Numpy科学计算（人口数据统计）

2-1 创建数组和相关运算

创建数组

```
#  创建一维数组
import numpy as np
arr1 = np.array([1,2,3,4,5,6])
print(arr1)
```

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

```
# 创建二维数组
arr2 = np.array([[1,2,3,4],[5,6,7,8]])
print(arr2)
arr2 = np.array([1,2,3,4,5,6,7,8,9,10,11,12]).reshape(3,4)
print(arr2)
```

```
[[1 2 3 4]
 [5 6 7 8]]
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

```
# 查看数组维度和形状
print(arr1.ndim, arr2.ndim)
print(arr1.shape, arr2.shape)
```

```
1 2
(6,) (3, 4)
```

```
# 创建特殊数组
arr3 = np.zeros(3)
arr4 = np.ones(4)
arr5 = np.zeros((3,4))
print(arr3)
print(arr4)
print(arr5)
```

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

```
# 创建数组序列
arr6 = np.arange(1,10)
print(arr6)
```

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

```
# 创建随机数数组
arr7 = np.random.random((3,3))
print(arr7)
```

```
[[0.65226412 0.74225169 0.42661526]
 [0.40497119 0.90266067 0.48289301]
 [0.87424003 0.0852752  0.91032022]]
```

项目：人口数据读取

```
# 从文本文件读取数组数据
import numpy as np
peoples = np.loadtxt("tmp/人口.csv",skiprows=2,delimiter=",",usecols=
(range(1,13)),encoding="utf8")
print(peoples)
# 读入全国各省市名称，文件的第一列（跳过前两行）
names = np.loadtxt("tmp/人口.csv",skiprows=2,delimiter=",",usecols=
(0),encoding="utf8", dtype=str)
print(names)
```

```
[[ 1035.7  1042.9  1048.7  1056.9  1068.2  1077.   1083.2  1092.3  1097.8
   1106.2  1113.5  1127.9]
 [  870.5   876.6   882.7   889.6   894.5   898.6   902.4   905.1   910.7
    916.2   918.7   923.7]
 [ 6116.8  6183.2  6249.3  6309.6  6366.   6420.5  6461.   6508.1  6555.3
   6602.2  6670.9  6702.5]
 [ 2845.2  2883.4  2919.1  2955.5  2990.9  3025.7  3059.2  3091.3  3113.3
   3145.1  3196.2  3220.3]
 [ 2149.4  2164.8  2178.5  2198.   2217.4  2237.2  2263.   2288.5  2310.2
   2329.5  2300.9  2319.2]
 [ 3917.4  3938.5  3957.9  3982.9  4007.2  4034.   4056.8  4077.1  4090.4
   4103.2  4135.3  4147. ]
 [ 2440.2  2459.7  2474.   2496.1  2515.6  2550.9  2579.1  2600.1  2603.2
   2616.1  2627.3  2637.1]
 [ 3488.9  3510.7  3526.2  3538.9  3557.6  3576.8  3605.1  3628.5  3642.
   3660.8  3698.1  3715.4]
 [ 1283.4  1287.2  1289.4  1294.7  1298.8  1301.4  1304.4  1305.5  1306.6
```

```
   1313.1  1321.6  1327.1]
 [ 6671.7  6733.9  6767.5  6800.7  6831.3  6868.4  6908.1  6948.4  6983.1
   7009.1  7069.3  7097. ]
 [ 4235.9  4261.4  4285.9  4313.3  4341.2  4369.6  4400.1  4422.3  4446.9
   4467.5  4501.2  4519.8]
 [ 5660.7  5744.   5817.5  5870.   5937.9  5999.6  6054.   6109.2  6152.2
   6205.5  6278.4  6325.2]
 [ 2999.8  3039.   3066.9  3099.2  3126.9  3164.6  3210.6  3237.1  3260.8
   3283.6  3304.6  3321. ]
 [ 3761.4  3801.9  3827.   3857.2  3893.7  3938.6  3981.   4026.   4070.6
   4117.   4164.5  4212. ]
 [ 8423.6  8534.   8579.8  8620.4  8652.6  8701.2  8747.   8809.7  8871.5
   8921.7  8975.5  9024. ]
 [ 8564.4  8687.   8811.5  8914.6  9005.   9108.8  9203.1  9292.7  9373.7
   9446.3  9526.5  9603.2]
 [ 5373.5  5446.8  5513.6  5590.5  5656.8  5727.1  5776.4  5838.8  5890.6
   5942.5  5936.   5956.7]
 [ 6110.6  6167.   6209.2  6248.   6305.9  6356.7  6403.9  6444.1  6482.2
   6520.6  6515.5  6539.8]
 [ 6246.3  6349.   6463.2  6581.6  6691.5  6788.7  6896.8  7013.7  7115.6
   7298.9  7498.5  7565.3]
 [ 4241.6  4294.5  4359.4  4408.8  4455.1  4502.1  4545.5  4588.5  4622.2
   4657.6  4723.6  4758. ]
 [  651.2   661.5   671.3   681.8   691.4   702.4   714.1   724.5   733.3
    743.2   760.9   769.5]
 [10813.4 10886.8 10942.9 11022.4 11084.3 11162.9 11238.2  8264.7  8315.7
   8358.6  8407.5  8436.6]
 [ 3237.   3271.4  3301.   3332.3  3380.6  3419.5  3459.5  3495.5  3536.5
   3582.   3676.6  3710.2]
 [ 3694.5  3734.7  3767.1  3802.1  3837.1  3873.5  3909.4  3944.6  3983.3
   4018.4  4076.6  4106.7]
 [  218.1   221.8   225.3   228.9   232.    235.6   239.3   242.7   245.4
    247.7   251.2   253.7]
 [ 3275.   3309.9  3340.3  3369.7  3401.6  3431.9  3457.7  3482.7  3501.1
   3519.2  3572.2  3589.5]
 [ 2229.9  2258.   2288.1  2318.6  2352.4  2388.4  2427.8  2456.6  2483.6
   2507.4  2533.7  2550.7]
 [  434.8   439.4   443.1   446.3   451.    456.2   462.6   466.5   470.3
    473.2   480.4   483.5]
 [  465.7   473.8   482.3   490.9   503.9   512.4   521.2   528.9   536.6
    543.3   554.3   566. ]
 [ 1498.7  1528.   1554.1  1577.1  1605.3  1637.3  1675.6  1705.8  1733.6
   1763.4  1791.5  1823.9]]
['北京' '天津' '河北' '山西' '内蒙古 ' '辽宁' '吉林' '黑龙江 ' '上海' '江苏' '浙江' '安
徽' '福建' '江西'
 '山东' '河南' '湖北' '湖南' '广东' '广西' '海南' '四川' '贵州' '云南' '西藏' '陕西' '甘
肃' '青海'
 '宁夏' '新疆']
```

```python
# 查看数组属性
print(peoples.ndim)
print(peoples.shape)
print(peoples.size)
print(peoples.itemsize)
print(peoples.dtype)
```

```
2
(30, 12)
360
8
float64
```

```
# 数据中的人数是以万为单位，假设改为以百万为单位，所有数据/100，保留两位小数
peoples_1 = np.round(peoples/100, 2)
print(peoples_1)
```

```
[[ 10.36  10.43  10.49  10.57  10.68  10.77  10.83  10.92  10.98  11.06
   11.14  11.28]
 [  8.7    8.77   8.83   8.9    8.94   8.99   9.02   9.05   9.11   9.16
    9.19   9.24]
 [ 61.17  61.83  62.49  63.1   63.66  64.2   64.61  65.08  65.55  66.02
   66.71  67.03]
 [ 28.45  28.83  29.19  29.56  29.91  30.26  30.59  30.91  31.13  31.45
   31.96  32.2 ]
 [ 21.49  21.65  21.78  21.98  22.17  22.37  22.63  22.88  23.1   23.3
   23.01  23.19]
 [ 39.17  39.38  39.58  39.83  40.07  40.34  40.57  40.77  40.9   41.03
   41.35  41.47]
 [ 24.4   24.6   24.74  24.96  25.16  25.51  25.79  26.    26.03  26.16
   26.27  26.37]
 [ 34.89  35.11  35.26  35.39  35.58  35.77  36.05  36.28  36.42  36.61
   36.98  37.15]
 [ 12.83  12.87  12.89  12.95  12.99  13.01  13.04  13.06  13.07  13.13
   13.22  13.27]
 [ 66.72  67.34  67.68  68.01  68.31  68.68  69.08  69.48  69.83  70.09
   70.69  70.97]
 [ 42.36  42.61  42.86  43.13  43.41  43.7   44.    44.22  44.47  44.68
   45.01  45.2 ]
 [ 56.61  57.44  58.18  58.7   59.38  60.    60.54  61.09  61.52  62.06
   62.78  63.25]
 [ 30.    30.39  30.67  30.99  31.27  31.65  32.11  32.37  32.61  32.84
   33.05  33.21]
 [ 37.61  38.02  38.27  38.57  38.94  39.39  39.81  40.26  40.71  41.17
   41.64  42.12]
 [ 84.24  85.34  85.8   86.2   86.53  87.01  87.47  88.1   88.72  89.22
   89.76  90.24]
 [ 85.64  86.87  88.12  89.15  90.05  91.09  92.03  92.93  93.74  94.46
   95.26  96.03]
 [ 53.74  54.47  55.14  55.9   56.57  57.27  57.76  58.39  58.91  59.42
   59.36  59.57]
 [ 61.11  61.67  62.09  62.48  63.06  63.57  64.04  64.44  64.82  65.21
   65.16  65.4 ]
 [ 62.46  63.49  64.63  65.82  66.92  67.89  68.97  70.14  71.16  72.99
   74.98  75.65]
 [ 42.42  42.94  43.59  44.09  44.55  45.02  45.46  45.88  46.22  46.58
   47.24  47.58]
 [  6.51   6.62   6.71   6.82   6.91   7.02   7.14   7.24   7.33   7.43
    7.61   7.7 ]
 [108.13 108.87 109.43 110.22 110.84 111.63 112.38  82.65  83.16  83.59
   84.08  84.37]
```

```
 [ 32.37  32.71  33.01  33.32  33.81  34.2   34.6   34.96  35.36  35.82
   36.77  37.1 ]
 [ 36.94  37.35  37.67  38.02  38.37  38.74  39.09  39.45  39.83  40.18
   40.77  41.07]
 [  2.18   2.22   2.25   2.29   2.32   2.36   2.39   2.43   2.45   2.48
    2.51   2.54]
 [ 32.75  33.1   33.4   33.7   34.02  34.32  34.58  34.83  35.01  35.19
   35.72  35.9 ]
 [ 22.3   22.58  22.88  23.19  23.52  23.88  24.28  24.57  24.84  25.07
   25.34  25.51]
 [  4.35   4.39   4.43   4.46   4.51   4.56   4.63   4.66   4.7    4.73
    4.8    4.84]
 [  4.66   4.74   4.82   4.91   5.04   5.12   5.21   5.29   5.37   5.43
    5.54   5.66]
 [ 14.99  15.28  15.54  15.77  16.05  16.37  16.76  17.06  17.34  17.63
   17.92  18.24]]
```

2-2 数组的索引

一维数组的索引

```python
import numpy as np
arr = np.array([1,2,3,4,5,6,7,8,9])
# 一维数组的正序索引
print(arr[0])
print(arr[3])
# 一维数组的逆序索引
print(arr[-1])
print(arr[-3])
# 一维数组的花式索引
print(arr[[1,3,5]])
```

```
1
4
9
7
[2 4 6]
```

二维数组的索引

```python
# 二维数组的索引
arr2 = np.array(range(16)).reshape(4,4)
print(arr2)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

```
# 数组某一行
print(arr2[0])
# 数组的某行某列
print(arr2[1,2])
# 二维数组的花式索引
print(arr2[[0,2],[1,3]])
```

```
[0 1 2 3]
6
[ 1 11]
```

项目：省市人口数据的读取

```
import numpy as np
# 从文本文件读取数组数据
peoples = np.loadtxt("tmp/人口.csv",skiprows=2,delimiter=",",usecols=
(range(1,13)),encoding="utf8")
# 读入全国各省市名称，文件的第一列（跳过前两行）
names = np.loadtxt("tmp/人口.csv",skiprows=2,delimiter=",",usecols=
(0),encoding="utf8", dtype=str)
#  读取北京的数据（第一行）
print(peoples[0])
# 读取上海1994年的数据(上海在第9行)
print(peoples[8,4])
# 读取北京、上海、广东的数据(一维花式索引)
print(peoples[[0,8,18]])
# 读取北京1991、上海1993、广东1995年的数据(二维花式索引)
print(peoples[[0,8,18],[1,3,5]])
```

```
[1035.7 1042.9 1048.7 1056.9 1068.2 1077.  1083.2 1092.3 1097.8 1106.2
 1113.5 1127.9]
1298.8
[[1035.7 1042.9 1048.7 1056.9 1068.2 1077.  1083.2 1092.3 1097.8 1106.2
  1113.5 1127.9]
 [1283.4 1287.2 1289.4 1294.7 1298.8 1301.4 1304.4 1305.5 1306.6 1313.1
  1321.6 1327.1]
 [6246.3 6349.  6463.2 6581.6 6691.5 6788.7 6896.8 7013.7 7115.6 7298.9
  7498.5 7565.3]]
[1042.9 1294.7 6788.7]
```

2-3 数组的切片

一维数组的切片

```
# 一维数组切片
import numpy as np
arr1 = np.arange(10)
print(arr1)
```

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

```
# 数组变量[开始:结束:步长]，步长缺省为1，
#省略开始默认为0，省略结束默认切片到最后一个元素（包含最后一个）
# 序号3到序号6的切片（不含序号6）
print(arr1[3:6])
# 从开始到序号6（不含序号6）
print(arr1[:6])
# 从序号3到结束（含最后一个）
print(arr1[3:])
# 序号2到序号8的切片,步长为2
print(arr1[2:8:2])
# 全部切片
print(arr1[:])
```

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

二维数组的切片

```
# 二维数组切片
import numpy as np
arr2 = np.arange(20).reshape(4,5)
print(arr2)
```

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

```
#  数组变量[行开始:行结束:行步长，列开始:列结束，列步长]
# 行序号1到3，列序号2:4
print(arr2[1:3,2:4])
# 前3行所有列
print(arr2[:3,:])
# 序号1到4列所有行
print(arr2[:,1:4])
# 取某一列
print(arr2[:,2])
```

```
[[ 7  8]
 [12 13]]
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
[[ 1  2  3]
 [ 6  7  8]
 [11 12 13]
 [16 17 18]]
[ 2  7 12 17]
```

项目：人口数据的切片

```
import numpy as np
# 从文本文件读取数组数据
peoples = np.loadtxt("tmp/人口.csv",skiprows=2,delimiter=",",usecols=
(range(1,13)),encoding="utf8")
# 读入全国各省市名称，文件的第一列（跳过前两行）
names = np.loadtxt("tmp/人口.csv",skiprows=2,delimiter=",",usecols=
(0),encoding="utf8", dtype=str)
```

```
# 前五行数据是哪些省市
print(names[:5])
```

```
['北京' '天津' '河北' '山西' '内蒙古 ']
```

```
# 1990年各省市人口数据（序号为0的列）
print(peoples[:,0])
```

```
[ 1035.7   870.5  6116.8  2845.2  2149.4  3917.4  2440.2  3488.9  1283.4
  6671.7  4235.9  5660.7  2999.8  3761.4  8423.6  8564.4  5373.5  6110.6
  6246.3  4241.6   651.2 10813.4  3237.   3694.5   218.1  3275.   2229.9
   434.8   465.7  1498.7]
```

```
# 山西省历年人口数据（序号为3的行）
print(peoples[3,:])
```

```
[2845.2 2883.4 2919.1 2955.5 2990.9 3025.7 3059.2 3091.3 3113.3 3145.1
 3196.2 3220.3]
```

```
# 河北、山西、内蒙古1994年到1999年的人口数据（行序号2到4，列序号4到9）
print(peoples[2:5,4:10])
```

```
[[6366.  6420.5 6461.  6508.1 6555.3 6602.2]
 [2990.9 3025.7 3059.2 3091.3 3113.3 3145.1]
 [2217.4 2237.2 2263.  2288.5 2310.2 2329.5]]
```

2-4 数组的通用函数（ufnc）

ufunc 函数全称为通用函数（universal function），是一种能够对数组中所有元素进行操 作的函数。ufunc 函数是针对数组进行操作的，并且都以NumPy 数组作为输出，因此不需 要对数组的每一个元素都进行操作。对一个数组进行重复运算时，使用ufunc 函数比使用 math 库中的函数效率要高很多。

```
import numpy as np
arr2 = np.arange(20).reshape(4,5)
print(arr2)
```

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

```
# 全部求和
print(arr2.sum())
# 按列求和
print(arr2.sum(axis=1))
# 按行求和
print(arr2.sum(axis=0))
# 求最大最小值
print(arr2.min(), arr2.max())
# 求最大最小值所在位置
print(arr2.argmin(), arr2.argmax())
# 找出某个值的位置索引
print(np.argwhere(arr2==7))
```

```
190
[10 35 60 85]
[30 34 38 42 46]
0 19
0 19
[[1 2]]
```

项目：人口数据的统计计算

```
#1990年全国人口总和(知识点：切片取一列、通用函数)
#注意：np.sum()和peoples[:,0].sum()两种调用方式
#print(np.sum(peoples[:,0]))
print(peoples[:,0].sum())
#2001年全国人口总和
print(peoples[:,-1].sum())
#2001年各省市人口的均值和方差（知识点：切片、通用函数mean和std）
print(peoples[:,-1].mean())
print(peoples[:,-1].std())
```

```
112955.3
121332.49999999999
4044.416666666666
2638.70154824721
```

```
#广东1990年人口(知识点：argwhere、索引)
ind = np.argwhere(names=="广东")[0][0]
```

```
print(peoples[ind,0])
# 广东2001年人口
print(peoples[ind,-1])
#广东人口每年增长值（知识点：数组切片）
print(peoples[ind,1:] - peoples[ind,:-1])
#东北三省1990年人口之和（知识点：通用函数sum等）
print(peoples[5:8,0].sum())
#东北三省2001年人口之和
print(peoples[5:8,-1].sum())
#东北人口每年增长值
print(peoples[5:8,1:] - peoples[5:8,:-1])
```

```
6246.3
7565.3
[102.7 114.2 118.4 109.9  97.2 108.1 116.9 101.9 183.3 199.6  66.8]
9846.5
10499.5
[[21.1 19.4 25.  24.3 26.8 22.8 20.3 13.3 12.8 32.1 11.7]
 [19.5 14.3 22.1 19.5 35.3 28.2 21.   3.1 12.9 11.2  9.8]
 [21.8 15.5 12.7 18.7 19.2 28.3 23.4 13.5 18.8 37.3 17.3]]
```

```
#1990年人口最多的省份（知识点：切片、argmax）
p2001 = peoples[:,0]
ind = np.argmax(p2001) #最大值的下标
print(names[ind])
##1990年人口最少的省份（知识点：切片、argmin）
ind = np.argmin(p2001) #最大值的下标
print(names[ind])
#2001年人口最大的值
p2001 = peoples[:,-1]
print(np.max(p2001))
#2001年人口最多的省份
ind = np.argmax(p2001) #最大值的下标
print(names[ind])
##2001年人口最少的省份
ind = np.argmin(p2001) #最大值的下标
print(names[ind])
```

```
四川
西藏
9603.2
河南
西藏
```