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

In [3]:
```
# boolean index
names = np.array(["a","b","c","a","b"])
data = np.random.randn(5,4)
print(names)
print(data)
```

```
['a' 'b' 'c' 'a' 'b']
[[ 0.81861299 -1.04386309  1.08290844  1.24947473]
 [-0.41444718 -1.88501587 -0.745801   -0.76376004]
 [-0.86987841  0.19856993 -0.08353096  1.26229515]
 [ 0.6431263  -0.16565641  1.17706194  0.46410421]
 [-0.60918971  0.29659153 -0.01256779  1.40143141]]
```

In [4]:
```
names=="a"
```

Out[4]:
```
array([ True, False, False,  True, False])
```

In [5]:
```
data[names=="a"]
```

Out[5]:
```
array([[ 0.81861299, -1.04386309,  1.08290844,  1.24947473],
       [ 0.6431263 , -0.16565641,  1.17706194,  0.46410421]])
```

In [6]:
```
a = np.array([[1,2,3],[2,4,5],[7,23,12]])
a
```

Out[6]:
```
array([[ 1,  2,  3],
       [ 2,  4,  5],
       [ 7, 23, 12]])
```

In [9]:
```
print(a<5)
```

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

In [10]:
```
a[a<5]
```

Out[10]:
```
array([1, 2, 3, 2, 4])
```

In [11]:
```
a[a>=5]
```

Out[11]:
```
array([ 5,  7, 23, 12])
```

In [13]:
```
a[a%2==0]
```

Out[13]:
```
array([ 2,  2,  4, 12])
```

In [14]:
```
a[(a>2) & (a<15)]
```

Out[14]:
```
array([ 3,  4,  5,  7, 12])
```

In [15]:
```
a[(a>5) | (a==5)]
```

Out[15]:
```
array([ 5,  7, 23, 12])
```

In [18]:
```
arr = np.array([41,34,23,45,67,89])

x = [True,False,True,False,True,True]

newarr = arr[x]

print(newarr)
```

```
[41 23 67 89]
```

- using generators
- fromiter(iterable,datatype)

In [22]:
```
iterable = (x*x for x in range(5) for y in range(10) if y<x)
np.fromiter(iterable,float)
```

Out[22]:
```
array([ 1.,  4.,  4.,  9.,  9.,  9., 16., 16., 16., 16.])
```

In [23]:
```
iterable = (x*x for x in range(5) for y in range(10) if y<x)
np.fromiter(iterable,int)
```

Out[23]:
```
array([ 1,  4,  4,  9,  9,  9, 16, 16, 16, 16])
```

In [25]:
```
iterable = (x*x for x in range(5) for y in range(10) if y<x)
np.fromiter(iterable,object)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-abcd6ac7a3d0> in <module>
      1 iterable = (x*x for x in range(5) for y in range(10) if y<x)
----> 2 np.fromiter(iterable,object)

ValueError: cannot create object arrays from iterator
```

In [21]:
```
for x in range(3):#0 to 2
    for y in range(5): # 0 to 4
        if y<x: #0<0,1<0,2<0,3<0,4<0#0<1,1<1,2<1,3<1,4<1
            print(x*x)
```

```
1
4
4
```

In [26]:
```
arr
```

Out[26]:
```
array([41, 34, 23, 45, 67, 89])
```

In [27]:
```
np.where(arr<40)
```

Out[27]:
```
(array([1, 2], dtype=int64),)
```

In [29]:
```
arr[np.where(arr<40)]
```

Out[29]:
```
array([34, 23])
```

In [38]:
```
arr1 = np.array([11,10,23,45])
np.nonzero(arr1<20)
```

Out[38]:
```
(array([0, 1], dtype=int64),)
```

In [39]:
```
arr1[np.nonzero(arr1<20)]
```

Out[39]:
```
array([11, 10])
```

In [ ]:
```

```

Statistical ang aggregate functions

In [40]:
```
arr1 = np.array([[10,20,30],[34,23,14]])
arr2 = np.array([[23,45,12,67],[23,1,45,67]])
print(arr1)
print(arr2)
```

```
[[10 20 30]
 [34 23 14]]
[[23 45 12 67]
 [23  1 45 67]]
```

In [42]:
```
print(arr1.sum())
print(arr2.sum())
```

```
131
283
```

In [43]:
```
print(arr1.sum(axis=0))
```

```
[44 43 44]
```

In [44]:
```
print(arr1.sum(axis=1))
```

```
[60 71]
```

In [45]:
```
arr1.sum(0)
```

Out[45]:
```
array([44, 43, 44])
```

In [46]:
```
np.average(arr1)
```

Out[46]:
```
21.833333333333332
```

In [47]:
```
np.average(arr1,0)
```

Out[47]:
```
array([22. , 21.5, 22. ])
```

In [48]:
```
np.average(arr1,axis=0)
```

Out[48]:
```
array([22. , 21.5, 22. ])
```

In [49]:
```
np.average(arr1,1)
```

Out[49]:
```
array([20.        , 23.66666667])
```

In [50]:
```

np.prod(arr2)
```

Out[50]:
```
1870173452
```

In [51]:
```
np.prod(arr2,1)
```

Out[51]:
```
array([832140,  69345])
```

In [52]:
```
arr2
```

Out[52]:
```
array([[23, 45, 12, 67],
       [23,  1, 45, 67]])
```

In [53]:
```
print(arr1.min())
print(arr2.min())
```

```
10
1
```

In [55]:
```
print(arr1.min(axis=0))
print(arr2.min(0))
```

```
[10 20 14]
[23  1 12 67]
```

In [56]:
```
print(arr1.max())
print(arr2.max())
```

```
34
67
```

In [57]:
```
print(arr1.max(0))
```

```
[34 23 30]
```

In [58]:
```
arr1
```

Out[58]:
```
array([[10, 20, 30],
       [34, 23, 14]])
```

In [59]:
```
arr1.mean()
```

Out[59]:
```
21.833333333333332
```

In [60]:
```
arr1.mean(axis=0)
```

Out[60]:
```
array([22. , 21.5, 22. ])
```

In [61]:
```
arr1.mean(axis=1)
```

Out[61]:
```
array([20.        , 23.66666667])
```

In [63]:
```
np.median(arr1)
```

Out[63]:
```
21.5
```

In [64]:
```
np.median(arr1,0)
```

Out[64]:
```
array([22. , 21.5, 22. ])
```

In [65]:
```
#variance = (item1-mean)^2+-----+(itemN-mean)^2/total items
arr1.var()
```

Out[65]:
```
70.13888888888887
```

In [66]:
```
arr1.var(0)
```

Out[66]:
```
array([144.  ,   2.25,  64.  ])
```

In [67]:
```
#square root of variance--std
arr1.std()
```

Out[67]:
```
8.374896350934074
```

In [68]:
```
arr1.std(0)
```

Out[68]:
```
array([12. ,  1.5,  8. ])
```

In [70]:
```
arr1.cumsum()
```

Out[70]:
```
array([ 10,  30,  60,  94, 117, 131], dtype=int32)
```

In [69]:
```
arr1
```

Out[69]:
```
array([[10, 20, 30],
       [34, 23, 14]])
```

In [71]:
```
arr1.cumsum(0)
```

Out[71]:
```
array([[10, 20, 30],
       [44, 43, 44]], dtype=int32)
```

In [72]:
```
arr1.cumprod()
```

Out[72]:
```
array([      10,      200,     6000,   204000,  4692000, 65688000],
      dtype=int32)
```

In [73]:
```
arr1.cumprod(1)
```

Out[73]:
```
array([[   10,   200,  6000],
       [   34,   782, 10948]], dtype=int32)
```

In [74]:
```
arr1
```

Out[74]:
```
array([[10, 20, 30],
       [34, 23, 14]])
```

In [75]:
```
arr1.min()
```

Out[75]:
```
10
```

In [76]:
```
arr1.argmin()
```

Out[76]:
```
0
```

In [77]:
```
arr1.argmin(0)
```

Out[77]:
```
array([0, 0, 1], dtype=int64)
```

In [78]:
```
arr1.argmax()
```

Out[78]:
```
3
```

In [79]:
```
arr1.argmax(axis=1)
```

Out[79]:
```
array([2, 0], dtype=int64)
```

In [80]:
```
np.corrcoef(arr1)
```

Out[80]:
```
array([[ 1.        , -0.99833749],
       [-0.99833749,  1.        ]])
```

In [81]:
```
np.corrcoef(arr2)
```

Out[81]:
```
array([[1.        , 0.29111125],
       [0.29111125, 1.        ]])
```

Saving data

In [84]:
```
data = np.linspace(0,100,200)
data
```

...

In [83]:
```
np.savetxt("data1.dat",data)
```

In [85]:
```
np.loadtxt("data1.dat")
```

...

In [86]:
```
#arrays to strings
np.array_str(np.arange(3))
```

Out[86]:
```
'[0 1 2]'
```

In [87]:
```
hex(67)
```

Out[87]:
```
'0x43'
```

In [88]:
```
bin(23)
```

Out[88]:
```
'0b10111'
```

In [ ]:
```

```

Pandas

- pandas is also one of the library in python which is used to perform data analysis
- reduce the complexity and make our works easier

- data analysis and data manipulation
- data manipulation --> adding,deleting,modification
- data analysis --> cleaning,processing,modeling and inspecting data

- create data in two type:
  - Series
  - DataFrame

In [91]:
```
import pandas as pd
```

In [94]:
```
# pd.Series(list)
a = pd.Series([1,2,3,4])
a
```

Out[94]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

In [95]:
```
# creating series by using numpy
b = pd.Series(np.array([1,2,4,5]))
b
```

Out[95]:
```
0    1
1    2
2    4
3    5
dtype: int32
```

In [97]:
```
# creating series by using numpy
b = pd.DataFrame(np.array([[1,2,4,5],[7,8,9,7]]))
b
```

Out[97]:

	0	1	2	3
0	1	2	4	5
1	7	8	9	7

In [98]:
```
# creating data series by using dictionary
d = {"a":1,"b":3}
pd.Series(d)
```

Out[98]:
```
a    1
b    3
dtype: int64
```

In [99]:
```
pd.Series([1,2,4,5],index=["a","b","c","d"])
```

Out[99]:
```
a    1
b    2
c    4
d    5
dtype: int64
```

In [100]:
```
a = np.array([[1,2,3],[9,8,7]])
```

In [101]:
```
a
```

Out[101]:
```
array([[1, 2, 3],
       [9, 8, 7]])
```

In [ ]:
```
axis=0-rows
axis=1-columns
```

In [102]:
```
a.min()
```

Out[102]:
```
1
```

In [104]:
```
a.min(axis=1)
```

Out[104]:
```
array([1, 7])
```

In [ ]:
```

```