Numpy入门

什么是numpy

numpy：number python，一个重要的python语言数学运算库，在数据分析，机器学习，科学运算，图像处理等方面有重要的作用。

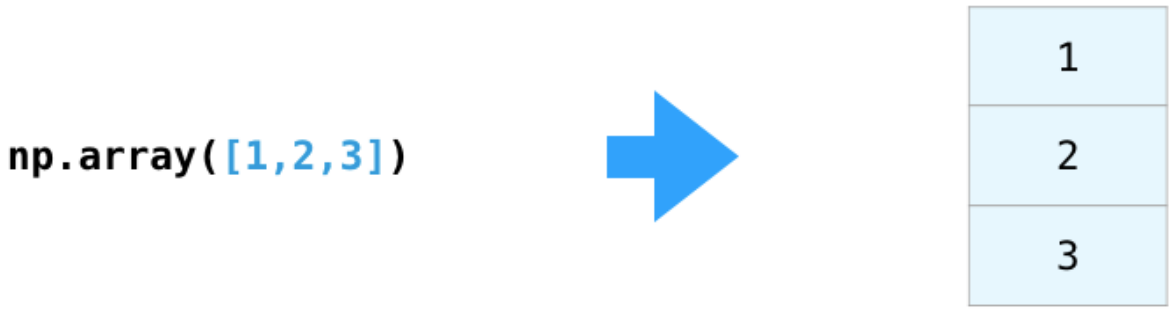
NumPy 是支持 Python 语言的数值计算扩充库，numpy让向量和矩阵的运算变得非常简单。

除此之外，NumPy 还内建了大量的函数，方便你快速构建数学模型。

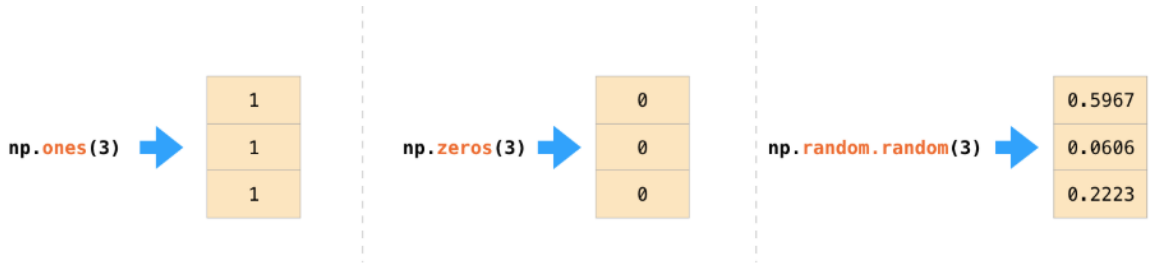
创建numpy数组

numpy的数组不同于普通的list数组，numpy的数组可以轻松的扩展到多个维度。在numpy中数组的名字叫ndarray（n个*dimension*的array）

```
np.array([1,2,3])
```




```
np.ones(3)
np.zeros(3)
np.random.random(3)
```




```
a = np.array([0, 1, 2, 3])
type(a)
a = np.array(0,1,2,3) #报错
a = np.array((0,1,2,3))
c = np.array([3, '4.5'])  # 数组的数据类型
```

创建1-6的数组

```
c = np.arange(6)
```

创建-2到1间隔为0.5的数组, 前开后闭

```
np.arange(-2, 1, 0.5)
```

把0-2的空间分割成5份，包含5

```
np.linspace(0, 2, 5)
np.linspace(0, 10, 10, endpoint=False)
```

二维0数组和1数组

```
np.ones([2, 3])
np.zeros([2, 3])
```

numpy对角单位矩阵

```
np.eye(2)
```

numpy斜对角矩阵

```
a = np.array([1, 2, 3])
d = np.diag(a)  # 2D的斜对角矩阵
```

numpy数组的属性

d的维度

```
d.ndim
```

d的形状

```
d.shape
```

d的元素个数

```
d.size
```

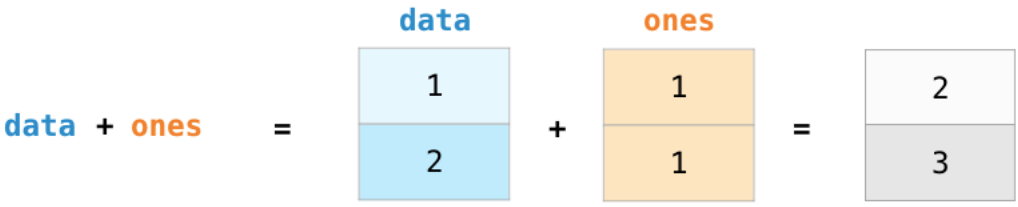
d的数据类型

```
d.dtype
```

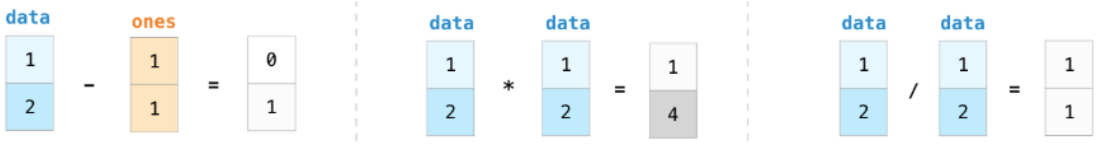
```
import numpy as np  # 导入 NumPy 模块

a = np.array([1.1, 2.2, 3.3], dtype=np.float64)  # 指定 1 维数组的数值类型为 float64
a, a.dtype  # 查看 a 及 dtype 类型

a.astype(int).dtype  # 将 a 的数值类型从 float64 转换为 int，并查看 dtype 类型
```

numpy数组的运算

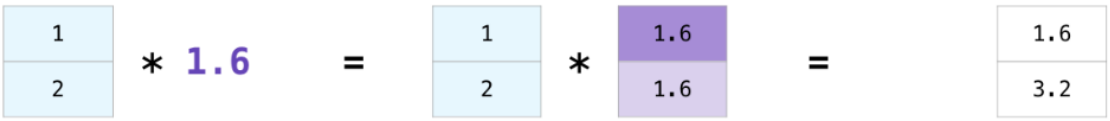
- 加



- 减 乘 除



- numpy数组乘以常量



- 平方

```
data ** 2
```

- 三角函数

```
np.sin(data)
```

- 开平方

```
np.sqrt(a)
```

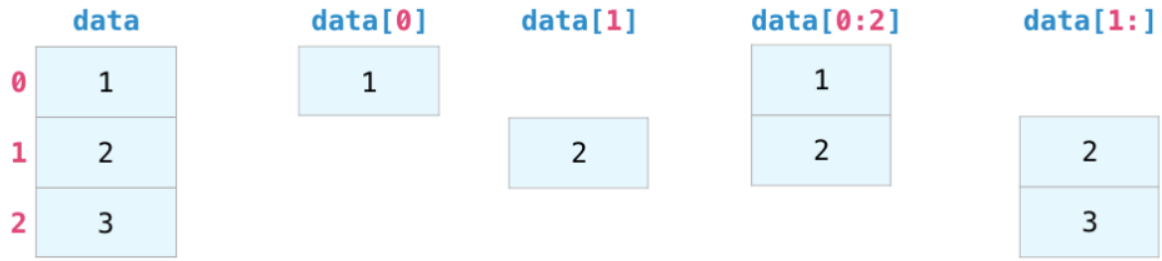
- 布尔过滤

```
a = np.array([20,30,40,50])
a <= 35
```

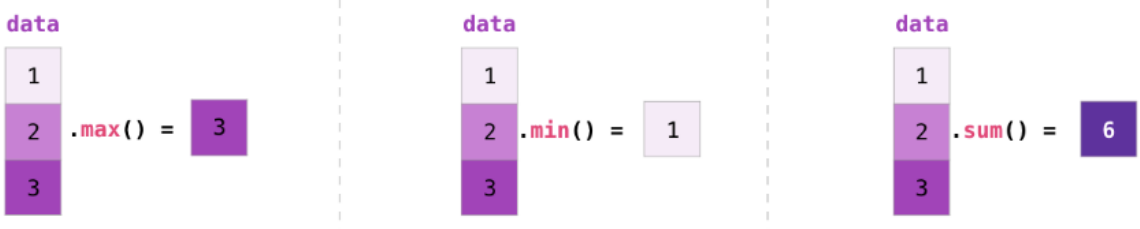
numpy数组和list比较

```
a = [20,30,40,50] #创建一个list
b = list（range(2,6)) #创建另外一个list 从2到6

a + b # 两个list进行concat操作 [20,30,40,50,2,3,4,5]
a - b # 报错
a * 2 # a所在的list repetition2次 [20,30,40,50,20,30,40,50]
a / b # 没有这样的操作
```

numpy数组的索引

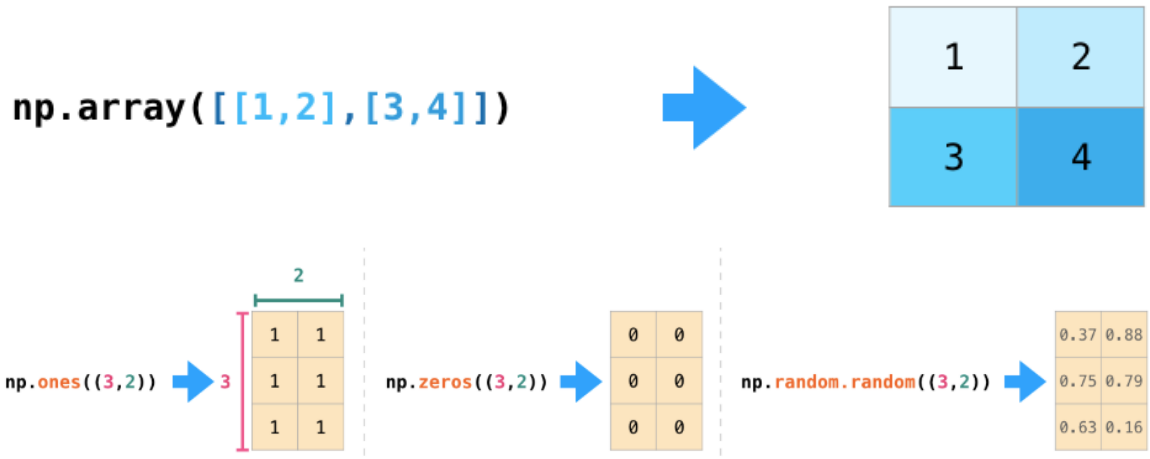


numpy数组的元素操作



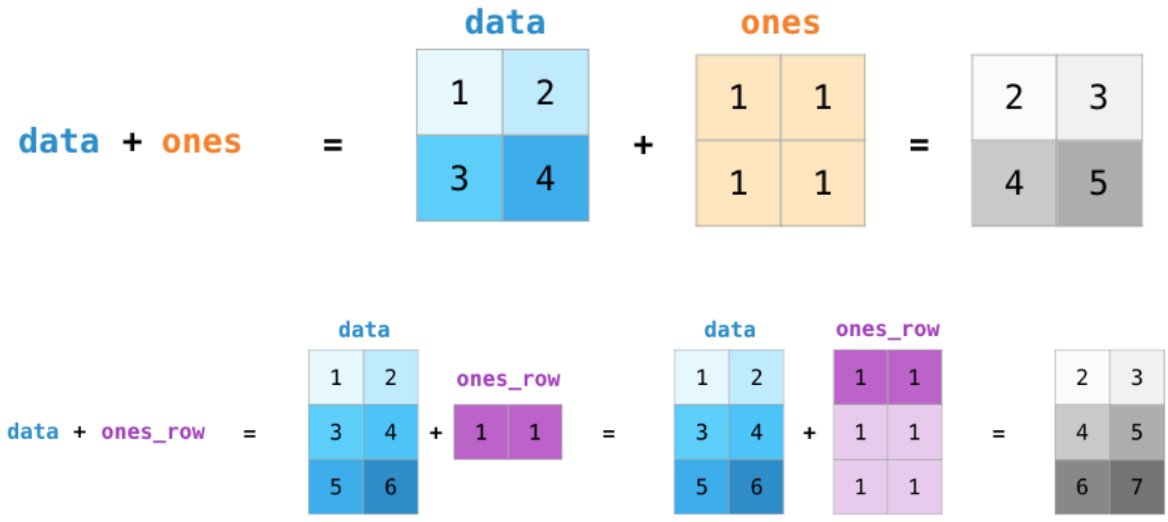
Numpy多维数组的运算

numpy二维数组

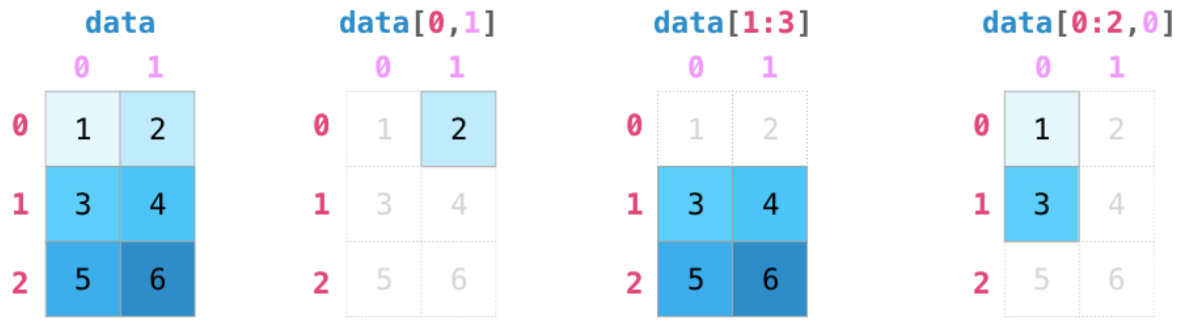



```
b = np.array([[0, 1], [2, 3]])
```

numpy数组运算

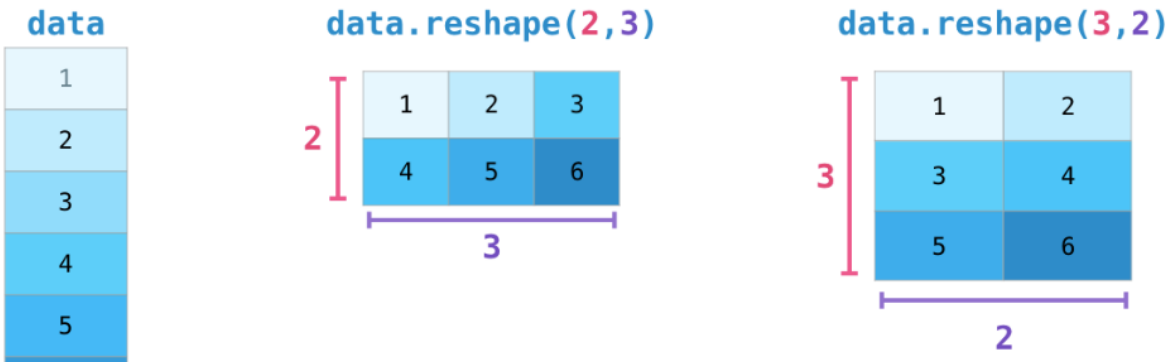


numpy数组切片索引



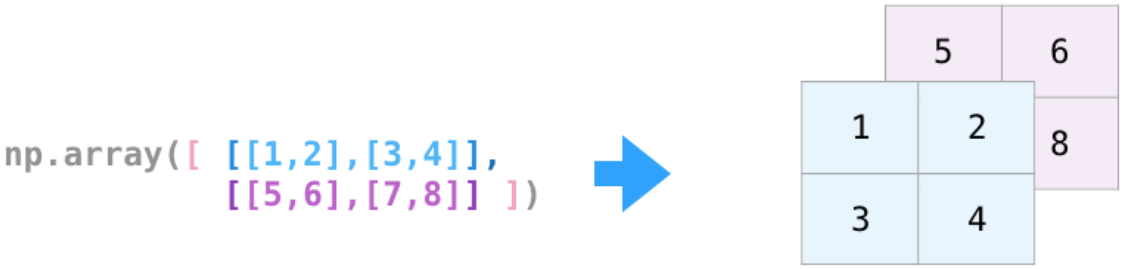
numpy数据转置

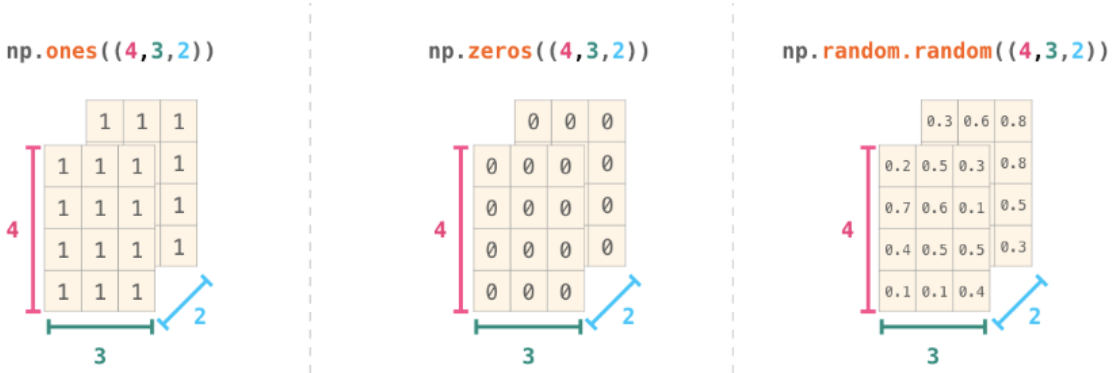
numpy数据变形




```
c = np.array([1,2,3,4,5,6])
c.reshape(2,3)
c.reshape(3,2)
c.reshape(3,-1)
c.ravel() # 降成一维
```

numpy高维数据



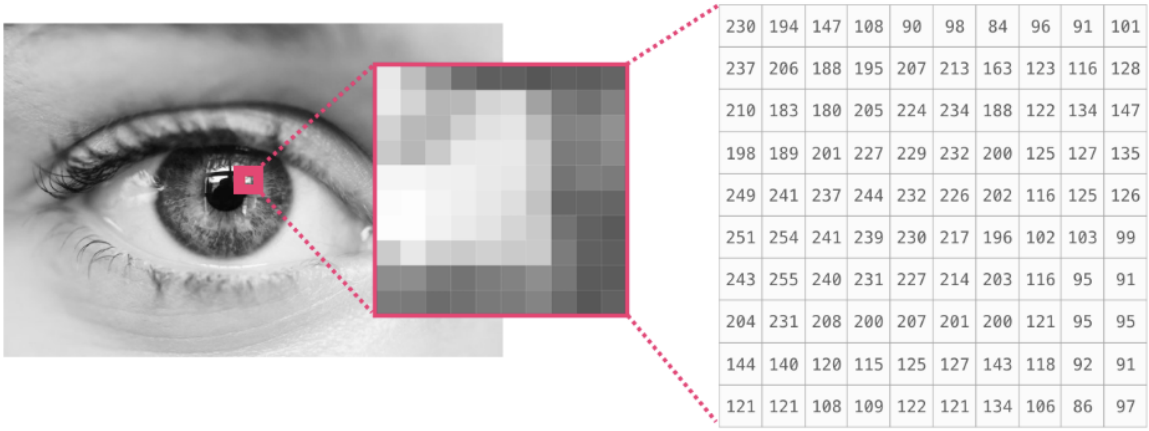



```
d.ndim
d.shape
```

numpy数据维度应用场景

计算机如何表示图像

1. 黑白照片



2. 彩色照片

Numpy统计分析

```
a = np.array([
    [0,1],
    [2,3]
])
```

- 求和

  ```
  np.sum(a)
  ```

- 按行求和

  ```
  np.sum(a,axis=0)
  ```

- 按列求和

  ```
  np.sum(a,axis=1)
  ```

- 求最大值

  ```
  np.max(a)
  ```

- 求最小值

  ```
  np.min(a)
  ```

- 求平均值

  ```
  np.mean(a) #所有数值加起来的平均值
  ```

- 求中位数

  ```
  np.median(a)  #去掉最大值和最小值剩下数的平均值
  ```

- 求方差，方差反应稳定程度

  > 方差越大，数据的波动越大；方差越小，数据的波动就越小
  > 方差是指一组数据中的各个数减这组数据的平均数的平方和的平均数
  >
  > 如（1，2，3,4,5）这组数据的方差，就先求出这组数据的平均数（1+2+3+4+5）÷5＝3，然后再求各个数与平均数的差的平方和，用（1-3）²+（2-3）²+（3-3）²+（4-3）²+（5-3）²＝10，再求平均数10÷5＝2，即这组数据的方差为2

  ```
  np.var(a)
  ```

- 求标准差，均方差

```
标准差是方差的开平方
np.std(a)
```

Numpy随机数

numpy 的随机数功能非常强大，主要由 `numpy.random` 模块完成。

首先，我们需要了解如何使用 NumPy 也就是生成一些满足基本需求的随机数据。主要由以下一些方法完成：

`numpy.random.rand(d0, d1, ..., dn)`：

并使用 `[0, 1)` 区间随机数据填充，这些数据均匀分布。

```
np.random.rand(2, 5)
```

`numpy.random.randn(d0, d1, ..., dn)` 与 `numpy.random.rand(d0, d1, ..., dn)` 的区别在于，前者是从标准正态分布中取一个或多个值，

```
np.random.randn(1, 10)
```

`randint(low, high, size, dtype)` 方法将会生成 `[low, high)` 的随机整数。注意这是一个半开半闭区间。

```
np.random.randint(2, 5, 10)
```

`random_sample(size)` 方法将会在 `[0, 1)` 区间内生成指定 `size` 的随机浮点数。

```
np.random.random_sample([10])
```

numpy随机数种子

```
np.random.seed(seed=3)
np.random.randn(2,2)
```

一旦定下来种子，每次随机的结果都一样

Numpy 逻辑操作

```
a = np.array([True,True,False])
```

判断是不是全都是True

```
np.all(a)
```

判断是不是有一个是True

```
np.any(a)
```

转换普通numpy数组到真值数组

```
a = np.zeros((3,3))
a == 0
a != 0
```

Numpy 排序操作

```
a = np.random.randn(5)
```

直接排序

```
a.sort()
```

排序索引

```
a.argsort()
```

按行排序, 从小到大

```
b = np.random.randn(3,3)
b.sort(axis = 1)
```

按列排序, 从小到大

```
b = np.random.randn(3,3)
b.sort(axis = 0)
```

按行排序索引, 从小到大

```
b = np.random.randn(3,3)
b.argsort(axis = 1)
```

按列排序索引, 从小到大

```
b = np.random.randn(3,3)
b.argsort(axis = 0)
```

行向量和列向量

- numpy中一维数组 行向量和列向量没有任何区别

  ```
  a = np.array([1,2,3])
  a.T
  a.transpose()
  ```

- 二维数组 行向量列向量变化

  ```
  b = np.array([[1],[2],[3]])
  b.T
  b.transpose()
  ```

- 一维数组变二维

  ```
  np.array([1,2,3]).reshape(3,1)
  ```

- 其他创建行向量或者列向量方式

  ```
  a = np.r_[-2, -1, 3,2,3,4,1:4]
  np.c_[np.array([1,2,3,4,5])]
  ```