1.NumPy概述

NumPy(Numerical Python)是用Python进行科学计算的基础软件包。包含以下特点：

1. 强大的N维数组对象Array
2. 成熟的函数库
3. 用于集成C/C++和Fortran代码的工具
4. 实用的线性代数、傅立叶变换和随机生成函数

2.NumPy安装

```
pip install numpy或pip3 install numpy
```

3.NumPy引入

```
import numpy as np#为了方便实用numpy 采用np简写
```

4.NumPy方法

```
array=np.array([[1,2,3],[4,5,6]])#将列表转换为矩阵 并转换为int类型
print(array)
'''
[[1 2 3]
 [4 5 6]]
 '''
```

4.1NumPy属性

```
print('array of dim:',array.ndim)#矩阵的维度
#array of dim:2
print('array of shape',array.shape)#矩阵的行数和列数
#array of shape:(2,3)
print('number of size:',array.size)#元素的个数
#number of size:6
```

4.2NumPy创建Array

- array:创建数组
- dtype:指定数据类型
- zeros:创建数据全为0
- ones:创建数据全为1
- empty:创建数据接近0
- arange:指定范围内创建数据
- linspace创建线段

创建数组

```
a=np.array([1,2,3])
print(a)
#[1,2,3]
```

指定数据dtype

```
a=np.array([1,2,3],dtype=np.int)#指定为int类型
print(a.dtype)
#int 64
b=np.array([1,2,3],dtype=np.float)#指定为float类型
print(b.dtype)
#float 64
```

创建特定数据

```
a=np.array([[1,2,3],[4,5,6]])#矩阵 2行3列
print(a)
'''
[[1 2 3]
 [4 5 6]]
 '''
```

创建全0数组

```
a=np.zeros((2,3))#数据全0 2行3列
print(a)
'''
[[0 0 0]
 [0 0 0]]
 '''
```

创建全1数组 指定特定类型dtype

```
a=np.zeros((2,3),dtype=np.int)#数据全1 2行3列 同时指定类型
print(a)
'''
[[1 1 1]
 [1 1 1]]
 '''
```

创建全空数组 每个值接近0

```
a=np.empty(2,3)#数据全为empty 3行4列
print(a)
'''
[[  0.00000000e+000   0.00000000e+000   2.12704693e-314]
 [  2.12706024e-314   2.12706024e-314   2.12706024e-314]]
 '''
```

用array创建连续数组

```
a=np.arange(1,10,2)#1到10的数据 2步长
print(a)
#[1 3 5 7 9]
```

用reshape改变数据形状

```
a=np.arange(6).reshape(2,3)
print(a)
'''
[[0 1 2]
 [3 4 5]]
 '''
```

用linspace创建线段形数据

```
a=np.linspace(1,10,20)#开始端1 结束端5 分割成10个数据 生成线段
print(a)
'''
[ 1.          1.44444444  1.88888889  2.33333333  2.77777778  3.22222222
  3.66666667  4.11111111  4.55555556  5.        ]
  '''
```

4.3NumPy基础运算

基础运算之加、减、三角函数等

```
a=np.array([10,20,30,40])
b=np.arange(4) #array[0,1,2,3]

c=a+b#加法运算
print(c)
#[10,21,32,43]

c=a-b#减法运算
print(c)
#[10.19,28,37]

c=10*np.sin(a)#三角函数运算
```

```
#[-5.44021111,  9.12945251, -9.88031624,  7.4511316 ]

print(b<3)#逻辑判断
#[ True  True  True False]

d=np.random.random((2,3))#随机生成2行3列的矩阵
print(d)
'''
[[ 0.21116981  0.0804489   0.51855475]
 [ 0.38359164  0.55852973  0.73218811]]
'''
print(np.sum(d))#元素求和
#2.48448292958
print(np.max(d))#元素求最大值
#0.732188108709
print(np.min(d))#元素求最小值
#0.0804488978886
```

多维矩阵运算

```
a=np.array([[1,1],[0,1]])
b=np.arange(4).reshape((2,2))

c=np.dot(a,b)#或c=a.dot(b)矩阵运算
print(c)
'''
[[2 4]
 [2 3]]
 '''
```

对行或列执行查找运算

```
a=np.array([[1,2],[3,4]])
print(a)
'''
[[1,2]
 [3,4]]
 '''
print(np.max(a,axis=0))#axis=0时是对列进行操作
#[3,4]
print(np.min(a,axis=1))#axis=1是对行进行操作
#[1,3]
```

矩阵索引操作

```
A=np.arange(2,14).reshape(3,4)
print(A)
'''
```

```
[[2,3,4,5]
 [6,7,8,9]
 [10,11,12,13]]
 '''
print(np.argmax(A))#矩阵中最大元素的索引
#11
print(np.argmin(A))#矩阵中最小元素的索引
#0
print(np.mean(A))#或者np.average(A)求解矩阵均值
#7.5
print(np.cumsum(A))#矩阵累加函数
#[2 5 9 14 20 27 35 44 54 65 77 90]
print(np.diff(A))#矩阵累差函数
'''
[[1 1 1]
 [1 1 1]
 [1 1 1]]
 '''
print(np.nonzero(A))#将非0元素的行与列坐标分割开来
#(array([0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2]), array([0, 1, 2, 3, 0, 1, 2,
3, 0, 1, 2, 3]))
```

矩阵排序、转置、替换操作

```
A=np.arange(14,2,-1).reshape((3,4))
print(A)
'''
[[14 13 12 11]
 [10  9  8  7]
 [ 6  5  4  3]]
 '''
print(np.sort(A))#排序
'''
[[11 12 13 14]
 [ 7  8  9 10]
 [ 3  4  5  6]]
 '''


print(np.transpose(A))
'''
[[14 10  6]
 [13  9  5]
 [12  8  4]
 [11  7  3]]
 '''


print(np.clip(A,5,9))#替换  判断当前矩阵元素是否比最小值小或比最大值大  若是则替换
'''
[[9 9 9 9]
```

```
 [9 9 8 7]
 [6 5 5 5]]
 '''
```

5.索引

一维索引

```
A=np.arange(0,12)
print(A)
#[ 0  1  2  3  4  5  6  7  8  9 10 11]
print(A[1])#一维索引
#1

A=np.arange(0,12).reshape((3,4))
print(A[0])
#[0,1,2,3]
```

二维索引

```
A=np.arange(0,12).reshape((3,4))
print(A)
'''
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
 '''
print(A[1][1])#或者A[1,1]
#5
print(A[1,1:3])#切片处理
#[5,6]

for row in A:
    print(A)
'''
[0 1 2 3]
[4 5 6 7]
[ 8  9 10 11]
 '''
for col in A:
    print(col)
'''
[0 4 8]
[1 5 9]
[ 2  6 10]
[ 3  7 11]
 '''
```

```
for item in A.flat:
    print(item)
'''
0
1
...
10
11
'''
```

6.NumPy之Array合并

```
A=np.array([1,1,1])
B=np.array([2,2,2])
print(np.vstack((A,B)))#上下合并
'''
[[1 1 1]
 [2 2 2]]
 '''
print(np.hstack((A,B)))#左右合并
#[1 1 1 2 2 2]
```

增加维度

```
A=np.array([1,1,1])
print(A.shape)
#(3,)
print(A[np.newaxis,:])
#[[1 1 1]]
print(A[np.newaxis,:].shape)#newaxis增加维度
#(1,3)

print(A[:,np.newaxis])
'''
[[1]
 [1]
 [1]]
 '''
print(A[:,np.newaxis].shape)
# (3,1)
```

多矩阵合并

```
A = np.array([1,1,1])[:,np.newaxis]
B = np.array([2,2,2])[:,np.newaxis]
print(np.concatenate((A,B,B,A),axis=0))#0表示上下合并
'''
```

```
[[1]
 [1]
 [1]
 [2]
 [2]
 [2]
 [2]
 [2]
 [2]
 [1]
 [1]
 [1]]
 '''
print(np.concatenate((A,B,B,A),axis=1))#1表示左右合并
'''
[[1 2 2 1]
 [1 2 2 1]
 [1 2 2 1]]
 '''
```

7.NumPy分割

```
A=np.arange(12).reshape((3,4))
print(A)
'''
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
 '''
print(np.split(A,3,axis=0))#横向分割成3部分 或者np.vsplit(A,3)
#[array([[0, 1, 2, 3]]), array([[4, 5, 6, 7]]), array([[ 8,  9, 10, 11]])]

print(np.split(A,2,axis=1))#竖向分割成2部分 或者np.hsplit(A,2)
'''
[array([[0, 1],
       [4, 5],
       [8, 9]]), array([[ 2,  3],
       [ 6,  7],
       [10, 11]])]
 '''

print(np.array_split(A,3,axis=1))#不等量分割成3部分
'''
[array([[0, 1],
       [4, 5],
       [8, 9]]), array([[ 2],
       [ 6],
       [10]]), array([[ 3],
```

```
       [ 7],
       [11]])]
'''
```

8.NumPy中copy和deep copy

'='赋值方式会带有关联性

```
a=np.arange(4)
print(a)
#[1 2 3 4]
b=a
c=a
d=b
print(b is a)
#True
print(c is a)
#True
print(d is a)
#True

b[0]=5#改变b的值，a,c,d同样会进行改变
print(a)
#[5 2 3 4]
```

'copy()'赋值方式没有关联性

```
a=np.arange(4)#deep copy
print(a)
#[0 1 2 3]
b=a.copy()
a[0]=5
print(b)#值并不发生改变
#[0 1 2 3]
```

- 知乎：@谓之小一
- 公众号：@谓之小一
- GitHub：@weizhixiaoyi
- 技术博客：https://weizhixiaoyi.com

长按关注微信公众号