1. Numpy简易入门

NumPy 的全称是“ **Numeric Python**”，它是 Python 的第三方扩展包，主要用来计算、处理一维或多维数组。

- NumPy 是 Python 科学计算基础库；
- NumPy 可以对数组进行高效的数学运算；
- NumPy 的 ndarray 对象可以用来构建多维数组；
- NumPy 能够执行傅立叶变换与重塑多维数组形状；
- NumPy 提供了线性代数，以及随机数生成的内置函数。

NumPy 通常与 SciPy（Python科学计算库）和 Matplotlib（Python绘图库）等软件包组合使用，这种组合方式被用来广泛地代替 MatLab 的使用。

本节代码来自: **黄海广-机器学习** https://github.com/fengdu78/WZU-machine-learning-course 推荐自学

菜鸟Numpy教程： https://www.runoob.com/numpy/numpy-tutorial.html

C语言中文网教程： http://c.biancheng.net/numpy/what-is-numpy.html

标准的Python中用list（列表）保存值，可以当做数组使用，但因为列表中的元素可以是任何对象，所以浪费了CPU运算时间和内存。

NumPy诞生为了弥补这些缺陷。它提供了两种基本的对象：

ndarray：全称（n-dimensional array object）是储存单一数据类型的多维数组。

ufunc：全称（universal function object）它是一种能够对数组进行处理的函数。

NumPy的官方文档： https://docs.scipy.org/doc/numpy/reference/

CMD命令行中安装：`python -m pip install numpy`

安装后导入这个库`import numpy as np`

NumPy 定义了一个 n 维数组对象，简称 ndarray 对象，它是一个一系列相同类型元素组成的数组集合。
数组中的每个元素都占有大小相同的内存块，您可以使用索引或切片的方式获取数组中的每个元素

1.1 创建NumPy数组

In [1]:
```
import numpy as np
```

In [2]:
```
data1 = np.array([1, 2, 3])                    # 创建一个一维数组
data1
```

Out[2]:
```
array([1, 2, 3])
```

In [3]:
```
data2 = np.array([[1, 2, 3], [4, 5, 6]])   # 创建一个二维数组
data2
```

Out[3]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [4]:
```
np.zeros((3, 4))#创建一个全0数组
```

Out[4]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [5]:
```
np.ones((3, 4))#创建全一数组
```

Out[5]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [6]:
```
np.empty((5, 2))# 创建全空数组，其实每个值都是接近于零的数
```

Out[6]:
```
array([[ 9.70176468e-312,  1.77863633e-322],
       [ 0.00000000e+000,  0.00000000e+000],
       [ 0.00000000e+000,  2.67818390e+184],
       [ 1.74254523e-076,  5.25159449e-090],
       [ 1.42106696e+161, -2.33593625e-310]])
```

In [7]:
```
np.arange(1, 20, 5)
```

Out[7]:
```
array([ 1,  6, 11, 16])
```

In [8]:
```
np.ones((2, 3), dtype='float64')
```

Out[8]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

1.2 ndarray数组对象类型

In [9]:
```
data = np.arange(12).reshape(3, 4)  # 创建一个3行4列的数组
data
```

Out[9]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [10]:
```
type(data)
```

Out[10]:
```
numpy.ndarray
```

In [11]:
```
data.ndim          # 数组维度的个数，输出结果2，表示二维数组
```

Out[11]:
```
2
```

In [12]:
```
data.shape         # 数组的维度，输出结果（3，4），表示3行4列
```

Out[12]:
```
(3, 4)
```

In [13]:
```
data.size          # 数组元素的个数，输出结果12，表示总共有12个元素
```

Out[13]:
```
12
```

In [14]:
```
data.dtype # 数组元素的类型，输出结果dtype('int32'),表示元素类型都是int32
```

Out[14]:
```
dtype('int32')
```

转换数据类型

In [15]:
```
float_data = data.astype(np.float64) # 数据类型转换为float64
```

```
float_data,float_data.dtype
```

Out[15]:
```
(array([[ 0.,  1.,  2.,  3.],
        [ 4.,  5.,  6.,  7.],
        [ 8.,  9., 10., 11.]]),
 dtype('float64'))
```

In [16]:
```
float_data = np.array([1.2, 2.3, 3.5])
float_data
```

Out[16]:
```
array([1.2, 2.3, 3.5])
```

In [17]:
```
int_data = float_data.astype(np.int64) # 数据类型转换为int64
int_data
```

Out[17]:
```
array([1, 2, 3], dtype=int64)
```

In [18]:
```
str_data = np.array(['1', '2', '3'])
str_data
```

Out[18]:
```
array(['1', '2', '3'], dtype='<U1')
```

In [19]:
```
int_data = str_data.astype(np.int64)
int_data
```

Out[19]:
```
array([1, 2, 3], dtype=int64)
```

1.3 数组运算

1.3.1 向量化运算

In [20]:
```
import numpy as np
```

In [21]:
```
data1 = np.array([[1, 2, 3], [4, 5, 6]])
data2 = np.array([[1, 2, 3], [4, 5, 6]])
```

In [22]:
```
data1 + data2        # 数组相加
```

Out[22]:
```
array([[ 2,  4,  6],
       [ 8, 10, 12]])
```

In [23]:
```
data1 - data2        # 数组相减
```

Out[23]:
```
array([[0, 0, 0],
       [0, 0, 0]])
```

In [24]:
```
data1 * data2        # 数组相乘
```

Out[24]:
```
array([[ 1,  4,  9],
       [16, 25, 36]])
```

In [25]:
```
data1 / data2       # 数组相除
```

Out[25]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

1.3.2 数组广播

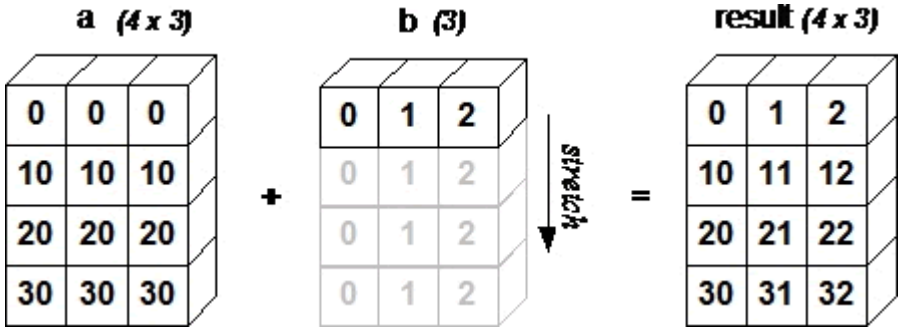
numpy数组间的基础运算是一对一，也就是`a.shape==b.shape`，

但是当两者不一样的时候，就会自动触发广播机制，如下例子：

In [26]:
```
import numpy as np
a = np.array([[ 0, 0, 0],
              [10,10,10],
              [20,20,20],
              [30,30,30]])
#b数组与a数组形状不同
b = np.array([1,2,3])
```

In [27]:
```
a+b
```

Out[27]:
```
array([[ 1,  2,  3],
       [11, 12, 13],
       [21, 22, 23],
       [31, 32, 33]])
```



1.3.3 数组与标量间的运算

In [28]:
```
data1 = np.array([[1, 2, 3], [4, 5, 6]])
data2 = 10
```

In [29]:
```
data1+data2
```

Out[29]:
```
array([[11, 12, 13],
       [14, 15, 16]])
```

In [30]:
```
data1 + data2      # 数组相加
```

Out[30]:
```
array([[11, 12, 13],
       [14, 15, 16]])
```

In [31]:
```
data1 - data2        # 数组相减
```

Out[31]:
```
array([[-9, -8, -7],
       [-6, -5, -4]])
```

In [32]:
```
data1 * data2       # 数组相乘
```

Out[32]:
```
array([[10, 20, 30],
       [40, 50, 60]])
```

In [33]:
```
data1 / data2       # 数组相除
```

Out[33]:
```
array([[0.1, 0.2, 0.3],
       [0.4, 0.5, 0.6]])
```

1.4 ndarray的索引和切片

ndarray的索引和切片与Python中可迭代对象（list,tuple,string）用法大致类似。

In [34]:
```
arr = np.arange(8) # 创建一个一维数组
arr[1],arr[2:5],arr[1:6:2] #   获取索引为1~6的元素，步长为2
```

Out[34]:
```
(1, array([2, 3, 4]), array([1, 3, 5]))
```

In [35]:
```
arr2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]]) # 创建二维数组
```

In [36]:
```
arr2d[1],arr2d[1][1]
```

Out[36]:
```
(array([4, 5, 6]), 5)
```

In [37]:
```
arr2d[:2],arr2d[1,:2]
```

Out[37]:
```
(array([[1, 2, 3],
        [4, 5, 6]]),
 array([4, 5]))
```

1.5 NumPy通用函数

In [38]:
```
arr = np.array([4, 9, 16])
```

In [39]:
```
np.sqrt(arr)#开方
```

Out[39]:
```
array([2., 3., 4.])
```

In [40]:
```
np.abs(arr)#求绝对值
```

Out[40]:
```
array([ 4,  9, 16])
```

In [41]:
```
np.square(arr)#求平方
```

Out[41]:
```
array([ 16,  81, 256])
```

In [42]:
```
x = np.array([12, 9, 13, 15])
```

In [43]:
```
y = np.array([11, 10, 4, 8])
```

In [44]:
```
# np.add(x, y) np.multiply(x, y)
```

In [45]:
```
np.maximum(x, y)  # 两个数组元素级最大值的比较
```

Out[45]:
```
array([12, 10, 13, 15])
```

In [46]:
```
np.greater(x, y)  # 执行元素级的比较操作
```

Out[46]:
```
array([ True, False,  True,  True])
```

1.6 利用NumPy数组进行数据处理

In [47]:
```
arr = np.arange(1,10)
arr
```

Out[47]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [48]:
```
arr.sum()      # 求和
```

Out[48]:
```
45
```

In [49]:
```
arr.mean()     # 求平均值
```

Out[49]:
```
5.0
```

In [50]:
```
arr.min()      # 求最小值
```

Out[50]:
```
1
```

In [51]:
```
arr.max()       # 求最大值
```

Out[51]:
```
9
```

In [52]:
```
arr.argmin()   # 求最小值的索引
```

Out[52]:
```
0
```

In [53]:
```
arr.cumsum()   # 计算元素的累计和
```

Out[53]:
```
array([ 1,  3,  6, 10, 15, 21, 28, 36, 45])
```

In [54]:
```
arr.cumprod()  # 计算元素的累计积
```

Out[54]:
```
array([     1,      2,      6,     24,    120,    720,   5040,  40320,
       362880])
```

1.7 随机数模块

In [55]:
```
import numpy as np
np.random.rand(3, 3)     # 随机生成一个二维数组
```

Out[55]:
```
array([[0.69576812, 0.58002339, 0.8111664 ],
       [0.64067189, 0.68919313, 0.70145569],
       [0.5967098 , 0.91630882, 0.76594956]])
```

In [56]:
```
np.random.rand(2, 3, 3) # 随机生成一个三维数组
```

Out[56]:
```
array([[[0.06026788, 0.35696925, 0.9643501 ],
        [0.73402283, 0.07446614, 0.76484565],
        [0.95269283, 0.42213954, 0.50329953]],

       [[0.89926545, 0.50240875, 0.27924173],
        [0.19635104, 0.66497443, 0.17619637],
        [0.03066481, 0.80543011, 0.41399586]]])
```

In [57]:
```
np.random.seed(0)   # 生成随机数的种子
```

In [58]:
```
np.random.rand(5)   # 随机生成包含5个元素的浮点数组
```

Out[58]:
```
array([0.5488135 , 0.71518937, 0.60276338, 0.54488318, 0.4236548 ])
```

In [59]:
```
np.random.seed()
```

In [60]:
```
np.random.rand(5)
```

Out[60]:
```
array([0.72612318, 0.07898037, 0.1804478 , 0.85119978, 0.8472926 ])
```