Numpy

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

(**hint**: import … as …)

```
import numpy as np
```

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

(**hint**: np.__version__, np.show_config)

```
print(np.__version__)
np.show_config()
```

```
1.21.6
blas_mkl_info:
  NOT AVAILABLE
blis_info:
  NOT AVAILABLE
openblas_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
    runtime_library_dirs = ['/usr/local/lib']
blas_opt_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
    runtime_library_dirs = ['/usr/local/lib']
lapack_mkl_info:
  NOT AVAILABLE
openblas_lapack_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
    runtime_library_dirs = ['/usr/local/lib']
lapack_opt_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
    runtime_library_dirs = ['/usr/local/lib']
Supported SIMD extensions in this NumPy install:
    baseline = SSE,SSE2,SSE3
```

```
found = SSSE3,SSE41,POPCNT,SSE42,AVX,F16C,FMA3,AVX2
not found = AVX512F,AVX512CD,AVX512_KNL,AVX512_KNM,AVX512_SKX,AVX512_CLX,AVX512_CNL,
```

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

(**hint**: np.zeros)

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

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

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

(**hint**: size, itemsize)

```
Z = np.zeros((10,10))
print("%d bytes" % (Z.size * Z.itemsize))
```

```
800 bytes
```

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

(**hint**: np.info)

```
%run  'python -c "import numpy; numpy.info(numpy.add)"'
```

```
ERROR:root:File `'python -c "import numpy; numpy.info(numpy.add)".py'` not found.
```

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

(**hint**: array[4])

```
Z = np.zeros(10)
Z[4] = 1
print(Z)
```

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

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

(**hint**: np.arange)

```
Z = np.arange(10,50)
print(Z)
```

```
[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 49]
```

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

(**hint**: array[::-1])

```
Z = np.arange(50)
Z = Z[::-1]
print(Z)
```

```
[49 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  8  7  6  5  4  3  2
  1  0]
```

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

(**hint**: reshape)

```
Z = np.arange(9).reshape(3,  3)
print(Z)
```

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

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

(**hint**: np.nonzero)

```
nz = np.nonzero([1,2,0,0,4,0])
print(nz)
```

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

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

(**hint**: np.eye)

```
Z =np.eye(3)
print(Z)
```

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

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

(**hint**: np.random.random)

```
Z = np.random.random((3,3,3))
print(Z)
```

```
[[[0.99251066 0.14862012 0.4161226 ]
  [0.69083956 0.71746249 0.55022612]
  [0.35796788 0.01936327 0.8844194 ]]

 [[0.08320042 0.55468105 0.87617262]
  [0.54508254 0.306775   0.17695722]
  [0.47731544 0.82876506 0.58229943]]

 [[0.37703969 0.98724505 0.17351072]
  [0.30747905 0.84774674 0.03000441]
  [0.81635611 0.98284099 0.20521686]]]
```

z#### 13. Create a 10x10 array with random values and find the minimum and maximum values (★☆☆) (**hint**: min, max)

```
Z = np.random.random((10,10))
Zmin, Zmax = Z.min(), Z.max()
print(Zmin, Zmax)
```

```
0.0030300874657973598 0.9888716090564297
```

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

(**hint**: mean)

```
Z = np.random.random(30)
m = Z.mean()
print(m)
```

```
0.5579366198466744
```

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

(**hint**: array[1:-1, 1:-1])

```
Z = np.ones((10,10))
Z[1:-1, 1:-1] = 0
print(Z)
```

```
[[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? (★☆☆)

(**hint**: np.pad)

```
Z = np.ones((5,5))
Z = np.pad(Z, pad_width=1, mode='constant',constant_values=0)
print(Z)
Z[:, [0, -1]] = 0
Z[[0, -1], :] = 0
print(Z)
```

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

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

(**hint**: NaN = not a number, inf = infinity)

```
0 * np.nan
np.nan == np.nan
np.inf > np.nan
```

```
np.nan - np.nan
0.3 == 3 * 0.1
```

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

```
nan
False
False
nan
True
False
```

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

(**hint**: np.diag)

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

```
[[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 (★☆☆)

(**hint**: array[::2])

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

```
[[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?

(**hint**: np.unravel_index)

```
print(np.unravel_index(99,(6,7,8)))
```

```
(1, 5, 3)
```

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

(**hint**: np.tile)

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

```
[[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 (★☆☆)

(**hint**: (x - min) / (max - min))

```
Z = np.random.random((5,5))
Z = (Z - np.mean (Z)) / (np.std (Z))
print(Z)
```

```
[[ 1.02053174  1.46989079 -1.18391412 -1.01220046 -0.80126663]
 [-0.82862373 -1.24694967  0.59845538  1.08167068  0.32272793]
 [-0.36026594 -1.4455928   1.85306833  0.80188572  1.16601031]
 [ 0.48193365  0.08408601 -0.61437195 -0.22756929 -1.08387672]
 [-0.10456265 -1.24402145  0.12973711  1.82215081 -0.67893304]]
```

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

(**hint**: np.dtype)

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

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

(**hint**: np.dot | @)

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

Z = np.ones((5,3)) @ np.ones((3,2))
print(Z)
```

```
[[3. 3.]
 [3. 3.]
 [3. 3.]
 [3. 3.]
 [3. 3.]]
[[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. (★☆☆)

(**hint**: >, <=)

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

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

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

(**hint**: np.sum)

```
# Author: Jake VanderPlas

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

```
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
```

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-37-4e3654d03fce> in <module>
----> 1 Z**Z
      2 2 << Z >> 2
      3 Z <- Z
      4 1j*Z
      5 Z/1/1

ValueError: Integers to negative integer powers are not allowed.
```

SEARCH STACK OVERFLOW

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)
```

```
print(np.array(0) / np.array(0))
print(np.array(0) // np.array(0))
```

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

```
nan
0
[-9.22337204e+18]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: RuntimeWarning: divide b
```

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

(**hint**: np.uniform, np.copysign, np.ceil, np.abs)

```
Z = np.random.uniform(-10,+10,10)
print(np.copysign(np.ceil(np.abs(Z)), Z))
print(np.where(Z>0, np.ceil(Z), np.floor(Z)))
```

```
[ 7.  8. -2. -9. -9. -7.  6. -4. -8. -1.]
[ 7.  8. -2. -9. -9. -7.  6. -4. -8. -1.]
```

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

(**hint**: np.intersect1d)

```
Z1 = np.random.randint(0,10,10)
Z2 = np.random.randint(0,10,10)
print(np.intersect1d(Z1,Z2))
```

```
[2 3 5 6]
```

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

(**hint**: np.seterr, np.errstate)

```
defaults = np.seterr(all="ignore")
Z = np.ones(1) / 0
_ = np.seterr(**defaults)
with np.errstate(all="ignore"):
  np.arange(3) / 0
```

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

(**hint**: imaginary number)

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

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid
  """Entry point for launching an IPython kernel.
False
```

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

(**hint**: np.datetime64, np.timedelta64)

```
yesterday = np.datetime64('today') - np.timedelta64(1)
today = np.datetime64('today')
tomorrow = np.datetime64('today') + np.timedelta64(1)
```

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

(**hint**: np.arange(dtype=datetime64['D']))

```
Z = np.arange('2016-07', '2016-08', dtype='datetime64[D]')
print(Z)
```

```
['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)? (★★☆)

(**hint**: np.add(out=), np.negative(out=), np.multiply(out=), np.divide(out=))

```
A = np.ones(3)*1
B = np.ones(3)*2
np.add(A,B,out=B)
np.divide(A,2,out=A)
np.negative(A,out=A)
np.multiply(A,B,out=A)
```

```
array([-1.5, -1.5, -1.5])
```

36. Extract the integer part of a random array using 5 different methods (★★☆)

(**hint**: %, np.floor, np.ceil, astype, np.trunc)

```
Z = np.random.uniform(0,10,10)
print(Z - Z%1)
print(Z // 1)
print(np.floor(Z))
print(Z.astype(int))
print(np.trunc(Z))
```

```
[8. 1. 9. 1. 3. 9. 7. 9. 9. 4.]
[8. 1. 9. 1. 3. 9. 7. 9. 9. 4.]
[8. 1. 9. 1. 3. 9. 7. 9. 9. 4.]
[8 1 9 1 3 9 7 9 9 4]
[8. 1. 9. 1. 3. 9. 7. 9. 9. 4.]
```

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

(**hint**: np.arange)

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

Z = np.title(np.arange(0,5), (5,1))
print(Z)
```

[[0 1 2 3 4]

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

(**hint**: np.fromiter)

<ipython-input-50-eeedd7512df2> in <module>

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

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

```
--> 314                                    "{!r}".format(  name  , attr))
```

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

(**hint**: np.linspace)

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

```
[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 (★★☆)

(**hint**: sort)

```
Z = np.random.random(10)
Z.sort()
print(Z)
```

```
[0.20129385 0.20993647 0.32247891 0.46430325 0.50964432 0.53640436
 0.67248531 0.85219591 0.92271984 0.98638123]
```

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

(**hint**: np.add.reduce)

```
Z = np.arange(10)
np.add.reduce(Z)
```

```
45
```

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

(**hint**: np.allclose, np.array_equal)

```
A = np.random.randint(0,2,5)
B = np.random.randint(0,2,5)
equal = np.allclose(A,B)
print(equal)

equal = np.array_equal(A,B)
print(equal)
```

```
False
False
```

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

(**hint**: flags.writeable)

```
Z = np.zeros(10)
Z.flags.writeable = False
Z[0] = 1
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-57-dcc5e7f145b5> in <module>
      1 Z = np.zeros(10)
      2 Z.flags.writeable = False
----> 3 Z[0] = 1

ValueError: assignment destination is read-only
```

SEARCH STACK OVERFLOW

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

(**hint**: np.sqrt, np.arctan2)

```
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)
```

```
[1.22110221 0.47990324 0.92865314 0.95890456 0.61579872 0.20295019
 0.5594253  0.961733   0.07567443 0.66182969]
```

```
[0.86361228 0.32780376 0.75233126 0.65458894 1.40890017 0.42577386
 0.34256306 0.41273576 0.64955786 1.15918034]
```

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

(**hint**: argmax)

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

```
[0.49967956 0.18632047 0.24254243 0.62701719 0.11721601 0.53412272
 0.56939999 0.09691906 0.         0.0204506 ]
```

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

(**hint**: np.meshgrid)

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

```
[[(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))

(**hint**: np.subtract.outer)

```
x = np.arange(8)
y = x+ 0.5
c = 1.0 / np. subtract.outer(x,y)
print(np.linalg.det(c))
```

```
3638.163637117973
```

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

(**hint**: np.iinfo, np.finfo, eps)

```
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).min)
  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? (★★☆)

(**hint**: np.set_printoptions)

```
np.set_printoptions(threshold=float("inf"))
z = np.zeros((40,40))
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. 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. 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. 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. 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. 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.
  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. 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. 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. 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. 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. 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. 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. 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.
 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. 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. 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. 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. 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. 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. 0. 0. 0. 0.]]
```

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

(**hint**: argmin)

```
z = np.arange(1000)
v = np.random.uniform(0,100)
index = (np.abs(z-v)).argmin()
print(z[index])
```

```
29
```

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

(**hint**: dtype)

```
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.))]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: FutureWarning: Passing (
  """
```

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

(**hint**: np.atleast_2d, T, np.sqrt)

```
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)
```

```
[[0.00000000e+00 3.20073125e-01 1.87915277e-01 4.30348607e-02
  3.98117670e-01 3.24031172e-01 3.95497885e-01 8.54255214e-02
  2.20692164e-01 8.58363562e-02]
 [3.20073125e-01 0.00000000e+00 1.32157848e-01 3.63107986e-01
  7.80445452e-02 3.95804688e-03 7.54247600e-02 2.34647604e-01
  9.93809609e-02 2.34236769e-01]
 [1.87915277e-01 1.32157848e-01 0.00000000e+00 2.30950138e-01
  2.10202393e-01 1.36115894e-01 2.07582608e-01 1.02489756e-01
  3.27768866e-02 1.02078921e-01]
 [4.30348607e-02 3.63107986e-01 2.30950138e-01 0.00000000e+00
  4.41152531e-01 3.67066033e-01 4.38532746e-01 1.28460382e-01
  2.63727025e-01 1.28871217e-01]
 [3.98117670e-01 7.80445452e-02 2.10202393e-01 4.41152531e-01
  0.00000000e+00 7.40864984e-02 2.61978528e-03 3.12692149e-01
  1.77425506e-01 3.12281314e-01]
 [3.24031172e-01 3.95804688e-03 1.36115894e-01 3.67066033e-01
  7.40864984e-02 0.00000000e+00 7.14667131e-02 2.38605650e-01
  1.03339008e-01 2.38194816e-01]
 [3.95497885e-01 7.54247600e-02 2.07582608e-01 4.38532746e-01
  2.61978528e-03 7.14667131e-02 0.00000000e+00 3.10072364e-01
  1.74805721e-01 3.09661529e-01]
 [8.54255214e-02 2.34647604e-01 1.02489756e-01 1.28460382e-01
  3.12692149e-01 2.38605650e-01 3.10072364e-01 0.00000000e+00
  1.35266643e-01 4.10834792e-04]
 [2.20692164e-01 9.93809609e-02 3.27768866e-02 2.63727025e-01
```

```
 1.77425506e-01 1.03339008e-01 1.74805721e-01 1.35266643e-01
 0.00000000e+00 1.34855808e-01]
[8.58363562e-02 2.34236769e-01 1.02078921e-01 1.28871217e-01
 3.12281314e-01 2.38194816e-01 3.09661529e-01 4.10834792e-04
 1.34855808e-01 0.00000000e+00]]
```

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

(**hint**: astype(copy=False))

```
Z = (np.random.rand(10)*100).astype(np.float32)
Y = Z.view(np.int32)
Y[:] = Z
print(Y)
```

```
[63 97 11 81 44 79 75 79  6 34]
```

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

(**hint**: np.genfromtxt)

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

```
from io import StringIO
s = StringIO('''1, 2, 3, 4, 5
                6,  ,  , 7, 8
                 ,  , 9,10,11
''')
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]]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: DeprecationWarning: `np
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
```

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

(**hint**: np.ndenumerate, np.ndindex)

```
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 (★★☆)

(**hint**: np.meshgrid, np.exp)

```
X, Y = np.meshgrid(np.linspace(-1,1,10), np.linspace(-1,1,10))
D = np.sqrt(X*X+Y*Y)
sigma, mu = 1.0, 0.0
G = np.exp(-( (D-mu)**2 / (2.0 * sigma**2 ) ) )
print(G)
```

```
[[0.60653066 0.7389913  0.85699689 0.94595947 0.99384617 0.99384617
  0.94595947 0.85699689 0.7389913  0.60653066]
 [0.7389913  0.83278885 0.91087038 0.96695078 0.99627278 0.99627278
  0.96695078 0.91087038 0.83278885 0.7389913 ]
 [0.85699689 0.91087038 0.95348657 0.98299939 0.99809662 0.99809662
  0.98299939 0.95348657 0.91087038 0.85699689]
 [0.94595947 0.96695078 0.98299939 0.99384617 0.99931436 0.99931436
  0.99384617 0.98299939 0.96695078 0.94595947]
 [0.99384617 0.99627278 0.99809662 0.99931436 0.9999238  0.9999238
  0.99931436 0.99809662 0.99627278 0.99384617]
 [0.99384617 0.99627278 0.99809662 0.99931436 0.9999238  0.9999238
  0.99931436 0.99809662 0.99627278 0.99384617]
 [0.94595947 0.96695078 0.98299939 0.99384617 0.99931436 0.99931436
  0.99384617 0.98299939 0.96695078 0.94595947]
 [0.85699689 0.91087038 0.95348657 0.98299939 0.99809662 0.99809662
  0.98299939 0.95348657 0.91087038 0.85699689]
 [0.7389913  0.83278885 0.91087038 0.96695078 0.99627278 0.99627278
  0.96695078 0.91087038 0.83278885 0.7389913 ]
```

```
[0.60653066 0.7389913  0.85699689 0.94595947 0.99384617 0.99384617
 0.94595947 0.85699689 0.7389913  0.60653066]]
```

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

(**hint**: np.put, np.random.choice)

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

```
[[0. 0. 0. 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. 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.]
 [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 (★★☆)

(**hint**: mean(axis=,keepdims=))

```
X = np.random.rand(5, 10)
Y = X - X.mean(axis=1, keepdims=True)
Y = X - X.mean(axis=1).reshape(-1,1)
print(Y)
```

```
[[ 5.80132948e-01  1.13012191e-01 -1.08936510e-01 -1.36051737e-01
   2.42709731e-02 -2.24651525e-01  1.57318664e-02 -6.82252163e-02
  -2.10452841e-01  1.51698513e-02]
 [ 4.99714648e-01 -2.09603131e-01 -1.07927073e-01 -4.21373839e-02
   1.85785610e-01 -7.06319183e-02 -1.34607769e-02 -1.51108829e-01
   6.35133730e-02 -1.54144519e-01]
 [ 1.13956884e-01 -1.48212655e-01  3.87695901e-01  1.97456028e-01
   4.06085456e-01 -8.03262047e-05 -1.13113347e-01 -1.37514190e-01
  -3.29520839e-01 -3.76752912e-01]
 [ 5.22356409e-04  2.99689796e-01 -2.17553715e-01  4.13070636e-01
  -2.03680881e-02 -2.41976718e-01 -2.86102991e-01  1.98303815e-01
   3.82934420e-01 -5.28519512e-01]
 [-2.72967214e-01  3.97264212e-01  1.01854901e-02 -4.54812459e-02
  -2.98694138e-01 -3.12703967e-01  6.31506992e-01  1.22730734e-01
  -2.20299554e-01 -1.15413084e-02]]
```

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

(**hint**: argsort)

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

```
[[5 6 7]
 [1 2 3]
 [6 9 5]]
[[1 2 3]
 [5 6 7]
 [6 9 5]]
```

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

(**hint**: any, ~)

```
Z = np.random.randint(0,3,(3,10))
print((~Z.any(axis=0)).any())
Z=np.array([
    [0,1,np.nan],
    [1,2,np.nan],
    [4,5,np.nan]
])
print(np.isnan(Z).all(axis=0))
```

```
False
[False False  True]
```

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

(**hint**: np.abs, argmin, flat)

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

```
0.46520439826407833
```

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

(**hint**: np.nditer)

```
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 (★★☆)

(**hint**: class method)

```
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.name = getattr(obj, 'name', "no name")

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-106-651c34b1660b> in <module>
----> 1 class NamedArray(np.ndarray):
      2   def __new__(cls, array, name="no name"):
      3     obj = np.asarray(array).view(cls)
      4     obj.name = name
      5     return obj

<ipython-input-106-651c34b1660b> in NamedArray()
      8      self.name = getattr(obj, 'name', "no name")
      9
---> 10   Z = NamedArray(np.arange(10), "range_10")
     11   print (Z.name)

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

SEARCH STACK OVERFLOW

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

(**hint**: np.bincount | np.add.at)

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

np.add.at(Z, I, 1)
print(Z)
```

```
[5. 3. 2. 1. 2. 3. 2. 2. 3. 7.]
[ 9.  5.  3.  1.  3.  5.  3.  3.  5. 13.]
```

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

(**hint**: np.bincount)

```
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 (★★★)

(**hint**: np.unique)

```
w, h = 256, 256
I = np.random.randint(0, 4, (h, w, 3)).astype(np.ubyte)
colors = np.unique(I.reshape(-1, 3), axis=0)
n = len(colors)
print(n)

# Faster version
# Author: Mark Setchell
# https://stackoverflow.com/a/59671950/2836621

w, h = 256, 256
I = np.random.randint(0,4,(h,w,3), dtype=np.uint8)

# View each pixel as a single 24-bit integer, rather than three 8-bit bytes
I24 = np.dot(I.astype(np.uint32),[1,256,65536])

# Count unique colours
```

```
n = len(np.unique(I24))
print(n)
```

```
64
64
```

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

(**hint**: sum(axis=(-2,-1)))

```
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)
```

```
[[53 47 53 35]
 [57 48 41 61]
 [59 55 57 59]]
[[53 47 53 35]
 [57 48 41 61]
 [59 55 57 59]]
```

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? (★★★)

(**hint**: np.bincount)

```
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())
```

```
[0.40806798 0.45420423 0.4111099  0.48318569 0.55796092 0.62364367
 0.30106128 0.43831555 0.1016394  0.40411565]
0    0.408068
```

```
1    0.454204
2    0.411110
3    0.483186
4    0.557961
5    0.623644
6    0.301061
7    0.438316
8    0.101639
9    0.404116
dtype: float64
```

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

(**hint**: np.diag)

```
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)
```

```
array([0.50418913, 1.42272117, 1.0342917 , 0.56407643, 0.94970817])
```

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

(**hint**: array[::4])

```
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)? (★★★)

(**hint**: array[:, :, None])

```
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? (★★★)

(**hint**: array[[]] = array[[]])

```
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 (★★★)

(**hint**: repeat, np.roll, np.sort, view, np.unique)

```
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)
```

```
[( 1, 82) ( 1, 99) ( 7, 21) ( 7, 29) (14, 20) (14, 82) (18, 50) (18, 60)
 (20, 82) (21, 29) (26, 43) (26, 92) (27, 33) (27, 42) (27, 72) (27, 74)
 (33, 74) (35, 45) (35, 85) (40, 57) (40, 84) (42, 72) (43, 92) (45, 85)
 (50, 60) (57, 84) (72, 85) (72, 98) (82, 99) (85, 98)]
```

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

(**hint**: np.repeat)

```
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? (★★★)

(**hint**: np.cumsum)

```
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))

# Author: Jeff Luo (@Jeff1999)
# make sure your NumPy >= 1.20.0

from numpy.lib.stride_tricks import sliding_window_view

Z = np.arange(20)
print(sliding_window_view(Z, window_shape=3).mean(axis=-1))
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.]
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 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]) (★★★)

(**hint**: from numpy.lib import stride_tricks)

```
from numpy.lib import stride_tricks

def rolling(a, window):
    shape = (a.size - window + 1, window)
    strides = (a.strides[0], a.strides[0])
    return stride_tricks.as_strided(a, shape=shape, strides=strides)
Z = rolling(np.arange(10), 3)
print(Z)

# Author: Jeff Luo (@Jeff1999)

Z = np.arange(10)
print(sliding_window_view(Z, window_shape=3))
```

```
     [[0 1 2]
      [1 2 3]
      [2 3 4]
      [3 4 5]
      [4 5 6]
      [5 6 7]
      [6 7 8]
      [7 8 9]]
     [[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? (★★★)

(**hint**: np.logical_not, np.negative)

```
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)
```

```
     array([ 0.63092902, -0.98436307, -0.97215174, -0.7027369 , -0.35949374,
```

```
-0.84048   , -0.11672294, -0.46603637,  0.47840666,  0.16702858,
-0.09749903, -0.08420075, -0.95446779,  0.39109019, -0.76572623,
 0.55528996,  0.01830981,  0.49098198,  0.56718352,  0.57221034,
-0.95985733, -0.90716885,  0.70870602, -0.0601778 ,  0.81704848,
-0.36316943, -0.60707573,  0.41543441,  0.29522177, -0.87106297,
-0.19375618,  0.80604805, -0.77253338, -0.93682892, -0.80761491,
 0.41138007, -0.40545055, -0.34525557,  0.76927445,  0.73509794,
 0.82868098, -0.29811923, -0.74197136,  0.33960457,  0.11897893,
 0.84614875,  0.99370613, -0.3103349 , -0.1528385 , -0.27992735,
-0.14979803,  0.13649683,  0.21103765,  0.78988541,  0.26289949,
-0.53082131, -0.86612501,  0.78379908, -0.08013812, -0.77089599,
 0.45100924,  0.49000354,  0.99343249, -0.57804365,  0.16461235,
 0.10147023, -0.64350599, -0.1936761 , -0.5440476 ,  0.32390334,
 0.11056388,  0.87238043,  0.65989684, -0.23006791, -0.79977635,
 0.40453075, -0.1215098 ,  0.10722015,  0.58745431, -0.1641337 ,
 0.9871075 , -0.8698384 , -0.35406785,  0.55193152, -0.5445269 ,
 0.43015992, -0.69988676,  0.0104265 , -0.9742748 ,  0.29460464,
-0.66222638, -0.6454917 , -0.62744371,  0.77313663, -0.94281646,
-0.7049324 ,  0.70749796, -0.38292099,  0.59385579,  0.13070186])
```

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])? (★★★)

```
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]) / L
    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))
```

```
[ 4.41574382  1.68538094  2.53790368  0.69901269  5.73276278 10.04277967
  6.18554029  3.13202785  0.79591638  4.05342961]
```

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])? (★★★)

```
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]))
```

```
[[1.52471112e+01 5.38766346e+00 1.73143454e+01 1.34852025e+01
  1.53205782e+01 9.96077960e+00 9.69342865e-01 5.01675493e+00
```

```
 1.63912599e+00 1.58172318e+01]
[5.02113455e+00 5.96922146e+00 5.29448110e+00 1.46825962e+00
 2.11443334e+00 4.36917298e-02 1.05396006e+01 7.70311186e-01
 1.97023673e-01 8.16968981e+00]
[7.53208327e+00 3.24987610e+00 7.36715702e+00 1.30924529e+00
 3.03175072e+00 2.50285645e+00 8.30528624e+00 3.64249328e+00
 2.59893919e+00 1.09616681e+01]
[1.54900673e+01 5.09601794e+00 1.75051661e+01 1.34353884e+01
 1.53835919e+01 1.02081766e+01 1.17898187e+00 5.31491794e+00
 1.93562259e+00 1.60975006e+01]
[3.39045900e-01 1.79537727e+01 3.78766633e+00 1.01093739e+01
 6.64388950e+00 5.03363759e+00 1.31181998e+01 9.49457393e+00
 1.17033910e+01 5.10543932e-01]
[1.35264463e+01 2.81287085e+00 1.43803022e+01 8.23746551e+00
 1.06863234e+01 8.37073178e+00 1.58092943e+00 6.19995264e+00
 3.76274073e+00 1.54786758e+01]
[1.32757490e+01 2.19074286e+00 1.38685814e+01 7.17478466e+00
 9.78821078e+00 8.14769146e+00 2.00921508e+00 6.54771530e+00
 4.29694711e+00 1.55169792e+01]
[1.55922169e+00 1.54663612e+01 6.71221868e-01 4.74620199e+00
 1.85002097e+00 6.80014014e+00 1.58506108e+01 8.44000097e+00
 9.68200497e+00 1.17480139e-02]
[1.41821922e+01 6.83215100e+00 1.65346138e+01 1.39040844e+01
 1.51943369e+01 8.87059244e+00 8.86132423e-02 3.59187607e+00
 1.88300398e-01 1.45298131e+01]
[3.47124885e+00 8.73935377e+00 4.38742467e+00 2.88391978e+00
 2.53342795e+00 1.65367726e+00 1.16680622e+01 1.77764973e+00
 2.91639219e+00 6.05903565e+00]]
```

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) (★★★)

(**hint**: minimum, maximum)

```
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))).tolist()
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)
```

```
[[6 2 2 2 9 0 4 6 2 6]
 [3 5 2 7 8 5 7 0 3 6]
 [5 9 5 1 1 8 1 3 3 7]
 [3 2 4 7 8 3 6 9 2 0]
 [8 6 8 2 7 7 4 5 4 0]
 [2 0 2 1 2 0 2 2 2 2]
 [6 1 0 2 9 5 9 0 7 3]
 [8 3 5 4 5 0 9 4 5 8]
 [7 1 5 3 9 7 9 4 8 4]
 [6 8 1 7 5 4 1 8 0 8]]
[[0 0 0 0 0]
 [0 6 2 2 2]
 [0 3 5 2 7]
 [0 5 9 5 1]
 [0 3 2 4 7]]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:23: FutureWarning: Using a
```

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]]? (★★★)

(**hint**: stride_tricks.as_strided)

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

# Author: Jeff Luo (@Jeff1999)

Z = np.arange(1, 15, dtype=np.uint32)
print(sliding_window_view(Z, window_shape=4))
```

```
[[ 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]]
[[ 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 (★★★)

(**hint**: np.linalg.svd) (suggestion: np.linalg.svd)

```
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)

# alternative solution:
# Author: Jeff Luo (@Jeff1999)

rank = np.linalg.matrix_rank(Z)
print(rank)
```

```
10
10
```

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

(**hint**: np.bincount, argmax)

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

```
3
```

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

(**hint**: stride_tricks.as_strided)

```
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.strides + Z.strides)
print(C)

# Author: Jeff Luo (@Jeff1999)

Z = np.random.randint(0,5,(10,10))
print(sliding_window_view(Z, window_shape=(3, 3)))
```

```
        [3 1 0]]

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

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

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

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

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

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

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


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

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

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

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

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

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

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

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

(**hint**: class method)

```
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(Symetric)

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

```
[[ 9  8 16  8 11]
 [ 8  1  9 10  5]
 [16  9  7 42 11]
 [ 8 10 42  0  9]
 [11  5 11  9  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)) (★★★)

(**hint**: np.tensordot)

```
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.]]
```

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

(**hint**: np.add.reduceat)

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

# alternative solution:
# Author: Sebastian Wallkötter (@FirefoxMetzger)

Z = np.ones((16,16))
k = 4

windows = np.lib.stride_tricks.sliding_window_view(Z, (k, k))
S = windows[::k, ::k, ...].sum(axis=(-2, -1))

# Author: Jeff Luo (@Jeff1999)

Z = np.ones((16, 16))
k = 4
print(sliding_window_view(Z, window_shape=(k, k))[::k, ::k].sum(axis=(-2, -1)))
```

```
     [[16. 16. 16. 16.]
      [16. 16. 16. 16.]
      [16. 16. 16. 16.]
      [16. 16. 16. 16.]]
     [[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? (★★★)

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

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

(**hint**: np.argsort | np.argpartition)

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

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

(**hint**: np.indices)

```
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? (★★★)

(**hint**: np.core.records.fromarrays)

```
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 (★★★)

(**hint**: np.power, *, np.einsum)

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

%timeit np.power(x,3)
```

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

```
2.18 s ± 13.1 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
79.1 ms ± 1.21 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
69.1 ms ± 1.78 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

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? (★★★)

(**hint**: np.where)

```
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)
```

```
[4 5 6 7]
```

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

```
Z = np.random.randint(0,5,(10,3))
print(Z)
# solution for arrays of all dtypes (including string arrays and record arrays)
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 columns in Z
U = Z[Z.max(axis=1) != Z.min(axis=1),:]
print(U)
```

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

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

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

(**hint**: np.unpackbits)

```
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? (★★★)

(**hint**: np.ascontiguousarray)

```
Z = np.random.randint(0,2,(6,3))
T = np.ascontiguousarray(Z).view(np.dtype((np.void, Z.dtype.itemsize * Z.shape[1])))
_, idx = np.unique(T, return_index=True)
uZ = Z[idx]
print(uZ)

# Author: Andreas Kouzelis
# NumPy >= 1.13
uZ = np.unique(Z, axis=0)
print(uZ)
```

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

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

(**hint**: np.einsum)

```
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)
```

```
array([[1.87049371e-02, 2.51885275e-01, 1.32406087e-01, 3.80289478e-01,
        3.10029407e-01, 3.51176952e-02, 3.90091061e-01, 3.23664880e-01,
        2.66426882e-01, 3.55150300e-01],
       [4.30163817e-02, 5.79269159e-01, 3.04498794e-01, 8.74564684e-01,
        7.12985201e-01, 8.07613614e-02, 8.97105718e-01, 7.44343163e-01,
        6.12710987e-01, 8.16751259e-01],
       [1.50702292e-02, 2.02939407e-01, 1.06677188e-01, 3.06392349e-01,
        2.49785081e-01, 2.82936914e-02, 3.14289307e-01, 2.60770935e-01,
        2.14655316e-01, 2.86138169e-01],
       [1.30430702e-03, 1.75641185e-02, 9.23275965e-03, 2.65178243e-02,
        2.16185453e-02, 2.44877895e-03, 2.72012948e-02, 2.25693554e-02,
        1.85781139e-02, 2.47648536e-02],
       [1.32582344e-02, 1.78538640e-01, 9.38506734e-02, 2.69552741e-01,
        2.19751744e-01, 2.48917510e-02, 2.76500194e-01, 2.29416695e-01,
        1.88845867e-01, 2.51733856e-01],
       [3.14413957e-02, 4.23397554e-01, 2.22563281e-01, 6.39234012e-01,
        5.21132855e-01, 5.90298351e-02, 6.55709632e-01, 5.44052915e-01,
        4.47840748e-01, 5.96977209e-01],
```

```
[5.03141367e-03, 6.77542519e-02, 3.56157197e-02, 1.02293511e-01,
 8.33943568e-02, 9.44625749e-03, 1.04930024e-01, 8.70621427e-02,
 7.16657774e-02, 9.55313600e-02],
[3.17689196e-03, 4.27808071e-02, 2.24881715e-02, 6.45892890e-02,
 5.26561477e-02, 5.96447468e-03, 6.62540136e-02, 5.49720295e-02,
 4.52505890e-02, 6.03195899e-02],
[7.99032763e-03, 1.07599714e-01, 5.65608968e-02, 1.62451096e-01,
 1.32437577e-01, 1.50014881e-02, 1.66638111e-01, 1.38262343e-01,
 1.13811560e-01, 1.51712206e-01],
[7.16638380e-04, 9.65042839e-03, 5.07284699e-03, 1.45699520e-02,
 1.18780925e-02, 1.34545698e-03, 1.49454780e-02, 1.24005055e-02,
 1.02075580e-02, 1.36067999e-02]])
```

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

(**hint**: np.cumsum, np.interp)

```
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. (★★★)

(**hint**: np.logical_and.reduce, np.mod)

```
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). (★★★)

(**hint**: np.percentile)

```
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)
```

```
[-0.19779758  0.21600864]
```