NumPy模块

电子信息工程系

袁羽

目录

CONTENTS

一 NumPy概述

- 图像在OpenCV中以二维或三维数组表示，数组中的每一个值就是图像的像素值。OpenCV中很多操作都要依赖NumPy模块，例如创建纯色图像、创建掩模和创建卷积核等。
- NumPy（Numerical Python）是Python的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。
- NumPy提供了一个高性能的数组对象，以及可以轻松创建一维数组、二维数组和多维数组等大量实用方法，帮助开发者轻松地进行数组计算，从而广泛地应用于数据分析、机器学习、图像处理和计算机图形学、数学任务等领域中。

一 NumPy概述

由于NumPy是由C语言实现的，所以其运算速度非常快。具体功能如下。

1、有一个强大的N维数组对象ndarray；

2、有比较成熟的（广播）函数库（广播(Broadcast)是 numpy 对不同形状(shape)的数组进行数值计算的方式， 对数组的算术运算通常在相应的元素上进行。）；

3、用于整合C/C++和Fortran代码的工具包；

4、有实用的线性代数、傅里叶变换，随机数生成函数和图形操作等功能。

二 NumPy的数据类型

为了区别于Python数据类型，NumPy中的bool、int、float、complex等数据类型名称末尾都加了短下画线“_”。

数据类型	描述
bool_	存储为一个字节的布尔值（真或假）
int_	默认整数，相当于 C 的 long，通常为 int32 或 int64
intc	相当于 C 语言的 int，通常为 int32 或 int64
intp	用于索引的整数，相当于 C 语言的 size_t，通常为 int32 或 int64
int8	字节（-128~127）
int16	16 位整数（-32768~32767）
int32	32 位整数（-2147483648~2147483647）
int64	64 位整数（-9223372036854775808~9223372036854775807）
uint8	8 位无符号整数（0~255）
uint16	16 位无符号整数（0~65535）
uint32	32 位无符号整数（0~4294967295）
uint64	64 位无符号整数（0~18446744073709551615）
float_	_float64 的简写
float16	半精度浮点：1 个符号位，5 位指数，10 位尾数
float32	单精度浮点：1 个符号位，8 位指数，23 位尾数
float64	双精度浮点：1 个符号位，11 位指数，52 位尾数
complex_	complex128 类型的简写
complex64	复数，由两个 32 位浮点表示（实部和虚部）
complex128	复数，由两个 64 位浮点表示（实部和虚部）
datatime64	日期时间类型
timedelta64	两个时间之间的间隔

三 创建数组

1．最常规的array()方法

NumPy创建简单的数组主要使用array()方法，通过传递列表、元组来创建NumPy数组，其中的元素可以是任何对象，语法如下：

numpy.array(object, dtype, copy, order, subok, ndmin)

参数说明：Object：数组或嵌套的数列

Dtype：数组元素的数据类型，可选

Copy：对象是否需要复制，可选

order：创建数组的样式，C为行方向，F为列方向，A为任意方向（默认）

Subok：默认返回一个与基类类型一致的数组

Ndmin：指定生成数组的最小维度

例：创建一维和二维数组

分别创建一维数组和二维数组，效果如图所示。

例：创建一维和二维数组

```
import numpy as np          #导入numpy模块
n1 = np.array([1,2,3])      #创建一个简单的一维数组
n2 = np.array([0.1,0.2,0.3]) #创建一个包含小数的一维数组
n3 = np.array([[1,2],[3,4]]) #创建一个简单的二维数组
```

例：创建浮点类型数组

NumPy支持比Python更多种类的数据类型，通过dtype参数可以指定数组的数据类型。

```
import numpy as np  # 导入numpy模块
list = [1, 2, 3]  # 列表
# 创建浮点型数组
n1 = np.array(list, dtype=np.float_)
# 或者n1 = np.array(list, dtype=float)
print(n1)
print(n1.dtype)
print(type(n1[0]))
```

运行结果如下：

```
[1. 2. 3.]
float64
<class 'numpy.float64'>
```

三 创建数组

2.创建指定维度和数据类型未初始化的数组

创建指定维度和数据类型未初始化的数组主要使用empty()方法，数组元素因为未被初始化会自动取随机值。如果要改变数组类型，可以使用dtype参数，如将数组类型设为整型，dtype=int。

例：创建2行3列的未初始化数组

创建2行3列的未初始化数组，具体代码如下：

```
import numpy as np

n = np.empty([2, 3])

print(n)
```

运行结果如下：

```
[[6.23042070e-307 3.56043053e-307 1.37961641e-306]
 [2.22518251e-306 1.33511969e-306 2.22523004e-307]
```

三 创建数组

3.创建用0 填充的数组

创建指定大小的数组，数组元素以 0 来填充，使用zeros()方法，OpenCV经常使用该方法创建纯黑图像，语法如下：

numpy.zeros(shape, dtype = float, order = 'C')

参数说明：shape：数组形状

dtype ： 数据类型，可选

order ： ‘C' 用于 C 的行数组，或者 'F' 用于 FORTRAN 的列数组

例：创建用0 填充的数组

创建2行3列，数字类型为默认的纯0数组；创建3行4列，数字类型为无符号8位整数的纯0数组。具体代码如下：

```
import numpy as np
# 默认为浮点数
n1 = np.zeros((2,3))
print(n1)
# 设置类型为整数
n2 = np.zeros((3,4), dtype = np.uint8)
print(n2)
```

运行结果如下：

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

三 创建数组

4.创建用1 填充的数组

创建用1填充的数组需要用ones()方法，该方法创建的数组元素均为1。OpenCV经常使用该方法创建纯掩模，卷积核等用于计算的二维数组。

例：创建用1 填充的数组

创建一个2行3列，数字类型为无符号8位整数的纯1数组；创建一个2行3列，数字类型为无符号8位整数的纯100数组。具体代码如下：

```
import numpy as np

# 设置类型为整数

n2 = np.ones((2,3), dtype = np.uint8)

print(n2)

# 改变数组填充的值

n3 = n2*100

print(n3)
```

运行结果如下：

```
[[1 1 1]
 [1 1 1]]
[[100 100 100]
 [100 100 100]]
```

三 创建数组

4.创建用1 填充的数组

创建用1填充的数组需要用ones()方法，该方法创建的数组元素均为1。OpenCV经常使用该方法创建纯掩模，卷积核等用于计算的二维数组。

三 创建数组

5.创建随机数组

randint()方法用于生成一定范围内的随机整数数组，左闭右开区间（[low,high)），语法如下：numpy.random.randint(low,high,size)

参数说明：

low：随机数最小取值范围。

high：可选参数，随机数最大取值范围。若high为空，取值范围为（0，low）。若high不为空，则生成的数值在[low, high)区间。

size：可选参数，输出随机数组的尺寸，比如size = (m, n, k)，则输出数组的shape = (m, n, k)，数组中的每个元素均满足要求。size默认为None，仅仅返回满足要求的单一随机数。

例：创建随机数组

生成一定范围内的随机数组，具体代码如下生成一定范围内的随机数组。

```
import numpy as np
 n1 = np.random.randint(low=1, high=3, size=10)
 print('随机生成10个1～3且不包括3的整数：')
 print(n1)
 n2 = np.random.randint(5, 10)
 print('size数组大小为空随机返回一个整数：')
 print(n2)
 n3 = np.random.randint(5, size=(2, 5))
 print('随机生成5以内二维数组：')
 print(n3)
```

运行结果如下：
随机生成10个1～3且不包括3的整数：
[2 1 1 2 2 1 1 2 2 1]
size数组大小为空随机返回一个整数：
6
随机生成5以内二维数组：
[[4 0 2 1 0]
 [3 4 4 1 0]]

三 创建数组

5.numpy.arange()函数：用于生成一维数组，使用频率非常高。也可作为线性序列生成器， 用于在线性空间中以均匀步长生成数字序列。

使用语法：numpy.arange(start, stop, step, dtype = None)

使用参数

Start：开始位置，数字，可选项，默认起始值为0

stop：停止位置，数字

step：步长，数字，可选项， 默认步长为1，如果指定了step，则还必须给出start。

dtype ：输出数组的类型。 如果未给出dtype，则从其他输入参数推断数据类型。

三 创建数组

使用实例

```
import numpy as np
c = np.arange(2,10,2,dtype=np.int32)
print(c)
d = np.arange(1,8,2,dtype=np.int32).reshape((2,2))
print(d)
```

运行结果:

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

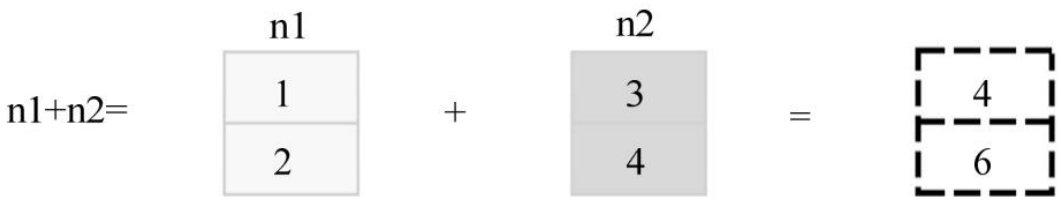
四 操作数组

不用编写循环即可对数据执行批量运算，这就是NumPy数组运算的特点，NumPy称为矢量化。大小相等的数组之间的任何算术运算都可以用NumPy实现。

四 操作数组

1．加法运算

加法运算是数组中对应位置的元素相加（即每行对应相加）。



例 对数组做加法运算：使用NumPy创建2个数组，并让2个数据进行加法运算。

```
import numpy as np
n1 = np.array([1, 2])  # 创建一维数组
n2 = np.array([3, 4])
print(n1 + n2)         # 加法运算
```

运行结果如下：

[4 6]

四 操作数组

2．减法和乘除法运算

除了加法运算，还可以实现数组的减法、乘法和除法。

n1−n2=

n1		n2		
1	−	3	=	−2
2		4		−2

n1*n2=

n1		n2		
1	*	3	=	3
2		4		8

n1/n2=

n1		n2		
1	/	3	=	0.333
2		4		0.5

例 对数组做减法、乘法和除法运算：使用NumPy创建2个数组，并让2个数组进行减法、乘法和除法运算。

```
import numpy as np
 n1 = np.array([1, 2])  # 创建一维数组
 n2 = np.array([3, 4])
 print(n1 - n2)          # 减法运算
 print(n1 * n2)          # 乘法运算
 print(n1 / n2)          # 除法运算
```

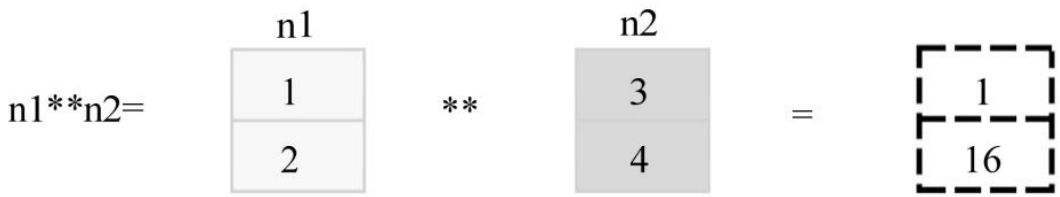
运行结果如下：

```
 [-2 -2]
[3 8]
[0.33333333 0.5  ]
```

四 操作数组

3．幂运算

幂是数组中对应位置元素的幂运算，使用“**”运算符进行运算。数组n1的元素1和数组n2的元素3，通过幂运算得到的是1的3次幂；数组n1的元素2和数组n2的元素4，通过幂运算得到的是2的4次幂。



例 两个数组做幂运算：使用NumPy创建2个数组，并让2个数组做幂运算。

```
import numpy as np
n1 = np.array([1, 2])  # 创建一维数组
n2 = np.array([3, 4])
print(n1 ** n2)        # 幂运算
```

运行结果如下：

```
[ 1 16]
```

四 操作数组

4．比较运算

NumPy创建的数组可以使用逻辑运算符进行比较运算，运算的结果是布尔值数组，数组中的布尔值为相比较的数组在相同位置元素的比较结果。

例 使用逻辑运算符比较数组：使用NumPy创建2个数组，分别使用“>=”“==”“<=”和“!=”运算符比较2个数组。

```
import numpy as np
n1 = np.array([1, 2]) # 创建一维数组
n2 = np.array([3, 4])
print(n1 >= n2) # 大于等于
print(n1 == n2) # 等于
print(n1 <=n2) # 小于等于
print(n1 != n2) # 不等于
```

运行结果如下：

```
[False False]
[False False]
[ True  True]
[ True  True]
```

四 操作数组

5．复制数组

NumPy提供的array()方法可以使用如下语法复制数据：

```
n2 = np.array(n1, copy=True)
```

但开发过程中更常用的是copy()方法，其语法如下：

```
n2 = n1.copy()
```

这两种方法都可以按照原数组的结构、类型、元素值创建出一个副本，修改副本中的元素不会影响到原数组。

例 复制数据，比较复制的结果与原数组是否相同。使用copy()方法复制数组，比较2个数组是否相同。修改副本数组中的元素值后，再查看2个数组是否相同。

```
import numpy as np
n1 = np.array([1, 2])  # 创建一维数组
n2 = n1.copy()  # 创建复制第一个数组
print(n1 == n2)  # 比较两个数组是否相同
n2[0] = 9  # 副本数组修改第一个元素
print(n1)  # 输出两个数组的元素值
print(n2)
print(n1 == n2)  # 比较两个数组是否相同
```

运行结果如下：

```
[ True  True]
[1 2]
[9 2]
[False  True]
```

五 数组的索引和切片

1．索引

所谓数组的索引，即用于标记数组中对应元素的唯一数字，从0开始，即数组中的第一个元素的索引是0，依次类推。NumPy数组可以使用标准Python语法x[obj]的语法对数组进行索引，其中x是数组，obj是选择方式。

例 查找一维数组索引为0的元素：查找数组n1索引为0的元素。

```
import numpy as np
 n1=np.array([1,2,3])  #创建一维数组
 print(n1[0])
```

运行结果如下：

1

五 数组的索引和切片

2．切片式索引

数组的切片可以理解为对数组的分割，按照等分或者不等分，将一个数组切割为多个片段，与Python中列表的切片操作一样。NumPy中用冒号分隔切片参数来进行切片操作，语法如下：

[start:stop:step]

参数说明：

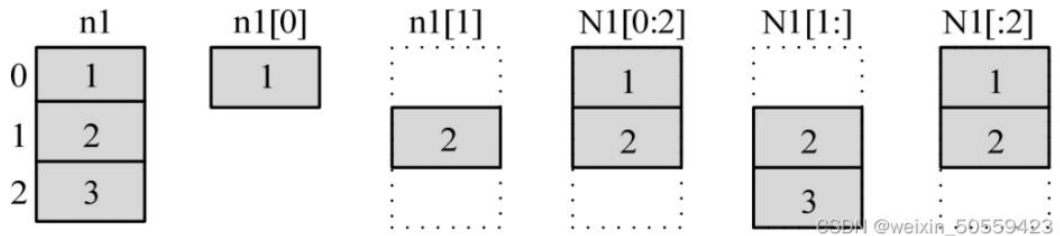
start：起始索引，若不写任何值，则表示从0开始的全部索引。

stop：终止索引，若不写任何值，则表示直到末尾的全部索引。

step：步长。

五 数组的索引和切片

对数组n1进行一系列切片式索引操作的示意图：



例 获取数组中某范围内的元素

```
import numpy as np
n1=np.array([1,2,3]) #创建一维数组
print(n1[0])
print(n1[1])
print(n1[0:2])
print(n1[1:])
print(n1[:2])
```

运行结果如下：

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

五 数组的索引和切片

切片式索引操作需要注意以下几点。

（1）索引是左闭右开区间，如上述代码中的n1[0:2]，只能取到索引从0～1的元素，而取不到索引为2的元素。

（2）当没有start参数时，代表从索引0开始取数，如上述代码中的n1[:2]。

（3）start、stop和step 3个参数都可以是负数，代表反向索引。以step参数为例，如图4.16所示。

五 数组的索引和切片

例 使用不同的切片式索引操作获取数组中的元素。分别演示start、stop、step 3种索引的切片场景。

```
import numpy as np
n = np.array([0,1,2,3,4,5,6,7,8,9])
print(n)
print(n[:3])          # 0 1 2
print(n[3:6])         # 3 4 5
print(n[6:])          # 6 7 8 9
print(n[::])          # 0 1 2 3 4 5 6 7 8 9
print(n[::2])         # 0 2 4 6 8
print(n[1::5])        # 1 6
print(n[2::6])        # 2 8
#start、stop、step为负数时
print(n[::-1])        # 9 8 7 6 5 4 3 2 1 0
print(n[:-3:-1])      # 9 8
print(n[-3:-5:-1])    # 7 6
print(n[-5::-1])      # 5 4 3 2 1 0
```

例 使用不同的切片式索引操作获取数组中的元素。分别演示start、stop、step 3种索引的切片场景。

运行结果如下：

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

[0 1 2]

[3 4 5]

[6 7 8 9]

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

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

[0 2 4 6 8]

[1 6]

[2 8]

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

[9 8]

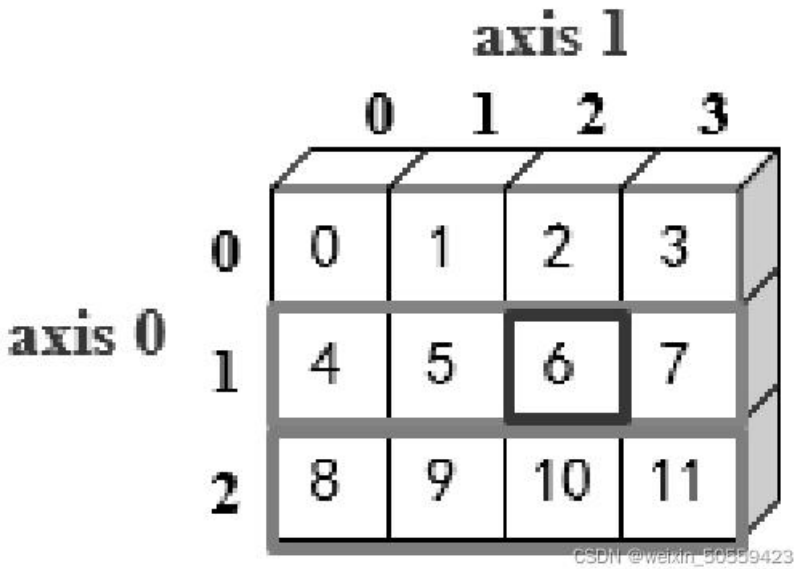
[7 6]

[5 4 3 2 1 0]

五 数组的索引和切片

3．二维数组索引

二维数组索引可以使用array[n,m]的方式，以逗号分隔，表示第n个数组的第m个元素。



例 用3种方式获取二维数组中的元素：分别获取二维数组中索引为1的元素、第2行第3列的元素、索引为-1的元素。

```
import numpy as np
#创建3行4列的二维数组
n=np.array([[0,1,2,3],[4,5,6,7],[8,9,10,11]])
print(n[1])
print(n[1,2])
print(n[-1])
```

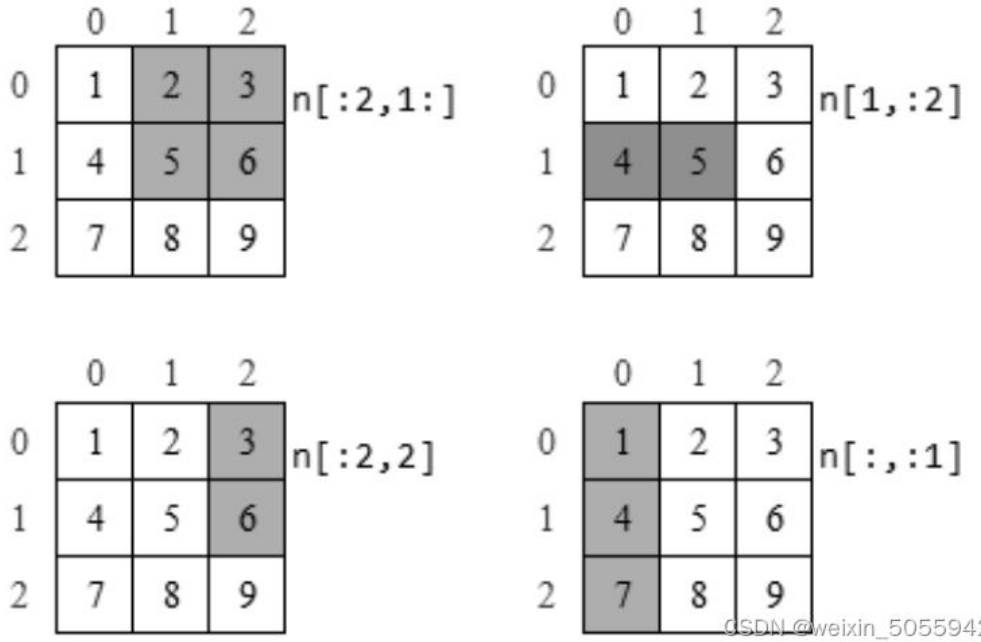
运行结果如下：

```
[4 5 6 7]
6
[ 8  9 10 11]
```

五 数组的索引和切片

4．二维数组切片式索引

二维数组也支持切片式索引操作，如图4.18所示就是获取二维数组中某一块区域的索引。



例 对二维数组进行切片式索引操作：创建二维数组，对该数组进行切片式索引操作

```
import numpy as np
# 创建3行3列的二维数组
n = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(n[:2, 1:])
print(n[1, :2])
print(n[:2, 2])
print(n[:, :1])
```

运行结果如下：

```
[[2 3]
 [5 6]]
[4 5]
[3 6]
[[1]
 [4]
 [7]]
```

六 数组形状操作

Numpy 有一个强大之处在于可以很方便的修改生成的N维数组的形状。

1. 改变数组的形状

2. 展平数组

3.转置数组

4.组合数组

5.分割数组

六 数组形状操作

1. 改变数组的形状

在对数组进行操作时，经常要改变数组的维度。在NumPy中，常用reshape函数改变数组的“形状”，也就是改变数组的维度。其参数为一个正整数元组，分别指定数组在每个维度上的大小。

例. 随机生成二维数组，然后改变数组的形状。

```
import numpy as np

n1=np.random.randint(10,size=(3,4))

print(n1)

print(n1.shape)

print(n1.reshape(2,6))
```

六 数组形状操作

1. 改变数组的形状

- reshape函数在改变原始数据的形状的同时不改变原始数据的值。如果指定的维度和数组的元素数目不吻合，则函数将抛出异常。
- 在使用 reshape 时，可以将其中的一个维度指定为 -1，Numpy 会自动计算出它的真实值。reshape函数当参数只有一个-1时表示将数组降为一维。

六 数组形状操作

2. 展平数组

在NumPy中，可以使用ravel函数完成数组展平工作。ravel()返回的是原始数组的视图，原始数组本身并没有发生变化。传入'F'参数表示列序优先。

例. 随机生成二维数组，然后将数组展平。

```
import numpy as np

n1=np.random.randint(10,size=(3,4))

print(n1)

print(n1.ravel())

print(n1)
```

六 数组形状操作

3.转置数组

数组转置是将矩阵的行列互换得到的新矩阵称为转置矩阵。x.T表示x的转置,行列互换。

例. 随机生成二维数组，然后将数组转置。

```
import numpy as np

n1=np.random.randint(10,size=(3,4))

print(n1)

print(n1.T)

print(n1)
```

六 数组形状操作

注意：

无论是ravel、reshape、T，它们都不会更改原有的数组形状，都是返回一个新的数组。

六 数组形状操作

4.组合数组

除了可以改变数组“形状”外，NumPy也可以对数组进行组合（堆叠）。组合主要有横向组合与纵向组合。可使用hstack函数、vstack函数以及concatenate函数来完成数组的组合。

横向组合是将ndarray对象构成的元组作为参数，传给hstack函数。纵向组合同样是将ndarray对象构成的元组作为参数，只不过传给vstack函数。

concatenate函数也可以实现数组的横向组合和纵向组合，其中参数axis=1时按照横轴组合，参数axis = 0时按照纵轴组合。

例. 生成两个维度相同的数组，然后将两数组组合在一起。

```
import numpy as np
arr1 = np.arange(12).reshape(3,4)
print('创建的数组1为：',arr1)
arr2 = arr1*3
print('创建的数组2为：',arr2)
print('横向组合为：',np.hstack((arr1,arr2)))  #hstack函数横向组合
print('纵向组合为：',np.vstack((arr1,arr2)))  #vstack函数纵向组合
print('横向组合为：',np.concatenate((arr1,arr2),axis = 1))  #concatenate函数横向组合
print('纵向组合为：',np.concatenate((arr1,arr2),axis = 0))  #concatenate函数纵向组合
```

六 数组形状操作

5.分割数组

还可以对数组进行分割。NumPy提供了hsplit、vsplit、dsplit和split函数，可以将数组分割成相同大小的子数组，也可以指定原数组中需要分割的位置。

使用hsplit函数可以对数组进行横向分割。

使用vsplit函数可以对数组进行纵向分割。

split函数同样可以实现数组分割。在参数axis=1时，可以进行横向分割；在参数axis=0时，可以进行纵向分割。

例. 生成一个的数组，然后将一个数组拆分为多个。

```
import numpy as np
arr = np.arange(16).reshape(4,4)
print('创建的二维数组为：\n',arr)
print('横向分割为：\n',np.hsplit(arr, 2)）#hsplit函数横向分割
print('纵向分割为：\n',np.vsplit(arr, 2)）#vsplit函数纵向分割
print('横向分割为：\n',np.split(arr, 2, axis=1)）#split函数横向分割
print('纵向分割为：\n',np.split(arr, 2, axis=0)）#split函数纵向分割
```

小结

1.创建数组：zeros()用0填充

ones()用1 填充

2.数组的索引：索引切片

THANKS