这100道练习，带你玩转Numpy

Datawhale 前天

Datawhale

Datawhale，和鲸社区编辑

Numpy是Python做数据分析所必须要掌握的基础库之一。以下为入门Numpy的100题小练习，原为github上的开源项目，由和鲸社区的小科翻译并整理（保留了部分原文作为参考）。受限于篇幅，小编在这里只提供了部分题目的运行结果。友情提示：代码虽好，自己动手才算学到。

1. 导入numpy库并简写为 np (★☆☆)

(提示: import ... as ...)

```
import numpy as np
```

2. 打印numpy的版本和配置说明 (★☆☆)

(提示: np.version, np.show_config)

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

3. 创建一个长度为10的空向量 (★☆☆)

(提示: np.zeros)

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

4. 如何找到任何一个数组的内存大小? (★☆☆)

(提示: size, itemsize)

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

5. 如何从命令行得到numpy中add函数的说明文档? (★☆☆)

(提示: np.info)

```
numpy.info(numpy.add)
```

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

6. 创建一个长度为10并且除了第五个值为1的空向量 (★☆☆)

(提示: array[4])

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

7. 创建一个值域范围从10到49的向量(★☆☆)

(提示: np.arange)

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

8. 反转一个向量(第一个元素变为最后一个) (★☆☆)

(提示: array[::-1])

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

9. 创建一个 3x3 并且值从0到8的矩阵(★☆☆)

(提示: reshape)

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

10. 找到数组[1,2,0,0,4,0]中非0元素的位置索引 (★☆☆)

(提示: np.nonzero)

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

11. 创建一个 3x3 的单位矩阵 (★☆☆)

(提示: np.eye)

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

12. 创建一个 3x3x3的随机数组 (★☆☆)

(提示: np.random.random)

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

13. 创建一个 10x10 的随机数组并找到它的最大值和最小值 (★☆☆)

(提示: min, max)

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

14. 创建一个长度为30的随机向量并找到它的平均值 (★☆☆)

(提示: mean)

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

15. 创建一个二维数组，其中边界值为1，其余值为0 (★☆☆)

(提示: array[1:-1, 1:-1])

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

16. 对于一个存在在数组，如何添加一个用0填充的边界? (★☆☆)

(提示: np.pad)

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

17. 以下表达式运行的结果分别是什么? (★☆☆)

(提示: 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(0.3 == 3 * 0.1)
```

18. 创建一个 5x5的矩阵，并设置值1,2,3,4落在其对角线下方位置 (★☆☆)

(提示: np.diag)

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

19. 创建一个8x8 的矩阵，并且设置成棋盘样式 (★☆☆)

(提示: array[::2])

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

20. 考虑一个 (6,7,8) 形状的数组，其第100个元素的索引(x,y,z)是什么?

(提示: np.unravel_index)

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

21. 用tile函数去创建一个 8x8的棋盘样式矩阵(★☆☆)

(提示: np.tile)

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

22. 对一个5x5的随机矩阵做归一化(★☆☆)

(提示: (x - min) / (max - min))

```
Z = np.random.random((5,5))
Zmax, Zmin = Z.max(), Z.min()
Z = (Z - Zmin)/(Zmax - Zmin)
print(Z)
```

23. 创建一个将颜色描述为(RGBA)四个无符号字节的自定义dtype？(★☆☆)

(提示: np.dtype)

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

24. 一个5x3的矩阵与一个3x2的矩阵相乘，实矩阵乘积是什么？(★☆☆)

(提示: np.dot | @)

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

25. 给定一个一维数组，对其在3到8之间的所有元素取反 (★☆☆)

(提示: >, <=)

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

26. 下面脚本运行后的结果是什么? (★☆☆)

(提示: np.sum)

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))
```

27. 考虑一个整数向量Z,下列表达合法的是哪个? (★☆☆)

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

```
Z = np.arange(5)
Z ** Z  # legal
```

array([1, 1, 4, 27, 256])

```
Z = np.arange(5)
2 << Z >> 2  # false
```

array([0, 1, 2, 4, 8])

```
Z = np.arange(5)
Z <- Z   # legal
```

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

```
Z = np.arange(5)
1j*Z   # legal
```

array([0.+0.j, 0.+1.j, 0.+2.j, 0.+3.j, 0.+4.j])

```
Z = np.arange(5)
Z/1/1   # legal
```

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

```
Z = np.arange(5)
```

```
Z<Z>Z    # false
```

ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()

28. 下列表达式的结果分别是什么?(★☆☆)

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))
```

29. 如何从零位对浮点数组做舍入 ? (★☆☆)

(提示: np.uniform, np.copysign, np.ceil, np.abs)

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

30. 如何找到两个数组中的共同元素? (★☆☆)

(提示: np.intersect1d)

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

31. 如何忽略所有的 numpy 警告(尽管不建议这么做)? (★☆☆)

(提示: np.seterr, np.errstate)

```
# Suicide mode on
defaults = np.seterr(all="ignore")
Z = np.ones(1) / 0

# Back to sanity
_ = np.seterr(**defaults)
```

An equivalent way, with a context manager:

```
with np.errstate(divide='ignore'):
    Z = np.ones(1) / 0
```

32. 下面的表达式是正确的吗? (★☆☆)

(提示: imaginary number)

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

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

False

33. 如何得到昨天，今天，明天的日期? (★☆☆)

(提示: np.datetime64, np.timedelta64)

```
yesterday = np.datetime64('today', 'D') - np.timedelta64(1, 'D')
today     = np.datetime64('today', 'D')
tomorrow  = np.datetime64('today', 'D') + np.timedelta64(1, 'D')
print ("Yesterday is " + str(yesterday))
print ("Today is " + str(today))
print ("Tomorrow is "+ str(tomorrow))
```

34. 如何得到所有与2016年7月对应的日期？(★★☆)

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

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

35. 如何直接在位计算(A+B)*(-A/2)(不建立副本)? (★★☆)

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

```
A = np.ones(3)*1
B = np.ones(3)*2
C = np.ones(3)*3
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. 用五种不同的方法去提取一个随机数组的整数部分(★★☆)

(提示: %, np.floor, np.ceil, astype, np.trunc)

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

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

37. 创建一个5x5的矩阵，其中每行的数值范围从0到4 (★★☆)

(提示: np.arange)

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

38. 通过考虑一个可生成10个整数的函数，来构建一个数组(★☆☆)

(提示: np.fromiter)

```
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.]

39. 创建一个长度为10的随机向量，其值域范围从0到1，但是不包括0和1 (★★☆)

(提示: np.linspace)

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

40. 创建一个长度为10的随机向量，并将其排序 (★★☆)

(提示: sort)

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

41.对于一个小数组，如何用比 np.sum更快的方式对其求和？(★★☆)

(提示: np.add.reduce)

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

42. 对于两个随机数组A和B，检查它们是否相等(★★☆)

(提示: np.allclose, np.array_equal)

```
A = np.random.randint(0,2,5)
B = np.random.randint(0,2,5)
# Assuming identical shape of the arrays and a tolerance for the comparison of value
equal = np.allclose(A,B)
print(equal)
```

False

```
# 方法2
# Checking both the shape and the element values, no tolerance (values have to be ex
equal = np.array_equal(A,B)
print(equal)
```

False

43. 创建一个只读数组(read-only) (★★☆)

(提示: flags.writeable)

```
# 使用如下过程实现
Z = np.zeros(10)
Z.flags.writeable = False
Z[0] = 1
```

44. 将笛卡尔坐标下的一个10x2的矩阵转换为极坐标形式(★★☆)

(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)
```

45. 创建一个长度为10的向量，并将向量中最大值替换为1 (★★☆)

(提示: argmax)

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

46. 创建一个结构化数组，并实现 x 和 y 坐标覆盖 [0,1]x[0,1] 区域 (★★☆)

(提示: 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)
```

47. 给定两个数组X和Y，构造Cauchy矩阵C (Cij =1/(xi - yj))

(提示: np.subtract.outer)

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

48. 打印每个numpy标量类型的最小值和最大值? (★★☆)

(提示: 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).max)
    print(np.finfo(dtype).eps)
```

49. 如何打印一个数组中的所有数值? (★★☆)

(提示: np.set_printoptions)

```
np.set_printoptions(threshold=np.nan)
Z = np.zeros((16,16))
print (Z)
```

50. 给定标量时，如何找到数组中最接近标量的值？(★★☆)

(提示: argmin)

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

51. 创建一个表示位置(x,y)和颜色(r,g,b)的结构化数组(★★☆)

(提示: dtype)

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

52. 对一个表示坐标形状为(100,2)的随机向量，找到点与点的距离(★★☆)

(提示: 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)

# 方法2
# Much faster with scipy
import scipy
# Thanks Gavin Heverly-Coulson (#issue 1)
```

```
import scipy.spatial
D = scipy.spatial.distance.cdist(Z,Z)
print (D)
```

53. 如何将32位的浮点数(float)转换为对应的整数(integer)?

(提示: astype(copy=False))

```
Z = np.arange(10, dtype=np.int32)
Z = Z.astype(np.float32, copy=False)
print (Z)
```

54. 如何读取以下文件? (★★☆)

(提示: np.genfromtxt)

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

参考链接：https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.genfromtxt.html

55. 对于numpy数组，enumerate的等价操作是什么？(★★☆)

(提示: 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])
```

56. 生成一个通用的二维Gaussian-like数组 (★★☆)

(提示: 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)
```

57. 对一个二维数组，如何在其内部随机放置p个元素? (★★☆)

(提示: np.put, np.random.choice)

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

58. 减去一个矩阵中的每一行的平均值 (★★☆)

(提示: mean(axis=,keepdims=))

```
X = np.random.rand(5, 10)
# Recent versions of numpy
Y = X - X.mean(axis=1, keepdims=True)
print(Y)

# 方法2
# Older versions of numpy
Y = X - X.mean(axis=1).reshape(-1, 1)
print (Y)
```

59. 如何通过第n列对一个数组进行排序? (★★☆)

(提示: argsort)

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

60. 如何检查一个二维数组是否有空列? (★★☆)

(提示: any, ~)

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

True

61. 从数组中的给定值中找出最近的值 (★★☆)

(提示: np.abs, argmin, flat)

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

0.5531249196891759

62. 如何用迭代器(iterator)计算两个分别具有形状(1,3)和(3,1)的数组? (★★☆)

(提示: 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])
```

63. 创建一个具有name属性的数组类(★★☆)

(提示: class方法)

```
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. 考虑一个给定的向量，如何对由第二个向量索引的每个元素加1(小心重复的索引)? (★★★)

(提示: 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)
```

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

```
# 方法2
np.add.at(Z, I, 1)
print(Z)
```

[5. 1. 9. 7. 5. 7. 3. 1. 7. 5.]

65. 根据索引列表(I)，如何将向量(X)的元素累加到数组(F)？(★★★)

(提示: 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. 考虑一个(dtype=ubyte) 的 (w,h,3)图像，计算其唯一颜色的数量(★★★)

(提示: np.unique)

```
w,h = 16,16
I = np.random.randint(0,2,(h,w,3)).astype(np.ubyte)
#Note that we should compute 256*256 first.
#Otherwise numpy will only promote F.dtype to 'uint16' and overfolw will occur
F = I[...,0]*(256*256) + I[...,1]*256 +I[...,2]
n = len(np.unique(F))
print (n)
```

8

67. 考虑一个四维数组，如何一次性计算出最后两个轴(axis)的和？(★★★)

(提示: 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)

# 方法2
sum = A.reshape(A.shape[:-2] + (-1,)).sum(axis=-1)
print (sum)
```

68. 考虑一个一维向量D，如何使用相同大小的向量S来计算D子集的均值？(★★★)

(提示: 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)

# 方法2
import pandas as pd
print(pd.Series(D).groupby(S).mean())
```

69. 如何获得点积 dot prodcut的对角线? (★★★)

(提示: 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))

# 方法2
# Fast version
np.sum(A * B.T, axis=1)

# 方法3
# Faster version
np.einsum("ij,ji->i", A, B)
```

70. 考虑一个向量[1,2,3,4,5],如何建立一个新的向量，在这个新向量中每个值之间有3个连续的零？(★★★)

(提示: 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. 考虑一个维度(5,5,3)的数组，如何将其与一个(5,5)的数组相乘？(★★★)

(提示: array[:, :, None])

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

72. 如何对一个数组中任意两行做交换? (★★★)

(提示: array[[]] = array[[]])

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

73. 考虑一个可以描述10个三角形的triplets，找到可以分割全部三角形的line segment

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

74. 给定一个二进制的数组C，如何产生一个数组A满足np.bincount(A)==C(★★★)

(提示: 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. 如何通过滑动窗口计算一个数组的平均数? (★★★)

(提示: 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))
```

[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]) (★★★)

(提示: from numpy.lib import stride_tricks)

```
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)
```

77. 如何对布尔值取反，或者原位(in-place)改变浮点数的符号(sign)？(★★★)

(提示: 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)
```

78. 考虑两组点集P0和P1去描述一组线(二维)和一个点p,如何计算点p到每一条线 i (P0[i],P1[i])的距离？(★★★)

```
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))
```

79.考虑两组点集P0和P1去描述一组线(二维)和一组点集P，如何计算每一个点 j(P[j]) 到每一条线 i (P0[i],P1[i])的距离？(★★★)

```
# based on distance function from previous question
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]))
```

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)
```

81. 考虑一个数组Z = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]，如何生成一个数组R = [[1,2,3,4], [2,3,4,5], [3,4,5,6], ...,[11,12,13,14]]? (★★★)

(提示: stride_tricks.as_strided)

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

82. 计算一个矩阵的秩(★★★)

(提示: 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)
```

83. 如何找到一个数组中出现频率最高的值?

(提示: np.bincount, argmax)

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

1

84. 从一个10x10的矩阵中提取出连续的3x3区块(★★★)

(提示: 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)
```

85. 创建一个满足 Z[i,j] == Z[j,i]的子类 (★★★)

(提示: class 方法)

```
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)
```

86. 考虑p个 nxn 矩阵和一组形状为(n,1)的向量，如何直接计算p个矩阵的乘积(n,1)？(★★★)

(提示: 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)
```

87. 对于一个16x16的数组，如何得到一个区域(block-sum)的和(区域大小为4x4)? (★★★)

(提示: 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)
```

88. 如何利用numpy数组实现Game of Life? (★★★)

(提示: Game of Life)

```
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)
```

89. 如何找到一个数组的第n个最大值? (★★★)

(提示: np.argsort | np.argpartition)

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

```
n = 5

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

[9995 9996 9997 9998 9999]

```
# 方法2
# Fast
print (Z[np.argpartition(-Z,n)[:n]])
```

[9999 9997 9998 9996 9995]

90. 给定任意个数向量，创建笛卡尔积(每一个元素的每一种组合)(★★★)

(提示: 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])))
```

91. 如何从一个正常数组创建记录数组(record array)? (★★★)

(提示: 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. 考虑一个大向量Z, 用三种不同的方法计算它的立方(★★★)

(提示: np.power, *, np.einsum)

```
x = np.random.rand()
np.power(x,3)

# 方法2
x*x*x

# 方法3
np.einsum('i,i,i->i',x,x,x)
```

93. 考虑两个形状分别为(8,3) 和(2,2)的数组A和B. 如何在数组A中找到满足包含B中元素的行？(不考虑B中每行元素顺序)？(★★★)

(提示: 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)
```

[0 1 4 5 6 7]

94. 考虑一个10x3的矩阵，分解出有不全相同值的行 (如 [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)
```

```
# 方法2
# 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)
```

95. 将一个整数向量转换为matrix binary的表现形式 (★★★)

(提示: 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])

# 方法2
print (np.unpackbits(I[:, np.newaxis], axis=1))
```

96. 给定一个二维数组，如何提取出唯一的(unique)行?(★★★)

(提示: 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)
```

97. 考虑两个向量A和B，写出用einsum等式对应的inner, outer, sum, mul函数(★★★)

(提示: np.einsum)

```
A = np.random.uniform(0,1,10)
B = np.random.uniform(0,1,10)
print ('sum')
print (np.einsum('i->', A))# np.sum(A)

print ('A * B')
print (np.einsum('i,i->i', A, B)) # A * B

print ('inner')
print (np.einsum('i,i', A, B))    # np.inner(A, B)

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

98. 考虑一个由两个向量描述的路径(X,Y)，如何用等距样例(equidistant samples)对其进行采样(sample)? (★★★)

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

(提示: 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. 对于一个一维数组X，计算它boostrapped之后的95%置信区间的平均值。

(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). (★★★)
(提示: 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)
```

原文地址：https://www.kesci.com/home/project/59f29f67c5f3f5119527a2cc