Numpy1

March 3, 2023

[242]:
```
import numpy as np
```

[3]:
```
l = [1,2,3,4,5,6,7,8,9]
```

Array is a data structure that stores a collection of elements of the same type. The elements are stored in contiguous memory locations and are accessed using an index or a subscript. Arrays are used to store and manipulate collections of data that can be represented as a list or a table

[5]:
```
ar = np.array(l)
```

[6]:
```
ar
```

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

[7]:
```
type(ar)
```

[7]:
```
numpy.ndarray
```

[8]:
```
np.array([[1,2],[3,4]])
```

[8]:
```
array([[1, 2],
       [3, 4]])
```

[9]:
```
np.asanyarray(l)
```

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

[10]:
```
a = [5,6,7]
```

[12]:
```
np.asanyarray(a)
```

[12]:
```
array([5, 6, 7])
```

In numpy, a matrix is represented as a two-dimensional array of numbers.

[13]:
```
b = np.matrix(l)
```

[14]:
```
b
```

[14]:
```
matrix([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

[15]:
```
np.asanyarray(b)
```

[15]:
```
matrix([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

Matrix is already a subset of array.

[16]:
```
a = np.array(l)
```

[17]:
```
a
```

[17]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In Python, a shallow copy of an object creates a new object that points to the same memory locations as the original object. This means that if any changes are made to the original object, those changes will also be reflected in the shallow copy.

[18]:
```
c = a
```

[19]:
```
c
```

[19]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[20]:
```
c[0] = 100
```

[21]:
```
c
```

[21]:
```
array([100,   2,   3,   4,   5,   6,   7,   8,   9])
```

[22]:
```
a
```

[22]:
```
array([100,   2,   3,   4,   5,   6,   7,   8,   9])
```

A deep copy, on the other hand, creates a new object with new memory addresses for all of its contents. Changes made to the original object will not affect the deep copy.

[23]:
```
d = np.copy(a)
```

[24]:
```
d
```

[24]:
```
array([100,   2,   3,   4,   5,   6,   7,   8,   9])
```

[25]:
```
a
```

[25]:
```
array([100,   2,   3,   4,   5,   6,   7,   8,   9])
```

[26]:
```
a[2] = 400
```

[27]:
```
a
```

[27]:
```
array([100,   2, 400,   4,   5,   6,   7,   8,   9])
```

[28]:
```
d
```

[28]:
```
array([100,   2,   3,   4,   5,   6,   7,   8,   9])
```

[29]:
```
np.fromfunction(lambda i, j : i==j,(3,3))
```

[29]:
```
array([[ True, False, False],
       [False,  True, False],
       [False, False,  True]])
```

[30]:
```
np.fromfunction(lambda i, j : i*j,(3,3))
```

[30]:
```
array([[0., 0., 0.],
       [0., 1., 2.],
       [0., 2., 4.]])
```

[31]:
```
iterable = (i * i for i in range(5))
```

[32]:
```
np.fromiter(iterable, float)
```

[32]:
```
array([ 0.,  1.,  4.,  9., 16.])
```

[36]:
```
np.fromstring('345 789', sep =' ')
```

[36]:
```
array([345., 789.])
```

[38]:
```
np.fromstring('5,6', sep =',')
```

[38]:
```
array([5., 6.])
```

Numpy Data Types

[39]:
```
l
```

[39]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[41]:
```
ar = np.array(l)
```

[42]:
```
ar
```

[42]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[43]:
```
ar.ndim
```

[43]: 1

[47]:
```
ar2 = np.array([[1,2,3,4],[5,6,7,8],[9,8,7,6]])
```

[48]:
```
ar2.ndim
```

[48]: 2

[49]:
```
ar.size
```

[49]: 9

[50]:
```
ar2.size
```

[50]: 12

[51]:
```
ar.shape
```

[51]: (9,)

[52]:
```
ar2.shape
```

[52]: (3, 4)

[53]:
```
ar.dtype
```

[53]: dtype('int64')

[54]:
```
ar2.dtype
```

[54]: dtype('int64')

[55]:
```
ar22 = np.array([(1.5,3,4,5),(6,7,8,9)])
```

[56]:
```
ar22
```

[56]:
```
array([[1.5, 3. , 4. , 5. ],
       [6. , 7. , 8. , 9. ]])
```

[57]:
```
ar22.dtype
```

[57]: dtype('float64')

[58]:
```
list(range(5))
```

[58]: [0, 1, 2, 3, 4]

[59]:
```
list(range(0.5,5))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[59], line 1
----> 1 list(range(0.5,5))

TypeError: 'float' object cannot be interpreted as an integer
```

[62]:
```
np.arange(0.5,5)
```

[62]:
```
array([0.5, 1.5, 2.5, 3.5, 4.5])
```

[63]:
```
np.arange(0.5,5,.2)
```

[63]:
```
array([0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9, 2.1, 2.3, 2.5, 2.7, 2.9,
       3.1, 3.3, 3.5, 3.7, 3.9, 4.1, 4.3, 4.5, 4.7, 4.9])
```

[64]:
```
list(np.arange(0.5,5,.2))
```

[64]:
```
[0.5,
 0.7,
 0.8999999999999999,
 1.0999999999999999,
 1.2999999999999998,
 1.4999999999999998,
 1.6999999999999997,
 1.8999999999999997,
 2.0999999999999996,
 2.3,
 2.4999999999999996,
 2.6999999999999993,
 2.8999999999999995,
 3.0999999999999996,
 3.2999999999999994,
 3.499999999999999,
 3.6999999999999993,
 3.8999999999999995,
 4.1,
 4.299999999999999,
 4.499999999999999,
 4.699999999999999,
 4.899999999999999]
```

[65]:
```
np.linspace(1,10,10)
```

[65]:
```
array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

[66]:
```
np.linspace(1,10,5)
```

[66]:
```
array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])
```

[67]:
```
np.zeros(5)
```

[67]:
```
array([0., 0., 0., 0., 0.])
```

[69]:
```
np.zeros((5,7))
```

[69]:
```
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., 0.],
       [0., 0., 0., 0., 0., 0., 0.]])
```

[71]:
```
np.zeros((4,5,2))
```

[71]:
```
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., 0.],
        [0., 0.]],

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

[73]:
```
ar4 = np.zeros((4,5,2,2))
```

[74]:
```
ar4
```

```
[74]: 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., 0.]],

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

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

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


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

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

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

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

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

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

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

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

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

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

[75]:
```
ar.dtype
```

[75]:
```
dtype('int64')
```

[77]:
```
np.ones((2,3))
```

[77]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

[78]:
```
on = np.ones((2,3,4))
```

[79]:
```
on
```

[79]:
```
array([[[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]],

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
```

[80]:
```
on + 5
```

[80]:
```
array([[[6., 6., 6., 6.],
        [6., 6., 6., 6.],
        [6., 6., 6., 6.]],

       [[6., 6., 6., 6.],
        [6., 6., 6., 6.],
        [6., 6., 6., 6.]]])
```

[81]:
```
on * 2
```

[81]:
```
array([[[2., 2., 2., 2.],
        [2., 2., 2., 2.],
        [2., 2., 2., 2.]],

       [[2., 2., 2., 2.],
        [2., 2., 2., 2.],
        [2., 2., 2., 2.]]])
```

[84]:
```
np.empty((3,5))
```

[84]:
```
array([[4.64829839e-310, 0.00000000e+000, 0.00000000e+000,
        0.00000000e+000, 0.00000000e+000],
       [0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
        0.00000000e+000, 0.00000000e+000],
       [0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
        0.00000000e+000, 6.32404027e-322]])
```

[85]:
```
np.eye(4)
```

[85]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

[86]:
```
np.linspace(2,20)
```

[86]:
```
array([ 2.        ,  2.36734694,  2.73469388,  3.10204082,  3.46938776,
        3.83673469,  4.20408163,  4.57142857,  4.93877551,  5.30612245,
        5.67346939,  6.04081633,  6.40816327,  6.7755102 ,  7.14285714,
        7.51020408,  7.87755102,  8.24489796,  8.6122449 ,  8.97959184,
        9.34693878,  9.71428571, 10.08163265, 10.44897959, 10.81632653,
       11.18367347, 11.55102041, 11.91836735, 12.28571429, 12.65306122,
       13.02040816, 13.3877551 , 13.75510204, 14.12244898, 14.48979592,
       14.85714286, 15.2244898 , 15.59183673, 15.95918367, 16.32653061,
       16.69387755, 17.06122449, 17.42857143, 17.79591837, 18.16326531,
       18.53061224, 18.89795918, 19.26530612, 19.63265306, 20.        ])
```

[101]:
```
np.logspace(2,5,10, base = 2)
```

[101]:
```
array([ 4.        ,  5.0396842 ,  6.34960421,  8.        , 10.0793684 ,
       12.69920842, 16.        , 20.1587368 , 25.39841683, 32.        ])
```

randn generates random numbers from a normal distribution with mean 0 and standard deviation 1. In other words, the numbers generated by randn are more likely to be closer to 0, with the probability decreasing as the distance from 0 increases.

[93]:
```
arr = np.random.randn(4,6)
```

[94]:
```
arr
```

[94]:
```
array([[-0.53688812, -0.21389285, -0.34690481, -1.42821269,  0.92075499,
        -0.1177727 ],
       [-0.24936593,  0.26390129,  0.56995875, -1.08992043, -0.19959973,
         1.78134032],
       [ 0.00782157, -0.14339131, -0.13329628,  1.4951467 ,  0.99550247,
        -0.61514573],
       [-0.9915527 , -0.93624207, -0.36077873, -0.2503121 ,  0.85293611,
         1.03630345]])
```

[95]:
```
import pandas as pd
```

[97]:
```
df = pd.DataFrame(arr)
```

[98]:
```
df
```

[98]:

	0	1	2	3	4	5
0	-0.536888	-0.213893	-0.346905	-1.428213	0.920755	-0.117773
1	-0.249366	0.263901	0.569959	-1.089920	-0.199600	1.781340
2	0.007822	-0.143391	-0.133296	1.495147	0.995502	-0.615146
3	-0.991553	-0.936242	-0.360779	-0.250312	0.852936	1.036303

rand generates random numbers uniformly distributed between 0 and 1. In other words, each number has an equal probability of being chosen.

[99]:
```
np.random.rand(4,6)
```

[99]:
```
array([[0.99797334, 0.12051452, 0.60464789, 0.99320695, 0.82882434,
        0.02354227],
       [0.2430047 , 0.77551957, 0.66307135, 0.08358323, 0.1293972 ,
        0.66476811],
       [0.30327706, 0.89280799, 0.24332848, 0.07479089, 0.54435452,
        0.0538428 ],
       [0.97048672, 0.37151432, 0.15563299, 0.6781778 , 0.45832453,
        0.34671433]])
```

[102]:
```
pd.DataFrame(np.random.randint(1,220,(400,500)))
```

[102]:

	0	1	2	3	4	5	6	7	8	9	…	490	491	492	\
0	33	116	186	33	189	109	73	197	76	93	…	38	125	7	
1	199	89	117	49	189	31	177	48	157	78	…	61	134	84	
2	112	37	151	188	165	207	52	59	135	109	…	133	203	63	
3	45	147	44	64	99	149	73	126	111	140	…	148	64	104	
4	60	114	179	139	114	89	111	74	54	152	…	212	161	45	

```
..   …  …  …  …  …  …  …  …  …  …  …  …  …  …
395  219   94  142  136   63  153  111  169  156  210  …  109  131   80
396   54   60  168  101  177   84  212   83   15  158  …  176    8   10
397  163  207  149   61  106  142  107  109   18  167  …  148   89  122
398  177    3   38   85   48    1  140  190  124  151  …   15   35   67
399  105  209  194   95   54  103  187  123  193   37  …   95  136  163

     493  494  495  496  497  498  499
0    193  153  208   46  179   94  105
1     89   46   93  163   20  212  134
2     75   46  158   67  218  205  179
3    143   66  197  120   30  118  176
4    165  217  199   12  141   49   59
..   …  …  …  …  …  …  …
395    2   19   62  170   96  107   12
396    7   97  114   40   43    6  165
397  195    8  130  125  136  175   51
398  154  141  198  180  123   18   88
399  182  206  157   90  194  113  155

[400 rows x 500 columns]
```

[103]:
```
#to same save file
pd.DataFrame(np.random.randint(1,220,(400,500))).to_csv('text.csv')
```

[104]:
```
arr = np.random.rand(4,6)
```

[105]:
```
arr
```

[105]:
```
array([[6.37922080e-01, 2.70978986e-02, 7.11086567e-01, 2.28835464e-01,
        6.40867737e-01, 9.65393687e-01],
       [7.33066527e-01, 6.75644638e-01, 3.97978512e-01, 1.18351171e-01,
        4.54314524e-01, 8.35701985e-04],
       [3.04433170e-01, 4.51185354e-01, 9.94174179e-01, 9.23079169e-01,
        4.11077309e-01, 7.81317926e-01],
       [5.06722067e-01, 7.24782698e-01, 6.41358577e-01, 4.52299758e-01,
        3.44136120e-01, 5.38114741e-01]])
```

[112]:
```
arr1 = arr.reshape(3,8)
```

[113]:
```
arr1
```

[113]:
```
array([[6.37922080e-01, 2.70978986e-02, 7.11086567e-01, 2.28835464e-01,
        6.40867737e-01, 9.65393687e-01, 7.33066527e-01, 6.75644638e-01],
       [3.97978512e-01, 1.18351171e-01, 4.54314524e-01, 8.35701985e-04,
        3.04433170e-01, 4.51185354e-01, 9.94174179e-01, 9.23079169e-01],
       [4.11077309e-01, 7.81317926e-01, 5.06722067e-01, 7.24782698e-01,
```

```
       6.41358577e-01, 4.52299758e-01, 3.44136120e-01, 5.38114741e-01]])
```

[114]:
```
arr1[1][1]
```

[114]:
```
0.11835117124042116
```

[121]:
```
arr1[2:5,5]
```

[121]:
```
array([0.45229976])
```

[138]:
```
arr = np.random.randint(1,100,(5,5))
```

[139]:
```
arr
```

[139]:
```
array([[ 7, 23, 13, 67, 80],
       [21, 45, 95, 74, 30],
       [11, 27, 27, 95, 82],
       [72, 75, 57, 49, 17],
       [61, 75, 82, 93, 62]])
```

[140]:
```
arr[arr > 50]
```

[140]:
```
array([67, 80, 95, 74, 95, 82, 72, 75, 57, 61, 75, 82, 93, 62])
```

[141]:
```
arr
```

[141]:
```
array([[ 7, 23, 13, 67, 80],
       [21, 45, 95, 74, 30],
       [11, 27, 27, 95, 82],
       [72, 75, 57, 49, 17],
       [61, 75, 82, 93, 62]])
```

[142]:
```
arr[2:4 ,[1,2] ]
```

[142]:
```
array([[27, 27],
       [75, 57]])
```

[145]:
```
arr[0][0] = 500
```

[146]:
```
arr
```

[146]:
```
array([[500,  23,  13,  67,  80],
       [ 21,  45,  95,  74,  30],
       [ 11,  27,  27,  95,  82],
       [ 72,  75,  57,  49,  17],
       [ 61,  75,  82,  93,  62]])
```

Mathematical Functions

[147]:
```
arr5 = np.random.randint(1,3,(3,3))
```

[148]:
```
arr5
```

[148]:
```
array([[1, 1, 1],
       [2, 2, 1],
       [1, 2, 2]])
```

[149]:
```
arr6 = arr5 = np.random.randint(1,3,(3,3))
```

[150]:
```
arr6
```

[150]:
```
array([[2, 2, 1],
       [2, 1, 2],
       [1, 1, 2]])
```

[151]:
```
arr5 + arr6
```

[151]:
```
array([[4, 4, 2],
       [4, 2, 4],
       [2, 2, 4]])
```

[152]:
```
arr5 * arr6
```

[152]:
```
array([[4, 4, 1],
       [4, 1, 4],
       [1, 1, 4]])
```

[153]:
```
arr5 - arr6
```

[153]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

[154]:
```
# matrix multiplication
arr5@arr6
```

[154]:
```
array([[9, 7, 8],
       [8, 7, 8],
       [6, 5, 7]])
```

[155]:
```
arr5/0
```

```
/tmp/ipykernel_135/4034504109.py:1: RuntimeWarning: divide by zero encountered
in divide
  arr5/0
```

[155]:
```
array([[inf, inf, inf],
       [inf, inf, inf],
       [inf, inf, inf]])
```

[156]:
```
arr5+100
```

[156]:
```
array([[102, 102, 101],
       [102, 101, 102],
       [101, 101, 102]])
```

[157]:
```
arr5**2
```

[157]:
```
array([[4, 4, 1],
       [4, 1, 4],
       [1, 1, 4]])
```

Numpy Broadcasting

[158]:
```
arr = np.zeros((4,4))
```

[159]:
```
arr
```

[159]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

[160]:
```
row = np.array([1,2,3,4])
```

[161]:
```
row
```

[161]:
```
array([1, 2, 3, 4])
```

[162]:
```
arr + row
```

[162]:
```
array([[1., 2., 3., 4.],
       [1., 2., 3., 4.],
       [1., 2., 3., 4.],
       [1., 2., 3., 4.]])
```

[164]:
```
col = np.array([[1,2,3,4]])
```

[165]:
```
col
```

[165]:
```
array([[1, 2, 3, 4]])
```

[166]:
```
col.T
```

[166]:
```
array([[1],
       [2],
       [3],
       [4]])
```

[167]:
```
arr + col
```

[167]:
```
array([[1., 2., 3., 4.],
       [1., 2., 3., 4.],
       [1., 2., 3., 4.],
       [1., 2., 3., 4.]])
```

[168]:
```
row + col
```

[168]:
```
array([[2, 4, 6, 8]])
```

[169]:
```
arr5
```

[169]:
```
array([[2, 2, 1],
       [2, 1, 2],
       [1, 1, 2]])
```

[170]:
```
np.sqrt(arr5)
```

[170]:
```
array([[1.41421356, 1.41421356, 1.        ],
       [1.41421356, 1.        , 1.41421356],
       [1.        , 1.        , 1.41421356]])
```

[171]:
```
np.exp(arr5)
```

[171]:
```
array([[7.3890561 , 7.3890561 , 2.71828183],
       [7.3890561 , 2.71828183, 7.3890561 ],
       [2.71828183, 2.71828183, 7.3890561 ]])
```

[172]:
```
np.log(arr5)
```

[172]:
```
array([[0.69314718, 0.69314718, 0.        ],
       [0.69314718, 0.        , 0.69314718],
       [0.        , 0.        , 0.69314718]])
```

Numpy Array Manipulation

[173]:
```
arr = np.random.randint(1,10,(3,4))
```

[174]:
```
arr
```

[174]:
```
array([[3, 3, 1, 7],
       [4, 6, 3, 7],
       [4, 4, 2, 6]])
```

[175]:
```
arr.reshape(6,2)
```

[175]:
```
array([[3, 3],
       [1, 7],
       [4, 6],
       [3, 7],
       [4, 4],
       [2, 6]])
```

[176]:
```
arr.reshape(12,1)
```

[176]:
```
array([[3],
       [3],
       [1],
       [7],
       [4],
       [6],
       [3],
       [7],
       [4],
       [4],
       [2],
       [6]])
```

[177]:
```
arr
```

[177]:
```
array([[3, 3, 1, 7],
       [4, 6, 3, 7],
       [4, 4, 2, 6]])
```

[178]:
```
arr.T
```

[178]:
```
array([[3, 4, 4],
       [3, 6, 4],
       [1, 3, 2],
       [7, 7, 6]])
```

[179]:
```
arr.flatten()
```

[179]:
```
array([3, 3, 1, 7, 4, 6, 3, 7, 4, 4, 2, 6])
```

[180]:
```
arr1 = np.array([1,2,3,4,5])
```

[181]:
```
arr1.ndim
```

[181]:
```
1
```

[182]:
```
np.expand_dims(arr1, axis = 1)
```

[182]:
```
array([[1],
       [2],
       [3],
       [4],
       [5]])
```

[183]:
```
np.expand_dims(arr1, axis = 0)
```

[183]:
```
array([[1, 2, 3, 4, 5]])
```

[184]:
```
arr
```

[184]:
```
array([[3, 3, 1, 7],
       [4, 6, 3, 7],
       [4, 4, 2, 6]])
```

[185]:
```
np.squeeze(arr)
```

[185]:
```
array([[3, 3, 1, 7],
       [4, 6, 3, 7],
       [4, 4, 2, 6]])
```

[186]:
```
data = np.array([[1],[2],[3],[4]])
```

[187]:
```
data
```

[187]:
```
array([[1],
       [2],
       [3],
       [4]])
```

[188]:
```
np.squeeze(data)
```

[188]:
```
array([1, 2, 3, 4])
```

[189]:
```
arr1
```

[189]:
```
array([1, 2, 3, 4, 5])
```

[190]:
```
np.repeat(arr1,3)
```

[190]:
```
array([1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5])
```

[191]:
```
np.roll(arr1, shift =2)
```

[191]:
```
array([4, 5, 1, 2, 3])
```

[192]:
```
np.diag(arr1)
```

[192]:
```
array([[1, 0, 0, 0, 0],
       [0, 2, 0, 0, 0],
       [0, 0, 3, 0, 0],
       [0, 0, 0, 4, 0],
       [0, 0, 0, 0, 5]])
```

Numpy Binary Operations

[218]:
```
arr4 = np.random.randint(1,10, (3,4))
```

[195]:
```
arr4
```

[195]:
```
array([[1, 5, 5, 4],
       [3, 3, 8, 5],
       [9, 2, 4, 1]])
```

[219]:
```
arr5 = np.random.randint(1,10, (3,4))
```

[197]:
```
arr5
```

[197]:
```
array([[5, 4, 1, 4],
       [7, 2, 6, 8],
       [7, 3, 2, 9]])
```

[198]:
```
arr4 + arr5
```

[198]:
```
array([[ 6,  9,  6,  8],
       [10,  5, 14, 13],
       [16,  5,  6, 10]])
```

[199]:
```
arr4/ arr5
```

[199]:
```
array([[0.2       , 1.25      , 5.        , 1.        ],
       [0.42857143, 1.5       , 1.33333333, 0.625     ],
       [1.28571429, 0.66666667, 2.        , 0.11111111]])
```

[200]:
```
arr4 * arr5
```

[200]:
```
array([[ 5, 20,  5, 16],
       [21,  6, 48, 40],
       [63,  6,  8,  9]])
```

[201]:
```
arr4%arr5
```

[201]:
```
array([[1, 1, 0, 0],
       [3, 1, 2, 5],
       [2, 2, 0, 1]])
```

[202]:
```
arr4 ** arr5
```

[202]:
```
array([[      1,     625,       5,     256],
       [   2187,       9,  262144,  390625],
       [4782969,       8,      16,       1]])
```

[203]:
```
arr4 & arr5
```

[203]:
```
array([[1, 4, 1, 4],
       [3, 2, 0, 0],
       [1, 2, 0, 1]])
```

[204]:
```
~arr4
```

[204]:
```
array([[ -2,  -6,  -6,  -5],
       [ -4,  -4,  -9,  -6],
       [-10,  -3,  -5,  -2]])
```

[205]:
```
arr4|arr5
```

[205]:
```
array([[ 5,  5,  5,  4],
       [ 7,  3, 14, 13],
       [15,  3,  6,  9]])
```

[206]:
```
arr4 > arr5
```

[206]:
```
array([[False,  True,  True, False],
       [False,  True,  True, False],
       [ True, False,  True, False]])
```

Numpy String Function

[217]:
```
arr = np.array(['chetan', "patil"])
```

[211]:
```
arr
```

[211]:
```
array(['chetan', 'patil'], dtype='<U6')
```

[213]:
```
np.char.upper(arr)
```

[213]:
```
array(['CHETAN', 'PATIL'], dtype='<U6')
```

[214]:
```
np.char.title(arr)
```

[214]:
```
array(['Chetan', 'Patil'], dtype='<U6')
```

[215]:
```
np.char.capitalize(arr)
```

[215]:
```
array(['Chetan', 'Patil'], dtype='<U6')
```

Numpy Mathematical Functions

[220]:
```
np.sin(arr4)
```

[220]:
```
array([[-0.7568025 ,  0.84147098, -0.2794155 ,  0.98935825],
       [-0.2794155 ,  0.84147098, -0.2794155 ,  0.6569866 ],
       [ 0.6569866 ,  0.14112001, -0.95892427,  0.98935825]])
```

[221]:
```
np.tan(arr4)
```

[221]:
```
array([[ 1.15782128,  1.55740772, -0.29100619, -6.79971146],
       [-0.29100619,  1.55740772, -0.29100619,  0.87144798],
       [ 0.87144798, -0.14254654, -3.38051501, -6.79971146]])
```

[222]:
```
np.cos(arr4)
```

[222]:
```
array([[-0.65364362,  0.54030231,  0.96017029, -0.14550003],
       [ 0.96017029,  0.54030231,  0.96017029,  0.75390225],
       [ 0.75390225, -0.9899925 ,  0.28366219, -0.14550003]])
```

[223]:
```
np.tanh(arr4)
```

[223]:
```
array([[0.9993293 , 0.76159416, 0.99998771, 0.99999977],
       [0.99998771, 0.76159416, 0.99998771, 0.99999834],
       [0.99999834, 0.99505475, 0.9999092 , 0.99999977]])
```

[224]:
```
np.log10(arr4)
```

[224]:
```
array([[0.60205999, 0.        , 0.77815125, 0.90308999],
       [0.77815125, 0.        , 0.77815125, 0.84509804],
       [0.84509804, 0.47712125, 0.69897   , 0.90308999]])
```

[225]:
```
np.exp(arr4)
```

[225]:
```
array([[5.45981500e+01, 2.71828183e+00, 4.03428793e+02, 2.98095799e+03],
       [4.03428793e+02, 2.71828183e+00, 4.03428793e+02, 1.09663316e+03],
       [1.09663316e+03, 2.00855369e+01, 1.48413159e+02, 2.98095799e+03]])
```

[226]:
```
np.sqrt(arr4)
```

[226]:
```
array([[2.        , 1.        , 2.44948974, 2.82842712],
       [2.44948974, 1.        , 2.44948974, 2.64575131],
       [2.64575131, 1.73205081, 2.23606798, 2.82842712]])
```

[228]:
```
np.power(arr4,2)
```

[228]:
```
array([[16,  1, 36, 64],
       [36,  1, 36, 49],
       [49,  9, 25, 64]])
```

[229]:
```
np.mean(arr4)
```

[229]:
```
5.166666666666667
```

[230]:
```
np.median(arr4)
```

[230]:
```
6.0
```

[231]:
```
np.std(arr4)
```

[231]:
```
2.3392781412696997
```

[234]:
```
np.var(arr4)
```

[234]:
```
5.472222222222221
```

[235]:
```
np.min(arr4)
```

[235]:
```
1
```

[236]:
```
np.max(arr4)
```

[236]:
```
8
```

Sort, Search and Counting Functions

[245]:
```
arr = np.array([4,6,8,9,23,45,72,498,216,945])
```

[239]:
```
arr
```

[239]:
```
array([  4,   6,   8,   9,  23,  45,  72, 498, 216, 945])
```

[240]:
```
np.sort(arr)
```

[240]:
```
array([  4,   6,   8,   9,  23,  45,  72, 216, 498, 945])
```

[241]:
```
np.searchsorted(arr, 24)
```

[241]:
```
5
```

[243]:
```
arr1 = np.array([0,3,5,0,0,0,0])
```

[244]:
```
np.count_nonzero(arr1)
```

[244]:
```
2
```

[246]:
```
arr
```

[246]:
```
array([  4,   6,   8,   9,  23,  45,  72, 498, 216, 945])
```

[248]:
```
np.where(arr >10)
```

[248]:
```
(array([4, 5, 6, 7, 8, 9]),)
```

[250]:
```
np.extract(arr >10, arr)
```

[250]:
```
array([ 23,  45,  72, 498, 216, 945])
```

Numpy Byte Swapping

[251]:
```
arr
```

[251]:
```
array([  4,   6,   8,   9,  23,  45,  72, 498, 216, 945])
```

[252]:
```
arr.byteswap()
```

[252]:
```
array([  288230376151711744,   432345564227567616,   576460752303423488,
          648518346341351424,  1657324662872342528,  3242591731706757120,
         5188146770730811392, -1008524841554280448, -2882303761517117440,
        -5691705504066174976])
```

Numpy Copies & Views

[253]:
```
arr
```

[253]:
```
array([  4,   6,   8,   9,  23,  45,  72, 498, 216, 945])
```

[254]:
```
a = np.copy(arr)
```

[255]:
```
a
```

[255]:
```
array([  4,   6,   8,   9,  23,  45,  72, 498, 216, 945])
```

[256]:
```
b = arr.view()
```

[257]:
```
b
```

[257]:
```
array([  4,   6,   8,   9,  23,  45,  72, 498, 216, 945])
```

[258]:
```
b[0]=78
```

[259]:
```
b
```

[259]:
```
array([ 78,   6,   8,   9,  23,  45,  72, 498, 216, 945])
```

[260]:
```
arr
```

[260]:
```
array([ 78,   6,   8,   9,  23,  45,  72, 498, 216, 945])
```

Numpy Matrix Library

[265]:
```
import numpy.matlib as nm
```

[266]:
```
nm.zeros(5)
```

[266]:
```
matrix([[0., 0., 0., 0., 0.]])
```

[267]:
```
nm.ones((3,4))
```

[267]:
```
matrix([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
```

[268]:
```
nm.eye(5)
```

[268]:
```
matrix([[1., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0.],
        [0., 0., 1., 0., 0.],
        [0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 1.]])
```

Numpy Linear Algebra

[269]:
```
arr1 = np.random.randint([[5,6],[8,9]])
```

[270]:
```
arr2 = np.random.randint([[6,6],[9,8]])
```

[271]:
```
arr1
```

[271]:
```
array([[4, 0],
       [5, 1]])
```

[272]:
```
arr2
```

[272]:
```
array([[4, 3],
       [6, 6]])
```

[273]:
```
np.dot(arr1, arr2)
```

[273]:
```
array([[16, 12],
       [26, 21]])
```

[274]:
```
arr1@arr2
```

[274]:
```
array([[16, 12],
       [26, 21]])
```

[ ]: