numpy其他

一、NAN值和INF值

首先我们要知道这两个英文单词代表的什么意思：

1. `NAN`：`Not A number`，不是一个数字的意思，但是他是属于浮点类型的，所以想要进行数据操作的时候需要注意他的类型。
2. `INF`：`Infinity`，代表的是无穷大的意思，也是属于浮点类型。`np.inf`表示正无穷大，`-np.inf`表示负无穷大。

1.NAN的特点：

1. NAN和NAN不相等。比如`np.NAN == np.NAN`返回结果是`False`。
2. NAN和任何值做运算，结果都是NAN。
3. 用`np.isnan()`函数判断NAN值

```
np.isnan(np.NAN)  #返回结果是True
```

有些时候，特别是从文件中读取数据的时候，经常会出现一些缺失值。缺失值的出现会影响数据的处理。因此我们在做数据分析之前，先要对缺失值进行一些处理。处理的方式有多种，需要根据实际情况来做。

2.删除缺失值：

有时候，我们想要将数组中的`NAN`删掉，有两种方法，示例代码如下：

```
# 1. 删除所有NAN的值，使用~删除，因为删除了值后数组将不知道该怎么变化，所以会被变成一维数组
data = np.random.randint(0,10,size=(3,5)).astype(float)
data[0,1] = np.nan
data = data[~np.isnan(data)] # 此时的data会没有nan，并且变成一个1维数组

# 2. 删除NAN所在的行
data = np.random.randint(0,10,size=(3,5)).astype(float)
# 将第(0,1)和(1,2)两个值设置为NAN
data[[0,1],[1,2]] = np.NAN
# 获取哪些行有NAN
lines = np.where(np.isnan(data))[0]    #[0]表示行索引，[1]表示列索引
# 使用delete方法删除指定的行,axis=0表示删除行，lines表示删除的行号
data1 = np.delete(data,lines,axis=0)
```

3.替换NAN值：

可以通过布尔索引将NAN值替换，一般根据需要替换成0或者平均值

```
data = np.random.randint(0,10,size=(3,5)).astype(float)
# 将第(0,1)和(1,2)两个值设置为NAN
data[[0,1],[1,2]] = np.NAN
data[np.isnan(data)] = 0. #通过不二索引将NAN值替换成0
```

二、random函数

`np.random`为我们提供了许多获取随机数的函数。这里统一来学习一下。

1.np.random.seed：

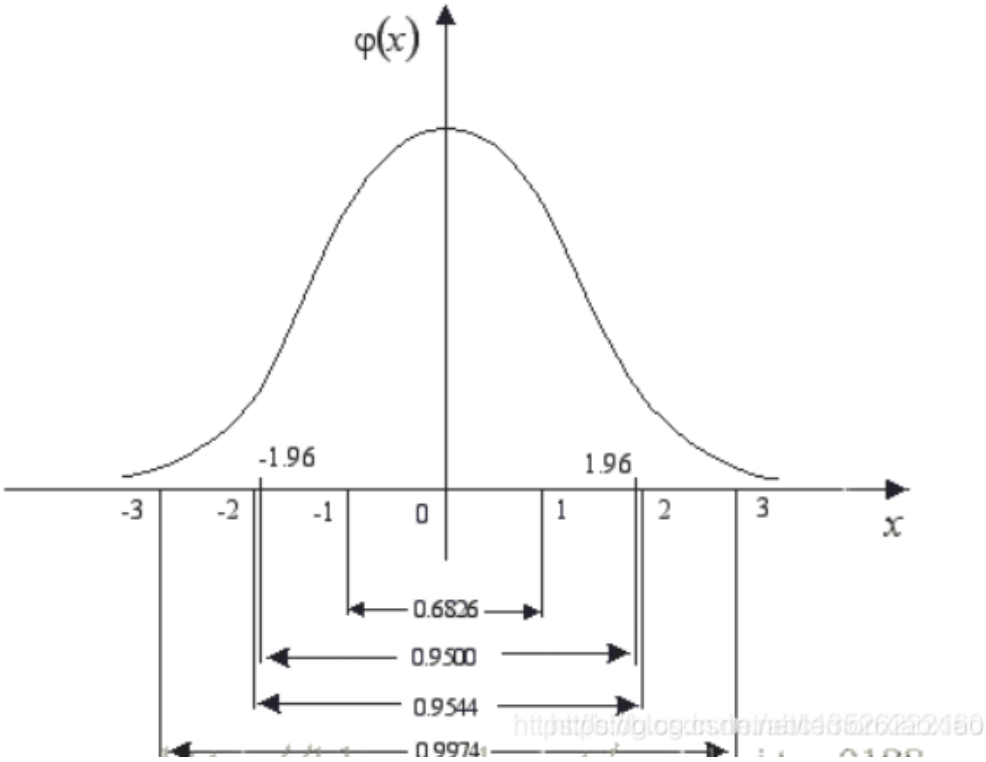
用于指定随机数生成时所用算法开始的整数值，如果使用相同的`seed()`值，则每次生成的随机数都相同，如果不设置这个值，则系统根据时间来自己选择这个值，此时每次生成的随机数因时间差异而不同。一般没有特殊要求不用设置。以下代码：

```
#如果不设置seed
print(np.random.random()) # 打印一个随机值
print(np.random.random()) # 打印其他的值，因为随机数种子只对下一次随机数的产生会有影响。
#设置了seed
np.random.seed(1)
print(np.random.random()) # 打印一个随机值
np.random.seed(1)
print(np.random.random()) # 打印之前相同的随机值
```

2.np.random.randn：

生成均值(μ)为0，标准差（σ）为1的标准正态分布的值。示例代码如下：

```
data = np.random.randn(2,3) #生成一个2行3列的数组，数组中的值都满足标准正态分布
```



3.np.random.randint：

生成指定范围内的随机整数，并且可以通过`size`参数指定维度。示例代码如下：

```
data1 = np.random.randint(10,size=(3,5)) #生成值在0-10之间，3行5列的数组
data2 = np.random.randint(1,20,size=(3,6)) #生成值在1-20之间，3行6列的数组
```

4.np.random.choice：

从一个列表或者数组中，随机进行采样。或者是从指定的区间中进行采样，采样个数可以通过参数指定：

```
data = [4,65,6,3,5,73,23,5,6]
result1 = np.random.choice(data,size=(2,3)) #从data中随机采样，生成2行3列的数组
result2 = np.random.choice(data,3) #从data中随机采样3个数据形成一个一维数组
result3 = np.random.choice(10,3) #从0-10之间随机取3个值
```

5.np.random.shuffle：

把原来数组的元素的位置打乱。示例代码如下：

```
a = np.arange(10)
np.random.shuffle(a) #将a的元素的位置都会进行随机更换
```

7.更多：

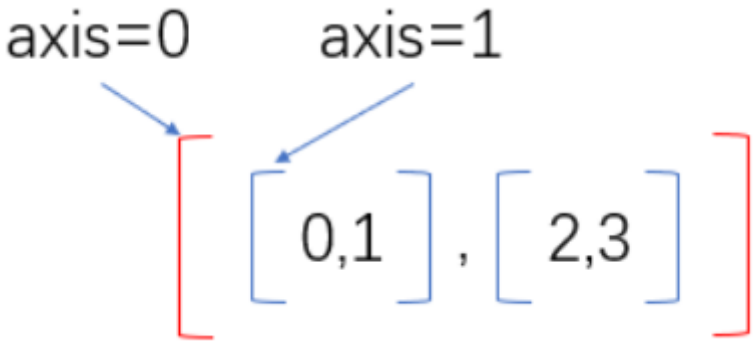
更多的random模块的文档，请参考`Numpy`的官方文档：
https://docs.scipy.org/doc/numpy/reference/routines.random.html

---

三、Axis理解

之前的课程中，为了方便大家理解，我们说`axis=0`代表的是行，`axis=1`代表的是列。但其实不是这么简单理解的。这里我们专门用一节来解释一下这个`axis`轴的概念。

简单来说，**最外面的括号代表着 axis=0，依次往里的括号对应的 axis 的计数就依次加 1**。什么意思呢？下面再来解释一下。



最外面的括号就是`axis=0`，里面两个子括号`axis=1`。
操作方式：如果指定轴进行相关的操作，那么他会使用轴下的每个直接子元素的第0个，第1个，第2个...分别进行相关的操作。

现在我们用刚刚理解的方式来做几个操作。比如现在有一个二维的数组：

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

1. 求`x`数组在`axis=0`和`axis=1`两种情况下的和：

    ```
    >>> x.sum(axis=0)
    array([2, 4])
    ```

    为什么得到的是[2,4]呢，原因是我们按照`axis=0`的方式进行相加，那么就会把**最外面轴下的所有直接子元素中的第0个位置进行相加，第1个位置进行相加...依此类推**，得到的就是`0+2`以及`2+3`，然后进行相加，得到的结果就是`[2,4]`。

    ```
    >>> x.sum(axis=1)
    array([1, 5])
    ```

    因为我们按照`axis=1`的方式进行相加，那么就会把**轴为1里面的元素拿出来进行求和**，得到的就是`0,1`，进行相加为`1`，以及`2,3`进行相加为`5`，所以最终结果就是`[1,5]`了。

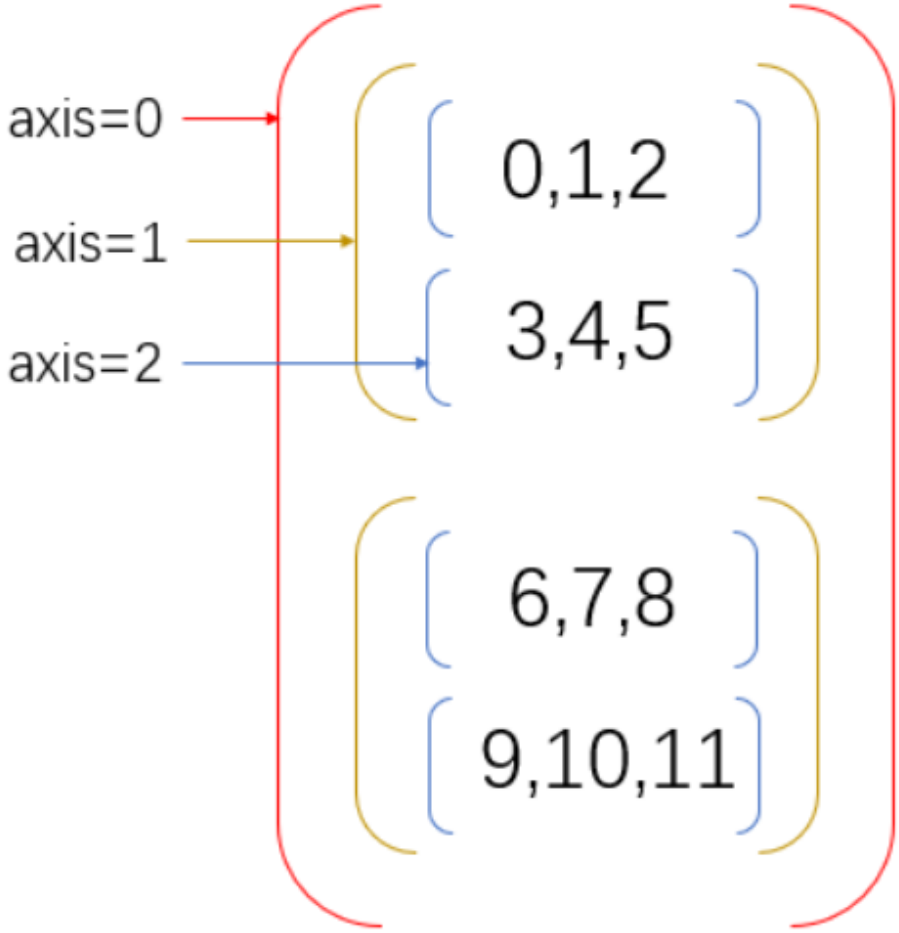
2. 用`np.max`求`axis=0`和`axis=1`两种情况下的最大值：

```
>>> np.random.seed(100)
>>> x = np.random.randint(0,10,size=(3,5))
>>> x.max(axis=0)
array([8, 8, 3, 7, 8])
```

因为我们是按照`axis=0`进行求最大值，那么就会在最外面轴里面找直接子元素，然后将每个子元素的第0个值放在一起求最大值，将第1个值放在一起求最大值，以此类推。而如果`axis=1`，那么就是拿到每个直接子元素，然后求每个子元素中的最大值：

```
>>> x.max(axis=1)
array([8, 5, 8])
```

三维以上数组



按照之前的理论，如果以上数组按照`axis=0`的方式进行相加，得到的结果如下：

如果是按照`axis=1`的方式进行相加，得到的结果如下：