Numpy 基础教程

为什么需要Numpy?

- 向量化操作，编程更加简单
- 扩充了List 的基本功能，如[:,index], a[a>10]
- 对于向量与矩阵的运算，底层通过CPP算法优化，运行效率更快

Ndarray对象 (n-dimension array)

- ndarray对象是封装过后的list列表对象
- ndarray对象与Python原生的list列表的区别在于
  - ndarray对象的元素必须是同类型，全是数值、字符串或对象
  - ndarray对象支持向量化运算以及更便捷的数据操作
  - ndarray对象经过底层算法优化，速度快
  - ndarray对象是Python科学计算领域的基石

创建Ndarray对象

由列表创建

In [ ]:
```
data = [1,2,3,4,5]
print(type(data))
import numpy as np
data = np.array(data)
print(type(data))
```

```
<class 'list'>
<class 'numpy.ndarray'>
```

生成特定序列

In [ ]:
```
x = np.arange(10, 20, 1)
print(x)
```

```
[10 11 12 13 14 15 16 17 18 19]
```

In [ ]:
```
x = np.linspace(10, 19, 10)
print(x)
```

```
[10. 11. 12. 13. 14. 15. 16. 17. 18. 19.]
```

生成多维数组

In [ ]:
```
x = [[1,2],[2,1],[3,3]]
x = np.array(x)
print(x)
```

```
[[1 2]
 [2 1]
 [3 3]]
```

In [ ]:
```
x = np.zeros((3,3))
print(x)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [ ]:
```
x = np.zeros((3,3,3))
print(x)
```

```
[[[0. 0. 0.]
  [0. 0. 0.]
  [0. 0. 0.]]

 [[0. 0. 0.]
  [0. 0. 0.]
  [0. 0. 0.]]

 [[0. 0. 0.]
  [0. 0. 0.]
  [0. 0. 0.]]]
```

In [ ]:
```
x = np.ones((3,5))
print(x)
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

In [ ]:
```
x = np.eye(3,3)
print(x)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [ ]:
```
x = np.random.random((5,5))
print(x)
```

```
[[0.97055975 0.27204919 0.42134443 0.89135736 0.1068979 ]
 [0.6090846  0.05051459 0.3017772  0.65262865 0.36001728]
 [0.12364218 0.59985068 0.28826965 0.44076935 0.14295756]
 [0.166454   0.01261358 0.34350451 0.16251287 0.2693441 ]
 [0.20121743 0.14070173 0.75580108 0.73987834 0.82384269]]
```

查看数据的维度与尺寸

In [ ]:
```
x = np.random.random((5,5))
print(f"shape:{x.shape}")
print(f"dimension:{x.ndim}")
print(f"type:{x.dtype}")
print(f"item size:{x.itemsize}")
print(f"total size:{x.nbytes}")
```

```
shape:(5, 5)
dimension:2
type:float64
item size:8
total size:200
```

随机数生成

In [ ]:
```
# 生成0-1的随机数组，均匀分布
x = np.random.random((3,3))
print(x)
```

```
[[0.5767288  0.82586158 0.96672993]
 [0.35584827 0.04155186 0.324823  ]
 [0.75157207 0.70833289 0.88318195]]
```

In [ ]:
```
# 生成特定范围的随机整数数组，均匀分布
x = np.random.randint(0,100, (5,5))
print(x)
```

```
[[66 83 54  7 90]
 [87 72 66 47 14]
 [39 66 32 28 99]
 [96 21 22 25 17]
 [43 48  8 28 96]]
```

In [ ]:
```
# 生成特定范围的随机数组，均匀分布
x = np.random.uniform(10,12, (3,3))
print(x)
```

```
[[10.51366991 11.00289412 10.29776608]
 [11.99792611 11.25838457 10.77710431]
 [11.05610487 11.60018468 10.57852758]]
```

In [ ]:
```
# 生成标准正态分布
x = np.random.randn(3,3)
print(x)
```

```
[[ 0.10229333 -0.72333052  1.29589992]
 [-1.25948835 -0.34124854 -0.69440162]
 [-1.43605793 -0.97088001  1.55028354]]
```

In [ ]:
```
# 生成指定均值和方差的正态分布
x = np.random.normal(3,3,(10,10))
print(x)
```

In [ ]:
```
# 打乱数组的元素
x = np.arange(0,10,1)
np.random.shuffle(x)
print(x)
```

```
[2 1 9 0 7 8 6 4 5 3]
```

In [ ]:
```
# 有放回抽样
samples = np.random.choice(x, 2)
print(samples)
```

```
[7 7]
```

In [ ]:
```
# 无放回抽样
samples = np.random.choice(x, 5, replace=False)
print(samples)
```

```
[7 1 6 3 9]
```

ndarray的切片

In [ ]:
```
x = np.random.randint(0,100, (5,5))
print(x)
print(x[1,1])
print(x[:,0])
print(x[2:4,3:5])
print(x[::3,::3])
```

```
[[89 86 31 74 13]
 [25 68 70 69 27]
 [10 89  8 73 29]
 [55 35 24 69 87]
 [58 69 50 13 94]]
68
[89 25 10 55 58]
[[73 29]
 [69 87]]
[[89 74]
 [55 69]]
```

花式索引

In [ ]:
```
x = np.arange(0, 100, 10)
print(x)
indices = [1, 3, -1]
print(x[indices])
```

```
[ 0 10 20 30 40 50 60 70 80 90]
[10 30 90]
```

布尔索引

In [ ]:
```
x = np.arange(0, 100, 10)
indices = np.random.choice([True, False], 10)
print(x)
print(indices)
print(x[indices])
```

```
[ 0 10 20 30 40 50 60 70 80 90]
[ True False False  True False False  True False  True  True]
[ 0 30 60 80 90]
```

数据筛选

In [ ]:
```
x = np.arange(0, 100, 10)
indices = x > 50
print(x)
print(indices)
print(x[indices])
print(np.where(indices))
print(np.where(x > 30))
```

```
[ 0 10 20 30 40 50 60 70 80 90]
[False False False False False False  True  True  True  True]
[60 70 80 90]
(array([6, 7, 8, 9], dtype=int64),)
(array([4, 5, 6, 7, 8, 9], dtype=int64),)
```

In [ ]:
```
x = np.arange(0, 100, 10)
indices =  (x > 50) & (x < 90)
print(x[indices])
```

```
[60 70 80]
```

数据转换

数据类型转换

In [ ]:
```
x = ['1', '2']
x = np.array(x)
print(x)
print(x.dtype)
x = x.astype('float')
print(x)
print(x.dtype)
```

```
['1' '2']
<U1
[1. 2.]
float64
```

数据形状变换

In [ ]:
```
x = np.arange(0,10,1)
print(x)
print(x.shape)
x = x.reshape(2,5)
print(x)
print(x.shape)
```

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

In [ ]:
```
x = np.arange(0,10,1)
print(x)
print(x.shape)
x = x.reshape(-1, len(x))
print(x)
print(x.shape)
```

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

In [ ]:
```
x = np.arange(0, 20, 1)
print(x)
print(x.shape)
x = x.reshape(-1, 5)
print(x)
print(x.shape)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
(20,)
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
(4, 5)
```

基础向量化运算

In [ ]:
```
x = np.array([1,2,3])
y = np.array([3,1,1])
z = x+y
print(z)
```

```
[4 3 4]
```

In [ ]:
```
z = x*y
print(z)
```

```
[3 2 3]
```

In [ ]:
```
print(x+y)
print(x-y)
print(x*y)
print(x/y)
print(x**2)
print(x>y)
print(x<y)
```

```
[4 3 4]
[-2  1  2]
[3 2 3]
[0.33333333 2.         3.        ]
[1 4 9]
[False  True  True]
[ True False False]
```

向量化运算的函数

In [ ]:
```
x = np.random.randint(0,10,10)
print(x)
print(np.max(x))
print(np.min(x))
print(np.argmax(x))
print(np.argmin(x))
print(np.median(x))
print(np.mean(x))
print(np.var(x))
print(np.std(x))
print(np.sort(x))
print(np.sum(x))
print(np.sin(x))
print(np.exp(x))
print(np.sqrt(x))
```

可参考官方介绍，不一一介绍。

https://numpy.org.cn/user/quickstart.html#%E9%80%9A%E5%87%BD%E6%95%B0

线性代数

内积运算 $z = \vec{x} * \vec{y} = x^T y = \sum_{i=1}^{n} x_i * y_i$

In [ ]:
```
x = np.array([1,2,3])
y = np.array([3,1,1])
z = x.dot(y)
print(z)
```

```
8
```

模长 $scale = ||x|| = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\vec{x} * \vec{x}}$

In [ ]:
```
scale = np.sqrt(np.sum(x**2))
print(scale)
```

```
3.7416573867739413
```

余弦 $cos\theta = \frac{\vec{x}*\vec{y}}{||x||||y||}$

In [ ]:
```
cosine = x.dot(y) / (np.sqrt(np.sum(x**2)) * np.sqrt(np.sum(y**2)))
print(cosine)
```

```
0.6446583712203042
```

叉积 $z = \vec{x} \times \vec{y} = (x_1 i + x_2 j + x_3 k) \times (y_1 i + y_2 j + y_3 k)$

In [ ]:
```
z = np.cross(x,y)
print(z)
```

```
[-1  8 -5]
```

外积 $z = \vec{x} \otimes \vec{y} = xy^T$

In [ ]:
```
z = np.outer(x,y)
print(z)
```

```
[[3 1 1]
 [6 2 2]
 [9 3 3]]
```

矩阵乘法

$$Ax = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{bmatrix}$$

$$C = A \cdot B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

In [ ]:
```
A = np.array([[1,2],[1,1]])
B = np.array([[1,1],[2,1]])
x = np.array([1,1])
print(A.dot(x))
print(A.dot(B))
```

```
[3 2]
[[5 3]
 [3 2]]
```

In [ ]:
```
print(A*B)
```

```
[[1 2]
 [2 1]]
```

其他线性代数操作，请查询官网或其他教程

广播

In [ ]:
```
x = np.array([1,2])
print(x+1)
```

```
[2 3]
```

In [ ]:
```
x = np.array([[1,2],[2,2]])
y = np.array([1,1])
print(x+y)
```

```
[[2 3]
 [3 3]]
```

Numpy广播机制 https://zhuanlan.zhihu.com/p/353987442