Entrée [7]:

```
import numpy as np
```

Entrée [8]:

```
list1 = [0,1,2,3,4]
```

Entrée [10]:

```
list1
```

Out[10]:

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

Entrée [11]:

```
arr1d = np.array(list1)
```

Entrée [12]:

```
arr1d
```

Out[12]:

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

Entrée [13]:

```
list1.append(5)
```

Entrée [14]:

```
list1
```

Out[14]:

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

Entrée [15]:

```
list1 + 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-a66ed14c8eee> in <module>
----> 1 list1 + 2

TypeError: can only concatenate list (not "int") to list
```

Entrée [16]:

```
arr1d + 2
```

Out[16]:

```
array([2, 3, 4, 5, 6])
```

Entrée [17]:

```
list2 = [[1, 1, 1], [2, 2 ,2], [3, 3, 3]]
arr2d = np.array(list2)
```

Entrée [19]:

```
type(arr2d)
```

Out[19]:

```
numpy.ndarray
```

Entrée [20]:

```
arr2d.dtype
```

Out[20]:

```
dtype('int32')
```

Entrée [21]:

```
arr2d = np.array(list2, dtype = 'float')
```

Entrée [22]:

```
arr2d
```

Out[22]:

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

Entrée [25]:

```
arr2d = arr2d.astype('int')
```

Entrée [26]:

```
arr2d.astype('str')
```

Out[26]:

```
array([['1', '1', '1'],
       ['2', '2', '2'],
       ['3', '3', '3']], dtype='<U11')
```

Entrée [30]:

```
list1.append('6')
```

Entrée [31]:

```
list1
```

Out[31]:

```
[0, 1, 2, 3, 4, 5, '6']
```

Entrée [ ]:

```

```

Entrée [32]:

```
np2list = arr2d.tolist()
```

Entrée [33]:

```
np2list
```

Out[33]:

```
[[1, 1, 1], [2, 2, 2], [3, 3, 3]]
```

Entrée [34]:

```
arr2d.tostring()
```

Out[34]:

```
b'\x01\x00\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00\x02\x00\x00\x00\x02\x00\x
00\x00\x02\x00\x00\x00\x03\x00\x00\x00\x03\x00\x00\x00\x03\x00\x00\x00'
```

Entrée [35]:

```
arr2d.tobytes()
```

Out[35]:

```
b'\x01\x00\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00\x02\x00\x00\x00\x02\x00\x
00\x00\x02\x00\x00\x00\x03\x00\x00\x00\x03\x00\x00\x00\x03\x00\x00\x00'
```

Entrée [ ]:

```

```

dtype and shape

Entrée [36]:

```
list2
```

Out[36]:

```
[[1, 1, 1], [2, 2, 2], [3, 3, 3]]
```

Entrée [38]:

```
arr2d = arr2d.astype('float' )
```

Entrée [39]:

```
arr2d
```

Out[39]:

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

Entrée [40]:

```
print('Shape: ', arr2d.shape) # python3
```

```
Shape:  (3, 3)
```

Entrée [41]:

```
arr2d.dtype
```

Out[41]:

```
dtype('float64')
```

Entrée [43]:

```
arr2d.size
```

Out[43]:

```
9
```

Entrée [44]:

```
arr1d.size
```

Out[44]:

```
5
```

Entrée [45]:

```
arr2d.ndim
```

Out[45]:

```
2
```

Entrée [46]:

```
arr1d.ndim
```

Out[46]:

```
1
```

Entrée [ ]:

```

```

Entrée [50]:

```
arr1d = arr1d * arr1d
```

Entrée [51]:

```
arr1d
```

Out[51]:

```
array([ 4,  9,  0,  1, 36], dtype=int32)
```

Entrée [55]:

```
arr1d[1]
```

Out[55]:

```
9
```

Entrée [60]:

```
arr2d[1][0] #[R][C]
```

Out[60]:

```
2.0
```

Entrée [59]:

```
arr2d
```

Out[59]:

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

Entrée [ ]:

```

```

Entrée [61]:

```
boolarr = arr2d<3
```

Entrée [62]:

```
boolarr
```

Out[62]:

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

Entrée [63]:

```
arr2d[boolarr]
```

Out[63]:

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

Entrée [ ]:

Entrée [64]:

```
arr2d
```

Out[64]:

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

Entrée [65]:

```
arr2d[::-1, ]
```

Out[65]:

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

Entrée [ ]:

Entrée [66]:

```
arr2d[::-1, ::-1]
```

Out[66]:

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

np.nan, np.inf

Entrée [67]:

```
np.nan
```

Out[67]:

```
nan
```

Entrée [68]:

```
np.inf
```

Out[68]:

```
inf
```

Entrée [69]:

```
arr2d
```

Out[69]:

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

Entrée [70]:

```
arr2d[0][0] = np.nan
arr2d[0][1] = np.inf
arr2d
```

Out[70]:

```
array([[nan, inf,  1.],
       [ 2.,  2.,  2.],
       [ 3.,  3.,  3.]])
```

Entrée [71]:

```
np.isnan(arr2d)
```

Out[71]:

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

Entrée [72]:

```
np.isinf(arr2d)
```

Out[72]:

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

Entrée [73]:

```
missing_flag = np.isnan(arr2d) | np.isinf(arr2d)
missing_flag
```

Out[73]:

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

Entrée [74]:

```
#replace inf and nan with 0
```

Entrée [75]:

```
arr2d[missing_flag]
```

Out[75]:

```
array([nan, inf])
```

Entrée [76]:

```
arr2d[missing_flag] = 0
```

Entrée [77]:

```
arr2d
```

Out[77]:

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

Entrée [ ]:

```

```

Statistical operations

Entrée [78]:

```
#mean, std, var
```

Entrée [79]:

```
arr2d.mean()
```

Out[79]:

```
1.7777777777777777
```

Entrée [80]:

```
arr2d.max()
```

Out[80]:

```
3.0
```

Entrée [81]:

```
arr2d.min()
```

Out[81]:

```
0.0
```

Entrée [82]:

```
arr2d.std()
```

Out[82]:

```
1.1331154474650633
```

Entrée [83]:

```
arr2d.var()
```

Out[83]:

```
1.2839506172839505
```

Entrée [85]:

```
arr2d.squeeze()
```

Out[85]:

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

Entrée [86]:

```
arr2d.cumsum()
```

Out[86]:

```
array([ 0.,  0.,  1.,  3.,  5.,  7., 10., 13., 16.])
```

Entrée [ ]:

```

```

Entrée [87]:

```
arr2d
```

Out[87]:

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

Entrée [ ]:

```

```

Entrée [88]:

```
arr = arr2d[:2, :2]
```

Entrée [89]:

```
arr
```

Out[89]:

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

Entrée [ ]:

```

```

Entrée [91]:

```
arr2d[1:3, 1:2]
```

Out[91]:

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

Entrée [ ]:

```

```

Entrée [92]:

```
arr2d
```

Out[92]:

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

```

```

Entrée [101]:

```
a = arr2d.reshape(1,9)
a
```

Out[101]:

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

Entrée [100]:

```
a.ndim
```

Out[100]:

```
2
```

Entrée [95]:

```
arr2d.reshape(9,1)
```

Out[95]:

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

Entrée [ ]:

```

```

Entrée [105]:

```
a = arr2d.flatten()
a #copy
```

Out[105]:

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

Entrée [106]:

```
b = arr2d.ravel()
b #reference
```

Out[106]:

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

Entrée [107]:

```
arr2d
```

Out[107]:

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

Entrée [109]:

```
b[0] = -1
```

Entrée [110]:

```
arr2d
```

Out[110]:

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

Entrée [ ]:

```

```

sequence, repetitions, and random numbers

Entrée [116]:

```
np.arange(1, 5, dtype = 'int')
```

Out[116]:

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

Entrée [118]:

```
np.arange(1, 50, 2)
```

Out[118]:

```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
       35, 37, 39, 41, 43, 45, 47, 49])
```

Entrée [ ]:

```

```

Entrée [123]:

```
np.linspace(1, 50, 9)
```

Out[123]:

```
array([ 1.   ,  7.125, 13.25 , 19.375, 25.5  , 31.625, 37.75 , 43.875,
       50.   ])
```

Entrée [ ]:

```

```

Entrée [124]:

```
np.logspace(1, 50, 10)
```

Out[124]:

```
array([1.00000000e+01, 2.78255940e+06, 7.74263683e+11, 2.15443469e+17,
       5.99484250e+22, 1.66810054e+28, 4.64158883e+33, 1.29154967e+39,
       3.59381366e+44, 1.00000000e+50])
```

Entrée [ ]:

```

```

Entrée [126]:

```
np.zeros([2,2])
```

Out[126]:

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

Entrée [127]:

```
np.ones([2, 2])
```

Out[127]:

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

Entrée [ ]:

Entrée [128]:

```
a = [1, 2, 3]
```

Entrée [129]:

```
a
```

Out[129]:

```
[1, 2, 3]
```

Entrée [130]:

```
np.tile(a, 3)
```

Out[130]:

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

Entrée [131]:

```
np.repeat(a, 3)
```

Out[131]:

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

Entrée [132]:

```
np.repeat(arr2d, 3)
```

Out[132]:

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

Entrée [ ]:

```

```

Entrée [133]:

```
arr2d
```

Out[133]:

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

Entrée [ ]:

```

```

Entrée [139]:

```
np.random.rand(3, 3)
```

Out[139]:

```
array([[0.03022941, 0.4897838 , 0.48808364],
       [0.06658789, 0.25569082, 0.30411541],
       [0.65788261, 0.16300932, 0.61885932]])
```

Entrée [138]:

```
np.random.randn(3, 3)
```

Out[138]:

```
array([[-0.95832052, -0.33374875,  0.46566206],
       [-1.04891141,  0.5784687 , -0.1718599 ],
       [ 0.27468945, -0.93154267,  0.65645431]])
```

Entrée [141]:

```
np.random.randint(0, 10, [3,3])
```

Out[141]:

```
array([[6, 4, 0],
       [2, 5, 9],
       [7, 3, 0]])
```

Entrée [ ]:

```

```

Entrée [157]:

```
np.random.seed(1)
np.random.randint(0, 10, [3,3])
```

Out[157]:

```
array([[5, 8, 9],
       [5, 0, 0],
       [1, 7, 6]])
```

Entrée [ ]:

```

```

Entrée [161]:

```
np.unique(arr2d)
```

Out[161]:

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

Entrée [162]:

```
arr2d
```

Out[162]:

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

Entrée [163]:

```
uniques, counts = np.unique(arr2d, return_counts= True)
```

Entrée [164]:

```
uniques
```

Out[164]:

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

Entrée [165]:

```
counts
```

Out[165]:

```
array([1, 1, 1, 3, 3], dtype=int64)
```

Entrée [ ]:

```

```

Numpy Crash Course Part 2

Entrée [166]:

```
arr = np.array([8,94,8,56,1,3,4,5,7])
print(arr)
```

```
[ 8 94  8 56  1  3  4  5  7]
```

Entrée [167]:

```
index_gt10 = np.where(arr>10)
index_gt10
```

Out[167]:

```
(array([1, 3], dtype=int64),)
```

Entrée [168]:

```
arr[index_gt10]
```

Out[168]:

```
array([94, 56])
```

Entrée [169]:

```
arr[arr>10]
```

Out[169]:

```
array([94, 56])
```

Entrée [170]:

```
arr>10
```

Out[170]:

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

Entrée [ ]:

Entrée [171]:

```
np.where(arr>10, 'gt10', 'lt10')
```

Out[171]:

```
array(['lt10', 'gt10', 'lt10', 'gt10', 'lt10', 'lt10', 'lt10', 'lt10',
       'lt10'], dtype='<U4')
```

Entrée [ ]:

Entrée [172]:

```
arr
```

Out[172]:

```
array([ 8, 94,  8, 56,  1,  3,  4,  5,  7])
```

Entrée [173]:

```
arr.max()
```

Out[173]:

```
94
```

Entrée [174]:

```
arr.argmax()
```

Out[174]:

```
1
```

Entrée [176]:

```
arr[arr.argmin()]
```

Out[176]:

```
1
```

Entrée [177]:

```
arr.argmin()
```

Out[177]:

```
4
```

read and write csv file

Entrée [ ]:

```
#np.genfromtxt(), np.loadtxt()
```

Entrée [181]:

```
data = np.genfromtxt('https://raw.githubusercontent.com/selva86/datasets/master/Auto.csv',
              filling_values= -1000, dtype = 'float')
```

Entrée [184]:

```
data.shape
```

Out[184]:

```
(392, 9)
```

Entrée [186]:

```
np.set_printoptions(suppress=True)
data[:3]
```

Out[186]:

```
array([[   18. ,      8. ,    307. ,    130. ,   3504. ,     12. ,     70. ,
          1. , -1000. ],
       [   15. ,      8. ,    350. ,    165. ,   3693. ,     11.5,     70. ,
          1. , -1000. ],
       [   18. ,      8. ,    318. ,    150. ,   3436. ,     11. ,     70. ,
          1. , -1000. ]])
```

Entrée [ ]:

```

```

Entrée [199]:

```
data2 = np.genfromtxt('https://raw.githubusercontent.com/selva86/datasets/master/Auto.csv',
data2[:3]
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: VisibleD
eprecationWarning: Reading unicode strings without specifying the encoding a
rgument is deprecated. Set the encoding, use None for the system default.
  """Entry point for launching an IPython kernel.
```

Out[199]:

```
array([(18., 8, 307., 130, 3504, 12. , 70, 1, b'"chevrolet chevelle malib
u"'),
       (15., 8, 350., 165, 3693, 11.5, 70, 1, b'"buick skylark 320"'),
       (18., 8, 318., 150, 3436, 11. , 70, 1, b'"plymouth satellite"')],
      dtype=[('f0', '<f8'), ('f1', '<i4'), ('f2', '<f8'), ('f3', '<i4'), ('f
4', '<i4'), ('f5', '<f8'), ('f6', '<i4'), ('f7', '<i4'), ('f8', 'S38')])
```

Entrée [ ]:

```

```

Entrée [200]:

```
np.savetxt('data.csv', data, delimiter=',')
```

Entrée [ ]:

```

```

Entrée [201]:

```
data
```

Out[201]:

```
array([[   18.,      8.,    307., ...,     70.,      1., -1000.],
       [   15.,      8.,    350., ...,     70.,      1., -1000.],
       [   18.,      8.,    318., ...,     70.,      1., -1000.],
       ...,
       [   32.,      4.,    135., ...,     82.,      1., -1000.],
       [   28.,      4.,    120., ...,     82.,      1., -1000.],
       [   31.,      4.,    119., ...,     82.,      1., -1000.]])
```

Entrée [202]:

```
np.save('data.npy', data)
```

Entrée [203]:

```
np.savez('data2.npz', data, data2)
```

Entrée [ ]:

```

```

Entrée [204]:

```
d = np.load('data.npy')
d
```

Out[204]:

```
array([[   18.,      8.,    307., ...,     70.,      1., -1000.],
       [   15.,      8.,    350., ...,     70.,      1., -1000.],
       [   18.,      8.,    318., ...,     70.,      1., -1000.],
       ...,
       [   32.,      4.,    135., ...,     82.,      1., -1000.],
       [   28.,      4.,    120., ...,     82.,      1., -1000.],
       [   31.,      4.,    119., ...,     82.,      1., -1000.]])
```

Entrée [206]:

```
d2 = np.load('data2.npz')
```

Entrée [208]:

```
d2.files
```

Out[208]:

```
['arr_0', 'arr_1']
```

Entrée [210]:

```
d2['arr_1']
```

Out[210]:

```
array([(18. , 8, 307. , 130, 3504, 12. , 70, 1, b'"chevrolet chevelle mali
bu"'),
       (15. , 8, 350. , 165, 3693, 11.5, 70, 1, b'"buick skylark 320"'),
       (18. , 8, 318. , 150, 3436, 11. , 70, 1, b'"plymouth satellite"'),
       (16. , 8, 304. , 150, 3433, 12. , 70, 1, b'"amc rebel sst"'),
       (17. , 8, 302. , 140, 3449, 10.5, 70, 1, b'"ford torino"'),
       (15. , 8, 429. , 198, 4341, 10. , 70, 1, b'"ford galaxie 500"'),
       (14. , 8, 454. , 220, 4354,  9. , 70, 1, b'"chevrolet impala"'),
       (14. , 8, 440. , 215, 4312,  8.5, 70, 1, b'"plymouth fury iii"'),
       (14. , 8, 455. , 225, 4425, 10. , 70, 1, b'"pontiac catalina"'),
       (15. , 8, 390. , 190, 3850,  8.5, 70, 1, b'"amc ambassador dpl"'),
       (15. , 8, 383. , 170, 3563, 10. , 70, 1, b'"dodge challenger se"'),
       (14. , 8, 340. , 160, 3609,  8. , 70, 1, b'"plymouth \'cuda 340"'),
       (15. , 8, 400. , 150, 3761,  9.5, 70, 1, b'"chevrolet monte carl
o"'),
       (14. , 8, 455. , 225, 3086, 10. , 70, 1, b'"buick estate wagon (s
w)"'),
       (24. , 4, 113. ,  95, 2372, 15. , 70, 3, b'"toyota corona mark i
```

Entrée [ ]:

```

```

concat with row and col wise

Entrée [211]:

```
arr1 = np.zeros([4, 4])
arr2 = np.ones([4, 4])
```

Entrée [212]:

```
arr1
```

Out[212]:

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

Entrée [213]:

```
arr2
```

Out[213]:

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

Entrée [214]:

```
#np.concatenate, np.vstack, np.r_
```

Entrée [219]:

```
np.concatenate([arr1, arr2], axis = 0)
np.vstack([arr1, arr2])
np.r_[arr1, arr2]
```

Out[219]:

```
array([[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.]])
```

Entrée [220]:

```
np.concatenate([arr1, arr2], axis = 1)
```

Out[220]:

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

Entrée [221]:

```
np.hstack([arr1, arr2])
```

Out[221]:

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

Entrée [222]:

```
np.c_[arr1, arr2]
```

Out[222]:

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

Entrée [ ]:

sort a numpy array

Entrée [223]:

```
arr = np.random.randint(1, 10, size = [10, 5])
```

Entrée [224]:

```
arr
```

Out[224]:

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

Entrée [226]:

```
np.sort(arr, axis = 0)
```

Out[226]:

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

Entrée [ ]:

```

```

Entrée [230]:

```
sorted_index = arr[:, 0].argsort()
```

Entrée [231]:

```
arr[sorted_index]
```

Out[231]:

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

Entrée [232]:

```
arr
```

Out[232]:

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

Entrée [ ]:

```

```

working with dates

Entrée [235]:

```
d = np.datetime64('2019-06-02 23:10:00')
```

Entrée [236]:

```
d
```

Out[236]:

```
numpy.datetime64('2019-06-02T23:10:00')
```

Entrée [240]:

```
d + 1000
```

Out[240]:

```
numpy.datetime64('2019-06-02T23:26:40')
```

Entrée [241]:

```
16*60+40
```

Out[241]:

```
1000
```

Entrée [243]:

```
oneday = np.timedelta64(1, 'D')
```

Entrée [244]:

```
oneday
```

Out[244]:

```
numpy.timedelta64(1,'D')
```

Entrée [245]:

```
d + oneday
```

Out[245]:

```
numpy.datetime64('2019-06-03T23:10:00')
```

Entrée [246]:

```
oneminute = np.timedelta64(1, 'm')
```

Entrée [247]:

```
d + oneminute
```

Out[247]:

```
numpy.datetime64('2019-06-02T23:11:00')
```

Entrée [ ]:

```

```

Entrée [249]:

```
dates = np.arange(np.datetime64('2019-06-02'), np.datetime64('2020-06-02'), 2)
dates
```

Out[249]:

```
array(['2019-06-02', '2019-06-04', '2019-06-06', '2019-06-08',
       '2019-06-10', '2019-06-12', '2019-06-14', '2019-06-16',
       '2019-06-18', '2019-06-20', '2019-06-22', '2019-06-24',
       '2019-06-26', '2019-06-28', '2019-06-30', '2019-07-02',
       '2019-07-04', '2019-07-06', '2019-07-08', '2019-07-10',
       '2019-07-12', '2019-07-14', '2019-07-16', '2019-07-18',
       '2019-07-20', '2019-07-22', '2019-07-24', '2019-07-26',
       '2019-07-28', '2019-07-30', '2019-08-01', '2019-08-03',
       '2019-08-05', '2019-08-07', '2019-08-09', '2019-08-11',
       '2019-08-13', '2019-08-15', '2019-08-17', '2019-08-19',
       '2019-08-21', '2019-08-23', '2019-08-25', '2019-08-27',
       '2019-08-29', '2019-08-31', '2019-09-02', '2019-09-04',
       '2019-09-06', '2019-09-08', '2019-09-10', '2019-09-12',
       '2019-09-14', '2019-09-16', '2019-09-18', '2019-09-20',
       '2019-09-22', '2019-09-24', '2019-09-26', '2019-09-28',
       '2019-09-30', '2019-10-02', '2019-10-04', '2019-10-06',
       '2019-10-08', '2019-10-10', '2019-10-12', '2019-10-14',
       '2019-10-16', '2019-10-18', '2019-10-20', '2019-10-22',
       '2019-10-24', '2019-10-26', '2019-10-28', '2019-10-30',
       '2019-11-01', '2019-11-03', '2019-11-05', '2019-11-07',
       '2019-11-09', '2019-11-11', '2019-11-13', '2019-11-15',
       '2019-11-17', '2019-11-19', '2019-11-21', '2019-11-23',
       '2019-11-25', '2019-11-27', '2019-11-29', '2019-12-01',
       '2019-12-03', '2019-12-05', '2019-12-07', '2019-12-09',
       '2019-12-11', '2019-12-13', '2019-12-15', '2019-12-17',
       '2019-12-19', '2019-12-21', '2019-12-23', '2019-12-25',
       '2019-12-27', '2019-12-29', '2019-12-31', '2020-01-02',
       '2020-01-04', '2020-01-06', '2020-01-08', '2020-01-10',
       '2020-01-12', '2020-01-14', '2020-01-16', '2020-01-18',
       '2020-01-20', '2020-01-22', '2020-01-24', '2020-01-26',
       '2020-01-28', '2020-01-30', '2020-02-01', '2020-02-03',
       '2020-02-05', '2020-02-07', '2020-02-09', '2020-02-11',
       '2020-02-13', '2020-02-15', '2020-02-17', '2020-02-19',
       '2020-02-21', '2020-02-23', '2020-02-25', '2020-02-27',
       '2020-02-29', '2020-03-02', '2020-03-04', '2020-03-06',
       '2020-03-08', '2020-03-10', '2020-03-12', '2020-03-14',
       '2020-03-16', '2020-03-18', '2020-03-20', '2020-03-22',
       '2020-03-24', '2020-03-26', '2020-03-28', '2020-03-30',
       '2020-04-01', '2020-04-03', '2020-04-05', '2020-04-07',
       '2020-04-09', '2020-04-11', '2020-04-13', '2020-04-15',
       '2020-04-17', '2020-04-19', '2020-04-21', '2020-04-23',
       '2020-04-25', '2020-04-27', '2020-04-29', '2020-05-01',
       '2020-05-03', '2020-05-05', '2020-05-07', '2020-05-09',
       '2020-05-11', '2020-05-13', '2020-05-15', '2020-05-17',
       '2020-05-19', '2020-05-21', '2020-05-23', '2020-05-25',
       '2020-05-27', '2020-05-29', '2020-05-31'], dtype='datetime64[D]')
```

Entrée [ ]:

```

```

Numpy Advanced Function

Entrée [250]:

```
#vectorize()
```

Entrée [252]:

```
def foo(x):
    if x%2 == 1:
        return x**2
    else:
        return x/2

foo(11)
```

Out[252]:

```
121
```

Entrée [ ]:

```

```

Entrée [253]:

```
foo_v = np.vectorize(foo, otypes=[float])
```

Entrée [254]:

```
arr
```

Out[254]:

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

Entrée [255]:

```
foo_v(arr)
```

Out[255]:

```
array([[ 9., 25.,  3.,  9., 25.],
       [ 9., 25.,  4.,  4.,  1.],
       [ 4.,  1., 49.,  4., 49.],
       [ 1.,  1.,  1., 81., 81.],
       [ 2., 81.,  4.,  2., 49.],
       [ 3.,  1.,  2., 25., 81.],
       [ 1., 25.,  1.,  2.,  9.],
       [ 1., 25.,  9.,  4.,  4.],
       [81., 49.,  2.,  4.,  4.],
       [25.,  3.,  2., 49., 81.]])
```

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]:

Entrée [ ]: