NumPy arrays

In [1]:

```
import numpy as np
```

In [2]:

```
data = np.random.randint(-10, 10, size = 9)
print(data)
```

```
[  7   4  -8   4  -3  -5 -10  -1   1]
```

In [3]:

```
print(type(data))
```

```
<class 'numpy.ndarray'>
```

In [4]:

```
print(data.shape)
```

```
(9,)
```

In [5]:

```
data_arr = np.reshape(data, (9, 1))
print(data_arr)
```

```
[[  7]
 [  4]
 [ -8]
 [  4]
 [ -3]
 [ -5]
 [-10]
 [ -1]
 [  1]]
```

In [6]:

```
print(data)
```

```
[  7   4  -8   4  -3  -5 -10  -1   1]
```

In [7]:

```
print(data_arr.shape)
```

```
(9, 1)
```

In [8]:

```
data_arr = np.reshape(data_arr, (1, 9))
print(data_arr)
```

```
[[  7   4  -8   4  -3  -5 -10  -1   1]]
```

In [9]:

```
print(data_arr.shape)
```

```
(1, 9)
```

In [10]:

```
print(data)
```

```
[  7   4  -8   4  -3  -5 -10  -1   1]
```

In [11]:

```
data_arr = np.reshape(data_arr, (3, 3))
print(data_arr)
print(data_arr.shape)
```

```
[[  7   4  -8]
 [  4  -3  -5]
 [-10  -1   1]]
(3, 3)
```

In [12]:

```
data_broken = np.reshape(data, (4, 4))
```

```
---------------------------------------------------------------------------
ValueError                                Trace
(most recent call last)
<ipython-input-12-17429e0c5040> in <module>
----> 1 data_broken = np.reshape(data, (4, 4))

/usr/local/lib/python3.7/site-packages/numpy/co
omnumeric.py in reshape(a, newshape, order)
    290                [5, 6]])
    291     """
--> 292     return _wrapfunc(a, 'reshape', news
order=order)
    293 
    294 

/usr/local/lib/python3.7/site-packages/numpy/co
omnumeric.py in _wrapfunc(obj, method, *args, *
)
     54 def _wrapfunc(obj, method, *args, **kwd
     55     try:
---> 56         return getattr(obj, method)(*ar
*kwds)
     57 
     58     # An AttributeError occurs if the o
does not have

ValueError: cannot reshape array of size 9 into
e (4,4)
```

In [13]:

```
data_broken = np.reshape(data, (2, 2))
```

```
-----------------------------------------------
```

```
-----------------------
ValueError                                Trace
(most recent call last)
<ipython-input-13-13c1dfcc8614> in <module>
----> 1 data_broken = np.reshape(data, (2, 2))

/usr/local/lib/python3.7/site-packages/numpy/co
omnumeric.py in reshape(a, newshape, order)
    290                 [5, 6]])
    291     """
--> 292     return _wrapfunc(a, 'reshape', news
order=order)
    293
    294

/usr/local/lib/python3.7/site-packages/numpy/co
omnumeric.py in _wrapfunc(obj, method, *args, *
)
     54 def _wrapfunc(obj, method, *args, **kwd
     55     try:
---> 56         return getattr(obj, method)(*ar
*kwds)
     57
     58     # An AttributeError occurs if the o
does not have
```

```
ValueError: cannot reshape array of size 9 into
e (2,2)
```

In [14]:

```
data_arr.shape
```

```
(3, 3)
```

In [15]:

```
print(data_arr)
```

```
[[  7   4  -8]
 [  4  -3  -5]
 [-10  -1   1]]
```

In [16]:

```
column_mean = np.mean(data_arr, axis = 0)
print(column_mean)
```

```
[ 0.33333333  0.         -4.        ]
```

In [17]:

```
print(column_mean.shape)
```

```
(3,)
```

In [18]:

```
row_mean = np.mean(data_arr, axis = 1)
print(row_mean)
```

```
[ 1.         -1.33333333 -3.33333333]
```

In [19]:

```
column_std = np.std(data_arr, axis = 0)
print (column_std)
```

```
[7.40870359 2.94392029 3.74165739]
```

In [21]:

```
column_std = np.var(data_arr, axis = 0)
```

In [22]:

```
print(column_std)
```

```
[54.88888889  8.66666667 14.        ]
```

Operations on arrays

In [23]:

```
print(data_arr)
```

```
[[  7   4  -8]
 [  4  -3  -5]
 [-10  -1   1]]
```

In [24]:

```
new_data = np.random.randint(-5, 5, size = (3, 3
print(new_data)
```

```
[[ 0  3 -5]
 [ 0  3 -1]
 [-3 -5  2]]
```

In [25]:

```
print(data_arr + new_data)
```

```
[[  7   7 -13]
 [  4   0  -6]
 [-13  -6   3]]
```

In [26]:

```
print(data_arr - new_data)
```

```
[[ 7  1 -3]
 [ 4 -6 -4]
 [-7  4 -1]]
```

In [27]:

```
print(data_arr * new_data)
```

```
[[ 0 12 40]
 [ 0 -9  5]
 [30  5  2]]
```

In [28]:

```
print(np.dot(data_arr, new_data))
```

```
[[ 24  73 -55]
 [ 15  28 -27]
 [ -3 -38  53]]
```

In [29]:

```
print(data_arr ** 3)
```

```
[[  343    64  -512]
 [   64   -27  -125]
 [-1000    -1     1]]
```

In [30]:

```
print(data_arr + 1)
```

```
[[ 8  5 -7]
 [ 5 -2 -4]
 [-9  0  2]]
```

In [32]:

```
print(data_arr)
```

```
[[  7   4  -8]
 [  4  -3  -5]
 [-10  -1   1]]
```

In [33]:

```
print(data_arr * 2)
```

```
[[ 14   8 -16]
 [  8  -6 -10]
 [-20  -2   2]]
```

In [34]:

```
print(column_mean)
```

```
[ 0.33333333  0.         -4.        ]
```

In [35]:

```
column_mean = np.reshape(column_mean, (1, 3))
print(column_mean)
```

```
[[ 0.33333333  0.         -4.        ]]
```

In [36]:

```
print(data_arr - column_mean)
```

```
[[  6.66666667   4.          -4.        ]
 [  3.66666667  -3.          -1.        ]
 [-10.33333333  -1.           5.        ]]
```

In [37]:

```python
print(row_mean)
```

```
[ 1.         -1.33333333 -3.33333333]
```

In [38]:

```python
row_mean = np.reshape(row_mean, (3, 1))
```

In [39]:

```python
print(data_arr - row_mean)
```

```
[[ 6.          3.         -9.        ]
 [ 5.33333333 -1.66666667 -3.66666667]
 [-6.66666667  2.33333333  4.33333333]]
```

NumPy Matrices

In [46]:

```
print(data_arr)
print(new_data)
```

```
[[  7   4  -8]
 [  4  -3  -5]
 [-10  -1   1]]
[[ 0  3 -5]
 [ 0  3 -1]
 [-3 -5  2]]
```

In [47]:

```
print(np.dot(data_arr, new_data))
```

```
[[ 24  73 -55]
 [ 15  28 -27]
 [ -3 -38  53]]
```

In [48]:

```
mtx1 = np.matrix(data_arr)
mtx2 = np.matrix(new_data)
```

In [49]:

```
print(mtx1)
```

```
[[  7   4  -8]
 [  4  -3  -5]
 [-10  -1   1]]
```

In [50]:

```
print(type(mtx1))
```

```
<class 'numpy.matrix'>
```

In [51]:

```
print(mtx1 * mtx2)
```

```
[[ 24  73 -55]
 [ 15  28 -27]
 [ -3 -38  53]]
```

In [52]:

```
mul = mtx1 * mtx2
```

In [53]:

```
print(type(mul))
```

```
<class 'numpy.matrix'>
```

In [54]:

```
print(mul.T)
```

```
[[ 24  15  -3]
 [ 73  28 -38]
 [-55 -27  53]]
```

In [55]:

```
print(mul.I)
```

```
[[-0.03180556  0.12354167  0.02993056]
 [ 0.04958333 -0.076875    0.01229167]
 [ 0.03375    -0.048125    0.029375  ]]
```

In [56]:

```
print((mtx1 * mtx2).T.I)
```

```
[[-0.03180556  0.04958333  0.03375    ]
 [ 0.12354167 -0.076875   -0.048125   ]
 [ 0.02993056  0.01229167  0.029375   ]]
```

In [ ]: