numpy 授课代码

In [5]:
```
import numpy as np
```

In [6]:
```
a = np.array([1,2,3,4])
b = np.array([[1,20], [5,3]])
c = np.array([[[5,3],[2,1]], [[6,7],[9,2]]])
```

In [30]:
```
print(f"a的最大值{a.max()} | b的最大值{b.max()} c的最大值{c.max()}")
```

```
a的最大值4 | b的最大值20 c的最大值9
```

In [31]:
```
print(f"a的最小值{a.min()} | b的最小值{b.min()} c的最小值{c.min()}")
```

```
a的最小值1 | b的最小值1 c的最小值1
```

In [20]:
```
print(a.shape) # shape形状，如果圆括号里有3个数，说明是3维的矩阵。
print(b.shape)
print(c.shape)
```

```
(4,)
(2, 2)
(2, 2, 2)
```

In [23]:
```
print(type(a)) # ndarray :  n dimension array: n维数组
```

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

作业说明：

没有标准答案，但是代码可以衡量好坏，衡量标准：

1 时间复杂度

2 空间复杂度

3 书写规范：命名清晰准确符合语言规则，代码整洁 《代码整洁之道》

In [26]:
```
print(a.ndim, b.ndim, c.ndim) # ndim : n dimension n维
```

```
1 2 3
```

In [34]:
```
print(a.size , b.size , c.size) # size大小：得到数组中元素的个数
```

```
4 4 8
```

In [54]:
```
a = np.array([1,2,127,4],dtype='int8') # int8范围： -128~127   dtype='int8'指定数据类型为8个位的整数
```

```
print(a.dtype,b.dtype) # dtype: data type 数据类型，数组中元素的数据类型
print(f"a={a}")
```

```
int8 int64
a=[  1   2 127   4]
```

numpy 创建n维数组

In [63]:
```
d = np.zeros((2,3)) # 创建了一个2维矩阵，用0填充
print(d)
```

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

In [66]:
```
d = np.ones((3,4)) # 创建了一个2维矩阵，用1填充
print(d)
```

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

In [71]:
```
d = np.full((5,5) , 100) # 创建了一个2维矩阵，用10填满
print(d)
```

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

In [74]:
```
d1 = np.full_like(d,5) # like指的是像d的shape
print(d1)
```

```
[[5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]]
```

numpy 随机数/随机矩阵

In [159…
```
d = np.random.rand(3,3)     # 创建一个随机(小数0～1)的矩阵，3x3
print(d)
```

```
[[0.62797201 0.10696749 0.04702019]
 [0.39218482 0.07141808 0.8426908 ]
 [0.40982573 0.78662572 0.80924801]]
```

In [217…
```
# 产生了一个随机数,范围是 [1,9] 或 [1,10).. 如何得到一个随机的整数？骰子
# 《计算机程序设计艺术》高德纳。 生成随机数。  伪随机
d = np.random.randint(1,10)
print(d)
```

```
2
```

In [291…
```
d = np.random.randint(5, size=(3,3)) # 生成一个随机的二维矩阵，数字范围 0-4
print(d)
```

```
[[4 1 1]
 [3 2 4]
 [4 0 0]]
```

In [313…
```
d = np.random.randint(1,5, size=(3,3)) # 生成一个随机的二维矩阵，数字范围 1-4  ()圆括号是元组类型
print(d)
```

```
[[1 2 4]
 [1 4 3]
 [1 2 4]]
```

In [318…
```
d1 = np.random.sample(d.shape)  # 按照d的shape创建一个新的随机矩阵
print(d1)
```

```
[[0.91667984 0.21671131 0.87056679 0.37716407 0.05187759]
 [0.82341433 0.55241154 0.54287287 0.91667294 0.08534548]
 [0.59712627 0.34105055 0.37626865 0.43414961 0.27433975]
 [0.63292811 0.49582586 0.09743218 0.89943159 0.32084477]
 [0.26122847 0.05422236 0.12827475 0.73738775 0.57180044]]
```

In [320…
```
d1 = np.random.randint(1,5, size=d.shape)
print(d1)
```

```
[[2 1 3]
 [3 4 1]
 [3 4 4]]
```

In [330…
```
a = np.array([[1,20], [5,3]])
b = np.repeat(a,3)   # repeat重复： 把a中的元素全部取出来放到新的【一维】数组b中，重复3次
print(a)
print(b)
```

```
[[ 1 20]
 [ 5  3]]
[ 1  1  1 20 20 20  5  5  5  3  3  3]
```

In [342…
```
a = np.array([[[5,3],[2,1]], [[6,7],[9,2]]])
b = np.repeat(a,3,axis=2) # axis层级,这里a是3维矩阵，所以 axis=的最大值是2 ，axis=0, axis=1 axis=2
print(a)
print('----------------------')
print(b)
```

```
[[[5 3]
  [2 1]]

 [[6 7]
  [9 2]]]
----------------------
[[[5 5 5 3 3 3]
  [2 2 2 1 1 1]]

 [[6 6 6 7 7 7]
  [9 9 9 2 2 2]]]
```

In [351…
```
a = np.identity(3) # Identity Matrix : 单位矩阵
print(a)
```

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

In [376…
```
a = np.array([2,3,4,5,6])  # 截取矩阵
a[1:-1]
```

Out[376]:
```
array([3, 4, 5])
```

In [370…
```
a = np.array([[11,12,13], [25,23,21]])  # 截取2维矩阵
print(a[1][0:2])
print(a[1,0:2])
```

```
[25 23]
[25 23]
```

练习 生成一个符合图片说明的矩阵

In [379…
```
a = np.ones((5,5))
b = np.zeros((3,3))
b[1,1] = 9
a[1:-1,1:-1] = b
print(a)
```

```
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 9. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

numpy矩阵的加减乘除 平方

In [407…
```
a = np.array([1, 2, 3, 4])
b = np.array([11,12,13,14])
#c = np.array([[1,20], [5,3]])
print(b/1)
print(a*b)  # TODO a*b 是矩阵的算术乘
#print(a+b[0:4]) #会报错，两个矩阵+的时候，shape要一样
```

```
[11. 12. 13. 14.]
[11 24 39 56]
```

In [412…
```
# 平方
a = np.array([1, 2, 3, 4])
a**2 # a*a
```

Out[412]:
```
array([ 1,  4,  9, 16])
```

In [413…
```
#矩阵的点乘：  左边矩阵的行 乘以 右边矩阵的列
a = np.array([[1,2,4],[2,3,5],[3,4,6]])
b = np.array([[3,6,9],[4,7,11],[5,8,15]])
print(np.dot(a,b)) # dot点：  矩阵的点乘，a和b交换位置后，点乘的结果不一样的！！！
print(np.dot(b,a))
```

```
[[ 31  52  91]
 [ 43  73 126]
 [ 55  94 161]]
[[ 42  60  96]
 [ 51  73 117]
 [ 66  94 150]]
```

In [414…
```
## numpy 加载txt文件中的数据
```

In [417…
```
# TODO 不用numpy ， 获取 1.txt中的数据
ftxt = open('1.txt','r',encoding='utf-8')

# java python c# c++ 本质上是同一个语言，因为都是面向对象的语言。 理解成方言

lines = ftxt.readlines() # lines是list类型，可以直接for迭代
for line in lines:
    #print(line)  # line是str类型   1,2,3,4,5
    line = line.replace('\n','')  # 替换掉 \n 换行符
    numbers = line.split(",") # 将字符串按照 ， 号切开。结果是一个list
    #print(numbers)
    for number in numbers:
        print(number,end="\t") # print不换行
    print()
```

```
1	2	3	4	5	6
7	16	4	3	2	1
5	6	7	8	9	3
5	4	3	4	5	6
```

In [418…

```
# 使用numpy加载 1.txt
import numpy as np

ftxt = np.genfromtxt("1.txt", delimiter=',') # ftxt是一个 numpy.ndarray类型
ftxt = ftxt.astype('int8')  # 修改数据类型。因为我们知道int8能装的下
print(ftxt)
print(ftxt.dtype)
```

```
[[ 1  2  3  4  5  6]
 [ 7 16  4  3  2  1]
 [ 5  6  7  8  9  3]
 [ 5  4  3  4  5  6]]
int8
```

In [421…

```
# numpy数组对象的复制
a = np.array([2,3,4,5])
b = a
b[0] = 1
print(a)
```

```
[1 3 4 5]
```

In [422…

```
# numpy数组对象的复制
a = np.array([2,3,4,5])
b = a.copy()
b[0] = 1
print(a)
```

```
[2 3 4 5]
```