In [1]:

```
!pwd
!ls -lh .
!python -V

%matplotlib inline

import sys
print(sys.version_info)

import numpy as np
print(np.__version__)
```

```
/notebooks/deeplearning.ai
total 4.0M
-rw-r--r--  1 root root  12K Aug 25 01:27 00-numpy-python3.ipynb
-rw-r--r--  1 root root  39K Aug 24 23:58 01-basics-numpy-python3.ipynb
-rw-r--r--  1 root root  27K Aug 25 01:28 02-quiz.ipynb
-rw-r--r--  1 root root 263K Aug 27 02:40 03-logistic-regression-python3.ipynb
-rw-r--r--  1 root root 556K Aug 25 08:50 04-classification.ipynb
-rw-r--r--  1 root root  65K Aug 25 08:07 05-deep-neural-network.ipynb
-rw-r--r--  1 root root 1.9M Aug 25 08:50 06-deep-neural-network.ipynb
drwxr-xr-x  5 root root  170 Aug 23 05:01 __pycache__
-rw-r--r--  1 root root 256K Aug 13 21:16 deep-learning-notations.pdf
drwxr-xr-x 44 root root 1.5K Aug 27 02:41 images
-rw-r--r--  1 root root 410K Aug 13 20:07 neural-network.pdf
-rw-r--r--  1 root root 443K Aug 13 20:15 supervised-learning.pdf
Python 2.7.12
sys.version_info(major=3, minor=5, micro=2, releaselevel='final', serial=0)
1.13.1
```

In [4]:

```
a = np.random.randn(5)
print(a) # rank 1 array <=== do not use!
print(a.shape)
print(a.T)
print(np.dot(a,a.T))

a = a.reshape((5,1))
print(a)
```

```
[-1.78689489  1.08261494  0.75955829 -0.14071355  0.28050073]
(5,)
[-1.78689489  1.08261494  0.75955829 -0.14071355  0.28050073]
5.04045824246
[[-1.78689489]
 [ 1.08261494]
 [ 0.75955829]
 [-0.14071355]
 [ 0.28050073]]
```

In [5]:

```
# use as column vector
a = np.random.randn(5,1)
assert(a.shape == (5,1))
print(a)
print(a.T)
print(np.dot(a,a.T))
```

```
[[-0.91507937]
 [-0.15836048]
 [ 0.74201373]
 [ 0.9058739 ]
 [ 1.12059087]]
[[-0.91507937 -0.15836048  0.74201373  0.9058739   1.12059087]]
[[ 0.83737024  0.14491241 -0.67900145 -0.82894652 -1.02542958]
 [ 0.14491241  0.02507804 -0.11750565 -0.14345463 -0.17745731]
 [-0.67900145 -0.11750565  0.55058438  0.67217088  0.83149381]
 [-0.82894652 -0.14345463  0.67217088  0.82060753  1.01511403]
 [-1.02542958 -0.17745731  0.83149381  1.01511403  1.2557239 ]]
```

In [6]:

```
print(np.array([1,2,3]))
print(np.asmatrix(np.array([1,2,3])))

# create an empty 2 x 2 matrix
print(np.empty([2,2], int))
```

```
[1 2 3]
[[1 2 3]]
[[1 1]
 [1 0]]
```

In [7]:

```
X = np.array([[1,2,3],[4,5,6]], np.int32)
print(X)
print(np.empty_like(X)) # Return a new array with the same shape and type as a given array.
print(np.zeros_like(X))
print(np.ones_like(X))
```

```
[[1 2 3]
 [4 5 6]]
[[         0 1072168960          0]
 [1072168960          0          0]]
[[0 0 0]
 [0 0 0]]
[[1 1 1]
 [1 1 1]]
```

In [8]:

```
print(np.diag(np.array([[0,1,2],[3,4,5],[6,7,8]])))
print(np.eye(3))
print(np.identity(3))
print(np.ones([3,2], float))
print(np.zeros([3,2], float))
print(np.full([2,2], 6, dtype=np.uint) * 5)
print(np.linspace(1., 10., 10)) # evenly spaced elements between 1. and 10.
print(np.logspace(1., 10., 10, endpoint=False))
```

```
[0 4 8]
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
[[ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]]
[[ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]]
[[30 30]
 [30 30]]
[  1.   2.   3.   4.   5.   6.   7.   8.   9.  10.]
[  1.00000000e+01   7.94328235e+01   6.30957344e+02   5.01187234e+03
   3.98107171e+04   3.16227766e+05   2.51188643e+06   1.99526231e+07
   1.58489319e+08   1.25892541e+09]
```

In [9]:

```
print(np.arange(5, dtype=np.int64))
print(np.arange(2,5,1))
print(np.arange(0,10).reshape(2,5))
print(np.arange(0,10).reshape(2,5).transpose())
print(np.ones([2, 2, 2]))
print(np.reshape(np.ones([2, 2, 2]), [-1, 2]))
```

```
[0 1 2 3 4]
[2 3 4]
[[0 1 2 3 4]
 [5 6 7 8 9]]
[[0 5]
 [1 6]
 [2 7]
 [3 8]
 [4 9]]
[[[ 1.  1.]
  [ 1.  1.]]

 [[ 1.  1.]
  [ 1.  1.]]]
[[ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]]
```

In [10]:

```
x = np.array([[1,2], [3,4]])
y = np.array([[1,2], [1,2]])
print(x.flatten())
print(y.flatten())
print(np.add(x, y))
print(np.multiply(x, y))
print(x)
print(y)
print(np.vdot(x, y)) # 1*1 + 2*2 + 3*1 + 4*2
```

```
[1 2 3 4]
[1 2 1 2]
[[2 4]
 [4 6]]
[[1 4]
 [3 8]]
[[1 2]
 [3 4]]
[[1 2]
 [1 2]]
16
```

In [11]:

```
# https://www.coursera.org/learn/neural-networks-deep-learning/lecture/NYnog/vectorization
import time

a = np.random.rand(1000000)
b = np.random.rand(1000000)

tic = time.time()
c = np.dot(a,b)
toc = time.time()
print(c)
print("Vectorized:" + str(1000*(toc-tic)) + "ms")

c = 0
tic = time.time()
for i in range(1000000):
    c += a[i]*b[i]
toc = time.time()
print(c)
print("For loop:" + str(1000*(toc-tic)) + "ms")
```

```
249887.776565
Vectorized:2.285480499267578ms
249887.776565
For loop:415.19737243652344ms
```

In [12]:

```
print(np.exp(np.array([1,2,3])))
print(np.log(np.array([1,2,3])))
```

```
[  2.71828183   7.3890561   20.08553692]
[ 0.          0.69314718  1.09861229]
```

In [13]:

```
# broadcast
print(np.array([[1,2,3],[4,5,6]]) + np.array([100,200,300]))
print(np.array([[1,2,3],[4,5,6]]) + np.array([100,200]).reshape(2,1))
```

```
[[101 202 303]
 [104 205 306]]
[[101 102 103]
 [204 205 206]]
```

In [14]:

```
A = np.array([[56.0,    0.0,   4.4,  68.0],
              [ 1.2,  104.0,  42.0,   8.0],
              [ 1.8,  135.0,  99.0,   0.9]])

print(A)
```

```
[[  56.     0.     4.4   68. ]
 [   1.2  104.    42.     8. ]
 [   1.8  135.    99.     0.9]]
```

In [15]:

```
# sum vertically (horizontal axis=1)
cal = A.sum(axis=0)

print(cal)
print(cal.reshape(1,4))
```

```
[  59.   239.   145.4   76.9]
[[  59.   239.   145.4   76.9]]
```

In [16]:

```
# broadcast
# cal is already 1x4 matrix, but reshape to be clear
# 3x4 / 1x4
pct = 100*A/cal.reshape(1,4)
print(pct)
```

```
[[ 94.91525424   0.           3.0261348   88.42652796]
 [  2.03389831  43.51464435  28.88583219  10.40312094]
 [  3.05084746  56.48535565  68.08803301   1.17035111]]
```