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

In [2]:
```
a = np.array([1,2,3,4])
a
```

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

In [3]:
```
a.ndim
```

Out[3]:
```
1
```

In [4]:
```
# 2-d array

b = np.array([[1,2,3,4],[5,6,7,8]])
b
```

Out[4]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [5]:
```
b.ndim
```

Out[5]:
```
2
```

In [6]:
```
b.nbytes
```

Out[6]:
```
32
```

In [7]:
```
b.itemsize
```

Out[7]:
```
4
```

In [8]:
```
len(b)
```

Out[8]:
```
2
```

In [9]:
```
b.size
```

Out[9]:
```
8
```

- 3-d array

In [10]:
```
c = np.array([[[1,2,3,4],[5,6,7,8]]])
c
```

Out[10]:
```
array([[[1, 2, 3, 4],
        [5, 6, 7, 8]]])
```

```
In [11]: c.ndim

Out[11]: 3

In [12]: c = np.array([[[1,2,3,4],[5,6,7,8]],[[7,8,9,10],[1,2,3,4]]])

In [13]: c

Out[13]: array([[[ 1,  2,  3,  4],
                 [ 5,  6,  7,  8]],

                [[ 7,  8,  9, 10],
                 [ 1,  2,  3,  4]]])

In [14]: c.ndim

Out[14]: 3

In [15]: a

Out[15]: array([1, 2, 3, 4])

In [16]: a[2]

Out[16]: 3

In [17]: a[-1]

Out[17]: 4

In [18]: a[1:]

Out[18]: array([2, 3, 4])

In [19]: # 2-d array

In [20]: b

Out[20]: array([[1, 2, 3, 4],
                [5, 6, 7, 8]])

In [21]: b[0]

Out[21]: array([1, 2, 3, 4])

In [22]: b[1]

Out[22]: array([5, 6, 7, 8])
```

In [23]:
```
b[0][1]
```
Out[23]:
```
2
```

In [24]:
```
b[0,1]#row index and column index
```
Out[24]:
```
2
```

In [25]:
```
b
```
Out[25]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [26]:
```
b[:,-2:]
```
Out[26]:
```
array([[3, 4],
       [7, 8]])
```

In [27]:
```
b[1:,-2:]
```
Out[27]:
```
array([[7, 8]])
```

In [28]:
```
# 3-d array
```

In [29]:
```
c # position,row index,column index
```
Out[29]:
```
array([[[ 1,  2,  3,  4],
        [ 5,  6,  7,  8]],

       [[ 7,  8,  9, 10],
        [ 1,  2,  3,  4]]])
```

In [30]:
```
c[0]
```
Out[30]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [31]:
```
c[0][1][3]
```
Out[31]:
```
8
```

In [32]:
```
c[0,1,3]
```
Out[32]:
```
8
```

```
In [33]: c

Out[33]: array([[[ 1,  2,  3,  4],
                 [ 5,  6,  7,  8]],

                [[ 7,  8,  9, 10],
                 [ 1,  2,  3,  4]]])

In [34]: c[1,0,3]

Out[34]: 10

In [35]: c[:,:,3:]

Out[35]: array([[[ 4],
                 [ 8]],

                [[10],
                 [ 4]]])

In [39]: c[:,:,-1:]

Out[39]: array([[[ 4],
                 [ 8]],

                [[10],
                 [ 4]]])

In [37]: # reshaping the arrays

In [38]: a

Out[38]: array([1, 2, 3, 4])

In [44]: a = np.array([1,2,3,4,5,6,7,8,9,10])

In [45]: a

Out[45]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])

In [46]: a.shape

Out[46]: (10,)
```

In [47]:
```
a.reshape(5,3)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-47-9bee1d10ce69> in <module>
----> 1 a.reshape(5,3)

ValueError: cannot reshape array of size 10 into shape (5,3)
```

In [48]:
```
a.reshape(5,2)
```

Out[48]:
```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10]])
```

In [49]:
```
a.reshape(2,5)
```

Out[49]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [50]:
```
a.reshape(5,2,1)
```

Out[50]:
```
array([[[ 1],
        [ 2]],

       [[ 3],
        [ 4]],

       [[ 5],
        [ 6]],

       [[ 7],
        [ 8]],

       [[ 9],
        [10]]])
```

In [51]:
```
a.reshape(10,1)
```

Out[51]:
```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]])
```

```
In [52]: b

Out[52]: array([[1, 2, 3, 4],
                [5, 6, 7, 8]])
```

```
In [53]: b.shape

Out[53]: (2, 4)
```

```
In [54]: b.reshape(4,2,1)

Out[54]: array([[[1],
                 [2]],

                [[3],
                 [4]],

                [[5],
                 [6]],

                [[7],
                 [8]]])
```

```
In [55]: a

Out[55]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
In [56]: a.reshape(5,-1)# -1 is unknown

Out[56]: array([[ 1,  2],
                [ 3,  4],
                [ 5,  6],
                [ 7,  8],
                [ 9, 10]])
```

```
In [57]: a.reshape(-1,5)

Out[57]: array([[ 1,  2,  3,  4,  5],
                [ 6,  7,  8,  9, 10]])
```

```
In [58]: a.reshape(6,-1)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-58-155e25d74457> in <module>
----> 1 a.reshape(6,-1)

ValueError: cannot reshape array of size 10 into shape (6,newaxis)
```

```
In [59]: a.reshape(-1,-1)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-59-aa43799da6cd> in <module>
----> 1 a.reshape(-1,-1)

ValueError: can only specify one unknown dimension
```

```
In [64]: a.reshape(2,5,2)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-64-cbb784d34fc5> in <module>
----> 1 a.reshape(2,5,2)

ValueError: cannot reshape array of size 10 into shape (2,5,2)
```

```
In [66]: a.reshape(2,5,1)

Out[66]: array([[[ 1],
                 [ 2],
                 [ 3],
                 [ 4],
                 [ 5]],

                [[ 6],
                 [ 7],
                 [ 8],
                 [ 9],
                 [10]]])
```

```
In [67]: a.reshape(2,1,5)

Out[67]: array([[[ 1,  2,  3,  4,  5]],

                [[ 6,  7,  8,  9, 10]]])
```

```
In [68]: # concatenation
```

```
In [69]: # 1-d array
```

```
In [72]: a1 = np.array([1,2,3,4])
         a2 = np.array([3,4,5,6])
         a3 = np.array([5,6,7,8])
         np.concatenate((a1,a2))

Out[72]: array([1, 2, 3, 4, 3, 4, 5, 6])
```

In [71]:
```
help(np.concatenate)
```

...

In [73]:
```
#2 - d array
b1 = np.array([[1,2],[3,4]])
b2 = np.array([[4,5],[9,10]])
np.concatenate((b1,b2))
```

Out[73]:
```
array([[ 1,  2],
       [ 3,  4],
       [ 4,  5],
       [ 9, 10]])
```

In [76]:
```
#3d - array
c1 = b1.reshape(2,2,1)
c2 = b2.reshape(2,2,1)
```

In [77]:
```
np.concatenate((c1,c2))
```

Out[77]:
```
array([[[ 1],
        [ 2]],

       [[ 3],
        [ 4]],

       [[ 4],
        [ 5]],

       [[ 9],
        [10]]])
```

In [78]:
```
a
```

Out[78]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [80]:
```
np.min(a)# output--- element
```

Out[80]:
```
1
```

In [81]:
```
np.max(a)
```

Out[81]:
```
10
```

In [82]:
```
np.argmin(a)# output-- index value
```

Out[82]:
```
0
```

In [83]:
```
np.median(a)
```

Out[83]:
```
5.5
```

In [84]:
```
np.mean(a)
```
Out[84]:
```
5.5
```
In [85]:
```
np.average(a)
```
Out[85]:
```
5.5
```
In [86]:
```
np.var(a)
```
Out[86]:
```
8.25
```
In [87]:
```
np.std(a)
```
Out[87]:
```
2.8722813232690143
```
In [88]:
```
np.sum(a)
```
Out[88]:
```
55
```
In [89]:
```
np.cumsum(a)
```
Out[89]:
```
array([ 1,  3,  6, 10, 15, 21, 28, 36, 45, 55], dtype=int32)
```
In [90]:
```
b
```
Out[90]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```
In [91]:
```
np.min(b)
```
Out[91]:
```
1
```
In [97]:
```
np.min(b,axis = 1) ## 1--row
```
Out[97]:
```
array([1, 5])
```
In [96]:
```
np.min(b,axis = 0) ## 0 --- columns
```
Out[96]:
```
array([1, 2, 3, 4])
```
In [95]:
```
np.argmax(b,1)
```
Out[95]:
```
array([3, 3], dtype=int64)
```
In [101]:
```
np.min(b[0])
```
Out[101]:
```
1
```

```
In [111]: b[0]

Out[111]: array([1, 2, 3, 4])

In [110]: np.min(b[1])

Out[110]: 5

In [112]: b[1]

Out[112]: array([5, 6, 7, 8])

In [103]: b

Out[103]: array([[1, 2, 3, 4],
                 [5, 6, 7, 8]])

In [108]: np.min(b[0])

Out[108]: 1

In [109]: np.min(b[1])

Out[109]: 5

In [113]: ## satcking-- arranging elements in proper order
          # horizontal stacking
          # vertical stacking

In [114]: b

Out[114]: array([[1, 2, 3, 4],
                 [5, 6, 7, 8]])

In [115]: np.hstack(b)

Out[115]: array([1, 2, 3, 4, 5, 6, 7, 8])

In [116]: a

Out[116]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
In [117]: np.vstack(a)

Out[117]: array([[ 1],
                 [ 2],
                 [ 3],
                 [ 4],
                 [ 5],
                 [ 6],
                 [ 7],
                 [ 8],
                 [ 9],
                 [10]])
```

```
In [118]: np.vstack(b)

Out[118]: array([[1, 2, 3, 4],
                 [5, 6, 7, 8]])
```

```
In [119]: b

Out[119]: array([[1, 2, 3, 4],
                 [5, 6, 7, 8]])
```

```
In [121]: np.sqrt(b)

Out[121]: array([[1.        , 1.41421356, 1.73205081, 2.        ],
                 [2.23606798, 2.44948974, 2.64575131, 2.82842712]])
```

```
In [122]: np.exp(b)

Out[122]: array([[2.71828183e+00, 7.38905610e+00, 2.00855369e+01, 5.45981500e+01],
                 [1.48413159e+02, 4.03428793e+02, 1.09663316e+03, 2.98095799e+03]])
```

```
In [123]: np.log(b)

Out[123]: array([[0.        , 0.69314718, 1.09861229, 1.38629436],
                 [1.60943791, 1.79175947, 1.94591015, 2.07944154]])
```

```
In [124]: np.log2(b)

Out[124]: array([[0.        , 1.        , 1.5849625 , 2.        ],
                 [2.32192809, 2.5849625 , 2.80735492, 3.        ]])
```

```
In [125]: np.log10(b)

Out[125]: array([[0.        , 0.30103   , 0.47712125, 0.60205999],
                 [0.69897   , 0.77815125, 0.84509804, 0.90308999]])
```

```
In [126]: np.remainder(b,4)

Out[126]: array([[1, 2, 3, 0],
                 [1, 2, 3, 0]], dtype=int32)

In [127]: b

Out[127]: array([[1, 2, 3, 4],
                 [5, 6, 7, 8]])

In [128]: np.power(b,5)

Out[128]: array([[    1,    32,   243,  1024],
                 [ 3125,  7776, 16807, 32768]], dtype=int32)

In [129]: np.multiply(b,6)

Out[129]: array([[ 6, 12, 18, 24],
                 [30, 36, 42, 48]])

In [130]: np.divide(b,5)

Out[130]: array([[0.2, 0.4, 0.6, 0.8],
                 [1. , 1.2, 1.4, 1.6]])

In [132]: x = np.array([1+2j,5+6j])

In [133]: x

Out[133]: array([1.+2.j, 5.+6.j])

In [134]: np.real(x)

Out[134]: array([1., 5.])

In [135]: np.imag(x)

Out[135]: array([2., 6.])

In [136]: ## ravel and flatten

In [137]: original = np.array([1,2,3,4])
          rav = original.ravel()
          rav

Out[137]: array([1, 2, 3, 4])

In [138]: rav[0]

Out[138]: 1
```

```
In [139]: rav[0] = 10
```

```
In [140]: rav
```

```
Out[140]: array([10,  2,  3,  4])
```

```
In [141]: original
```

```
Out[141]: array([10,  2,  3,  4])
```

```
In [142]: fla = original.flatten()
          fla
```

```
Out[142]: array([10,  2,  3,  4])
```

```
In [143]: fla[2] = 30
```

```
In [144]: fla
```

```
Out[144]: array([10,  2, 30,  4])
```

```
In [145]: original
```

```
Out[145]: array([10,  2,  3,  4])
```

Random methods

```
In [149]: np.random.random()
          # it returns a value between 0 and 1
```

```
Out[149]: 0.3701898006099572
```

```
In [150]: np.random.random(10)
```

```
Out[150]: array([0.93833579, 0.05513562, 0.02002892, 0.9417876 , 0.08836406,
                 0.51165736, 0.61834679, 0.79020979, 0.43235515, 0.89634256])
```

```
In [153]: np.random.random((2,5,2))
```

```
Out[153]: array([[[0.55628674, 0.48546299],
                  [0.60422187, 0.43556903],
                  [0.49513796, 0.11991529],
                  [0.45505345, 0.64554314],
                  [0.55811126, 0.98493573]],

                 [[0.21417782, 0.22618311],
                  [0.42835868, 0.24248508],
                  [0.81927552, 0.69032868],
                  [0.50518307, 0.1400043 ],
                  [0.94250264, 0.98062634]]])
```

```
In [154]: np.random.rand()
          # it is similar to random

Out[154]: 0.1592553950646859
```

```
In [155]: np.random.randint()

          ---------------------------------------------------------------------------
          TypeError                                 Traceback (most recent call last)
          <ipython-input-155-c77516ad7597> in <module>
          ----> 1 np.random.randint()

          mtrand.pyx in mtrand.RandomState.randint()

          TypeError: randint() takes at least 1 positional argument (0 given)
```

```
In [157]: np.random.randint(5) # range

Out[157]: 3
```

```
In [158]: np.random.randint(10,20)

Out[158]: 15
```

Filtering

```
In [159]: marks = np.array([55,34,89,90,56])
          marks

Out[159]: array([55, 34, 89, 90, 56])
```

```
In [160]: marks>35

Out[160]: array([ True, False,  True,  True,  True])
```

```
In [161]: marks[marks>35]

Out[161]: array([55, 89, 90, 56])
```

```
In [162]: x = np.arange(50,100)
```

```
In [163]: x

Out[163]: array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
                 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
                 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

In [164]:
```
x % 2 == 0
```

Out[164]:
```
array([ True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False,  True, False,  True, False,
        True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False,  True, False,  True, False,
        True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False])
```

In [165]:
```
x[x%2==0]
```

Out[165]:
```
array([50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82,
       84, 86, 88, 90, 92, 94, 96, 98])
```

In [167]:
```
x[(x>60)&(x<90)]
```

Out[167]:
```
array([61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
       78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89])
```

In [171]:
```
x[((x>60) & (x<90) & (x%2==0))]
```

Out[171]:
```
array([62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88])
```

Pandas

- analysize and manipulate the data
- import / export the data

Different types of files (https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html)

- Series - 1d array--list,tuple,numpy array
- data types -- float,int,string/object
- DataFrame - 2d array
- list,tuple,dict,numpy array
- data types -- float,int,string/object

In [172]:
```
import pandas as pd
```

In [173]:
```
# pip install pandas
```

In [178]:
```
s1 = pd.Series([1,2,3,7.9,"a"])
```

In [179]:
```
s1
```

Out[179]:
```
0      1
1      2
2      3
3    7.9
4      a
dtype: object
```

In [180]:
```
s2 = pd.Series((1,2,3))
s2
```

Out[180]:
```
0    1
1    2
2    3
dtype: int64
```

In [182]:
```
s3 = pd.Series(np.array([1,2,3,4]))
```

In [183]:
```
s3
```

Out[183]:
```
0    1
1    2
2    3
3    4
dtype: int32
```

In [184]:
```
s2 = pd.Series((1,2,3),index=["a","b","c"])
s2
```

Out[184]:
```
a    1
b    2
c    3
dtype: int64
```

In [185]:
```
s2.index
```

Out[185]:
```
Index(['a', 'b', 'c'], dtype='object')
```

In [186]:
```
s2.columns
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-186-9007aaaa321c> in <module>
----> 1 s2.columns

~\Anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5061         if (name in self._internal_names_set or name in self._metadata
 or
   5062                 name in self._accessors):
-> 5063             return object.__getattribute__(self, name)
   5064         else:
   5065             if self._info_axis._can_hold_identifiers_and_holds_name(nam
e):

AttributeError: 'Series' object has no attribute 'columns'
```

In [187]:
```
s2["a"]
```

Out[187]:
```
1
```

In [188]:
```
s2["c"]
```

Out[188]:
```
3
```

In [194]:
```
a = np.array([11,2,3])
```

In [195]:
```
np.min(a)
```

Out[195]:
```
2
```

In [196]:
```
np.argmin(a)
```

Out[196]:
```
1
```

In [ ]: