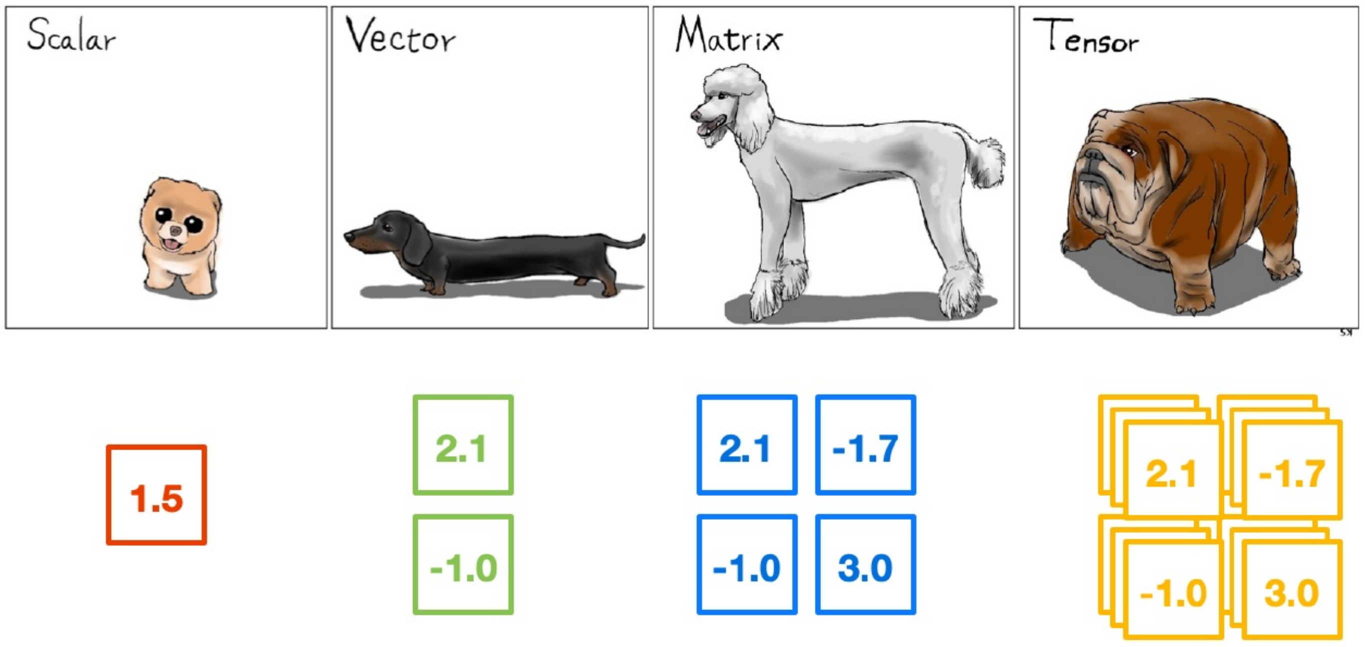
Numpy

- **Basic library used in scientific calculations**
- **Linear algebra, machine learning, data science**
- **Multi-dimensional arrays**
- **Fast access to multidimensional arrays**
- **The difference from the lists is having a fixed size.**



In [1]:

```
import numpy as np
```

In [3]:

```
a = np.array([1,2,3,4,5])    # 1st-degree array, vector
```

In [4]:

```
type(a)
```

Out[4]:

```
numpy.ndarray
```

In [6]:

```
a.shape
```

Out[6]:

```
(5,)
```

In [7]:

```
a.ndim    # return the dimension of an array
```

Out[7]:

```
1
```

In [8]:

```
a
```

Out[8]:

```
array([1, 2, 3, 4, 5])
```

In [9]:

```
print(a[0])
print(a[3])
print(a[2])
```

```
1
4
3
```

In [10]:

```
a[2] = 8

print(a)
```

```
[1 2 8 4 5]
```

In [23]:

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

In [25]:

```
b
```

Out[25]:

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

In [26]:

```
b.ndim
```

Out[26]:

```
2
```

In [27]:

```
b.shape     # 2nd-degree array with 2 row, 4 col.
```

Out[27]:

```
(2, 4)
```

In [30]:

```
b
```

Out[30]:

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

In [31]:

```
print(b[0,0])
print(b[1,0])
print(b[1,1])      #1st item of 1st row
```

```
1
5
6
```

In [32]:

```
print(b[0,0],b[1,0],b[1,1])
```

```
1 5 6
```

In [33]:

```
c = np.array([[1,2,3],
              [4,5,6],
              [7,8,9]])
```

In [35]:

```
c
```

Out[35]:

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

In [36]:

```
c.ndim
```

Out[36]:

```
2
```

In [37]:

```
c.shape
```

Out[37]:

```
(3, 3)
```

In [52]:

```
d = np.array([[[1,2,3],
               [4,5,6],
               [7,8,9]]])
```

In [53]:

```
d
```

Out[53]:

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

In [54]:

```
d.ndim   # 3-dimension array
```

Out[54]:

```
3
```

In [55]:

```
d.shape
```

Out[55]:

```
(1, 3, 3)
```

In [56]:

```
d[0,1,1]
```

Out[56]:

5

Arrays with Special Values

In [65]:

```
#Zero Array


s = np.zeros((2,2))

print(s)
```

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

In [68]:

```
s2 = np.ones((2,3)).astype("int64").dtype

print(s2)
```

```
int64
```

In [71]:

```
s2 = np.ones((2,3))
print(s2)
```

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

In [74]:

```
s3 = np.full((3,3),8)

print(s3)
```

```
[[8 8 8]
 [8 8 8]
 [8 8 8]]
```

In [75]:

```
# It creates a series of randomly determined elements according to the state of the memor
y.

s4 = np.empty((4,5))

print(s4)
```

```
[[9.27679712e-312 6.27463370e-322 0.00000000e+000 0.00000000e+000
  1.42417900e-306]
 [5.30276956e+180 1.57076922e-076 4.57487963e-071 4.66499561e-086
  3.35959356e-143]
 [6.01433264e+175 6.93885958e+218 5.56218858e+180 3.94356143e+180
  3.72782491e-057]
 [7.80064249e-043 6.52055591e-042 9.35008708e-067 4.41198586e-143
  1.50008929e+248]]
```

In [80]:

```
#diagonal array

s5 = np.eye(4)

print(s5)
```

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

In [86]:

```
s7 = np.arange(0,10,1)

print(s7)
```

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

In [93]:

```
s8 = np.linspace(2,3,5)

print(s8)
```

```
[2.   2.25 2.5  2.75 3.  ]
```

In [95]:

```
s6 = np.random.random((5,5))

print(s6)
```

```
[[0.76340755 0.53471369 0.46762585 0.85171004 0.95582832]
 [0.70299103 0.15913386 0.7042347  0.38874101 0.93434943]
 [0.78667396 0.61890382 0.4776483  0.22138073 0.45374143]
 [0.57427484 0.7634074  0.71862918 0.15050893 0.37722183]
 [0.27245849 0.2875156  0.25576563 0.28632725 0.64980458]]
```

In [101]:

```
array_random = np.random.randint(5,10, size = 10)
array_random.shape
```

Out[101]:

```
(10,)
```

In [105]:

```
np.random.randint(5,10, size= (4,4))
```

Out[105]:

```
array([[8, 9, 7, 7],
       [7, 8, 7, 8],
       [8, 6, 8, 6],
       [6, 8, 6, 8]])
```

In [106]:

```
#reshape

d2 = np.random.randint(5,10, size = (5,3))

print(d2)

print(d2.shape)
```

```
[[7 6 6]
 [7 7 9]
 [8 6 8]
 [9 5 8]
 [8 7 9]]
(5, 3)
```

In [110]:

```
d2.reshape(3,5)    #The original matrix and the new one must have the same number of items
.
```

Out[110]:

Out[110]:

```
array([[7, 6, 6, 7, 7],
       [9, 8, 6, 8, 9],
       [5, 8, 8, 7, 9]])
```

In [112]:

```
d2.reshape(15,1)
```

Out[112]:

```
array([[7],
       [6],
       [6],
       [7],
       [7],
       [9],
       [8],
       [6],
       [8],
       [9],
       [5],
       [8],
       [8],
       [7],
       [9]])
```

In [113]:

```
d3 =  np.random.randint(5,10, size = (5,3))

print(d3)
```

```
[[9 9 8]
 [5 8 6]
 [5 5 7]
 [9 7 8]
 [9 9 7]]
```

In [116]:

```
d3 = d3.ravel()

print(d3)
```

```
[9 9 8 5 8 6 5 5 7 9 7 8 9 9 7]
```

In [117]:

```
d3.shape
```

Out[117]:

```
(15,)
```

In [118]:

```
d3.dtype
```

Out[118]:

```
dtype('int32')
```

In [120]:

```
d3.astype("int64").dtype
d3
```

Out[120]:

```
array([9, 9, 8, 5, 8, 6, 5, 5, 7, 9, 7, 8, 9, 9, 7])
```

In [121]:

```
d3 = d3.reshape(3,5)
```

In [129]:

```
d3
```

Out[129]:

```
array([[9, 9, 8, 5, 8],
       [6, 5, 5, 7, 9],
       [7, 8, 9, 9, 7]])
```

In [130]:

```
d3.max()
```

Out[130]:

```
9
```

In [131]:

```
d3.min()
```

Out[131]:

```
5
```

In [132]:

```
d3[::-1]
```

Out[132]:

```
array([[7, 8, 9, 9, 7],
       [6, 5, 5, 7, 9],
       [9, 9, 8, 5, 8]])
```

In [140]:

```
news = np.random.randint(1,100,10)

print(news)
```

```
[79 20 99 85 42 69 54 24 42 10]
```

In [141]:

```
type(news)
```

Out[141]:

```
numpy.ndarray
```

In [142]:

```
news.ndim
```

Out[142]:

```
1
```

In [143]:

```
news.shape
```

Out[143]:

```
(10,)
```

In [144]:

```
news.argmax()
```

Out[144]:

```
2
```

In [145]:

```
news.argmin()
```

Out[145]:

```
9
```

In [146]:

```
news.mean()
```

Out[146]:

```
52.4
```

Stacking

In [147]:

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

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

In [148]:

```
a
```

Out[148]:

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

In [149]:

```
b
```

Out[149]:

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

In [150]:

```
np.vstack((a,b))  #vertical stacking
```

Out[150]:

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

In [151]:

```
np.hstack((a,b))  #horizontal stacking
```

Out[151]:

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

Concatenation

In [152]:

```
myArray = np.array([0,1,2,3,4,5,6,7,8,9]).reshape(5,2)

print(myArray)
```

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

In [153]:

```
print(np.concatenate([myArray,myArray], axis = 0))   #vertical
```

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

In [154]:

```
print(np.concatenate([myArray,myArray], axis = 1))  #horizontal
```

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

Slicing

In [156]:

```
a = np.array([[1,2,3,4],
              [5,6,7,8],
              [9,10,11,12]])

b = a[:2, 1:3]

print(b)
```

```
[[2 3]
 [6 7]]
```

In [158]:

```
print(a[0,1])
```

```
2
```

In [161]:

```
b[0,0] = 77

print(a[0,1])
```

```
77
```

In [162]:

```
a
```

Out[162]:

```
array([[ 1, 77,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [163]:

```
line1 = a[1,:]
line2 = a[1:2, :]
line3 = a[[1],:]
```

In [164]:

```
print(line1, line1.shape)
print(line2, line2.shape)
print(line3, line3.shape)
```

```
[5 6 7 8] (4,)
[[5 6 7 8]] (1, 4)
[[5 6 7 8]] (1, 4)
```

In [165]:

```
a
```

Out[165]:

```
array([[ 1, 77,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [166]:

```
col1 = a[:,1]
col2 = a[:, 1:2]

print(col1, col1.shape)
print(col2, col2.shape)
```

```
[77  6 10] (3,)
[[77]
 [ 6]
 [10]] (3, 1)
```

In [167]:

```
col2.ndim
```

Out[167]:

```
2
```

In [168]:

```
col1.ndim
```

Out[168]:

```
1
```

In [169]:

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

print(t[[0,1,2],[0,1,0]])
```

```
[1 4 5]
```

In [170]:

```
print(np.array([t[0,0], t[1,1], t[2,0]]))
```

```
[1 4 5]
```

In [171]:

```
s = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])

print(s)
```

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

In [172]:

```
indis = np.array([0,2,0,1])
```

In [173]:

```
indis
```

Out[173]:

```
array([0, 2, 0, 1])
```

In [174]:

```
print(s[np.arange(4), indis]) #([0,1,2,3],[0,2,0,1])
```

```
[ 1  6  7 11]
```

Aritmetic Operations

In [2]:

```
x = np.array([[1,2],[3,4]], dtype= np.float64)
y = np.array([[5,6],[7,8]], dtype= np.float64)


print(x+y)

print(np.add(x,y))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-732c4c718d67> in <module>
----> 1 x = np.array([[1,2],[3,4]], dtype= np.float64)
      2 y = np.array([[5,6],[7,8]], dtype= np.float64)
      3 
      4 
      5 print(x+y)

NameError: name 'np' is not defined
```

In [3]:

```
print(x-y)

print(np.subtract(x,y))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-fb9c15d00e93> in <module>
----> 1 print(x-y)
      2 
      3 print(np.subtract(x,y))

NameError: name 'x' is not defined
```

In [177]:

```
print(x*y)

print(np.subtract(x,y))
```

```
[[ 5. 12.]
 [21. 32.]]
[[-4. -4.]
 [-4. -4.]]
```

In [178]:

```
print(np.dot(x,y))  #This function returns the dot product of two arrays.
```

```
[[19. 22.]
 [43. 50.]]
```

In [179]:

```
# (2,3) ve (3,2)
# (2,2) ve (2,3)
```

In [180]:

```
print(x/y)

print(np.divide(x,y))
```

```
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
```

In [181]:

```
s = np.array([[4,9],[16,81]], dtype = np.float64)

print(np.sqrt(s))
```

```
[[2. 3.]
 [4. 9.]]
```

In [182]:

```
s = np.array([[4,9],[16,81]], dtype = np.float64)

print(np.square(s))
```

```
[[  16.   81.]
 [ 256. 6561.]]
```

In [ ]:

```
#Calculate the exponential of all elements in the input array

s = np.array([[4,9],[16,81]], dtype = np.float64)
print(np.exp(s))
```

In [186]:

```
v = np.array([10,100,1000,10000,100000,1000000])

print(np.log(v))
```

```
[ 2.30258509  4.60517019  6.90775528  9.21034037 11.51292546 13.81551056]
```

In [188]:

```
t = np.array([np.pi/6, np.pi/2, np.pi/3])

np.sin(t)
```

Out[188]:

```
array([0.5       , 1.        , 0.8660254])
```

Dot Product

Dot Product

$$\begin{bmatrix} a & b \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax+by \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax+by \\ cx+dy \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} aw+by & ax+bz \\ cw+dy & cx+dz \end{bmatrix}$$

credit: https://algebra1course.files.wordpress.com/

In [189]:

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

a = np.array([9,10])
b = np.array([11,12])


print(a.dot(b))

print(np.dot(a,b))
```

```
219
219
```

In [190]:

```
print(x.dot(a))

print(np.dot(a,x))
```

```
[29 67]
[39 58]
```

In [191]:

```
print(y.dot(b))

print(np.dot(y,b))
```

```
[127 173]
[127 173]
```

In [192]:

```
x = np.array([[1,2],[3,4]])

print(np.sum(x))

print(np.sum(x, axis = 0))

print(np.sum(x, axis = 1))
```

```
10
[4 6]
[3 7]
```

Transpose

In [193]:

```
x = np.array([[1,2],
              [3,4]])


print(x.T)
```

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

In [194]:

```
v = np.array([[1,2,3]])


print(v.T)
```

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

In [195]:

```
t = np.array([[1,2,3]])


print(t)

print(t.shape)

print(t.T)

v = t.T

print(v.shape)
```

```
[[1 2 3]]
(1, 3)
[[1]
 [2]
 [3]]
(3, 1)
```

In [198]:

```
#Data Type Conversion

x = np.array([1,2,2.5])
```

```
print(x)

x = x.astype(int)

print(x)
```

```
[1.  2.  2.5]
[1 2 2]
```

In [199]:

```
#Dimension Expansion

y = np.array([1,2])

print(y.shape)

y = np.expand_dims(y, axis = 0)

print(y.shape)

y = np.expand_dims(y, axis = 0)

print(y.shape)

y = np.expand_dims(y, axis = 0)

print(y.shape)
print(type(y))
print(y.ndim)
print(y.reshape(2,1,1,1))
```

```
(2,)
(1, 2)
(1, 1, 2)
(1, 1, 1, 2)
<class 'numpy.ndarray'>
4
[[[[1]]]


 [[[2]]]]
```

In [200]:

```
x = np.array([1,2])

print(x.shape)

x = np.expand_dims(x, axis = 1)

print(x.shape)

x = np.expand_dims(x, axis = 1)

print(x.shape)

x = np.expand_dims(x, axis = 1)

print(x.shape)
```

```
(2,)
(2, 1)
(2, 1, 1)
(2, 1, 1, 1)
```

In [ ]: