numpy简单用法总结

一个用于科学计算的库，便于创建多维数组对象并提供了很多科学计算函数。

常用基础函数简介

1. 创建数组对象：`n=numpy.array(list)`
2. 获取对象形状（行数，列数）：`n.shape`
3. 获取元素数据类型：`n.dtype`
4. 获取对象维度：`n.ndim`
5. 创建range数组对象：

```
>>> import numpy as np
>>> np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

改变数组形状

1. reshape函数,参数详解：

```
n.reshape(tuple)
```

```
>>> n = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> n.shape
(3, 3)
>>> n = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
>>> n.shape
(4, 3)
>>> n.reshape(2,3,2)  # 将数组形状变为(2,3,2)
array([[[ 1,  2],
        [ 3,  4],
        [ 5,  6]],

       [[ 7,  8],
        [ 9, 10],
        [11, 12]]])
>>> n
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])  # 这里打印发现n未变化，说明reshape函数的返回值并不是原数组，而是另外一个新数组。
>>> n.shape
(4, 3)
    >>> n.reshape(2,-1,2)  # 这里参数中使用了-1，表示这个位置的维度空出来，由函数根据元素个数和其他维度来计算得出。
array([[[ 1,  2],
        [ 3,  4],
        [ 5,  6]],

       [[ 7,  8],
        [ 9, 10],
```

```
        [11, 12]]])  # 可以看到这里形状还是变成了(2,3,2)
```

2.

功能之矢量化运算

```
>>> a=np.array(range(10))
>>> b=np.array(range(10))
>>> print(a+b)
[ 0  2  4  6  8 10 12 14 16 18]
>>> print(a*b)
[ 0  1  4  9 16 25 36 49 64 81]
```

即高等数学中的向量（矩阵）运算。

注意：==矩阵相乘的规则是（1,1）位置=a矩阵第一行元素与b矩阵第一列元素的积之和，又称笛卡尔积，然而这里的乘仅仅是对应位置元素相乘，并不适用于矩阵乘法。==

功能之花式索引 Fancy indexing

如下所示：

```
>>> c=[i*a for i in range(10)]
>>> c=np.array(c)
>>> print(c)
[[ 0  0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8  9]
 [ 0  2  4  6  8 10 12 14 16 18]
 [ 0  3  6  9 12 15 18 21 24 27]
 [ 0  4  8 12 16 20 24 28 32 36]
 [ 0  5 10 15 20 25 30 35 40 45]
 [ 0  6 12 18 24 30 36 42 48 54]
 [ 0  7 14 21 28 35 42 49 56 63]
 [ 0  8 16 24 32 40 48 56 64 72]
 [ 0  9 18 27 36 45 54 63 72 81]]
 >>> print(c[[3,2]])
[[ 0  3  6  9 12 15 18 21 24 27]
 [ 0  2  4  6  8 10 12 14 16 18]]
>>> print(c[3,2])
6
```

即对矩阵a进行遍历或取值时加一层中括号a[m,n]表示取出==下标==为m的行（即第m+1行）中==下标==为n的元素，加两层中括号a[[m,n]]表示取出第m,n行元素组成新矩阵。可以进行切片操作，如下所示：

```
# 打印第（1+1）行
>>> print(c[1])
[0 1 2 3 4 5 6 7 8 9]
# 将列表切片，切出下标为(0:1)行即第1行
>>> print(c[:1])
[[0 0 0 0 0 0 0 0 0 0]]
# 将列表切片，切出下标为(0:2)行即第1行第二行
>>> print(c[:2])
[[0 0 0 0 0 0 0 0 0 0]
 [0 1 2 3 4 5 6 7 8 9]]
```

```
# 取第一行元素并切片
>>> print(c[0:2,1])
[0 1]
# 分别取(1,5),(2,2),(5,1),(3,4)位置的元素
>>> print(c[[1,2,5,3],[5,2,1,4]])
[ 5  4  5 12]
# 取[1,2,5,3]行并对列进行切片操作
>>> print(c[[1,2,5,3]])
[[ 0  1  2  3  4  5  6  7  8  9]
 [ 0  2  4  6  8 10 12 14 16 18]
 [ 0  5 10 15 20 25 30 35 40 45]
 [ 0  3  6  9 12 15 18 21 24 27]]
>>> print(c[[1,2,5,3]][:,0:5])
[[ 0  1  2  3  4]
 [ 0  2  4  6  8]
 [ 0  5 10 15 20]
 [ 0  3  6  9 12]]
# 取[1,2,5,3]行再取[3,3]列
>>> print(c[[1,2,5,3]][:,[3,3]])
[[ 3  3]
 [ 6  6]
 [15 15]
 [ 9  9]]
```

一些数学方面的应用

常用数学运算

```
# 生成随机数矩阵
>>> a=np.random.randn(4,4)
>>> print(a)
[[-0.9955816   1.25785268 -0.23840713  1.51680747]
 [ 0.6785716  -1.71382548  0.72625388 -1.51999554]
 [-1.85061495  0.65180261  0.85115619  1.02627664]
 [ 0.80240893 -0.97497829 -1.42018605  0.73221539]]

# 求平均值
>>> print(a.mean())
-0.02939022830093141

# 求和
>>> print(a.sum())
-0.47024365281490255

# 求标准差
>>> print(a.std())
1.1436627070159053

# 返回自然常数e的幂次方
>>> a=np.array([0,1,2,1])
>>> np.exp(a)
array([1.        , 2.71828183, 7.3890561 , 2.71828183])

# 返回每个元素的开方
>>> a=np.array([2,5,3,4,9])
>>> np.sqrt(a)
array([1.41421356, 2.23606798, 1.73205081, 2.        , 3.        ])
```

```
# 求矩阵a与b笛卡尔积，即矩阵乘法a*b
>>> b=np.random.randn(4,4)
>>> print(a.dot(b))
[[-0.99094067 -1.27108874  1.80369754  1.04415964]
 [ 0.93837969  0.71852828 -1.99157078 -2.10084864]
 [-1.59684577 -2.41748927  2.22634146 -2.32705004]
 [ 2.08818752 -0.13605088 -3.65382844  4.27860794]]
```

argsort()函数，元素排序

argsort函数是常用的数组排序函数。
代码示例：

```
>>> a=np.array([2,5,3,4,9])
>>> np.argsort(a)
array([0, 2, 3, 1, 4], dtype=int64)   # 这里返回的是将a从小到大排序后元素的索引值
>>> np.argsort(-a)
array([4, 1, 3, 2, 0], dtype=int64)   # 这里变成了从大到小排序
```

官方文档

中文官方文档