In [1]:

```
import numpy as np
```

In [4]:

```
bin(12)
```

Out[4]:

```
'0b1100'
```

In [7]:

```
arr = np.arange(1, 17).reshape(4,4)
arr
```

Out[7]:

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

In [9]:

```
vec_bin = np.vectorize(bin)
```

In [10]:

```
vec_bin(arr)
```

Out[10]:

```
array([['0b1', '0b10', '0b11', '0b100'],
       ['0b101', '0b110', '0b111', '0b1000'],
       ['0b1001', '0b1010', '0b1011', '0b1100'],
       ['0b1101', '0b1110', '0b1111', '0b10000']], dtype='<U7')
```

In [ ]:

```

```

In [15]:

```
arr = np.array([int(i) for i in "1 1 2 5 7 6 7 7 6 3".split()])
arr
```

Out[15]:

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

In [18]:

```
uniq, freq = np.unique(arr, return_counts= True)
```

In [19]:

```
uniq
```

Out[19]:

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

In [20]:

```
freq
```

Out[20]:

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

In [23]:

```
zeros = np.zeros(10, dtype='int')
zeros
```

Out[23]:

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

In [26]:

```
zeros[uniq-1] = freq
```

In [27]:

```
zeros
```

Out[27]:

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

In [28]:

```
def convert(arr):
    uniq, freq = np.unique(arr, return_counts= True)
    zeros = np.zeros(10, dtype='int')
    zeros[uniq-1] = freq

    return zeros
```

In [30]:

```
convert = np.vectorize(convert)
```

In [32]:

```
# [convert(row) for row in matrix]
```

In [ ]:

```

```

New question

In [59]:

```
arr = np.arange(16).reshape(4,4)
arr[0,1] = 2
arr
```

Out[59]:

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

In [60]:

```
arr%2==0
```

Out[60]:

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

In [61]:

```
res = arr[arr%2==0]
res
```

Out[61]:

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

In [62]:

```
size = np.sqrt(arr[arr%2==0].size)
size
```

Out[62]:

```
2.8284271247461903
```

In [63]:

```
int(size)
```

Out[63]:

```
2
```

In [64]:

```
size == int(size)
```

Out[64]:

```
False
```

In [57]:

```
res
```

Out[57]:

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

In [58]:

```
res.reshape((int(size),int(size)))
```

Out[58]:

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

In [65]:

```
res.reshape((2,2))
```

```
-----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
 last)
<ipython-input-65-93f37efed883> in <module>
----> 1 res.reshape((2,2))

ValueError: cannot reshape array of size 8 into shape (2,2)
```

In [67]:

```
X = np.arange(12).reshape((3, 4))
X
```

Out[67]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [68]:

```
row = np.array([0, 1, 2])
row
```

Out[68]:

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

In [69]:

```
mask = np.array([1, 0, 1, 0], dtype=bool)
mask
```

Out[69]:

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

In [74]:

```
X
```

Out[74]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [73]:

```
X[ [0, 1, 2] , 1 ]
```

Out[73]:

```
array([1, 5, 9])
```

In [80]:

```
X[ row[:, np.newaxis] , mask ]
```

Out[80]:

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

In [76]:

```
row.shape
```

Out[76]:

```
(3,)
```

In [81]:

```
row[:, np.newaxis].shape
```

Out[81]:

```
(3, 1)
```

Original matrix retrivel

In [88]:

```
a = np.linspace(0, 2, num=4).reshape(2,2)
a
```

Out[88]:

```
array([[0.        , 0.66666667],
       [1.33333333, 2.        ]])
```

In [89]:

```
b = np.linspace(5, 7, num=4).reshape(2,2)
b
```

Out[89]:

```
array([[5.        , 5.66666667],
       [6.33333333, 7.        ]])
```

In [90]:

```
c = np.linspace(1, 3, num=4).reshape(2,2)
c
```

Out[90]:

```
array([[1.        , 1.66666667],
       [2.33333333, 3.        ]])
```

In [91]:

```
d = np.linspace(6, 9, num=4).reshape(2,2)
d
```

Out[91]:

```
array([[6., 7.],
       [8., 9.]])
```

In [94]:

```
e = np.hstack((a, b))
e
```

Out[94]:

```
array([[0.        , 0.66666667, 5.        , 5.66666667],
       [1.33333333, 2.        , 6.33333333, 7.        ]])
```

In [95]:

```
f = np.hstack((c,d))
f
```

Out[95]:

```
array([[1.        , 1.66666667, 6.        , 7.        ],
       [2.33333333, 3.        , 8.        , 9.        ]])
```

In [98]:

```
np.vstack((e, f)).round(1)
```

Out[98]:

```
array([[0. , 0.7, 5. , 5.7],
       [1.3, 2. , 6.3, 7. ],
       [1. , 1.7, 6. , 7. ],
       [2.3, 3. , 8. , 9. ]])
```

In [99]:

```
np.concatenate((a, b), axis=1)
```

Out[99]:

```
array([[0.        , 0.66666667, 5.        , 5.66666667],
       [1.33333333, 2.        , 6.33333333, 7.        ]])
```

In [ ]:

```

```

In [101]:

```
# np.bmat?
```

Entropy

In [124]:

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

In [125]:

```
size = A.size
size
```

Out[125]:

```
12
```

In [126]:

```
eles, freq = np.unique(A, return_counts = True)
```

In [127]:

```
eles
```

Out[127]:

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

In [128]:

```
freq
```

Out[128]:

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

In [129]:

```
prob = freq/size
prob
```

Out[129]:

```
array([0.25      , 0.33333333, 0.41666667])
```

In [130]:

```
log_prob = np.log2(prob)
log_prob
```

Out[130]:

```
array([-2.        , -1.5849625 , -1.26303441])
```

In [131]:

```
-1*np.sum(prob*log_prob)
```

Out[131]:

```
1.5545851693377997
```

New ques

In [136]:

```
def func(x):
    return 1/(1 + np.exp(-x))
```

In [137]:

```
func(0)
```

Out[137]:

```
0.5
```

In [138]:

```
func(20)
```

Out[138]:

```
0.9999999979388463
```

In [134]:

```
np.e**(-1)
```

Out[134]:

```
0.36787944117144233
```

In [135]:

```
np.exp(-1)
```

Out[135]:

```
0.36787944117144233
```

In [ ]:

```

```

In [141]:

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

Out[141]:

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

In [142]:

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

Out[142]:

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

In [144]:

```
a / b
```

Out[144]:

```
array([0.5       , 0.66666667, 3.        ])
```

In [145]:

```
np.log(a)
```

Out[145]:

```
array([0.        , 0.69314718, 1.09861229])
```

In [ ]:

```

```

In [147]:

```
A  = np.random.randint(0, 2, size=(3,32))
A.shape
```

Out[147]:

```
(3, 32)
```

In [149]:

```
B= np.array([1,2,3]).reshape(3,1)
B.shape
```

Out[149]:

```
(3, 1)
```

In [151]:

```
(A+B).shape
```

Out[151]:

```
(3, 32)
```

In [152]:

```
a = np.zeros(27).reshape(3, 3, 3)
b = np.arange(9).reshape(3, 3)
```

In [153]:

```
a
```

Out[153]:

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

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

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

In [154]:

```
b
```

Out[154]:

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

In [157]:

```
a+b
```

Out[157]:

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

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

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

In [158]:

```
b.shape
```

Out[158]:

```
(3, 3)
```

In [159]:

```
b.reshape(3,3,1)
```

Out[159]:

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

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

       [[6],
        [7],
        [8]]])
```

In [162]:

```
b.shape
```

Out[162]:

```
(3, 3)
```

In [161]:

```
b[:, :, np.newaxis].shape
```

Out[161]:

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

In [169]:

```
np.random.random(size=4)
```

Out[169]:

```
array([0.05524224, 0.28051832, 0.35141423, 0.96800566])
```

In [170]:

```
# np.random.randint()
```

In [172]:

```
a = np.arange(6).reshape(3, 2)
a
```

Out[172]:

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

In [173]:

```
np.hstack((a,a))
```

Out[173]:

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

In [176]:

```
a
```

Out[176]:

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

In [178]:

```
np.stack([a,a])
```

Out[178]:

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

In [184]:

```
np.stack([a,a], axis=0)
```

Out[184]:

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

       [[0, 1],
        [2, 3],
        [4, 5]]])
```

In [185]:

```
arr = 2*np.arange(0,2,0.5)
```

In [190]:

```
arr <=0.6
```

Out[190]:

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

In [188]:

```
np.any(arr <= 0.6)
```

Out[188]:

```
True
```

In [189]:

```
np.all(arr <= 0.6)
```

Out[189]:

```
False
```

In [ ]:

```

```