Python编程库
--Numpy

目 录

CONTENTS

Python编程库(包)

- Numpy http://www.numpy.org
- Scipy http://www.scipy.org
- Matplotlib http://matplotlib.org
- Scikit-learn http://scikit-learn.org
- Pandas http://pandas.pydata.org
- TensorFlow https://www.tensorflow.org/
- ……

Numpy简介

■使用Numpy，开发人员可以执行以下操作：

➢数组的算术和逻辑运算。

➢傅立叶变换和用于图形操作的例程。

➢与线性代数(矩阵)有关的操作。

■Numpy 通常与 **SciPy**（Scientific Python）和 **Matplotlib**（绘图库）一起使用，这种组合广泛用于替代 MATLAB。 NumPy除了提供一些高级的数学运算机制以外，还具备非常高效的向量和矩阵运算功能。

■Numpy是开源的，这是其额外的优势。

- http://www.numpy.org
- https://www.numpy.org.cn

Numpy举例

- 例子

np.arange([start,]stop, [step,]dtype=None)

arange函数用于创建等差数组

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
```

reshape（）是数组对象中的方法，用于改变数组的形状

数组的轴（维度）的个数

```
>>> a.dtype.name
'int64'
>>> a.itemsize
8
>>> a.size
15
>>> type(a)
<type 'numpy.ndarray'>
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<type 'numpy.ndarray'>
```

描述数组中元素类型的对象

数组中每个元素的字节大小。

数组元素的总数。

array函数从常规Python列表或元组中创建数组

Numpy基础

- 数组的创建

```
>>> a = np.array(1,2,3,4)    # WRONG
>>> a = np.array([1,2,3,4])  # RIGHT
```

一个常见的错误在于使用多个数值参数调用 `array` 函数，

```
>>> b = np.array([(1.5,2,3), (4,5,6)])
>>> b
array([[ 1.5,  2. ,  3. ],
       [ 4. ,  5. ,  6. ]])
```

`array` 将序列的序列转换成二维数组

Numpy基础

- 有初始占位符内容的数组

```
>>> np.zeros( (3,4) )
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

函数 `zeros` 创建一个由0组成的数组

Numpy基础

- 有初始占位符内容的数组

```
>>> np.ones( (2,3,4), dtype=np.int16 )
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]]], dtype=int16)
```

函数 `ones` 创建一个由1数组的数组

Numpy基础

- 有初始占位符内容的数组

```
>>> np.empty( (2,3) )                                  # uninitialize
array([[  3.73603959e-262,   6.02658058e-154,   6.55490914e-260],
       [  5.30498948e-313,   3.14673309e-307,   1.00000000e+000]])
```

函数 `empty` 内容是 随机的并且取决于存储器的状态。

Numpy基础

np.linspace主要用来创建等差数列。

```
>>> np.linspace( 0, 2, 9 )                 # 9 numbers from 0 to 2
array([ 0.  ,  0.25,  0.5 ,  0.75,  1.  ,  1.25,  1.5 ,  1.75,  2.  ])
```

numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)

start:返回样本数据开始点
stop:返回样本数据结束点
num:生成的样本数据量，默认为50
endpoint：True则包含stop；False则不包含stop
retstep：If True, return (samples, step), where step is the spacing between samples.(即如果为True则结果会给出数据间隔)
dtype：输出数组类型

Numpy基础

- 基本操作

```
>>> a = np.array( [20,30,40,50] )
>>> b = np.arange( 4 )
>>> b
array([0, 1, 2, 3])
>>> c = a-b
>>> c
array([20, 29, 38, 47])
>>> b**2
array([0, 1, 4, 9])
>>> 10*np.sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])
>>> a<35
array([ True, True, False, False])
```

Numpy基础

- 矩阵*乘

```
>>> A = np.array( [[1,1],
...               [0,1]] )
>>> B = np.array( [[2,0],
...               [3,4]] )
>>> A*B
array([[2, 0],
       [0, 4]])
```

星乘表示矩阵内各对应位置相乘

Numpy基础

- 矩阵点乘

```
>>> A.dot(B)
array([[5, 4],
       [3, 4]])
>>> np.dot(A, B)
array([[5, 4],
       [3, 4]])
```

点乘表示求矩阵内积

Numpy基础

- 基本操作+=/*=

```
>>> a = np.ones((2,3), dtype=int)
>>> b = np.random.random((2,3))
>>> a *= 3
>>> a
array([[3, 3, 3],
       [3, 3, 3]])
>>> b += a
>>> b
array([[ 3.417022  ,  3.72032449,  3.00011437],
       [ 3.30233257,  3.14675589,  3.09233859]])
```

np.random.random()

通过size参数来指定维数生成[0,1)之间的浮点数

Numpy基础

- 总和、最小、最大

```
>>> a = np.random.random((2,3))
>>> a
array([[ 0.18626021,  0.34556073,  0.39676747],
       [ 0.53881673,  0.41919451,  0.6852195 ]])
>>> a.sum()
2.5718191614547998
>>> a.min()
0.1862602113776709
>>> a.max()
0.6852195003967595
```

Numpy基础

`axis` 参数，你可以沿着数组的指定轴应用操作

```
>>> b = np.arange(12).reshape(3,4)
>>> b
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>>
>>> b.sum(axis=0)                          # sum of each column
array([12, 15, 18, 21])
>>>
>>> b.min(axis=1)                          # min of each row
array([0, 4, 8])
```

Numpy基础

- 索引、切片、迭代

```
>>> a = np.arange(10)**3
>>> a
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729])
>>> a[2]
8
>>> a[2:5]
array([ 8, 27, 64])
>>> a[:6:2] = -1000    # equivalent to a[0:6:2] = -1000; from start to positic
>>> a
array([-1000,     1, -1000,    27, -1000,   125,   216,   343,   512,   729])
>>> a[ : :-1]                                 # reversed a
array([  729,   512,   343,   216,   125, -1000,    27, -1000,     1, -1000])
```

Numpy基础

- 索引、切片、迭代

```
>>> b
array([[ 0,  1,  2,  3],
       [10, 11, 12, 13],
       [20, 21, 22, 23],
       [30, 31, 32, 33],
       [40, 41, 42, 43]])
>>> b[2,3]
23
>>> b[0:5, 1]
array([ 1, 11, 21, 31, 41])
>>> b[ : ,1]
array([ 1, 11, 21, 31, 41])
>>> b[1:3, : ]
array([[10, 11, 12, 13],
       [20, 21, 22, 23]])
```

```
>>> b[-1]
array([40, 41, 42, 43])
```

Numpy形状操作

- 更改数组的形状

```
>>> a = np.floor(10*np.random.random((3,4)))
>>> a
array([[ 2.,  8.,  0.,  6.],
       [ 4.,  5.,  1.,  1.],
       [ 8.,  9.,  3.,  6.]])
>>> a.shape
(3, 4)
```

np.floor 返回不大于输入参数的最大整数。

```
>>> a.ravel()  # returns the array, flattened
array([ 2.,  8.,  0.,  6.,  4.,  5.,  1.,  1.,  8.,  9.,  3.,  6.])
```

将多维数组转换为一维数组

Numpy形状操作

- 更改数组的形状

```
>>> a.reshape(6,2)  # returns the array with a modified shape
array([[ 2.,  8.],
       [ 0.,  6.],
       [ 4.,  5.],
       [ 1.,  1.],
       [ 8.,  9.],
       [ 3.,  6.]])
>>> a.T  # returns the array, transposed
array([[ 2.,  4.,  8.],
       [ 8.,  5.,  9.],
       [ 0.,  1.,  3.],
       [ 6.,  1.,  6.]])
>>> a.T.shape
(4, 3)
>>> a.shape
(3, 4)
```

Numpy形状操作

```
>>> a
array([[ 2.,  8.,  0.,  6.],
       [ 4.,  5.,  1.,  1.],
       [ 8.,  9.,  3.,  6.]])
>>> a.resize((2,6))
>>> a
array([[ 2.,  8.,  0.,  6.,  4.,  5.],
       [ 1.,  1.,  8.,  9.,  3.,  6.]])
```

`reshape` 函数返回具有修改形状的参数，而 `ndarray.resize` 方法修改数组本身

Numpy形状操作

```
>>> a.reshape(3,-1)
array([[ 2.,  8.,  0.,  6.],
       [ 4.,  5.,  1.,  1.],
       [ 8.,  9.,  3.,  6.]])
```

在reshape操作中将维度指定为-1，则会自动计算其他维度

Numpy形状操作

- 数组堆叠

```
>>> a = np.floor(10*np.random.random((2,2)))
>>> a
array([[ 8.,  8.],
       [ 0.,  0.]])
>>> b = np.floor(10*np.random.random((2,2)))
>>> b
array([[ 1.,  8.],
       [ 0.,  4.]])
>>> np.vstack((a,b))
array([[ 8.,  8.],
       [ 0.,  0.],
       [ 1.,  8.],
       [ 0.,  4.]])
>>> np.hstack((a,b))
array([[ 8.,  8.,  1.,  8.],
       [ 0.,  0.,  0.,  4.]])
```

Numpy形状操作

- 数组拆分

```
>>> a = np.floor(10*np.random.random((2,12)))
>>> a
array([[ 9.,  5.,  6.,  3.,  6.,  8.,  0.,  7.,  9.,  7.,  2.,  7.],
       [ 1.,  4.,  9.,  2.,  2.,  1.,  0.,  6.,  2.,  2.,  4.,  0.]])
>>> np.hsplit(a,3)   # Split a into 3
[array([[ 9.,  5.,  6.,  3.],
       [ 1.,  4.,  9.,  2.]]), array([[ 6.,  8.,  0.,  7.],
       [ 2.,  1.,  0.,  6.]]), array([[ 9.,  7.,  2.,  7.],
       [ 2.,  2.,  4.,  0.]])]
```

`hsplit`，可以沿其水平轴拆分数组

`vsplit` 沿纵轴分割

小 结

CONTENTS

Numpy简介

Numpy基础

Numpy形状操作

作业2

人工智能学院
School of Artificial Intelligence

在python命令行中完成以下6题，并将作业2提交职教云平台

1、创建从0到9的一维数组

2、创建一个3×3的二维数组，值域为0到8

3、将一维数组np.arange(10)转换为2行的二维数组

4、将数组a = np.arange(10).reshape(2,-1)和数组b = np.repeat(1, 10).reshape(2,-1)垂直堆叠

5、将数组a = np.arange(10).reshape(2,-1)和数组b = np.repeat(1, 10).reshape(2,-1)水平堆叠

6、**找到二维数组**np.arange(9).reshape(3,3)**每一行中的最大值**