Numpy Tutorials

Author: Jiawen Yan

Version 1.0

Date: 2018-04-22

In [1]:
```
# https://docs.scipy.org/doc/numpy-1.14.2/user/quickstart.html
import numpy as np
```

In [19]:
```
# np.array([]) 向量
# reshape 函数将向量或矩阵变形成指定形状
a = np.array([1,2,3,4,5,6,7,9,0,3,4,4]).reshape(2,3,-1) # reshape()
```

In [22]:
```
print(a)
print(a.shape) # m*n*k 矩阵a的形状
print(a.ndim)  # dimension 矩阵a的维度
print(a.size)  # number of elements 矩阵a中元素的数量
print(a.dtype) # numpy.int32, numpy.int16, and numpy.float64 矩阵A的数据类型
```

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

 [[7 9]
  [0 3]
  [4 4]]]
(2, 3, 2)
3
12
int64
```

Create a numpy array

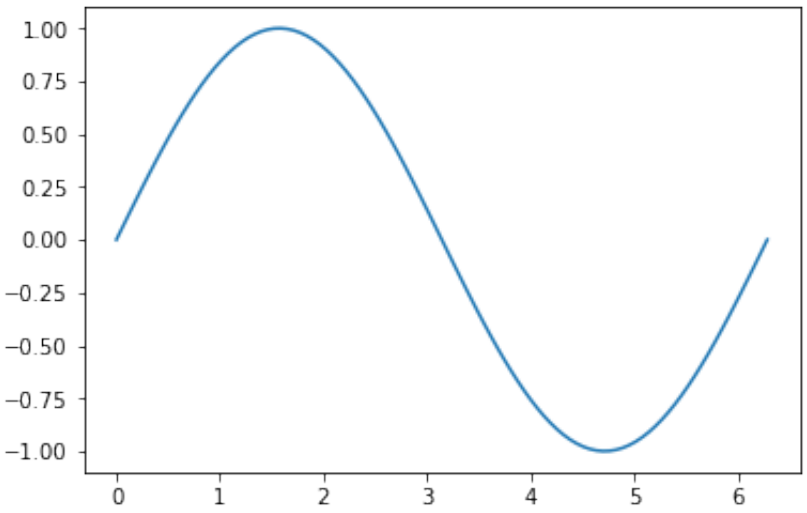
In [5]:
```
# 创建一个指定形状，每个元素都为0的矩阵
a = np.zeros( (3,4) )
print(a)
# 创建一个指定形状，每个元素都为1的矩阵
a = np.ones( (3,4) )
print(a)
# 创建一个指定形状，每个元素都为[0,1]随机数的矩阵
a = np.random.random( (3,4) )
print(a)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
[[1.41923866e-02 7.97421598e-04 2.68786558e-01 1.44278683e-01]
 [4.45507969e-01 5.35468580e-01 2.28216012e-01 9.07474433e-01]
 [1.46790833e-01 1.64885757e-02 5.07247449e-01 5.82221024e-01]]
[-0.26827145  0.58457802 -0.75477255 -0.27584747  0.54427174 -0.69
007269
 -0.42479335  0.66946597  0.81423216  0.60971879]
```

In [2]:
```
# 考虑如何生成(a, b) 的随机数?
# 伸缩变换+平移变换 a*random() + b
new_a = 5*np.random.random( (3,4) )+3 # 生成 形状为3*4， 每个元素随机分布在（3，8）中的矩阵
print(new_a)
#如果要求变形为整数?
new_a = new_a.astype(int)
print(new_a)
```

```
[[3.58412397 6.11366195 7.87930814 6.13918572]
 [3.16746377 3.0699855  3.36801692 4.23453586]
 [3.21413586 5.99163117 7.4290251  4.18977011]]
[[3 6 7 6]
 [3 3 3 4]
 [3 5 7 4]]
```

In [42]:
```
from numpy import pi # pi = 3.1415926...
import matplotlib.pyplot as plt
# linspace (a, b, num) 为线性空间函数，在 (a,b)的范围内，均匀选取 num个数字
x = np.linspace( 0, 2*pi, 100 ) # 100 numbers from 0 to 2*pi
y = np.sin(x)
plt.plot(x, y) # plot 是折线图，但是微分的思想，看起来类似于曲线图
plt.show()
```



array operations

In [43]:
```
a = np.array( [20,30,40,50] )
b = np.arange(4)
```

```
[20 30 40 50]
[0 1 2 3]
[20 29 38 47]
```

In [47]:
```
# minus
print(a-b)
print(b*2)
print(b**2)
print(b<2)
```

```
[20 29 38 47]
[0 2 4 6]
[0 1 4 9]
[ True  True False False]
```

In [57]:
```
# product
a = np.array([ [1,2], [4,5]]) # a 2-by-2 matrix
b = np.array([ [3,4], [2,3]]) # a 2-by-2 matrix
print(a)
print(b)
print(a*b) # element wise product,  a,b 两个矩阵每一个元素对应相乘，要求a,b 必须有相同的形状
print(a.dot(b)) # dot product 线性代数中的内积（点乘）
print(np.dot(a,b)) # dot product 线性代数中的内积（点乘）两种写法都是可以的
```

```
[[1 2]
 [4 5]]
[[3 4]
 [2 3]]
[[ 3  8]
 [ 8 15]]
[[ 7 10]
 [22 31]]
[[ 7 10]
 [22 31]]
```

In [7]:
```
a = np.random.random( (3,4) )
print(a)
print(np.exp(a)) # 每个元素e^x
print(np.sqrt(a))# 每个元素开平方
a = np.random.normal(0, 0.1, (3,4)) # 正态分布 normal(mu, sigma, array-shape)
print(a)
a = np.random.uniform(low=-1, high =1, size=(3,4)) # 随机分布
print(a)
```

```
[[0.92333618 0.27509202 0.25532298 0.55025102]
 [0.04944378 0.16157017 0.14629418 0.94684432]
 [0.18536094 0.96704214 0.75736187 0.98601488]]
[[2.51767582 1.31665183 1.29087848 1.73368815]
 [1.05068652 1.17535493 1.15753667 2.57756285]
 [1.2036528  2.63015332 2.1326426  2.68053092]]
[[0.96090384 0.52449216 0.50529494 0.74178906]
 [0.22235957 0.40195792 0.38248423 0.97305926]
 [0.43053564 0.98338301 0.8702654  0.99298282]]
[[ 0.08729897  0.2893323  -0.08692005 -0.06323378]
 [ 0.20540578 -0.02785456 -0.07925581  0.05698712]
 [ 0.04542889 -0.03253632 -0.11260597  0.12601933]]
[[ 0.83373929 -0.25346685 -0.48787024  0.24466814]
 [-0.64786638  0.45680377 -0.63412777  0.83859353]
 [ 0.10336068 -0.81555691  0.40049565 -0.58252311]]
```

Summary and Statistics

In [5]:
```
a = np.random.random( (3,4) )
print(a)
```

```
[[0.84308465 0.75695801 0.44886365 0.17495366]
 [0.52862907 0.48862976 0.63847202 0.88668085]
 [0.42007231 0.99692862 0.62720988 0.30248421]]
```

In [6]:
```
print(a.max()) # 一个矩阵中的最大值
print(a.min()) # 一个矩阵中的最小值
print(a.mean()) # 一个矩阵中的平均值
print(a.std()) # 一个矩阵中的标准差

# by axis: axis = 0 -> by col; axis = 1 -> by row
print(a.max(axis=0)) # 一个矩阵中每一列的最大值
print(a.min(axis=0)) # 一个矩阵中每一列的最小值
print(a.mean(axis=0)) # 一个矩阵中每一列的平均值
print(a.std(axis=1)) # 一个矩阵中每一行的标准差
```

```
0.9969286229264294
0.17495366243182153
0.5927472244821163
0.2357517736304155
[0.84308465 0.99692862 0.63847202 0.88668085]
[0.42007231 0.48862976 0.44886365 0.17495366]
[0.59726201 0.74750546 0.57151519 0.45470624]
[0.26432755 0.15499425 0.26385222]
```

In [7]:
```
# numpy copy 对矩阵进行拷贝
a = np.array([ [3,4,1], [2,7,5], [5,1,3] ])
print(a)
b = a # a赋值给b，只是将a的“地址” 给了b，两者指向同一个矩阵。如果对b进行操作，a矩阵也会发生变化
```

```
[[3 4 1]
 [2 7 5]
 [5 1 3]]
```

In [8]:
```
b is a  # 赋值后，a对象 和 b对象 是一个东西，所以 a is b == True
```

Out[8]:
```
True
```

In [9]:
```
b == a # a 和 b 中的每个元素也相同
```

Out[9]:
```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

In [11]:
```
b[0][0] = 999 # 这时我们对b 矩阵进行修改
```

In [13]:
```
a # 发现 a 矩阵也会被修改
```

Out[13]:
```
array([[999,   4,   1],
       [  2,   7,   5],
       [  5,   1,   3]])
```

In [14]:
```
c = a.copy() # 重新开辟一份内存，把 a 中的元素都完全复制给c，这使我们一般意义上的拷贝
```

In [15]:
```
c[0][0] = -8
```

In [16]:
```
c
```

Out[16]:
```
array([[-8,  4,  1],
       [ 2,  7,  5],
       [ 5,  1,  3]])
```

In [17]:
```
a
```

Out[17]:
```
array([[999,   4,   1],
       [  2,   7,   5],
       [  5,   1,   3]])
```