Numpy中的聚合操作

In [1]:
```
import numpy as np
```

In [2]:
```
x = np.random.random(100)
x
```

Out[2]:
```
array([0.83070488, 0.10377653, 0.86798037, 0.87345999, 0.36335185,
       0.29204955, 0.20396895, 0.43117501, 0.84694152, 0.78917441,
       0.98037256, 0.87453556, 0.94749672, 0.33023242, 0.91605717,
       0.88410896, 0.07366249, 0.89339501, 0.91195063, 0.77474178,
       0.96647958, 0.02250557, 0.02683088, 0.52596375, 0.80316678,
       0.70171046, 0.92778364, 0.14829842, 0.40947151, 0.83323356,
       0.03374536, 0.17451065, 0.53699872, 0.28127525, 0.08339391,
       0.02912533, 0.35179514, 0.45790103, 0.71627623, 0.03705884,
       0.18668608, 0.44053849, 0.49074088, 0.14107436, 0.87461207,
       0.23181681, 0.75801603, 0.17805497, 0.94422599, 0.73853698,
       0.94534336, 0.86211233, 0.68849968, 0.19678275, 0.95506773,
       0.60629861, 0.02171716, 0.19421199, 0.6824897 , 0.9699418 ,
       0.55749062, 0.47211211, 0.75756314, 0.22447047, 0.22813579,
       0.73618626, 0.42108869, 0.3823739 , 0.36385779, 0.38401728,
       0.14541901, 0.66286773, 0.81841185, 0.74944538, 0.18249296,
       0.34479754, 0.86962713, 0.76106057, 0.05508117, 0.91809021,
       0.00832434, 0.05745199, 0.58762447, 0.9740064 , 0.24089121,
       0.34387339, 0.21369638, 0.10775419, 0.32583031, 0.1757822 ,
       0.07428164, 0.13753188, 0.66784333, 0.24598231, 0.58240694,
       0.45628319, 0.5494578 , 0.46021338, 0.69449794, 0.25251885]
)
```

求和

In [3]:
```
sum(x)
```

Out[3]:
```
49.55427045739091
```

In [4]:
```
np.sum(x)
```

Out[4]:
```
49.55427045739091
```

In [5]:
```
arr = np.random.random(10**7)
```

In [6]:
```
%timeit sum(arr)
```

```
989 ms ± 5.11 ms per loop (mean ± std. dev. of 7 runs, 1 loop each
)
```

In [7]:
```
%timeit np.sum(arr)
```

```
5.2 ms ± 39.5 µs per loop (mean ± std. dev. of 7 runs, 100 loops e
ach)
```

In [8]:
```
np.min(x)
```

Out[8]:
```
0.00832433878452088
```

In [9]:
```
np.max(x)
```

Out[9]:
```
0.9803725573373384
```

In [10]:
```
x.sum()        # 不推荐这种写法，不能明显看出来使用了numpy
```

Out[10]:
```
49.55427045739091
```

In [11]:
```
X = np.arange(16).reshape(4, -1)
```

In [12]:
```
X
```

Out[12]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [13]:
```
np.sum(X)
```

Out[13]:
```
120
```

In [14]:
```
np.sum(X, axis=0)    # 求每一列的和
```

Out[14]:
```
array([24, 28, 32, 36])
```

In [15]:
```
np.sum(X, axis=1)      # 求每一行的和
```

Out[15]:
```
array([ 6, 22, 38, 54])
```

相乘

In [16]:
```
np.prod(X)
```

Out[16]:
```
0
```

In [17]:
```
np.prod(X + 1)
```

Out[17]:
```
20922789888000
```

In [18]:
```
X + 1
```

Out[18]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

In [19]:
```
np.prod(X+1, axis=1)
```

Out[19]:
```
array([   24,  1680, 11880, 43680])
```

求平均值

In [20]:
```
np.mean(X)
```

Out[20]:
```
7.5
```

In [21]:
```
X
```

Out[21]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [23]:
```
np.mean(X, axis=0)
```

Out[23]:
```
array([6., 7., 8., 9.])
```

求中位数

In [25]:
```
np.median(X, axis=0)
```

Out[25]:
```
array([6., 7., 8., 9.])
```

案例:

In [26]:
```
m = np.array([8000, 9000, 14000, 12000, 10000, 90000])
```

In [27]:
```
np.mean(m)
```

Out[27]:
```
23833.333333333332
```

In [28]:
```
np.median(m)
```

Out[28]:
```
11000.0
```

百分位点

In [29]:
```
np.percentile(m, 50)
```

Out[29]: 11000.0

In [30]:
```
np.percentile(m, 100)
```

Out[30]: 90000.0

In [31]:
```
np.max(m)
```

Out[31]: 90000

In [32]:
```
for i in [0, 25, 50, 75, 100]:
    print(i, np.percentile(m, i))
```

```
0 8000.0
25 9250.0
50 11000.0
75 13500.0
100 90000.0
```

方差

In [38]:
```
np.var(m)
```

Out[38]: 879472222.2222223

In [39]:
```
np.sum((m-np.mean(m))**2)/np.size(m)
```

Out[39]: 879472222.2222223

标准差

In [40]:
```
np.std(m)
```

Out[40]: 29655.896921560514

In [41]:
```
np.var(m)**0.5
```

Out[41]: 29655.896921560514

案例：

In [42]:
```
a = np.random.normal(0, 1, size=100000)          # 构造平均值为0,  标准差为1的标准分布数组
```

In [43]:
```
np.mean(a)
```

Out[43]: 0.006222418715093957

In [44]:
```
np.std(a)
```

Out[44]: 1.0018694397195966