalar if both `x1` and `x2` are scalars.

Notes
-----
Equivalent to `x1` + `x2` in terms of array broadcasting.

Examples
--------
>>> np.add(1.0, 4.0)
5.0
>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> np.add(x1, x2)
array([[  0.,   2.,   4.],
       [  3.,   5.,   7.],
       [  6.,   8.,  10.]])

The ``+`` operator can be used as a shorthand for ``np.add`` on nd
arrays.

>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> x1 + x2
array([[ 0.,  2.,  4.],
       [ 3.,  5.,  7.],
       [ 6.,  8., 10.]])
```

6. Create a null vector of size 10 but the fifth value which is 1 (★☆☆)

In [11]:
```
a=np.zeros(10)
a[4]=1
a
```

Out[11]:
```
array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])
```

7. Create a vector with values ranging from 10 to 49 (★☆☆)

```
In [14]: b=np.arange(10,49)
         b
```

```
Out[14]: array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
         25, 26,
                27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
         42, 43,
                44, 45, 46, 47, 48])
```

8. Reverse a vector (first element becomes last) (★☆☆)

```
In [15]: print(b[::-1])
```

```
[48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27
26 25
 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10]
```

9. Create a 3x3 matrix with values ranging from 0 to 8 (★☆☆)

```
In [21]: c =np.arange(0,9).reshape(3,3)
         print(c)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

10. Find indices of non-zero elements from [1,2,0,0,4,0] (★☆☆)

```
In [24]: d1=[1,2,0,0,4,0]
         d2=[]
         for i in range(len(d1)):
             if d1[i]!=0:
                 d2.append(i)
         print(d2)
```

```
[0, 1, 4]
```

11. Create a 3x3 identity matrix (★☆☆)

```
In [26]: e=np.eye(3,3)
         e
```

```
Out[26]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])
```

12. Create a 3x3x3 array with random values (★☆☆)

```
In [32]: f=np.random.random((3,3,3))
         f
```

```
Out[32]: array([[[0.65214666, 0.6581492 , 0.24172204],
                 [0.50164251, 0.08316561, 0.30708692],
                 [0.53643665, 0.62223953, 0.45768691]],

                [[0.50834829, 0.28676943, 0.88436828],
                 [0.3859957 , 0.1132351 , 0.45466697],
                 [0.36542809, 0.34620432, 0.30278338]],

                [[0.35875132, 0.04633132, 0.57123506],
                 [0.98282023, 0.534639  , 0.96448144],
                 [0.40692733, 0.99766714, 0.97998978]]])
```

13. Create a 10x10 array with random values and find the minimum and maximum values (★☆☆)

In [41]:
```
g=np.random.random((10,10))
g
```

Out[41]:
```
array([[0.90005705, 0.98328121, 0.62492011, 0.5548771 , 0.15823363
,
         0.45346475, 0.78492157, 0.63039117, 0.62585446, 0.74783865
],
        [0.10541635, 0.65169988, 0.63391122, 0.08704064, 0.06833723
,
         0.18896972, 0.78591857, 0.71903419, 0.36682912, 0.60941882
],
        [0.01620981, 0.13648657, 0.30303598, 0.05032591, 0.15526704
,
         0.0935427 , 0.43038619, 0.86181424, 0.00410034, 0.32979194
],
        [0.56925352, 0.17392608, 0.68759999, 0.12550366, 0.09906728
,
         0.15756249, 0.00977323, 0.6652983 , 0.82467368, 0.89527452
],
        [0.58717968, 0.27050513, 0.44464497, 0.65575782, 0.2633674
,
         0.97541926, 0.91973792, 0.39358055, 0.54347658, 0.43969733
],
        [0.96967941, 0.98316041, 0.55209943, 0.15309357, 0.35142155
,
         0.72455887, 0.89746538, 0.31718343, 0.20153086, 0.6506147
],
        [0.34783547, 0.64027027, 0.14562874, 0.07793643, 0.59874019
,
         0.40576042, 0.90147811, 0.76538156, 0.32340242, 0.25710718
],
        [0.98121789, 0.07974413, 0.88416242, 0.68318328, 0.24519343
,
         0.11250313, 0.07314364, 0.01931734, 0.04764178, 0.526018
],
        [0.49835323, 0.12524231, 0.11738205, 0.76565308, 0.21122262
,
         0.96658682, 0.21328417, 0.83383692, 0.48285392, 0.91807776
],
        [0.3507424 , 0.40167599, 0.9339651 , 0.2851172 , 0.02406186
,
         0.2898851 , 0.06123243, 0.60298135, 0.09045495, 0.50594052
]])
```

In [42]:
```
g.min()
```

Out[42]:
```
0.004100338073978249
```

In [43]:
```
g.max()
```

Out[43]:
```
0.9832812110467951
```

14. Create a random vector of size 30 and find the mean value (★☆☆)

```
In [46]: h=np.random.random((30))
         h
```

```
Out[46]: array([0.21640756, 0.9306141 , 0.35624436, 0.3351772 , 0.44661212,
                0.38907482, 0.64706016, 0.11199821, 0.04123379, 0.57979128,
                0.8755312 , 0.17077043, 0.3808254 , 0.22471358, 0.09363036,
                0.02129857, 0.62565545, 0.93539476, 0.05718409, 0.15040503,
                0.92437644, 0.3619423 , 0.23287199, 0.83635233, 0.13457537,
                0.64506826, 0.00543304, 0.9375695 , 0.88485084, 0.61266722]
         )
```

```
In [47]: h.mean()
```

```
Out[47]: 0.4388443253745781
```

15. Create a 2d array with 1 on the border and 0 inside (★☆☆)

```
In [50]: i = np.ones((10,10))
         i[1:-1,1:-1]=0
         i
```

```
Out[50]: array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
                [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]])
```

16. How to add a border (filled with 0's) around an existing array? (★☆☆)

In [51]:
```
j = np.pad(i, pad_width=1, mode='constant', constant_values=0)
j
```

Out[51]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

17. What is the result of the following expression? (★☆☆)

```
0 * np.nan
np.nan == np.nan
np.inf > np.nan
np.nan - np.nan
np.nan in set([np.nan])
0.3 == 3 * 0.1
```

In [52]:
```
0 * np.nan
np.nan == np.nan
np.inf > np.nan
np.nan - np.nan
np.nan in set([np.nan])
0.3 == 3 * 0.1
```

Out[52]:
```
False
```

18. Create a 5x5 matrix with values 1,2,3,4 just below the diagonal (★☆☆)

In [55]:
```
#k = np.diag([1, 2, 3, 4])

k = np.diag(1+np.arange(4), k = -1)
k
```

Out[55]:
```
array([[0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [0, 2, 0, 0, 0],
       [0, 0, 3, 0, 0],
       [0, 0, 0, 4, 0]])
```

19. Create a 8x8 matrix and fill it with a checkerboard pattern (★☆☆)

In [56]:

```

l = np.zeros ((8,8), dtype=int)
l[1::2, ::2]= 1
l[::2, 1::2] = 1
l
```

Out[56]:

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

20. Consider a (6,7,8) shape array, what is the index (x,y,z) of the 100th element? (★☆☆)

In [57]:

```
print (np.unravel_index(100, (6,7,8)))
```

```
(1, 5, 4)
```

21. Create a checkerboard 8x8 matrix using the tile function (★☆☆)

In [58]:

```
array= np.array([[0,1], [1,0]])
m = np.tile(array,(4,4))
print (m)
```

```
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

22. Normalize a 5x5 random matrix (★☆☆)

In [59]:

```
n= np.random.random((5,5))
nmax, nmin = n.max(), n.min()
n= (n-nmin)/(nmax-nmin)
n
```

Out[59]:

```
array([[0.77366124, 0.06753766, 0.01645899, 0.75370846, 0.89329812
],
       [0.81062597, 0.99997005, 0.26790638, 0.30326908, 0.97495381
],
       [0.        , 0.92118784, 0.76258788, 0.5192764 , 0.34051246
],
       [0.70342156, 0.75650845, 0.47981917, 0.60596282, 0.87470609
],
       [0.25728337, 1.        , 0.46400273, 0.82758396, 0.21802154
]])
```

23. Create a custom dtype that describes a color as four unsigned bytes (RGBA) (★☆☆)

In [61]:

```
color = np.dtype([("r", np.ubyte, 1),
                  ("g", np.ubyte, 1),
                  ("b", np.ubyte, 1),
                  ("a", np.ubyte, 1)])
color
```

```
/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1285/14
89744843.py:1: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  color = np.dtype([("r", np.ubyte, 1),
```

Out[61]:

```
dtype([('r', 'u1'), ('g', 'u1'), ('b', 'u1'), ('a', 'u1')])
```

24. Multiply a 5x3 matrix by a 3x2 matrix (real matrix product) (★☆☆)

In [63]:

```
o= np.dot(np.ones((5,3)), np.ones((3,2)))
o
```

Out[63]:

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

25. Given a 1D array, negate all elements which are between 3 and 8, in place. (★☆☆)

In [64]:

```
p = np.arange(11)
p[(3 < p) & (p <= 8)] *= -1
print(p)
```

```
[ 0  1  2  3 -4 -5 -6 -7 -8  9 10]
```

26. What is the output of the following script? (★☆☆)

```
# Author: Jake VanderPlas

print(sum(range(5),-1))
from numpy import *
print(sum(range(5),-1))
```

In [65]:

```
print(sum(range(5),-1))
from numpy import *
print(sum(range(5),-1))
```

```
9
10
```

27. Consider an integer vector Z, which of these expressions are legal? (★☆☆)

```
Z**Z
2 << Z >> 2
Z <- Z
1j*Z
Z/1/1
Z<Z>Z
```

In [70]:

```
Z**Z
2 << Z >> 2
Z <- Z
1j*Z
Z/1/1
Z<Z>Z
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1285/803206331.py in <module>
----> 1 Z**Z
      2 2 << Z >> 2
      3 Z <- Z
      4 1j*Z
      5 Z/1/1

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

28. What are the result of the following expressions? (★☆☆)

```
np.array(0) / np.array(0)
np.array(0) // np.array(0)
np.array([np.nan]).astype(int).astype(float)
```

In [71]:

```
np.array(0) / np.array(0)
np.array(0) // np.array(0)
np.array([np.nan]).astype(int).astype(float)
```

```
/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1285/548293649.py:1: RuntimeWarning: invalid value encountered in true_divide
  np.array(0) / np.array(0)
/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1285/548293649.py:2: RuntimeWarning: divide by zero encountered in floor_divide
  np.array(0) // np.array(0)
```

Out[71]:

```
array([-9.22337204e+18])
```

29. How to round away from zero a float array ? (★☆☆)

In [72]:

```




r = np.random.uniform(-10,+10,10)
print (np.copysign(np.ceil(np.abs(r)), r))
```

```
[-4.   6. 10.   9.   8.   8.   1.   9.   5. -7.]
```

30. How to find common values between two arrays? (★☆☆)

In [73]:

```
s1 = np.random.randint(0,10,10)
s2 = np.random.randint(0,10,10)
print(np.intersect1d(s1,s2))
```

```
[0 1 7 8]
```

31. How to ignore all numpy warnings (not recommended)? (★☆☆)

In [76]:

```
defaults = np.seterr(all="ignore")
t = np.ones(1) / 0

_ = np.seterr(**defaults)
t
```

Out[76]:

```
array([inf])
```

32. Is the following expressions true? (★☆☆)

```
np.sqrt(-1) == np.emath.sqrt(-1)
```

In [77]:

```
np.sqrt(-1) == np.emath.sqrt(-1)
```

```
/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1285/24
4602691.py:1: RuntimeWarning: invalid value encountered in sqrt
  np.sqrt(-1) == np.emath.sqrt(-1)
```

Out[77]:

```
False
```

33. How to get the dates of yesterday, today and tomorrow? (★☆☆)

In [78]:
```
yesterday = np.datetime64('today', 'D') - np.timedelta64(1, 'D')
today     = np.datetime64('today', 'D')
tomorrow  = np.datetime64('today', 'D') + np.timedelta64(1, 'D')
```

34. How to get all the dates corresponding to the month of July 2016? (★★☆)

In [79]:
```
u = np.arange('2016-07', '2016-08', dtype='datetime64[D]')
print(u)
```

```
['2016-07-01' '2016-07-02' '2016-07-03' '2016-07-04' '2016-07-05'
 '2016-07-06' '2016-07-07' '2016-07-08' '2016-07-09' '2016-07-10'
 '2016-07-11' '2016-07-12' '2016-07-13' '2016-07-14' '2016-07-15'
 '2016-07-16' '2016-07-17' '2016-07-18' '2016-07-19' '2016-07-20'
 '2016-07-21' '2016-07-22' '2016-07-23' '2016-07-24' '2016-07-25'
 '2016-07-26' '2016-07-27' '2016-07-28' '2016-07-29' '2016-07-30'
 '2016-07-31']
```

35. How to compute ((A+B)*(-A/2)) in place (without copy)? (★★☆)

In [80]:
```
v = np.ones(3)*1
x = np.ones(3)*2
y = np.ones(3)*3
np.add(v,x,out=y)
np.divide(v,2,out=x)
np.negative(v,out=v)
np.multiply(v,x,out=v)
```

Out[80]:
```
array([-0.5, -0.5, -0.5])
```

36. Extract the integer part of a random array of positive numbers using 4 different methods (★★☆)

In [82]:

```
z = np.random.uniform(0,10,10)

print (z- z%1)
print (np.floor(z))
print (np.ceil(z)-1)
print (z.astype(int))
```

```
[0. 6. 4. 9. 6. 5. 8. 2. 1. 4.]
[0. 6. 4. 9. 6. 5. 8. 2. 1. 4.]
[0. 6. 4. 9. 6. 5. 8. 2. 1. 4.]
[0 6 4 9 6 5 8 2 1 4]
```

37. Create a 5x5 matrix with row values ranging from 0 to 4 (★★☆)

In [83]:

```
A = np.zeros((5,5))
A += np.arange(5)
print(A)
```

```
[[0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]]
```

38. Consider a generator function that generates 10 integers and use it to build an array (★☆☆)

In [84]:

```
def generate():
    for x in range(10):
        yield x
B = np.fromiter(generate(),dtype=float,count=-1)
print(B)
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

39. Create a vector of size 10 with values ranging from 0 to 1, both excluded (★★☆)

In [85]:

```
C = np.linspace(0,1,11,endpoint=False)[1:]
print(C)
```

```
[0.09090909 0.18181818 0.27272727 0.36363636 0.45454545 0.54545455
 0.63636364 0.72727273 0.81818182 0.90909091]
```

40. Create a random vector of size 10 and sort it (★★☆)

In [86]:
```
D = np.random.random(10)
D.sort()
print(D)
```

```
[0.07073528 0.09213961 0.1740076  0.3511049  0.49282776 0.54447535
 0.54858057 0.78595241 0.80182775 0.96082067]
```

41. How to sum a small array faster than np.sum? (★★☆)

In [87]:
```
E = np.arange(10)
np.add.reduce(E)
```

Out[87]:
```
45
```

42. Consider two random array A and B, check if they are equal (★★☆)

In [88]:
```
F = np.random.randint(0,2,5)
G = np.random.randint(0,2,5)

# Assuming identical shape of the arrays and a tolerance for the co
equal = np.allclose(F,G)
print(equal)

# Checking both the shape and the element values, no tolerance (val
equal = np.array_equal(F,G)
print(equal)
```

```
False
False
```

43. Make an array immutable (read-only) (★★☆)

In [90]:
```
H = np.zeros(10)
H.flags.writeable = False
H[0] =1
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent c
all last)
/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1285/33
58469309.py in <module>
      1 H = np.zeros(10)
      2 H.flags.writeable = False
----> 3 H[0] =1

ValueError: assignment destination is read-only
```

44. Consider a random 10x2 matrix representing cartesian coordinates, convert them to polar coordinates (★★☆)

In [91]:

```
Z = np.random.random((10,2))
X,Y = Z[:,0], Z[:,1]
R = np.sqrt(X**2+Y**2)
T = np.arctan2(Y,X)
print(R)
print(T)
```

```
[1.26617288 1.06735756 0.76156466 1.02911027 0.98644419 0.95277227
 0.52599805 0.89303859 0.57637303 0.72953261]
[0.78830904 1.18269249 1.36306527 0.53022826 0.53643567 1.36229739
 1.45061017 0.33495232 1.33862223 0.99957451]
```

45. Create random vector of size 10 and replace the maximum value by 0 (★★☆)

In [92]:

```
I = np.random.random(10)
I[I.argmax()] = 0
print(I)
```

```
[0.         0.89868593 0.331076   0.40759222 0.50564007 0.88842885
 0.64825794 0.79051618 0.2235078  0.91180214]
```

46. Create a structured array with x and y coordinates covering the [0,1]x[0,1] area (★★☆)

In [93]:

```
J = np.zeros((5,5), [('x',float),('y',float)])
J['x'], J['y'] = np.meshgrid(np.linspace(0,1,5),
                             np.linspace(0,1,5))
print(J)
```

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

47. Given two arrays, X and Y, construct the Cauchy matrix C (Cij =1/(xi - yj)) (★★☆)

In [94]:

```
X = np.arange(8)
Y = X + 0.5
C = 1.0 / np.subtract.outer(X, Y)
print(np.linalg.det(C))
```

```
3638.1636371179666
```

48. Print the minimum and maximum representable value for each numpy scalar type (★★☆)

In [95]:

```
for dtype in [np.int8, np.int32, np.int64]:
   print(np.iinfo(dtype).min)
   print(np.iinfo(dtype).max)
for dtype in [np.float32, np.float64]:
   print(np.finfo(dtype).min)
   print(np.finfo(dtype).max)
   print(np.finfo(dtype).eps)
```

```
-128
127
-2147483648
2147483647
-9223372036854775808
9223372036854775807
-3.4028235e+38
3.4028235e+38
1.1920929e-07
-1.7976931348623157e+308
1.7976931348623157e+308
2.220446049250313e-16
```

49. How to print all the values of an array? (★★☆)

In [ ]:

```
np.set_printoptions(threshold=np.nan)
Z = np.zeros((25,25))
print(Z)
```

50. How to find the closest value (to a given scalar) in a vector? (★★☆)

In [100]:

```
Z = np.arange(100)
v = np.random.uniform(0,100)
index = (np.abs(Z-v)).argmin()
print(Z[index])
```

```
62
```

51. Create a structured array representing a position (x,y) and a color (r,g,b) (★★☆)

In [101]:

```
Z = np.zeros(10, [ ('position', [ ('x', float, 1),
                                  ('y', float, 1)]),
                   ('color',    [ ('r', float, 1),
                                  ('g', float, 1),
                                  ('b', float, 1)])])
print(Z)
```

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

/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1285/27
4409719.py:1: FutureWarning: Passing (type, 1) or '1type' as a syn
onym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  Z = np.zeros(10, [ ('position', [ ('x', float, 1),
```

52. Consider a random vector with shape (100,2) representing coordinates, find point by point distances (★★☆)

In [102]:

```
Z = np.random.random((10,2))
X,Y = np.atleast_2d(Z[:,0], Z[:,1])
D = np.sqrt( (X-X.T)**2 + (Y-Y.T)**2)
print(D)

# Much faster with scipy
import scipy
# Thanks Gavin Heverly-Coulson (#issue 1)
import scipy.spatial

Z = np.random.random((10,2))
D = scipy.spatial.distance.cdist(Z,Z)
print(D)
```

```
[[0.         0.45574237 0.71065675 0.5532571  0.37305855 0.5832279
7
  0.09648755 0.22007657 0.4296158  0.22956487]
 [0.45574237 0.         0.25671212 0.32311608 0.67454655 0.8271582
3
  0.51773205 0.47437548 0.10209215 0.48001645]
 [0.71065675 0.25671212 0.         0.38625712 0.91617241 1.0073706
3
  0.76672147 0.69898085 0.29632386 0.70273743]
 [0.5532571  0.32311608 0.38625712 0.         0.88337931 0.6452834
2
  0.56118623 0.42825599 0.2298198  0.42714353]
 [0.37305855 0.67454655 0.91617241 0.88337931 0.         0.9179338
6
  0.43678669 0.59118398 0.70036168 0.60032696]
 [0.58322797 0.82715823 1.00737063 0.64528342 0.91793386 0.
  0.49408561 0.39201172 0.73667296 0.38281314]
 [0.09648755 0.51773205 0.76672147 0.56118623 0.43678669 0.4940856
1
  0.         0.16551707 0.47521292 0.17317005]
 [0.22007657 0.47437548 0.69898085 0.42825599 0.59118398 0.3920117
2
  0.16551707 0.         0.40374093 0.00985697]
 [0.4296158  0.10209215 0.29632386 0.2298198  0.70036168 0.7366729
6
  0.47521292 0.40374093 0.         0.40796753]
 [0.22956487 0.48001645 0.70273743 0.42714353 0.60032696 0.3828131
4
  0.17317005 0.00985697 0.40796753 0.        ]]
[[0.         0.57028851 0.69491963 0.58108204 0.35937161 0.4822617
3
  0.30437074 0.69547421 0.19932133 0.30988446]
 [0.57028851 0.         0.49856277 0.6117666  0.92829177 0.6272497
8
  0.31932044 0.82667247 0.69975511 0.27778885]
 [0.69491963 0.49856277 0.         1.04266412 1.00091598 1.0134583
7
  0.65423162 0.42740365 0.8910471  0.46174516]
 [0.58108204 0.6117666  1.04266412 0.         0.77829784 0.1257814
5
  0.39759519 1.21822043 0.50829251 0.59590401]
 [0.35937161 0.92829177 1.00091598 0.77829784 0.         0.6541189
6
  0.63995068 0.86390395 0.27832238 0.66807487]
 [0.48226173 0.62724978 1.01345837 0.12578145 0.65411896 0.
  0.35947152 1.14744485 0.38782859 0.55394261]
 [0.30437074 0.31932044 0.65423162 0.39759519 0.63995068 0.3594715
2
  0.         0.82988234 0.38841133 0.19858491]
 [0.69547421 0.82667247 0.42740365 1.21822043 0.86390395 1.1474448
5
  0.82988234 0.         0.88112098 0.64500481]
 [0.19932133 0.69975511 0.8910471  0.50829251 0.27832238 0.3878285
9
  0.38841133 0.88112098 0.         0.47469145]
 [0.30988446 0.27778885 0.46174516 0.59590401 0.66807487 0.5539426
1
  0.19858491 0.64500481 0.47469145 0.        ]]
```

53. How to convert a float (32 bits) array into an integer (32 bits) in place?

In [103]:
```
Z = np.arange(10, dtype=np.int32)
Z = Z.astype(np.float32, copy=False)
print(Z)
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

54. How to read the following file? (★★☆)

```
1, 2, 3, 4, 5
6,  ,  , 7, 8
 ,  , 9,10,11
```

In [104]:
```
from io import StringIO

# Fake file
s = StringIO("""1, 2, 3, 4, 5\n
                6,  ,  , 7, 8\n
                 ,  , 9,10,11\n""")
Z = np.genfromtxt(s, delimiter=",", dtype=np.int)
print(Z)
```

```
[[ 1  2  3  4  5]
 [ 6 -1 -1  7  8]
 [-1 -1  9 10 11]]

/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1285/12
71915251.py:7: DeprecationWarning: `np.int` is a deprecated alias
for the builtin `int`. To silence this warning, use `int` by itsel
f. Doing this will not modify any behavior and is safe. When repla
cing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` t
o specify the precision. If you wish to review your current use, c
heck the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://n
umpy.org/devdocs/release/1.20.0-notes.html#deprecations
(https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations)
  Z = np.genfromtxt(s, delimiter=",", dtype=np.int)
```

55. What is the equivalent of enumerate for numpy arrays? (★★☆)

In [3]:
```
import numpy as np
Z = np.arange(9).reshape(3,3)
for index, value in np.ndenumerate(Z):
    print(index, value)
for index in np.ndindex(Z.shape):
    print(index, Z[index])
```

```
(0, 0) 0
(0, 1) 1
(0, 2) 2
(1, 0) 3
(1, 1) 4
(1, 2) 5
(2, 0) 6
(2, 1) 7
(2, 2) 8
(0, 0) 0
(0, 1) 1
(0, 2) 2
(1, 0) 3
(1, 1) 4
(1, 2) 5
(2, 0) 6
(2, 1) 7
(2, 2) 8
```

56. Generate a generic 2D Gaussian-like array (★★☆)

In [4]:

```
Z = np.arange(9).reshape(3,3)
for index, value in np.ndenumerate(Z):
    print(index, value)
for index in np.ndindex(Z.shape):
    print(index, Z[index])
```

```
(0, 0) 0
(0, 1) 1
(0, 2) 2
(1, 0) 3
(1, 1) 4
(1, 2) 5
(2, 0) 6
(2, 1) 7
(2, 2) 8
(0, 0) 0
(0, 1) 1
(0, 2) 2
(1, 0) 3
(1, 1) 4
(1, 2) 5
(2, 0) 6
(2, 1) 7
(2, 2) 8
```

57. How to randomly place p elements in a 2D array? (★★☆)

In [6]:

```
n = 10
p = 3
Y = np.zeros((n,n))
np.put(Y, np.random.choice(range(n*n), p, replace=False),1)
print(Y)
```

```
[[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

58. Subtract the mean of each row of a matrix (★★☆)

In [7]:

```

X = np.random.rand(5, 10)

# Recent versions of numpy
Y = X - X.mean(axis=1, keepdims=True)

# Older versions of numpy
Y = X - X.mean(axis=1).reshape(-1, 1)

print(Y)
```

```
[[-0.40459155  0.20833087 -0.11312464 -0.22090915  0.03904833 -0.0
8105609
   0.48472398  0.41874404 -0.39249926  0.06133345]
 [-0.3181533   0.37084311 -0.24625812  0.06504     0.38515546 -0.3
2173493
   0.15294926  0.03859459 -0.54766357  0.42122749]
 [-0.07047032  0.46348719 -0.06872773  0.18831735 -0.09063542 -0.1
1750908
  -0.32166971 -0.02904471 -0.02019143  0.06644385]
 [-0.37076738  0.15068995  0.46966491  0.46233958 -0.2445305   0.4
3542208
  -0.10439545 -0.25732393 -0.32267347 -0.21842578]
 [-0.59261501  0.05993769  0.01968782  0.03289092  0.39717162  0.2
1174831
  -0.5212097  -0.22806755  0.23308934  0.38736656]]
```

59. How to sort an array by the nth column? (★★☆)

In [8]:

```
Z = np.random.randint(0,10,(3,3))
print(Z)
print(Z[Z[:,1].argsort()])
```

```
[[3 4 6]
 [5 6 3]
 [9 5 1]]
[[3 4 6]
 [9 5 1]
 [5 6 3]]
```

60. How to tell if a given 2D array has null columns? (★★☆)

In [9]:

```
Z = np.random.randint(0,3,(3,10))
print((~Z.any(axis=0)).any())
```

```
True
```

61. Find the nearest value from a given value in an array (★★☆)

In [10]:

```
Z = np.random.uniform(0,1,10)
z = 0.5
m = Z.flat[np.abs(Z - z).argmin()]
print(m)
```

```
0.3416144101832661
```

62. Considering two arrays with shape (1,3) and (3,1), how to compute their sum using an iterator? (★★☆)

In [11]:

```
A = np.arange(3).reshape(3,1)
B = np.arange(3).reshape(1,3)
it = np.nditer([A,B,None])
for x,y,z in it: z[...] = x + y
print(it.operands[2])
```

```
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

63. Create an array class that has a name attribute (★★☆)

In [12]:

```
class NamedArray(np.ndarray):
    def __new__(cls, array, name="no name"):
        obj = np.asarray(array).view(cls)
        obj.name = name
        return obj
    def __array_finalize__(self, obj):
        if obj is None: return
        self.info = getattr(obj, 'name', "no name")

Z = NamedArray(np.arange(10), "range_10")
print (Z.name)
```

```
range_10
```

64. Consider a given vector, how to add 1 to each element indexed by a second vector (be careful with repeated indices)? (★★★)

In [13]:

```
Z = np.ones(10)
I = np.random.randint(0,len(Z),20)
Z += np.bincount(I, minlength=len(Z))
print(Z)

# Another solution
# Author: Bartosz Telenczuk
np.add.at(Z, I, 1)
print(Z)
```

```
[1. 3. 3. 4. 5. 1. 3. 2. 5. 3.]
[1. 5. 5. 7. 9. 1. 5. 3. 9. 5.]
```

65. How to accumulate elements of a vector (X) to an array (F) based on an index list (I)? (★★★)

In [14]:

```
X = [1,2,3,4,5,6]
I = [1,3,9,3,4,1]
F = np.bincount(I,X)
print(F)
```

```
[0. 7. 0. 6. 5. 0. 0. 0. 0. 3.]
```

66. Considering a (w,h,3) image of (dtype=ubyte), compute the number of unique colors (★★☆)

In [15]:

```
w,h = 16,16
I = np.random.randint(0,2,(h,w,3)).astype(np.ubyte)
F = I[...,0]*256*256 + I[...,1]*256 +I[...,2]
n = len(np.unique(F))
print(np.unique(I))
```

```
[0 1]
```

67. Considering a four dimensions array, how to get sum over the last two axis at once? (★★★)

In [16]:

```
A = np.random.randint(0,10,(3,4,3,4))
# solution by passing a tuple of axes (introduced in numpy 1.7.0)
sum = A.sum(axis=(-2,-1))
print(sum)
# solution by flattening the last two dimensions into one
# (useful for functions that don't accept tuples for axis argument)
sum = A.reshape(A.shape[:-2] + (-1,)).sum(axis=-1)
print(sum)
```

```
[[55 53 70 53]
 [63 56 55 63]
 [47 56 44 54]]
[[55 53 70 53]
 [63 56 55 63]
 [47 56 44 54]]
```

68. Considering a one-dimensional vector D, how to compute means of subsets of D using a vector S of same size describing subset indices? (★★★)

In [17]:

```


D = np.random.uniform(0,1,100)
S = np.random.randint(0,10,100)
D_sums = np.bincount(S, weights=D)
D_counts = np.bincount(S)
D_means = D_sums / D_counts
print(D_means)

# Pandas solution as a reference due to more intuitive code
import pandas as pd
print(pd.Series(D).groupby(S).mean())
```

```
[0.57146854 0.60286441 0.40416054 0.42371912 0.4494612  0.60707258
 0.37493685 0.50487551 0.39462024 0.38841574]
0    0.571469
1    0.602864
2    0.404161
3    0.423719
4    0.449461
5    0.607073
6    0.374937
7    0.504876
8    0.394620
9    0.388416
dtype: float64
```

69. How to get the diagonal of a dot product? (★★★)

In [18]:

```


A = np.random.uniform(0,1,(5,5))
B = np.random.uniform(0,1,(5,5))

# Slow version
np.diag(np.dot(A, B))

# Fast version
np.sum(A * B.T, axis=1)

# Faster version
np.einsum("ij,ji->i", A, B)
```

Out[18]:

```
array([2.53885442, 0.70691255, 0.59769867, 1.27727734, 1.17424763]
)
```

70. Consider the vector [1, 2, 3, 4, 5], how to build a new vector with 3 consecutive zeros interleaved between each value? (★★★)

In [19]:

```

Z = np.array([1,2,3,4,5])
nz = 3
Z0 = np.zeros(len(Z) + (len(Z)-1)*(nz))
Z0[::nz+1] = Z
print(Z0)
```

```
[1. 0. 0. 0. 2. 0. 0. 0. 3. 0. 0. 0. 4. 0. 0. 0. 5.]
```

71. Consider an array of dimension (5,5,3), how to mulitply it by an array with dimensions (5,5)? (★★★)

In [20]:
```
A = np.ones((5,5,3))
B = 2*np.ones((5,5))
print(A * B[:,:,None])
```

```
[[[2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]]

 [[2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]]

 [[2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]]

 [[2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]]

 [[2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]
  [2. 2. 2.]]]
```

72. How to swap two rows of an array? (★★★)

In [21]:
```
A = np.arange(25).reshape(5,5)
A[[0,1]] = A[[1,0]]
print(A)
```

```
[[ 5  6  7  8  9]
 [ 0  1  2  3  4]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

73. Consider a set of 10 triplets describing 10 triangles (with shared vertices), find the set of unique line segments composing all the triangles (★★★)

In [22]:
```
faces = np.random.randint(0,100,(10,3))
F = np.roll(faces.repeat(2,axis=1),-1,axis=1)
F = F.reshape(len(F)*3,2)
F = np.sort(F,axis=1)
G = F.view( dtype=[('p0',F.dtype),('p1',F.dtype)] )
G = np.unique(G)
print(G)
```

```
[( 4, 22) ( 4, 80) ( 6, 24) ( 6, 68) ( 8, 73) ( 8, 82) ( 8, 83) (
8, 91)
 (10, 21) (10, 32) (10, 54) (10, 80) (21, 32) (22, 80) (24, 68) (3
6, 84)
 (36, 85) (40, 48) (40, 73) (48, 73) (51, 78) (51, 82) (54, 80) (7
3, 82)
 (75, 93) (75, 99) (78, 82) (83, 91) (84, 85) (93, 99)]
```

74. Given a sorted array C that corresponds to a bincount, how to produce an array A such that np.bincount(A) == C? (★★★)

In [23]:
```
C = np.bincount([1,1,2,3,4,4,6])
A = np.repeat(np.arange(len(C)), C)
print(A)
```

```
[1 1 2 3 4 4 6]
```

75. How to compute averages using a sliding window over an array? (★★★)

In [24]:
```
def moving_average(a, n=3) :
    ret = np.cumsum(a, dtype=float)
    ret[n:] = ret[n:] - ret[:-n]
    return ret[n - 1:] / n
Z = np.arange(20)
print(moving_average(Z, n=3))
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 1
7. 18.]
```

76. Consider a one-dimensional array Z, build a two-dimensional array whose first row is (Z[0],Z[1],Z[2]) and each subsequent row is shifted by 1 (last row should be (Z[-3],Z[-2],Z[-1]) (★★★)

In [25]:

```
from numpy.lib import stride_tricks

def rolling(a, window):
    shape = (a.size - window + 1, window)
    strides = (a.itemsize, a.itemsize)
    return stride_tricks.as_strided(a, shape=shape, strides=strides
Z = rolling(np.arange(10), 3)
print(Z)
```

```
[[0 1 2]
 [1 2 3]
 [2 3 4]
 [3 4 5]
 [4 5 6]
 [5 6 7]
 [6 7 8]
 [7 8 9]]
```

77. How to negate a boolean, or to change the sign of a float inplace? (★★★)

In [26]:

```
Z = np.random.randint(0,2,100)
np.logical_not(Z, out=Z)

Z = np.random.uniform(-1.0,1.0,100)
np.negative(Z, out=Z)
```

Out[26]:

```
array([ 0.3497504 , -0.3979124 ,  0.13582173,  0.44657538,  0.6957
1546,
        0.74750678,  0.69142115, -0.9056327 , -0.94051209, -0.7827
0662,
        0.75072325, -0.1378537 , -0.50240359, -0.23983862,  0.5425
1255,
       -0.68801251,  0.29876285, -0.91887031,  0.78986207, -0.1546
7315,
       -0.74501757,  0.04297443,  0.47587592,  0.57771489,  0.8273
1932,
        0.23476873,  0.45941788, -0.51928372, -0.9266702 ,  0.9742
2173,
        0.20294609,  0.99930156, -0.39814051,  0.26113002, -0.0445
9141,
       -0.08265899, -0.86554058,  0.26216449,  0.21055264,  0.9352
5242,
       -0.38390414, -0.81326614,  0.86864029, -0.80876403, -0.3014
1593,
       -0.14139024, -0.90190665, -0.94683187, -0.38874918,  0.4188
5534,
       -0.03536736,  0.30872982,  0.88987136,  0.0861563 ,  0.3715
6055,
       -0.9301718 , -0.65574918,  0.14198597, -0.98227561,  0.9625
2119,
        0.90004964,  0.75517361,  0.84218896,  0.37114059, -0.3877
448 ,
        0.88559001,  0.2760651 ,  0.09361562,  0.86786971,  0.6948
5986,
       -0.19893208, -0.31311717,  0.49581271,  0.75279381, -0.8065
677 ,
        0.01472876, -0.92101422,  0.95522871, -0.66887861, -0.4006
4936,
       -0.25710423, -0.30294865, -0.61665386, -0.16228169,  0.8462
9874,
       -0.36508614,  0.77920276,  0.91277102,  0.99224436, -0.8300
5601,
       -0.16186958, -0.95255491,  0.80210863,  0.91917702, -0.0992
0456,
       -0.20340782, -0.2487168 , -0.29088956,  0.63109706,  0.4381
4519])
```

78. Consider 2 sets of points P0,P1 describing lines (2d) and a point p, how to compute distance from p to each line i (P0[i],P1[i])? (★★★)

In [27]:
```
def distance(P0, P1, p):
    T = P1 - P0
    L = (T**2).sum(axis=1)
    U = -((P0[:,0]-p[...,0])*T[:,0] + (P0[:,1]-p[...,1])*T[:,1]) /
    U = U.reshape(len(U),1)
    D = P0 + U*T - p
    return np.sqrt((D**2).sum(axis=1))

P0 = np.random.uniform(-10,10,(10,2))
P1 = np.random.uniform(-10,10,(10,2))
p  = np.random.uniform(-10,10,( 1,2))
print(distance(P0, P1, p))
```

```
[ 2.75609682 13.16625876  0.47734244 11.89799392  7.44924655  8.52
266477
 13.58367067  9.07258314  6.23717516  9.79416119]
```

79. Consider 2 sets of points P0,P1 describing lines (2d) and a set of points P, how to compute distance from each point j (P[j]) to each line i (P0[i],P1[i])? (★★★)

In [28]:

```
P0 = np.random.uniform(-10, 10, (10,2))
P1 = np.random.uniform(-10,10,(10,2))
p = np.random.uniform(-10, 10, (10,2))
print(np.array([distance(P0,P1,p_i) for p_i in p]))
```

```
[[ 2.88602843  4.71728439  8.82952128  1.63097634  5.23119758  3.5
3893427
   9.38288165  7.12863047  0.32453436  1.77310483]
 [15.41529823 11.57901067  3.30721455  1.5861924   2.95997521 14.0
6281563
   4.28412707  1.07990146 13.36592607 11.66165921]
 [16.24883565 12.22066272  3.65609164  1.42326463  3.65580733 15.0
3815535
   3.71976195  1.77643945 13.77367572 12.09563496]
 [ 8.56596738  3.31715399  5.62021585  6.27341995  6.83622718 11.2
1038924
   0.9511647   4.94783381  4.47995871  2.80719541]
 [ 3.86791554  2.39264116  2.12863313  2.92858494  5.72564889  1.0
6431812
  11.12075757  7.61575189  7.16671986  5.12369395]
 [ 1.60542747  0.7480818   2.87875128  3.6695826   7.92596928  1.7
507296
  12.97328803  9.81790641  6.23974351  4.11750886]
 [ 8.56139292  3.50809695  5.22786241  5.6741333   6.20827979 10.8
6922085
   0.30580496  4.32004644  4.83782084  3.14884478]
 [ 3.19639135  6.76099181 12.61856145  7.12389437  0.30214956  0.7
8544552
   3.63228634  1.59706723  3.16931706  5.12758914]
 [ 8.25293133  6.84619193  1.89229791  5.38859035  5.52485362  4.3
3375929
  11.71209773  7.41033472 11.30795581  9.31382247]
 [14.95002389 13.06583501  6.85311096  7.35441451  3.34988305 10.3
3232087
  10.69264706  5.22879585 16.55478772 14.68342781]]
```

80. Consider an arbitrary array, write a function that extract a subpart with a fixed shape and centered on a given element (pad with a `fill` value when necessary) (★★★)

In [30]:

```
 Z= np.random.randint(0,10,(10,10))
shape = (5,5)
fill  = 0
position = (1,1)

R = np.ones(shape, dtype=Z.dtype)*fill
P  = np.array(list(position)).astype(int)
Rs = np.array(list(R.shape)).astype(int)
Zs = np.array(list(Z.shape)).astype(int)

R_start = np.zeros((len(shape),)).astype(int)
R_stop  = np.array(list(shape)).astype(int)
Z_start = (P-Rs//2)
Z_stop  = (P+Rs//2)+Rs%2

R_start = (R_start - np.minimum(Z_start,0)).tolist()
Z_start = (np.maximum(Z_start,0)).tolist()
R_stop = np.maximum(R_start, (R_stop - np.maximum(Z_stop-Zs,0))).to
Z_stop = (np.minimum(Z_stop,Zs)).tolist()

r = [slice(start,stop) for start,stop in zip(R_start,R_stop)]
z = [slice(start,stop) for start,stop in zip(Z_start,Z_stop)]
R[r] = Z[z]
print(Z)
print(R)
```

```
[[9 5 8 9 1 0 6 2 6 3]
 [7 1 3 8 5 5 8 9 0 7]
 [3 9 1 3 9 3 8 0 6 3]
 [4 4 0 0 0 7 1 8 5 9]
 [2 0 6 5 2 8 0 6 9 0]
 [9 4 9 8 8 5 9 9 3 4]
 [8 1 8 9 2 6 7 1 1 7]
 [0 0 7 5 5 3 0 3 1 3]
 [4 5 4 6 3 5 0 8 2 6]
 [8 4 1 9 3 5 1 0 1 3]]
[[0 0 0 0 0]
 [0 9 5 8 9]
 [0 7 1 3 8]
 [0 3 9 1 3]
 [0 4 4 0 0]]

/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1308/20
87351065.py:23: FutureWarning: Using a non-tuple sequence for mult
idimensional indexing is deprecated; use `arr[tuple(seq)]` instead
of `arr[seq]`. In the future this will be interpreted as an array
index, `arr[np.array(seq)]`, which will result either in an error
or a different result.
  R[r] = Z[z]
```

81. Consider an array Z = [1,2,3,4,5,6,7,8,9,10,11,12,13,14], how to generate an array R = [[1,2,3,4], [2,3,4,5], [3,4,5,6], ..., [11,12,13,14]]? (★★★)

In [31]:

```
Z = np.arange(1,15,dtype=np.uint32)
R = stride_tricks.as_strided(Z,(11,4),(4,4))
print(R)
```

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

82. Compute a matrix rank (★★★)

In [33]:

```
Z = np.random.uniform(0,1,(10,10))
U, S, V = np.linalg.svd(Z) # Singular Value Decomposition
rank = np.sum(S > 1e-10)
print(rank)
```

```
10
```

83. How to find the most frequent value in an array?

In [34]:

```
Z = np.random.randint(0,10,50)
print(np.bincount(Z).argmax())
```

```
4
```

84. Extract all the contiguous 3x3 blocks from a random 10x10 matrix (★★★)

In [35]:
```
Z = np.random.randint(0,5,(10,10))
n = 3
i = 1 + (Z.shape[0]-3)
j = 1 + (Z.shape[1]-3)
C = stride_tricks.as_strided(Z, shape=(i, j, n, n), strides=Z.strid
print(C)
```

```
   [4 2 3]
   [0 2 0]]]


 [[[2 2 0]
   [3 2 0]
   [1 1 1]]

  [[2 0 1]
   [2 0 4]
   [1 1 3]]

  [[0 1 0]
   [0 4 2]
   [1 3 3]]

  [[1 0 2]
   [4 2 2]
   [3 3 0]]
```

85. Create a 2D array subclass such that Z[i,j] == Z[j,i] (★★★)

In [36]:
```
class Symetric(np.ndarray):
    def __setitem__(self, index, value):
        i,j = index
        super(Symetric, self).__setitem__((i,j), value)
        super(Symetric, self).__setitem__((j,i), value)

def symetric(Z):
    return np.asarray(Z + Z.T - np.diag(Z.diagonal())).view(Symetri

S = symetric(np.random.randint(0,10,(5,5)))
S[2,3] = 42
print(S)
```

```
[[ 1  7 15 10  7]
 [ 7  4  7 16 14]
 [15  7  0 42  6]
 [10 16 42  6 12]
 [ 7 14  6 12  6]]
```

86. Consider a set of p matrices wich shape (n,n) and a set of p vectors with shape (n,1). How to compute the sum of of the p matrix products at once? (result has shape (n,1)) (★★★)

In [37]:

```
p, n = 10, 20
M = np.ones((p,n,n))
V = np.ones((p,n,1))
S = np.tensordot(M, V, axes=[[0, 2], [0, 1]])
print(S)
```

```
[[200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]
 [200.]]
```

87. Consider a 16x16 array, how to get the block-sum (block size is 4x4)? (★★★)

In [39]:

```
Z= np.ones((16,16))
k = 4
S = np.add.reduceat(np.add.reduceat(Z, np.arange(0, Z.shape[0], k),
                                       np.arange(0, Z.shape[1], k),
print(S)
```

```
[[16. 16. 16. 16.]
 [16. 16. 16. 16.]
 [16. 16. 16. 16.]
 [16. 16. 16. 16.]]
```

88. How to implement the Game of Life using numpy arrays? (★★★)

In [41]:

```
def iterate(Z):
    # Count neighbours
    N = (Z[0:-2,0:-2] + Z[0:-2,1:-1] + Z[0:-2,2:] +
         Z[1:-1,0:-2]                + Z[1:-1,2:] +
         Z[2:  ,0:-2] + Z[2:  ,1:-1] + Z[2:  ,2:])

    # Apply rules
    birth = (N==3) & (Z[1:-1,1:-1]==0)
    survive = ((N==2) | (N==3)) & (Z[1:-1,1:-1]==1)
    Z[...] = 0
    Z[1:-1,1:-1][birth | survive] = 1
    return Z

Z = np.random.randint(0,2,(50,50))
for i in range(100): Z = iterate(Z)
print(Z)
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

89. How to get the n largest values of an array (★★★)

In [42]:

```
Z = np.arange(10000)
np.random.shuffle(Z)
n = 5

# Slow
print (Z[np.argsort(Z)[-n:]])

# Fast
print (Z[np.argpartition(-Z,n)[:n]])
```

```
[9995 9996 9997 9998 9999]
[9999 9998 9996 9997 9995]
```

90. Given an arbitrary number of vectors, build the cartesian product (every combinations of every item) (★★★)

In [43]:

```
def cartesian(arrays):
    arrays = [np.asarray(a) for a in arrays]
    shape = (len(x) for x in arrays)

    ix = np.indices(shape, dtype=int)
    ix = ix.reshape(len(arrays), -1).T

    for n, arr in enumerate(arrays):
        ix[:, n] = arrays[n][ix[:, n]]

    return ix

print (cartesian(([1, 2, 3], [4, 5], [6, 7])))
```

```
[[1 4 6]
 [1 4 7]
 [1 5 6]
 [1 5 7]
 [2 4 6]
 [2 4 7]
 [2 5 6]
 [2 5 7]
 [3 4 6]
 [3 4 7]
 [3 5 6]
 [3 5 7]]
```

91. How to create a record array from a regular array? (★★★)

In [44]:

```
Z = np.array([("Hello", 2.5, 3),
              ("World", 3.6, 2)])
R = np.core.records.fromarrays(Z.T,
                               names='col1, col2, col3',
                               formats = 'S8, f8, i8')
print(R)
```

```
[(b'Hello', 2.5, 3) (b'World', 3.6, 2)]
```

92. Consider a large vector Z, compute Z to the power of 3 using 3 different methods (★★★)

In [48]:

```
x = np.random.rand(5e7)

%timeit np.power(x,3)
%timeit x*x*x
%timeit np.einsum('i,i,i->i',x,x,x)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent c
all last)
/var/folders/cj/m_37th0926n097rrzhpj0n1h0000gn/T/ipykernel_1308/33
6921275.py in <module>
----> 1 x = np.random.rand(5e7).xdatatype=float
      2 
      3 get_ipython().run_line_magic('timeit', 'np.power(x,3)')
      4 get_ipython().run_line_magic('timeit', 'x*x*x')
      5 get_ipython().run_line_magic('timeit', "np.einsum('i,i,i->
i',x,x,x)")

mtrand.pyx in numpy.random.mtrand.RandomState.rand()

mtrand.pyx in numpy.random.mtrand.RandomState.random_sample()

_common.pyx in numpy.random._common.double_fill()

TypeError: 'float' object cannot be interpreted as an integer
```

93. Consider two arrays A and B of shape (8,3) and (2,2). How to find rows of A that contain elements of each row of B regardless of the order of the elements in B? (★★★)

In [49]:

```
A = np.random.randint(0,5,(8,3))
B = np.random.randint(0,5,(2,2))

C = (A[..., np.newaxis, np.newaxis] == B)
rows = np.where(C.any((3,1)).all(1))[0]
print(rows)
```

```
[0 3 7]
```

94. Considering a 10x3 matrix, extract rows with unequal values (e.g. [2,2,3]) (★★★)

In [50]:

```
Z = np.random.randint(0,5,(10,3))
print(Z)
# solution for arrays of all dtypes (including string arrays and re
E = np.all(Z[:,1:] == Z[:,:-1], axis=1)
U = Z[~E]
print(U)
# soluiton for numerical arrays only, will work for any number of c
U = Z[Z.max(axis=1) != Z.min(axis=1),:]
print(U)
```

```
[[3 0 1]
 [3 1 3]
 [3 2 0]
 [0 3 3]
 [0 4 4]
 [1 3 0]
 [4 1 3]
 [1 2 4]
 [2 0 1]
 [4 2 0]]
[[3 0 1]
 [3 1 3]
 [3 2 0]
 [0 3 3]
 [0 4 4]
 [1 3 0]
 [4 1 3]
 [1 2 4]
 [2 0 1]
 [4 2 0]]
[[3 0 1]
 [3 1 3]
 [3 2 0]
 [0 3 3]
 [0 4 4]
 [1 3 0]
 [4 1 3]
 [1 2 4]
 [2 0 1]
 [4 2 0]]
```

95. Convert a vector of ints into a matrix binary representation (★★★)

In [51]:
```
I = np.array([0, 1, 2, 3, 15, 16, 32, 64, 128])
B = ((I.reshape(-1,1) & (2**np.arange(8))) != 0).astype(int)
print(B[:,::-1])

# Author: Daniel T. McDonald

I = np.array([0, 1, 2, 3, 15, 16, 32, 64, 128], dtype=np.uint8)
print(np.unpackbits(I[:, np.newaxis], axis=1))
```

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1 1]
 [0 0 0 0 1 1 1 1]
 [0 0 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1 1]
 [0 0 0 0 1 1 1 1]
 [0 0 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
```

96. Given a two dimensional array, how to extract unique rows? (★★★)

In [52]:
```
Z = np.random.randint(0,2,(6,3))
T = np.ascontiguousarray(Z).view(np.dtype((np.void, Z.dtype.itemsiz
_, idx = np.unique(T, return_index=True)
uZ = Z[idx]
print(uZ)
```

```
[[0 1 0]
 [0 1 1]
 [1 0 1]
 [1 1 0]
 [1 1 1]]
```

97. Considering 2 vectors A & B, write the einsum equivalent of inner, outer, sum, and mul function (★★★)

In [53]:
```
A = np.random.uniform(0,1,10)
B = np.random.uniform(0,1,10)

np.einsum('i->', A)       # np.sum(A)
np.einsum('i,i->i', A, B) # A * B
np.einsum('i,i', A, B)    # np.inner(A, B)
np.einsum('i,j->ij', A, B)    # np.outer(A, B)
```

Out[53]:
```
array([[0.31276462, 0.30089477, 0.45198479, 0.40211532, 0.26163358
,
         0.1003606 , 0.18592724, 0.25024273, 0.4183349 , 0.48905826
],
       [0.10086815, 0.09704007, 0.14576736, 0.1296842 , 0.08437814
,
         0.0323668 , 0.05996247, 0.08070453, 0.1349151 , 0.15772374
],
       [0.26565585, 0.25557385, 0.38390661, 0.3415485 , 0.2222262
,
         0.08524424, 0.15792278, 0.21255104, 0.35532508, 0.41539606
],
       [0.27287885, 0.26252272, 0.39434477, 0.35083497, 0.22826838
,
         0.08756197, 0.1622166 , 0.21833015, 0.36498612, 0.42669039
],
       [0.09325224, 0.08971319, 0.1347614 , 0.11989258, 0.07800729
,
         0.02992299, 0.05543508, 0.07461105, 0.12472852, 0.14581503
],
       [0.44942489, 0.4323686 , 0.64947633, 0.57781674, 0.37595252
,
         0.14421246, 0.26716682, 0.3595845 , 0.60112336, 0.70274879
],
       [0.13689038, 0.13169521, 0.19782408, 0.17599727, 0.11451142
,
         0.04392569, 0.08137638, 0.10952589, 0.18309624, 0.21405034
],
       [0.44501804, 0.428129  , 0.64310787, 0.57215094, 0.3722661
,
         0.14279837, 0.26454711, 0.35605859, 0.59522902, 0.69585796
],
       [0.03330262, 0.03203874, 0.04812654, 0.04281653, 0.02785828
,
         0.01068622, 0.0197972 , 0.0266454 , 0.04454356, 0.05207406
],
       [0.05833635, 0.05612241, 0.08430347, 0.0750019 , 0.04879947
,
         0.0187191 , 0.03467885, 0.04667487, 0.07802715, 0.09121836
]])
```

98. Considering a path described by two vectors (X,Y), how to sample it using equidistant samples (★★★)?

In [55]:

```
phi = np.arange(0, 10*np.pi, 0.1)
a = 1
x = a*phi*np.cos(phi)
y = a*phi*np.sin(phi)

dr = (np.diff(x)**2 + np.diff(y)**2)**.5 # segment lengths
r = np.zeros_like(x)
r[1:] = np.cumsum(dr)                # integrate path
r_int = np.linspace(0, r.max(), 200) # regular spaced path
x_int = np.interp(r_int, r, x)       # integrate path
y_int = np.interp(r_int, r, y)
```

99. Given an integer n and a 2D array X, select from X the rows which can be interpreted as draws from a multinomial distribution with n degrees, i.e., the rows which only contain integers and which sum to n. (★★★)

In [56]:

```

X = np.asarray([[1.0, 0.0, 3.0, 8.0],
                [2.0, 0.0, 1.0, 1.0],
                [1.5, 2.5, 1.0, 0.0]])
n = 4
M = np.logical_and.reduce(np.mod(X, 1) == 0, axis=-1)
M &= (X.sum(axis=-1) == n)
print(X[M])
```

```
[[2. 0. 1. 1.]]
```

100. Compute bootstrapped 95% confidence intervals for the mean of a 1D array X (i.e., resample the elements of an array with replacement N times, compute the mean of each sample, and then compute percentiles over the means). (★★★)

In [57]:

```
X = np.random.randn(100) # random 1D array
N = 1000 # number of bootstrap samples
idx = np.random.randint(0, X.size, (N, X.size))
means = X[idx].mean(axis=1)
confint = np.percentile(means, [2.5, 97.5])
print(confint)
```

```
[-0.22626447  0.16936667]
```