In [1]:
```
a = 'abcdefg'
b = 'fgklmno'

import re
```

In [4]:
```
re.findall(r'12(34)|(56)', '1234')
```

Out[4]:
```
[('34', '')]
```

In [5]:
```
re.findall(r'12(34)|(56)', '56')
```

Out[5]:
```
[('', '56')]
```

In [7]:
```
re.findall(r'1234|56', '565656512346')
```

Out[7]:
```
['56', '56', '56', '1234']
```

In [11]:
```
re.findall(r'12(?:34|56)', '1234aslkjdfha1256lksd') # non-capturing parenth
```

Out[11]:
```
['1234', '1256']
```

In [12]:
```
re.findall(r'12(34|56)', '1234aslkjdfha1256lksd')
```

Out[12]:
```
['34', '56']
```

In [13]:
```
m = re.search(r'12(?:34|56)', '1234aslkjdfha1256lksd')
```

In [15]:
```
m.group(1)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-15-6a9fdd5b6bf1> in <module>
----> 1 m.group(1)

IndexError: no such group
```

In [18]:
```
m = re.search(r'12(34|56)', '1234aslkjdfha1256lksd')
```

In [19]:
```
m.group(1)
```

Out[19]:
```
'34'
```

In [22]:
```
re.findall(r'\w\w\d\d\d\d', 'AK1234  KK9876')
```

Out[22]:
```
['AK1234', 'KK9876']
```

In [23]:
```
re.findall(r'\w\w(\d\d\d\d)', 'AK1234  KK9876')
```

Out[23]:
```
['1234', '9876']
```

In [24]:
```
import numpy as np
```

In [25]:
```
a = np.array([1,2,3,2,5,4,3,2,3])
```

In [26]:
```
type(a)
```

Out[26]:
```
numpy.ndarray
```

In [27]:
```
b = [1,2,3,2,5,4,3,2,3]
```

In [28]:
```
[x for x in b if x>2]
```

Out[28]:
```
[3, 5, 4, 3, 3]
```

In [29]:
```
a
```

Out[29]:
```
array([1, 2, 3, 2, 5, 4, 3, 2, 3])
```

In [30]:
```
[x>2 for x in a]
```

Out[30]:
```
[False, False, True, False, True, True, True, False, True]
```

In [31]:
```
a>2
```

Out[31]:
```
array([False, False,  True, False,  True,  True,  True, False,  True])
```

In [35]:
```
a[1:4]
```

Out[35]:
```
array([2, 3, 2])
```

In [36]:
```
a[a>2]
```

Out[36]:
```
array([3, 5, 4, 3, 3])
```

In [37]:
```
a = np.random.random((10,4))
```

In [38]:
```
a
```

Out[38]:
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [39]:
```
a.shape
```

Out[39]:
```
(10, 4)
```

In [40]:
```
a.ndim
```

Out[40]:
```
2
```

In [41]:
```
a.size
```

Out[41]:
```
40
```

In [42]:
```
a[ 0:3 , 0:3]
```

Out[42]:
```
array([[0.91034426, 0.43457504, 0.51284057],
       [0.65390162, 0.30811191, 0.32252729],
       [0.18198257, 0.06664977, 0.08885812]])
```

In [43]:
```
a[ 1:8:2 , -1::-2]
```

Out[43]:
```
array([[0.56937793, 0.30811191],
       [0.07395098, 0.00725407],
       [0.27879045, 0.84526954],
       [0.88974561, 0.5370648 ]])
```

In [44]:
```
a
```

Out[44]:
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [49]:
```
a[ [1,4,5]  , ]
```

Out[49]:
```
array([[0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045]])
```

In [50]:
```
a[ [1,1,1,4,5]  , ]
```

Out[50]:
```
array([[0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045]])
```

In [53]:
```
b = a[ [1,1,1,4,5]  , ]
```

In [57]:
```
b[ :, [0,3]]
```

Out[57]:
```
array([[0.65390162, 0.56937793],
       [0.65390162, 0.56937793],
       [0.65390162, 0.56937793],
       [0.75784909, 0.92841833],
       [0.59983431, 0.27879045]])
```

In [59]:
```
a[ [1,1,1,4,5]  , ][:, [0,3]]
```

Out[59]:
```
array([[0.65390162, 0.56937793],
       [0.65390162, 0.56937793],
       [0.65390162, 0.56937793],
       [0.75784909, 0.92841833],
       [0.59983431, 0.27879045]])
```

In [61]:
```
a[ [1,1,1,4,5]  , ]
```

Out[61]:
```
array([[0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045]])
```

In [65]:
```
a[ [1,1,1,4,5] ,   ]
```

Out[65]:
```
array([[0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045]])
```

In [66]:
```
a[ [1,2,1,2,1] ,  ]
```

Out[66]:
```
array([[0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
```

```
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
```

In [67]:
```
a[ [True, True,True,True,True,True,True,True,True,True,]  , ]
```

Out[67]:
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [68]:
```
a[ [True, True,True,True,True,True,False,True,True,True,]  , ]
```

Out[68]:
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [69]:
```
c = np.array([1,2,3,2,5,4,3,2,3])
```

In [70]:
```
c>2
```

Out[70]:
```
array([False, False,  True, False,  True,  True,  True, False,  True])
```

In [71]:
```
c[ [False, False,  True, False,  True,  True,  True, False,  True]  ]
```

Out[71]:
```
array([3, 5, 4, 3, 3])
```

In [72]:
```
c[ c>2 ]
```

Out[72]:
```
array([3, 5, 4, 3, 3])
```

In [74]:
```
a[ : , : ]
```

Out[74]:
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [76]:
```
a[ 2:4 , 1:4 ]
```

Out[76]:
```
array([[0.06664977, 0.08885812, 0.22529457],
       [0.00725407, 0.90873155, 0.07395098]])
```

In [77]:
```
b = np.random.random((10,4))
```

In [78]:
```
a
```

Out[78]:
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
```

```
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
```

In [79]:
```
b
```

Out[79]:
```
array([[0.33216846, 0.02345675, 0.73491138, 0.12538281],
       [0.2708553 , 0.35841031, 0.50413981, 0.72588268],
       [0.41192295, 0.13768707, 0.67711436, 0.51872878],
       [0.43153449, 0.96566799, 0.63437696, 0.68992349],
       [0.77446576, 0.20970367, 0.06099859, 0.40308158],
       [0.4648441 , 0.26774465, 0.60392709, 0.49758574],
       [0.22481413, 0.56128597, 0.1667581 , 0.14674038],
       [0.86026691, 0.34353633, 0.26591266, 0.68680905],
       [0.88411492, 0.63039135, 0.39313626, 0.37378974],
       [0.01889116, 0.89421852, 0.815371  , 0.43527436]])
```

In [80]:
```
a+b
```

Out[80]:
```
array([[1.24251273, 0.4580318 , 1.24775195, 1.00807057],
       [0.92475692, 0.66652222, 0.8266671 , 1.29526061],
       [0.59390553, 0.20433683, 0.76597247, 0.74402335],
       [0.48732114, 0.97292206, 1.54310851, 0.76387447],
       [1.53231484, 0.47969761, 0.73880032, 1.33149991],
       [1.0646784 , 1.1130142 , 1.05067619, 0.77637619],
       [1.02997855, 0.86586094, 0.37419369, 0.55796238],
       [0.9897242 , 0.88060113, 0.42238116, 1.57655466],
       [1.34810384, 0.94780671, 1.23561651, 1.04922557],
       [1.0140874 , 1.7457198 , 1.34851134, 1.14394904]])
```

In [81]:
```
a-b
```

Out[81]:
```
array([[ 0.5781758 ,  0.41111829, -0.22207081,  0.75730495],
       [ 0.38304632, -0.05029839, -0.18161252, -0.15650475],
       [-0.22994038, -0.0710373 , -0.58825624, -0.29343421],
       [-0.37574783, -0.95841392,  0.27435459, -0.61597251],
       [-0.01661667,  0.06029028,  0.61680314,  0.52533675],
       [ 0.13499021,  0.57752489, -0.15717799, -0.21879529],
       [ 0.5803503 , -0.25671099,  0.04067748,  0.26448162],
       [-0.73080963,  0.19352847, -0.10944415,  0.20293657],
       [-0.420126  , -0.312976  ,  0.44934398,  0.30164609],
       [ 0.97630509, -0.04271725, -0.28223067,  0.27340033]])
```

In [82]:
```
a*b
```

Out[82]:
```
array([[0.30238765, 0.01019372, 0.37689237, 0.11067387],
       [0.17711272, 0.11043049, 0.16259885, 0.41330158],
       [0.0749628 , 0.00917681, 0.06016711, 0.11686678],
       [0.02407386, 0.00700502, 0.57647836, 0.05102052],
       [0.58692817, 0.05661872, 0.04134495, 0.37422833],
       [0.27882944, 0.2263164 , 0.26980389, 0.13872215],
       [0.18101234, 0.17095366, 0.03459156, 0.06034287],
       [0.11136782, 0.18450127, 0.04160696, 0.61108534],
       [0.41021952, 0.20009589, 0.33120953, 0.25247098],
       [0.01880041, 0.76142821, 0.43470717, 0.30846792]])
```

In [83]:
```
np.ones((3,6))
```

Out[83]:
```
array([[1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.]])
```

In [84]:
```
6 * np.ones((3,6))
```

Out[84]:
```
array([[6., 6., 6., 6., 6., 6.],
       [6., 6., 6., 6., 6., 6.],
       [6., 6., 6., 6., 6., 6.]])
```

In [85]:
```
np.zeros((3,6))
```

Out[85]:
```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

In [86]:
```
np.zeros((3,6)) + 6
```

Out[86]:
```
array([[6., 6., 6., 6., 6., 6.],
       [6., 6., 6., 6., 6., 6.],
       [6., 6., 6., 6., 6., 6.]])
```

In [91]:
```
np.zeros((3,6))
```

Out[91]:
```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

In [92]:
```
np.zeros((3,6), dtype=np.int)
```

Out[92]:
```
array([[0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0]])
```

In [ ]:
```
np.arange(1, 10, 3)
```

In [94]:
```
5-3/1
```

Out[94]:
```
2.0
```

In [95]:
```
0.1 + 0.1 + 0.1 == 0.3
```

Out[95]:
```
False
```

In [97]:
```
0.1 + 0.1 + 0.1 - 0.3
```

Out[97]:
```
5.551115123125783e-17
```

In [98]:
```
15
```

Out[98]:
```
15
```

In [100…
```
1*10**1 + 5*10**0
```

Out[100…
```
15
```

In [102…
```
1*10**(-1)
```

Out[102…
```
0.1
```

In [ ]:
```
1*2**(-1) + 1*2**(-2) + 1*2**(-3) + 1*2**(-4)
```

In [103…
```
1/3
```

Out[103…
```
0.3333333333333333
```

In [104…
```
np.arange(1,10,0.3)
```

Out[104…
```
array([1. , 1.3, 1.6, 1.9, 2.2, 2.5, 2.8, 3.1, 3.4, 3.7, 4. , 4.3, 4.6,
       4.9, 5.2, 5.5, 5.8, 6.1, 6.4, 6.7, 7. , 7.3, 7.6, 7.9, 8.2, 8.5,
       8.8, 9.1, 9.4, 9.7])
```

In [105…
```
np.linspace(10,20,30)
```

Out[105…
```
array([10.        , 10.34482759, 10.68965517, 11.03448276, 11.37931034,
       11.72413793, 12.06896552, 12.4137931 , 12.75862069, 13.10344828,
       13.44827586, 13.79310345, 14.13793103, 14.48275862, 14.82758621,
       15.17241379, 15.51724138, 15.86206897, 16.20689655, 16.55172414,
       16.89655172, 17.24137931, 17.5862069 , 17.93103448, 18.27586207,
       18.62068966, 18.96551724, 19.31034483, 19.65517241, 20.        ])
```

In [106…
```
a
```

Out[106…
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [107…
```
np.sin(a)
```

Out[107…
```
array([[0.78971498, 0.42102498, 0.49065436, 0.7724486 ],
       [0.60828781, 0.30326001, 0.31696455, 0.53910822],
       [0.18097976, 0.06660043, 0.08874123, 0.2233935 ],
       [0.05575772, 0.00725401, 0.7887246 , 0.07388359],
       [0.68736079, 0.2667256 , 0.62708219, 0.80067336],
       [0.56450571, 0.74814999, 0.43203598, 0.275193  ],
       [0.7209446 , 0.29988774, 0.20595114, 0.39972975],
       [0.12909599, 0.51161623, 0.15583083, 0.77691161],
       [0.44751885, 0.31211207, 0.7462963 , 0.62523752],
       [0.83886581, 0.75227037, 0.50824035, 0.65082813]])
```

In [108…
```
np.log(a)
```

Out[108…
```
array([[-9.39324387e-02, -8.33386638e-01, -6.67790265e-01,
        -1.24783755e-01],
       [-4.24798369e-01, -1.17729220e+00, -1.13156753e+00,
        -5.63210864e-01],
       [-1.70384436e+00, -2.70830375e+00, -2.42071436e+00,
        -1.49034652e+00],
       [-2.88622063e+00, -4.92619234e+00, -9.57055529e-02,
        -2.60435285e+00],
       [-2.77271006e-01, -1.30935574e+00, -3.88900466e-01,
        -7.42728648e-02],
       [-5.11101815e-01, -1.68099719e-01, -8.05758140e-01,
        -1.27729485e+00],
       [-2.16708762e-01, -1.18883799e+00, -1.57293442e+00,
        -8.88622060e-01],
       [-2.04440429e+00, -6.21636521e-01, -1.85490054e+00,
        -1.16819684e-01],
       [-7.67894610e-01, -1.14754410e+00, -1.71405068e-01,
        -3.92397118e-01],
       [-4.81533098e-03, -1.60754285e-01, -6.28970593e-01,
        -3.44358695e-01]])
```

In [109…
```
np.log10(a)
```

Out[109…
```
array([[-4.07943398e-02, -3.61935218e-01, -2.90017627e-01,
        -5.41928961e-02],
       [-1.84487588e-01, -5.11291508e-01, -4.91433533e-01,
        -2.44599370e-01],
       [-7.39970202e-01, -1.17620137e+00, -1.05130289e+00,
        -6.47249270e-01],
       [-1.25346969e+00, -2.13941815e+00, -4.15643935e-02,
        -1.13105607e+00],
       [-1.20417268e-01, -5.68645971e-01, -1.68897326e-01,
        -3.22562953e-02],
       [-2.21968698e-01, -7.30047806e-02, -3.49936314e-01,
        -5.54722105e-01],
       [-9.41154196e-02, -5.16305778e-01, -6.83116740e-01,
        -3.85923657e-01],
       [-8.87873502e-01, -2.69973311e-01, -8.05573068e-01,
        -5.07341441e-02],
```

```
       [-3.33492392e-01, -4.98372071e-01, -7.44402750e-02,
        -1.70415903e-01],
       [-2.09127168e-03, -6.98146990e-02, -2.73158458e-01,
        -1.49553081e-01]])
```

In [110…
```
np.pi
```

Out[110…
```
3.141592653589793
```

In [111…
```
np.e
```

Out[111…
```
2.718281828459045
```

In [112…
```
a
```

Out[112…
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [113…
```
np.max(a)
```

Out[113…
```
0.9951962441346142
```

In [114…
```
np.max(a, axis=0)
```

Out[114…
```
array([0.99519624, 0.85150127, 0.90873155, 0.92841833])
```

In [115…
```
np.max(a, axis=1)
```

Out[115…
```
array([0.91034426, 0.65390162, 0.22529457, 0.90873155, 0.92841833,
       0.84526954, 0.80516443, 0.88974561, 0.84248024, 0.99519624])
```

In [116…
```
a
```

Out[116…
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [125…
```
a.ravel()
```

Out[125…
```
array([0.91034426, 0.43457504, 0.51284057, 0.88268776, 0.65390162,
       0.30811191, 0.32252729, 0.56937793, 0.18198257, 0.06664977,
       0.08885812, 0.22529457, 0.05578665, 0.00725407, 0.90873155,
       0.07395098, 0.75784909, 0.26999395, 0.67780173, 0.92841833,
       0.59983431, 0.84526954, 0.4467491 , 0.27879045, 0.80516443,
       0.30457498, 0.20743558, 0.411222  , 0.12945729, 0.5370648 ,
       0.1564685 , 0.88974561, 0.46398892, 0.31741535, 0.84248024,
       0.67543583, 0.99519624, 0.85150127, 0.53314034, 0.70867468])
```

In [126…
```
a
```

Out[126…
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
```

```
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [127…
```
a.reshape((20,2))
```

Out[127…
```
array([[0.91034426, 0.43457504],
       [0.51284057, 0.88268776],
       [0.65390162, 0.30811191],
       [0.32252729, 0.56937793],
       [0.18198257, 0.06664977],
       [0.08885812, 0.22529457],
       [0.05578665, 0.00725407],
       [0.90873155, 0.07395098],
       [0.75784909, 0.26999395],
       [0.67780173, 0.92841833],
       [0.59983431, 0.84526954],
       [0.4467491 , 0.27879045],
       [0.80516443, 0.30457498],
       [0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 ],
       [0.1564685 , 0.88974561],
       [0.46398892, 0.31741535],
       [0.84248024, 0.67543583],
       [0.99519624, 0.85150127],
       [0.53314034, 0.70867468]])
```

In [128…
```
a.reshape((20,-1))
```

Out[128…
```
array([[0.91034426, 0.43457504],
       [0.51284057, 0.88268776],
       [0.65390162, 0.30811191],
       [0.32252729, 0.56937793],
       [0.18198257, 0.06664977],
       [0.08885812, 0.22529457],
       [0.05578665, 0.00725407],
       [0.90873155, 0.07395098],
       [0.75784909, 0.26999395],
       [0.67780173, 0.92841833],
       [0.59983431, 0.84526954],
       [0.4467491 , 0.27879045],
       [0.80516443, 0.30457498],
       [0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 ],
       [0.1564685 , 0.88974561],
       [0.46398892, 0.31741535],
       [0.84248024, 0.67543583],
       [0.99519624, 0.85150127],
       [0.53314034, 0.70867468]])
```

In [129…
```
a.reshape((5,-1))
```

Out[129…
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776, 0.65390162,
        0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457, 0.05578665,
        0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833, 0.59983431,
        0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  , 0.12945729,
        0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583, 0.99519624,
        0.85150127, 0.53314034, 0.70867468]])
```

In [130…
```
a.reshape((5,10))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-130-e1fc005408fe> in <module>
----> 1 a.reshape((5,10))

ValueError: cannot reshape array of size 40 into shape (5,10)
```

In [131…
```
a.reshape((5,8))
```

Out[131…
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776, 0.65390162,
        0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457, 0.05578665,
        0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833, 0.59983431,
        0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  , 0.12945729,
        0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583, 0.99519624,
        0.85150127, 0.53314034, 0.70867468]])
```

In [133…
```
a.sum()
```

Out[133…
```
19.83654724805626
```

In [134…
```
a.sum(axis=0)
```

Out[134…
```
array([5.55350538, 3.94241069, 4.69703303, 5.64359815])
```

In [135…
```
a
```

Out[135…
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793],
       [0.18198257, 0.06664977, 0.08885812, 0.22529457],
       [0.05578665, 0.00725407, 0.90873155, 0.07395098],
       [0.75784909, 0.26999395, 0.67780173, 0.92841833],
       [0.59983431, 0.84526954, 0.4467491 , 0.27879045],
       [0.80516443, 0.30457498, 0.20743558, 0.411222  ],
       [0.12945729, 0.5370648 , 0.1564685 , 0.88974561],
       [0.46398892, 0.31741535, 0.84248024, 0.67543583],
       [0.99519624, 0.85150127, 0.53314034, 0.70867468]])
```

In [136…
```
a.sum(axis=1)
```

Out[136…
```
array([2.74044763, 1.85391875, 0.56278503, 1.04572325, 2.63406309,
       2.1706434 , 1.72839699, 1.71273621, 2.29932035, 3.08851254])
```

In [137…
```
a.mean()
```

Out[137…
```
0.4959136812014065
```

In [138…
```
a.mean(axis=0)
```

Out[138…
```
array([0.55535054, 0.39424107, 0.4697033 , 0.56435982])
```

In [139…
```
a.mean(axis=1)
```

Out[139…
```
array([0.68511191, 0.46347969, 0.14069626, 0.26143081, 0.65851577,
       0.54266085, 0.43209925, 0.42818405, 0.57483009, 0.77212813])
```

In [146…
```
a[[0,1], [0,1]]
```

Out[146…
```
array([0.91034426, 0.30811191])
```

In [147…
```
a[ [0,1] , :]
```

Out[147…
```
array([[0.91034426, 0.43457504, 0.51284057, 0.88268776],
       [0.65390162, 0.30811191, 0.32252729, 0.56937793]])
```

In [159…
```
np.empty((100,100))
```

Out[159…
```
array([[6.95183186e-310, 6.95183186e-310, 6.95182817e-310, ...,
        6.95183186e-310, 6.95182718e-310, 6.95183181e-310],
       [6.95183186e-310, 6.95182723e-310, 6.95183173e-310, ...,
        6.95183186e-310, 6.95183186e-310, 6.95183186e-310],
       [6.95183189e-310, 6.95183189e-310, 6.95183186e-310, ...,
        6.95183189e-310, 6.95183173e-310, 6.95183189e-310],
       ...,
       [6.95183094e-310, 6.95183094e-310, 6.95183094e-310, ...,
        6.95183094e-310, 6.95183094e-310, 6.95183094e-310],
       [6.95183094e-310, 6.95183094e-310, 6.95183094e-310, ...,
        6.95183094e-310, 6.95183094e-310, 6.95183094e-310],
       [6.95183094e-310, 6.95183094e-310, 6.95183094e-310, ...,
        6.95183094e-310, 6.95183094e-310, 6.95183094e-310]])
```

In [157…
```
np.info(np.arange)
```

```
arange([start,] stop[, step,], dtype=None)

Return evenly spaced values within a given interval.

Values are generated within the half-open interval ``[start, stop)``
(in other words, the interval including `start` but excluding `stop`).
For integer arguments the function is equivalent to the Python built-in
`range` function, but returns an ndarray rather than a list.

When using a non-integer step, such as 0.1, the results will often not
be consistent.  It is better to use `numpy.linspace` for these cases.

Parameters
----------
start : number, optional
    Start of interval.  The interval includes this value.  The default
    start value is 0.
stop : number
    End of interval.  The interval does not include this value, except
    in some cases where `step` is not an integer and floating point
    round-off affects the length of `out`.
step : number, optional
    Spacing between values.  For any output `out`, this is the distance
    between two adjacent values, ``out[i+1] - out[i]``.  The default
    step size is 1.  If `step` is specified as a position argument,
    `start` must also be given.
dtype : dtype
    The type of the output array.  If `dtype` is not given, infer the data
    type from the other input arguments.

Returns
-------
arange : ndarray
    Array of evenly spaced values.

    For floating point arguments, the length of the result is
    ``ceil((stop - start)/step)``.  Because of floating point overflow,
    this rule may result in the last element of `out` being greater
    than `stop`.

See Also
--------
numpy.linspace : Evenly spaced numbers with careful handling of endpoints.
numpy.ogrid: Arrays of evenly spaced numbers in N-dimensions.
numpy.mgrid: Grid-shaped arrays of evenly spaced numbers in N-dimensions.

Examples
--------
>>> np.arange(3)
array([0, 1, 2])
>>> np.arange(3.0)
```

```
array([ 0.,  1.,  2.])
>>> np.arange(3,7)
array([3, 4, 5, 6])
>>> np.arange(3,7,2)
```

In [156…
```
print (np.arange.__doc__)
```

```
arange([start,] stop[, step,], dtype=None)

    Return evenly spaced values within a given interval.

    Values are generated within the half-open interval ``[start, stop)``
    (in other words, the interval including `start` but excluding `stop`).
    For integer arguments the function is equivalent to the Python built-in
    `range` function, but returns an ndarray rather than a list.

    When using a non-integer step, such as 0.1, the results will often not
    be consistent.  It is better to use `numpy.linspace` for these cases.

    Parameters
    ----------
    start : number, optional
        Start of interval.  The interval includes this value.  The default
        start value is 0.
    stop : number
        End of interval.  The interval does not include this value, except
        in some cases where `step` is not an integer and floating point
        round-off affects the length of `out`.
    step : number, optional
        Spacing between values.  For any output `out`, this is the distance
        between two adjacent values, ``out[i+1] - out[i]``.  The default
        step size is 1.  If `step` is specified as a position argument,
        `start` must also be given.
    dtype : dtype
        The type of the output array.  If `dtype` is not given, infer the d
ata
        type from the other input arguments.

    Returns
    -------
    arange : ndarray
        Array of evenly spaced values.

        For floating point arguments, the length of the result is
        ``ceil((stop - start)/step)``.  Because of floating point overflow,
        this rule may result in the last element of `out` being greater
        than `stop`.

    See Also
    --------
    numpy.linspace : Evenly spaced numbers with careful handling of endpoin
ts.
    numpy.ogrid: Arrays of evenly spaced numbers in N-dimensions.
    numpy.mgrid: Grid-shaped arrays of evenly spaced numbers in N-dimension
s.

    Examples
    --------
    >>> np.arange(3)
    array([0, 1, 2])
    >>> np.arange(3.0)
    array([ 0.,  1.,  2.])
    >>> np.arange(3,7)
    array([3, 4, 5, 6])
    >>> np.arange(3,7,2)
    array([3, 5])
```

In [160…
```
help(np.arange)
```

```
Help on built-in function arange in module numpy:

arange(...)
    arange([start,] stop[, step,], dtype=None)

    Return evenly spaced values within a given interval.
```

```
    Values are generated within the half-open interval ``[start, stop)``
    (in other words, the interval including `start` but excluding `stop`).
    For integer arguments the function is equivalent to the Python built-in
    `range` function, but returns an ndarray rather than a list.

    When using a non-integer step, such as 0.1, the results will often not
    be consistent.  It is better to use `numpy.linspace` for these cases.

    Parameters
    ----------
    start : number, optional
        Start of interval.  The interval includes this value.  The default
        start value is 0.
    stop : number
        End of interval.  The interval does not include this value, except
        in some cases where `step` is not an integer and floating point
        round-off affects the length of `out`.
    step : number, optional
        Spacing between values.  For any output `out`, this is the distance
        between two adjacent values, ``out[i+1] - out[i]``.  The default
        step size is 1.  If `step` is specified as a position argument,
        `start` must also be given.
    dtype : dtype
        The type of the output array.  If `dtype` is not given, infer the d
ata
        type from the other input arguments.

    Returns
    -------
    arange : ndarray
        Array of evenly spaced values.

        For floating point arguments, the length of the result is
        ``ceil((stop - start)/step)``.  Because of floating point overflow,
        this rule may result in the last element of `out` being greater
        than `stop`.

    See Also
    --------
    numpy.linspace : Evenly spaced numbers with careful handling of endpoin
ts.
    numpy.ogrid: Arrays of evenly spaced numbers in N-dimensions.
    numpy.mgrid: Grid-shaped arrays of evenly spaced numbers in N-dimension
s.

    Examples
    --------
    >>> np.arange(3)
    array([0, 1, 2])
    >>> np.arange(3.0)
    array([ 0.,  1.,  2.])
    >>> np.arange(3,7)
    array([3, 4, 5, 6])
    >>> np.arange(3,7,2)
    array([3, 5])
```

In [161…
```
a = np.random.random((10,4))
```

In [162…
```
a
```

Out[162…
```
array([[0.79683391, 0.25347974, 0.92577563, 0.38656224],
       [0.63168278, 0.09892285, 0.67796582, 0.74420502],
       [0.45661152, 0.48652765, 0.77131846, 0.82170743],
       [0.48671481, 0.92323741, 0.61871866, 0.11600489],
       [0.36461906, 0.9185336 , 0.94501384, 0.46753965],
       [0.51922117, 0.95833315, 0.11351766, 0.19240031],
       [0.4867943 , 0.57543986, 0.02064402, 0.10252277],
       [0.17175174, 0.08612459, 0.48906271, 0.43351026],
       [0.06018079, 0.54431405, 0.90772228, 0.96892279],
       [0.23176151, 0.87935775, 0.39122227, 0.78592687]])
```

In [164…
```
a>0.6
```

Out[164…
```
array([[ True, False,  True, False],
       [ True, False,  True,  True],
       [False, False,  True,  True],
       [False,  True,  True, False],
       [False,  True,  True, False],
       [False,  True, False, False],
       [False, False, False, False],
       [False, False, False, False],
       [False, False,  True,  True],
       [False,  True, False,  True]])
```

In [165…
```
a[a>0.6]
```

Out[165…
```
array([0.79683391, 0.92577563, 0.63168278, 0.67796582, 0.74420502,
       0.77131846, 0.82170743, 0.92323741, 0.61871866, 0.9185336 ,
       0.94501384, 0.95833315, 0.90772228, 0.96892279, 0.87935775,
       0.78592687])
```

In [166…
```
a[a>0.6] = 4
```

In [167…
```
a
```

Out[167…
```
array([[4.        , 0.25347974, 4.        , 0.38656224],
       [4.        , 0.09892285, 4.        , 4.        ],
       [0.45661152, 0.48652765, 4.        , 4.        ],
       [0.48671481, 4.        , 4.        , 0.11600489],
       [0.36461906, 4.        , 4.        , 0.46753965],
       [0.51922117, 4.        , 0.11351766, 0.19240031],
       [0.4867943 , 0.57543986, 0.02064402, 0.10252277],
       [0.17175174, 0.08612459, 0.48906271, 0.43351026],
       [0.06018079, 0.54431405, 4.        , 4.        ],
       [0.23176151, 4.        , 0.39122227, 4.        ]])
```

In [168…
```
a = np.random.random((10,4))
```

In [169…
```
a
```

Out[169…
```
array([[0.11664114, 0.90081776, 0.76977479, 0.0725833 ],
       [0.98007171, 0.76313216, 0.52170905, 0.91533419],
       [0.84525193, 0.24917769, 0.17264222, 0.70513479],
       [0.72593005, 0.81587464, 0.2714711 , 0.1839582 ],
       [0.28367563, 0.16678939, 0.05534161, 0.91322394],
       [0.00403207, 0.94703751, 0.97902811, 0.43846949],
       [0.83043016, 0.8182949 , 0.91601623, 0.44531797],
       [0.15918545, 0.70754974, 0.01450686, 0.3241225 ],
       [0.40790456, 0.66964418, 0.28835226, 0.62179632],
       [0.11769992, 0.34173566, 0.79710396, 0.3527701 ]])
```

In [171…
```
a[:2, :2] = np.ones((2,2))*8
```

In [172…
```
a
```

Out[172…
```
array([[8.00000000e+00, 8.00000000e+00, 7.69774792e-01, 7.25832967e-02],
       [8.00000000e+00, 8.00000000e+00, 5.21709052e-01, 9.15334193e-01],
       [8.45251934e-01, 2.49177687e-01, 1.72642222e-01, 7.05134788e-01],
       [7.25930048e-01, 8.15874642e-01, 2.71471100e-01, 1.83958199e-01],
       [2.83675632e-01, 1.66789390e-01, 5.53416108e-02, 9.13223941e-01],
       [4.03206965e-03, 9.47037514e-01, 9.79028114e-01, 4.38469488e-01],
       [8.30430158e-01, 8.18294901e-01, 9.16016232e-01, 4.45317973e-01],
       [1.59185446e-01, 7.07549741e-01, 1.45068593e-02, 3.24122500e-01],
       [4.07904557e-01, 6.69644181e-01, 2.88352264e-01, 6.21796324e-01],
       [1.17699916e-01, 3.41735664e-01, 7.97103965e-01, 3.52770099e-01]])
```

In [173…
```
a = np.random.random((10,4))
```

In [174…
```
a
```

Out[174…
```
array([[0.82164049, 0.19352902, 0.05711344, 0.28260897],
       [0.59726084, 0.51859015, 0.39009567, 0.08538083],
       [0.58735639, 0.40942708, 0.2441224 , 0.03564297],
```

```
       [0.61365861, 0.9384137 , 0.15112556, 0.86910738],
       [0.99333749, 0.99460423, 0.56210331, 0.60220333],
       [0.04833048, 0.07312763, 0.68074445, 0.35236327],
       [0.82015041, 0.27351077, 0.25745896, 0.62515886],
       [0.12450278, 0.62692996, 0.86326096, 0.08718809],
       [0.32797452, 0.56749623, 0.15775214, 0.31211542],
       [0.21574227, 0.7574958 , 0.78618495, 0.71996386]])
```

In [175…
```
a[a>0.8]
```

Out[175…
```
array([0.82164049, 0.9384137 , 0.86910738, 0.99333749, 0.99460423,
       0.82015041, 0.86326096])
```

In [176…
```
a[  (a>0.8) | (a<0.2)  ]
```

Out[176…
```
array([0.82164049, 0.19352902, 0.05711344, 0.08538083, 0.03564297,
       0.9384137 , 0.15112556, 0.86910738, 0.99333749, 0.99460423,
       0.04833048, 0.07312763, 0.82015041, 0.12450278, 0.86326096,
       0.08718809, 0.15775214])
```

In [177…
```
a[ ~ ((a>0.8) | (a<0.2))  ]
```

Out[177…
```
array([0.28260897, 0.59726084, 0.51859015, 0.39009567, 0.58735639,
       0.40942708, 0.2441224 , 0.61365861, 0.56210331, 0.60220333,
       0.68074445, 0.35236327, 0.27351077, 0.25745896, 0.62515886,
       0.62692996, 0.32797452, 0.56749623, 0.31211542, 0.21574227,
       0.7574958 , 0.78618495, 0.71996386])
```

In [178…
```
a = np.random.random((2,3))
```

In [179…
```
b = np.random.random((2,3))
```

In [180…
```
a
```

Out[180…
```
array([[0.09967707, 0.48059882, 0.99304391],
       [0.43176898, 0.02422939, 0.38226235]])
```

In [181…
```
b
```

Out[181…
```
array([[0.16805678, 0.57776481, 0.63451924],
       [0.47654105, 0.32328421, 0.38376588]])
```

In [182…
```
np.hstack((a,b))
```

Out[182…
```
array([[0.09967707, 0.48059882, 0.99304391, 0.16805678, 0.57776481,
        0.63451924],
       [0.43176898, 0.02422939, 0.38226235, 0.47654105, 0.32328421,
        0.38376588]])
```

In [183…
```
np.hstack((a,b)).shape
```

Out[183…
```
(2, 6)
```

In [185…
```
np.vstack((a,b))
```

Out[185…
```
array([[0.09967707, 0.48059882, 0.99304391],
       [0.43176898, 0.02422939, 0.38226235],
       [0.16805678, 0.57776481, 0.63451924],
       [0.47654105, 0.32328421, 0.38376588]])
```

In [186…
```
np.block([a,b])
```

Out[186…
```
array([[0.09967707, 0.48059882, 0.99304391, 0.16805678, 0.57776481,
        0.63451924],
       [0.43176898, 0.02422939, 0.38226235, 0.47654105, 0.32328421,
        0.38376588]])
```

In [187…
```
np.block([[a], [b]])
```

Out[187…
```
array([[0.09967707, 0.48059882, 0.99304391],
       [0.43176898, 0.02422939, 0.38226235],
       [0.16805678, 0.57776481, 0.63451924],
       [0.47654105, 0.32328421, 0.38376588]])
```

```
  A       B
  0 1     5 6
  2 3     7 8

  A B
  B A

  0 1 5 6
  2 3 7 8
  5 6 0 1
  7 8 2 3
```

In [188…
```
a = np.array([0,1,2,3]).reshape((2,-1))
```

In [189…
```
a
```

Out[189…
```
array([[0, 1],
       [2, 3]])
```

In [190…
```
b=  np.array([5,6,7,8]).reshape((2,-1))
```

In [191…
```
np.block( [[a,b], [b,a]] )
```

Out[191…
```
array([[0, 1, 5, 6],
       [2, 3, 7, 8],
       [5, 6, 0, 1],
       [7, 8, 2, 3]])
```

In [192…
```
a
```

Out[192…
```
array([[0, 1],
       [2, 3]])
```

In [196…
```
np.linalg.det(a)
```

Out[196…
```
-2.0
```

In [197…
```
np.linalg.inv(a)
```

Out[197…
```
array([[-1.5,  0.5],
       [ 1. ,  0. ]])
```

In [198…
```
a.dot(np.linalg.inv(a))
```

Out[198…
```
array([[1., 0.],
       [0., 1.]])
```

In [199…
```
a*b
```

Out[199…
```
array([[ 0,  6],
       [14, 24]])
```

In [200…
```
np.matmul(a, np.linalg.inv(a))
```

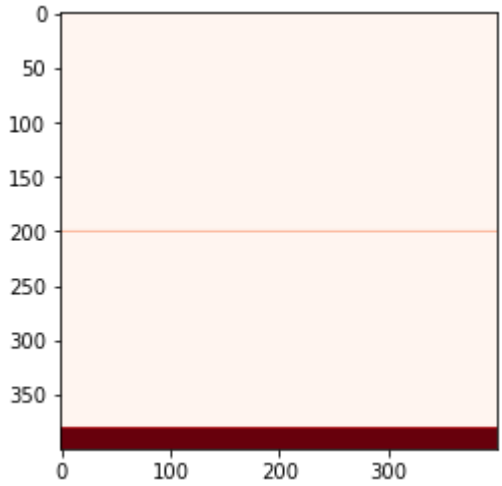
Out[200…
```
array([[1., 0.],
       [0., 1.]])
```

In [201…
```
import matplotlib.pyplot as plt
```

In [215…
```
#a = np.random.random((400, 400))
a = np.zeros((400, 400))
a[200, :] = 0.5 * np.ones((1,400))
a[380:, :] = np.ones((20, 400))
```

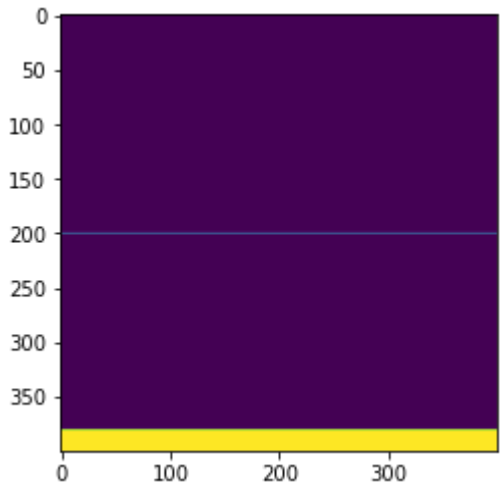
In [222…
```
plt.imshow(a, cmap='Reds')
```

Out[222…
```
<matplotlib.image.AxesImage at 0x7ff8d32a8dc0>
```



In [223…
```
plt.imshow(a, cmap='viridis')
```

Out[223…
```
<matplotlib.image.AxesImage at 0x7ff8d42a7220>
```

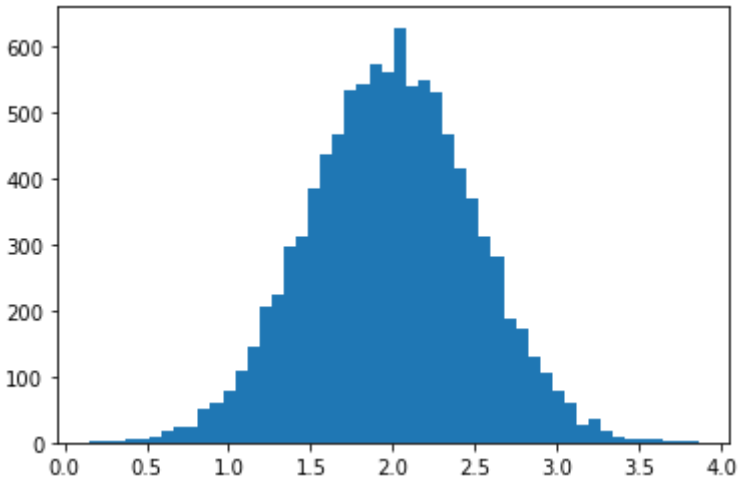


In [225…
```
b = np.random.normal(2, 0.5, 10000)
```

In [231…
```
plt.hist(b, bins=50, )
```

Out[231…
```
(array([  1.,   3.,   2.,   4.,   5.,   8.,  17.,  24.,  23.,  50.,  61.,
         77., 108., 144., 205., 223., 296., 312., 386., 437., 466., 535.,
        542., 572., 560., 629., 540., 548., 530., 466., 414., 371., 311.,
        282., 189., 173., 130., 106.,  78.,  59.,  27.,  36.,  19.,   9.,
          6.,   6.,   6.,   1.,   1.,   2.]),
 array([0.14632218, 0.22066297, 0.29500376, 0.36934455, 0.44368534,
        0.51802612, 0.59236691, 0.6667077 , 0.74104849, 0.81538928,
        0.88973007, 0.96407086, 1.03841165, 1.11275244, 1.18709323,
        1.26143402, 1.33577481, 1.41011559, 1.48445638, 1.55879717,
        1.63313796, 1.70747875, 1.78181954, 1.85616033, 1.93050112,
        2.00484191, 2.0791827 , 2.15352349, 2.22786427, 2.30220506,
        2.37654585, 2.45088664, 2.52522743, 2.59956822, 2.67390901,
```

```
        2.7482498 , 2.82259059, 2.89693138, 2.97127217, 3.04561295,
        3.11995374, 3.19429453, 3.26863532, 3.34297611, 3.4173169 ,
        3.49165769, 3.56599848, 3.64033927, 3.71468006, 3.78902085,
        3.86336163]),
<BarContainer object of 50 artists>)
```



In [237…
```
np.save('numbers.npy', a, allow_pickle=False)
```

In [238…
```
bb = np.load('numbers.npy')
```

In [239…
```
bb
```

Out[239…
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [1., 1., 1., ..., 1., 1., 1.],
       [1., 1., 1., ..., 1., 1., 1.],
       [1., 1., 1., ..., 1., 1., 1.]])
```

In [240…
```
np.eye(3)
```

Out[240…
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [247…
```
a = np.random.random((2,3))
b = np.random.random((2,3))
```

In [249…
```
a
```

Out[249…
```
array([[0.02901079, 0.24198614, 0.63418021],
       [0.36267935, 0.96943709, 0.51991906]])
```

In [250…
```
b
```

Out[250…
```
array([[0.97738408, 0.14605648, 0.64308621],
       [0.7227673 , 0.34126365, 0.75843498]])
```

In [256…
```
f = np.vstack((a,b))
```

In [257…
```
l = np.vsplit(f, 2)
```

In [258…
```
l[0]
```

Out[258…
```
array([[0.02901079, 0.24198614, 0.63418021],
       [0.36267935, 0.96943709, 0.51991906]])
```

In [259…
```
l[1]
```

Out[259…
```
array([[0.97738408, 0.14605648, 0.64308621],
       [0.7227673 , 0.34126365, 0.75843498]])
```

In [261…
```
a = np.random.random((4,3,2))
```

In [262…
```
a
```

Out[262…
```
array([[[0.97802065, 0.7498867 ],
        [0.44254369, 0.25451652],
        [0.71827602, 0.59238266]],

       [[0.87701873, 0.56530363],
        [0.75442683, 0.24924573],
        [0.54135247, 0.4586013 ]],

       [[0.40365882, 0.71294405],
        [0.37863548, 0.97681321],
        [0.47045034, 0.75938253]],

       [[0.74222558, 0.07796156],
        [0.72495488, 0.78410846],
        [0.32311795, 0.42584033]]])
```

In [266…
```
a.sum(axis=0)
```

Out[266…
```
array([[3.00092379, 2.10609594],
       [2.30056088, 2.26468391],
       [2.05319677, 2.23620682]])
```

In [267…
```
a.sum(axis=1)
```

Out[267…
```
array([[2.13884036, 1.59678589],
       [2.17279804, 1.27315066],
       [1.25274463, 2.44913978],
       [1.79029841, 1.28791034]])
```

In [270…
```
a
```

Out[270…
```
array([[[0.97802065, 0.7498867 ],
        [0.44254369, 0.25451652],
        [0.71827602, 0.59238266]],

       [[0.87701873, 0.56530363],
        [0.75442683, 0.24924573],
        [0.54135247, 0.4586013 ]],

       [[0.40365882, 0.71294405],
        [0.37863548, 0.97681321],
        [0.47045034, 0.75938253]],

       [[0.74222558, 0.07796156],
        [0.72495488, 0.78410846],
        [0.32311795, 0.42584033]]])
```

In [271…
```
a [ :2 , -1 , :1 ]
```

Out[271…
```
array([[0.71827602],
       [0.54135247]])
```

In [273…
```
from numpy.random import normal
```

In [282…
```
a = np.array(10 * np.random.random((3,4)), dtype=np.int)
a
```

Out[282…
```
array([[0, 5, 2, 8],
       [6, 2, 6, 0],
       [7, 8, 1, 8]])
```

In [287…
```
a[2,:][a[1,:] > a[0, :]].sum()
```

Out[287…
```
8
```

In [289…
```
a
```

Out[289…
```
array([[0, 5, 2, 8],
       [6, 2, 6, 0],
       [7, 8, 1, 8]])
```

In [291…
```
a.T
```

Out[291…
```
array([[0, 6, 7],
       [5, 2, 8],
       [2, 6, 1],
       [8, 0, 8]])
```

In [292…
```
np.transpose(a)
```

Out[292…
```
array([[0, 6, 7],
       [5, 2, 8],
       [2, 6, 1],
       [8, 0, 8]])
```

In [277…
```
a.copy(dtype=np.int)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-277-bf1f579d9bed> in <module>
----> 1 a.copy(dtype=np.int)

TypeError: 'dtype' is an invalid keyword argument for copy()
```

In [293…
```
a
```

Out[293…
```
array([[0, 5, 2, 8],
       [6, 2, 6, 0],
       [7, 8, 1, 8]])
```

In [294…
```
a.sum(axis=1)
```

Out[294…
```
array([15, 14, 24])
```

In [298…
```
np.median(a, axis=1)
```

Out[298…
```
array([3.5, 4. , 7.5])
```

In [299…
```
np.inf
```

Out[299…
```
inf
```

In [300…
```
np.inf > 1112134123412
```

Out[300…
```
True
```

In [301…
```
np.nan
```

Out[301…
```
nan
```

In [302…
```
a = np.array([1,2,3])
b = np.array([1,0,2])
```

In [303…
```
a/b
```

```
<ipython-input-303-aae42d317509>:1: RuntimeWarning: divide by zero encounte
red in true_divide
```

```
  a/b
```

Out[303…
```
array([1. , inf, 1.5])
```

In [304…
```
c = a/b
```

```
<ipython-input-304-a3c10b518fb0>:1: RuntimeWarning: divide by zero encounte
red in true_divide
  c = a/b
```

In [305…
```
np.isinf(c)
```

Out[305…
```
array([False,  True, False])
```

In [306…
```
c[np.isinf(c)] = 10
```

In [307…
```
c
```

Out[307…
```
array([ 1. , 10. ,  1.5])
```

In [ ]:
```

```