4.Python Numpy模块

本课目标

- Numpy模块中的矩阵数据类型
- Numpy模块中的运算库
- 随机数生成

课程目录

Course catalogue

1 Numpy模块中的数组对象 ndarray

2 Numpy模块中的运算库

3 随机数生成

4 利用NumPy进行统计分析

NumPy(Numerical Python)

NumPy：目前Python 科学计算中最重要的基础包

- 快速高效的多维数组对象 ndarray。
- 对数据进行元素级的矩阵计算，而无需编写循环程序。
- 读写硬盘上基于数组的数据集的工具。
- 线性代数运算、傅里叶变换，以及随机数生成的功能。
- 将 C、C++、Fortran 代码集成到 Python 的工具。

NumPy数组对象

NumPy数组方法与Python列表性能对比

In [121]:
```
import time
import numpy as np
my_arr  =  np.arange(1000000)
my_list = list(range(1000000))
```

In [119]:
```
%timeit for _ in range(10):my_arr2 = my_arr*2
%timeit for _ in range(10):my_list2 = my_list*2
```

NumPy数组对象

1．数组创建：

```
#导入NumPy包
import numpy as np

#通过Python列表创建数组：
arr1 = np.array([1,4,2,5,3])
arr2 = np.array([[1,2,3],[4,5,6]])

#生成随机数组
arr = np.random.randn(2,3)
```

N维数组对象--ndarray（数组）是存储单一数据类型的多维数组；NumPy的核心特征之一。

Ndarray是Python中一个快速、灵活的大型数据集容器。

最简单方式用array()函数创建；array函数接受任意的序列型对象。

NumPy数组对象

1．数组创建：

```
#创建一个长度为10 的数组，数组的值都是0
np.zeros(10,dtype=int)
#创建一个3*5的浮点型数组，值都为1
np.ones((3,5),dtype=float)
#创建一个线性序列组，从0开始，到20结束，步长为2
np.arange(0,20,2)
#创建一个5个元素的数组，5个数均匀分配在0-1
np.linspace(0,1,5)
#创建一个3*3，在0-1之间均匀分布的随机数数组
np.random.random((3,3))
#创建3*3的单位阵
np.eye(3)
```

数组允许用类似于标量的操作语法在整块数据上进行数学计算。

arange()是Python内建函数range的数组版

NumPy数组对象

2．数组属性:

属性	说明
ndim	返回 int。表示数组的维数
shape	返回 tuple。表示数组的尺寸，对于 n 行 m 列的矩阵，形状为(n,m)
size	返回 int。表示数组的元素总数，等于数组形状的乘积
dtype	返回 data-type。描述数组中元素的类型
itemsize	返回 int。表示数组的每个元素的大小（以字节为单位）。

NumPy数组对象

2．数组属性:

```
import numpy as np
#设置随机数种子
np.random.seed(0)
#一维数组
x1 = np.random.randint(10,size=6)
#二维数组
x2 = np.random.randint(10,size=(3,4))
#三维数组
x3 = np.random.randint(10,size=(3,4,5))
print("x3 ndim:",x3.ndim)
print("x1 shape:",x1.shape)
print("x2 shape:",x2.shape)
print("x3 shape:",x3.shape)
print("x3 size:",x3.size)
print("x3 dtype:",x3.dtype)
print("x3 itemsize:",x3.itemsize)
```

NumPy数组对象

3．数组数据类型：

NumPy基本数据类型与其取值范围（只展示一部分）

类型	描述
bool	用一位存储的布尔类型（值为TRUE或FALSE）
inti	由所在平台决定其精度的整数（一般为int32或int64）
int8	整数，范围为−128至127
int16	整数，范围为−32768至32767
int32	整数，范围为-2^{31}至$2^{32}-1$
……	……

NumPy数组对象

4．数组的索引与切片(一维）：

In[29]:	arr = np.arange(10) print('索引结果为：',arr[5]) #用整数作为下标可以获取数组中的某个元素
Out[29]:	索引结果为： 5
In[30]:	print('索引结果为：',arr[3:5]) #用范围作为下标获取数组的一个切片，包括arr[3]不包括arr[5]
Out[30]:	索引结果为： [3 4]
In[31]:	print('索引结果为：',arr[:5]) #省略开始下标，表示从arr[0]开始
Out[31]:	索引结果为： [0 1 2 3 4]
In[32]:	print('索引结果为：',arr[-1]) #下标可以使用负数，-1表示从数组后往前数的第一个元素
Out[32]:	索引结果为： 9

一维数组索引和列表相似

数组的切片是原数组的视图，不是复制，任何对于视图的修改都会反映到原数组上。

这种设计适合于处理非常大的数组，减少内存

NumPy数组对象

4．数组的索引与切片(多维)：

In[36]:	arr = np.array([[1, 2, 3, 4, 5],[4, 5, 6, 7, 8], [7, 8, 9, 10, 11]]) print('创建的二维数组为：',arr)
Out[36]:	创建的二维数组为： [[1 2 3 4 5] [4 5 6 7 8] [7 8 9 10 11]]
In[37]:	print('索引结果为：',arr[0,3:5]) #索引第0行中第3和4列的元素
Out[37]:	索引结果为： [4 5]

二维数组中，每个索引值对应的元素不是一个值，而是一个一维数组。

NumPy数组对象

4．数组的索引与切片(多维)：

```
In[38]: #索引第2和3行中第3 ~ 5列的元素
        print('索引结果为：',arr[1:,2:])
```

```
Out[38]: 索引结果为： [[ 6  7  8]
          [ 9 10 11]]
```

```
In[39]: print('索引结果为：',arr[:,2])  #索引第2列的元素
```

```
Out[39]: 索引结果为： [3 6 9]
```

二维数组中，每个索引值对应的元素不是一个值，而是一个一维数组。

NumPy数组对象

5．改变数组形状：重新设置数组的 shape 属性，ravel()展平数组

```
data = np.arange(10)
print(data.shape)
print(data)
data1 = data.reshape(2,5)
#data.shape = 2,5
print(data1)
print(data1.shape)
print(data1.ravel())
```

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

shape的返回值是一个元组，里面每个数字表示每一维的长度

NumPy数组对象

6．数组组合/拼接

```
import numpy as np

arr1 = np.arange(12).reshape(3,4)
print("arr1:\n",arr1)
arr2 = arr1*3
print("arr2:\n",arr2)
print("hstack:\n",np.hstack([arr1,arr2]))
print("vstack:\n",np.vstack((arr1,arr2)))
print("concatenate:\n",np.concatenate((arr1,arr2),axis=1))
```

使用hstack函数实现数组横向组合：

np.hstack((arr1,arr2))

使用vstack函数实现数组纵向组合：

np.vstack((arr1,arr2))

使用concatenate函数实现数组横向组合：

np.concatenate((arr1,arr2),axis = 1))

使用concatenate函数实现数组纵向组合：

np.concatenate((arr1,arr2),axis = 0))

Concatenate函数将数组元组或者数组列表作为第一个参数。

NumPy数组对象

7．数组分裂

```
import numpy as np

grid = np.arange(16).reshape(4,4)
print("grid:\n",grid)

upper,lower = np.vsplit(grid,[2])
print("upper:\n",upper)
print("lower:\n",lower)

left,right = np.hsplit(grid,[2])
print("left:\n",left)
print("right:\n",right)
```

使用hsplit函数实现数组横向分割：

np.hsplit(arr1, 2)

使用vsplit函数实现数组纵向分割：

np.vsplit(arr, 2)

使用split函数实现数组横向分割：

np.split(arr, 2, axis=1)

使用split函数实现数组纵向分割：

np.split(arr, 2, axis=0)

课程目录

Course catalogue

1 Numpy模块中的数组对象 ndarray

2 Numpy模块中的通用函数

3 随机数生成

4 利用NumPy进行统计分析

Numpy的通用函数

NumPy提供的两个基本对象： （1）N维数组 ndarray （2）通用函数对象（ufunc: universal function）

通用函数（ufunc）：目的：对NumPy数组中的值执行更快的重复操作。

通用函数是一种能够在ndarray数据中进行元素级处理的函数，即对数组的操作将会被用于数组中的每个元素（向量操作）。

- 四则运算：加（+）、减（-）、乘（*）、除（/）、幂（**）。数组间的四则运算表示对每个数组中的元素分别进行四则运算，所以形状必须相同。
- 比较运算：>、<、==、>=、<=、!=。比较运算返回的结果是一个布尔数组，每个元素为每个数组对应元素的比较结果。
- 逻辑运算：np.any函数表示逻辑“or”，np.all函数表示逻辑“and”。运算结果返回布尔值。

Numpy的通用函数

通用函数:

```
import numpy as np

arr = np.arange(6).reshape(2,3)
print("arr:\n",arr)

print("arr+5:\n",arr+5)
print("arr**2:\n",arr**2)

print("arr sqrt:\n",np.sqrt(arr))
print("arr exp:\n:",np.exp(arr))
```

```
import timeit
import numpy as np

big_array = np.random.rand(1000000)
%timeit sum(big_array)
%timeit np.sum(big_array)
```

Numpy的通用函数

ufunc函数的广播机制

广播（broadcasting）是指不同形状的数组之间执行算术运算的方式。需要遵循4个原则。

- 让所有输入数组都向其中shape最长的数组看齐，shape中不足的部分都通过在前面加1补齐。
- 输出数组的shape是输入数组shape的各个轴上的最大值。
- 如果输入数组的某个轴和输出数组的对应轴的长度相同或者其长度为1时，这个数组能够用来计算，否则出错。
- 当输入数组的某个轴的长度为1时，沿着此轴运算时都用此轴上的第一组值。

Numpy的通用函数

ufunc函数的广播机制

- 一维数组的广播机制

$$\begin{matrix}[0\ 0\ 0]\\ [1\ 1\ 1]\\ [2\ 2\ 2]\\ [3\ 3\ 3]\end{matrix} + [1\ 2\ 3] \rightarrow \begin{matrix}[0\ 0\ 0]\\ [1\ 1\ 1]\\ [2\ 2\ 2]\\ [3\ 3\ 3]\end{matrix} + \begin{matrix}[1\ 2\ 3]\\ [1\ 2\ 3]\\ [1\ 2\ 3]\\ [1\ 2\ 3]\end{matrix} = \begin{matrix}[1\ 2\ 3]\\ [2\ 3\ 4]\\ [3\ 4\ 5]\\ [4\ 5\ 6]\end{matrix}$$

课程目录

Course catalogue

1 Numpy模块中的数组对象 ndarray

2 Numpy模块中的通用函数

3 随机数生成

4 利用NumPy进行统计分析

随机数生成

- 无约束条件下生成随机数

```
In[25]:  print('生成的随机数组为：',np.random.random(100))
```

```
Out[25]: 生成的随机数组为： [ 0.15343184 0.51581585
0.07228451 ... 0.24418316
0.92510545 0.57507965]
```

- 生成服从均匀分布的随机数

```
In[26]:  print('生成的随机数组为：\n',np.random.rand(10,5))
```

```
Out[26]: 生成的随机数组为：
[[ 0.39830491 0.94011394 0.59974923 0.44453894
0.65451838]
...
[ 0.1468544 0.82972989 0.58011115 0.45157667
0.32422895]]
```

随机数生成

- 生成服从正态分布的随机数

```
In[27]:  print('生成的随机数组为：\n',np.random.randn(10,5))
```

```
Out[27]: 生成的随机数组为：
 [[-0.60571968  0.39034908 -1.63315513  0.02783885 -
1.84139301]
...,
 [-0.27500487  1.41711262  0.6635967   0.35486644 -
0.26700703]]
```

- 根据给定的由低到高的范围生成随机样本，
  如创建一个最小值不低于 2、最大值不高于 10 的 2 行 5 列数组

```
In[28]:  print('生成的随机数组为：',np.random.randint(2,10,size
= [2,5]))
```

```
Out[28]: 生成的随机数组为：  [[6 6 6 6 8]
[9 6 6 8 4]]
```

随机数生成

- random模块常用随机数生成函数

函数	说明
seed	确定随机数生成器的种子。
permutation	返回一个序列的随机排列或返回一个随机排列的范围。
rand	从均匀分布中生成样本
normal	产生正态（高斯）分布的随机数。
randn	从均值0方差1的正态分布抽取样本
randint	根据给定的由到高的范围抽取随机样本
beta	产生beta分布的随机数。
gamma	产生gamma分布的随机数。
uniform	产生在[0,1)中均匀分布的随机数。

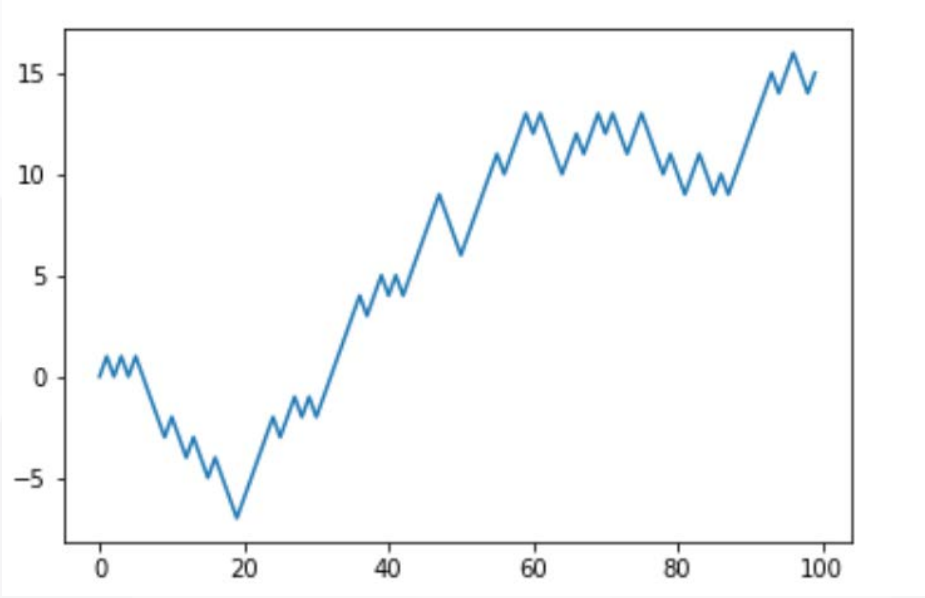
随机数生成

- 示例：简单随机漫步

```
import matplotlib.pyplot as plt
import numpy as np
import random
position = 0
walk = [position]
steps = 1000
for i in range(steps):
    step = 1 if random.randint(0,1) else -1
    position += step
    walk.append(position)
plt.plot(walk[:100])
```

简单的随机漫步：从0开始，步进为1和-1，且两种步进发生概率相等。

对前100步进行可视化



课程目录

Course catalogue

1/ Numpy模块中的数组对象 ndarray

2/ Numpy模块中的通用函数

3/ 随机数生成

4/ 利用NumPy进行统计分析

利用NumPy进行统计分析

NumPy文件读写主要有二进制的文件读写和文件列表形式的数据读写两种形式

- save函数是以二进制的格式保存数据。 np.save("../tmp/save_arr",arr)
- load函数是从二进制的文件中读取数据。 np.load("../tmp/save_arr.npy")
- savez函数可以将多个数组保存到一个文件中。 np.savez('../tmp/savez_arr',arr1,arr2)
- 存储时可以省略扩展名，但读取时不能省略扩展名。

利用NumPy进行统计分析

直接排序

- sort函数是最常用的排序方法。 arr.sort()
- sort函数也可以指定一个axis参数，使得sort函数可以沿着指定轴对数据集进行排序。axis=1为沿横轴排序； axis=0为沿纵轴排序。

```
arr = np.random.randint(1,10,size=10)
print(arr)
arr.sort()
print(arr)
arr = np.random.randint(1,10,size=(3,3))
print(arr)
#沿着横轴排序
arr.sort(axis=1)
print(arr)
```

利用NumPy进行统计分析

间接排序

- argsort函数返回值为重新排序值的下标。 arr.argsort()
- lexsort函数返回值是按照最后一个传入数据排序的。 np.lexsort((a,b,c))

```
arr = np.array([2,3,6,8,0,7])
print(arr)
print("argsort:",arr.argsort())
```

```
[2 3 6 8 0 7]
argsort: [4 0 1 2 5 3]
```

利用NumPy进行统计分析

去重与重复数据

- 去重：通过unique函数可以找出数组中的唯一值并返回已排序的结果。
- 重复数据：
  - tile函数主要有两个参数，参数“A”指定重复的数组，参数“reps”指定重复的次数。np.tile(A，reps)
  - repeat函数主要有三个参数，参数“a”是需要重复的数组元素，参数“repeats”是重复次数，参数“axis”指定沿着哪个轴进行重复，axis = 0表示按行进行元素重复；axis = 1表示按列进行元素重复。

    numpy.repeat(a, repeats, axis=None)
- 这两个函数的主要区别在于，tile函数是对数组进行重复操作，repeat函数是对数组中的每个元素进行重复操作。

利用NumPy进行统计分析

去重与重复数据

```
arr = np.arange(5)
print("重复后数组:",np.tile(arr,3))

arr = np.random.randint(0,10,size=(3,3))
print(arr)
#按照行进行元素重复
print("重复后: ",arr.repeat(2,axis=0))
```

利用NumPy进行统计分析

常用的统计函数

- 当axis=0时，表示沿着纵轴计算。当axis=1时，表示沿着横轴计算。默认时计算一个总值。

函数	说明
sum	计算数组的和
mean	计算数组均值
std	计算数组标准差
var	计算数组方差
min	计算数组最小值
max	计算数组最大值
argmin	返回数组最小元素的索引
argmax	返回数组最小元素的索引
cumsum	计算所有元素的累计和
cumprod	计算所有元素的累计积