

# PHÂN TÍCH DỮ LIỆU BẰNG PYTHON

## LECTURE 2: NumPy



## Nội dung

- ❖ Giới thiệu tổng quan (Python, Matlab,...)
- ❖ Mảng một chiều
- ❖ Mảng nhiều chiều
- ❖ Một số tính toán quan trọng ở Numpy

## Giới thiệu



### □ NumPy (Numerical Python)

- Là một thư viện bao gồm các **đối tượng mảng đa chiều** (multidimensional array) và **một tập hợp các phương thức** để xử lý mảng đa chiều đó.
- Sử dụng Numpy, các toán tử toán học và logic có thể được thực hiện.
- Thay thế cho List
- Cài đặt: pip install numpy

## Giới thiệu

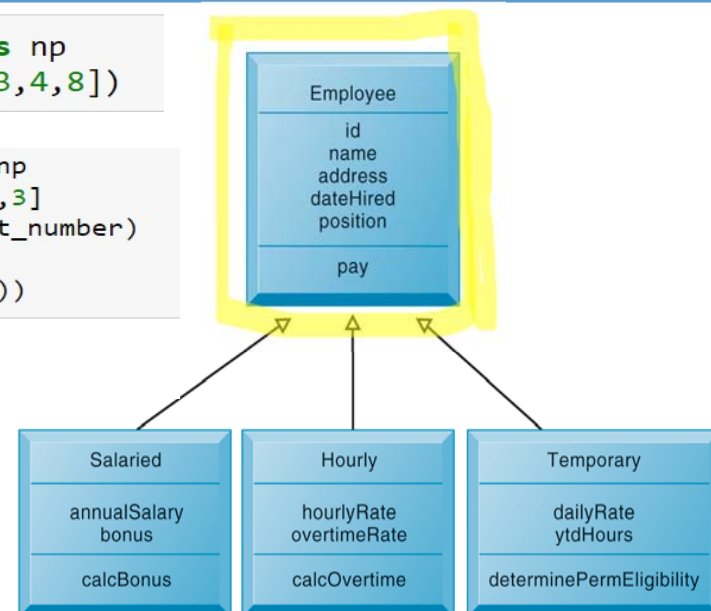


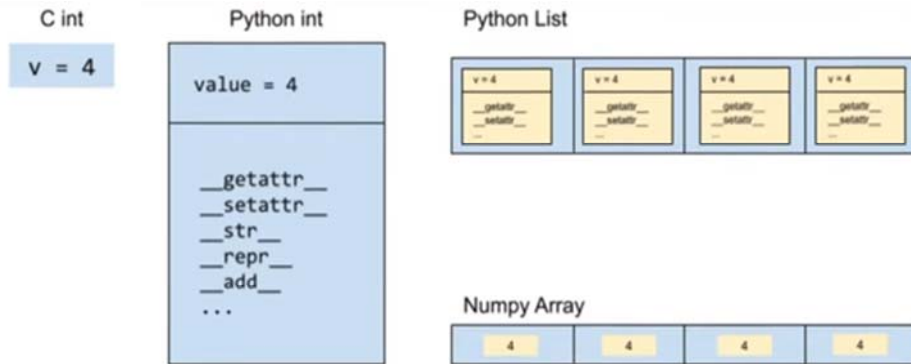
```
import numpy as np
arr=np.array([3,4,8])
```

```
import numpy as np
list_number=[1,2,3]
arr=np.array(list_number)
print(sum(arr))
print(np.sum(arr))
```

6

6





## Ưu điểm

- Chứa bộ các giá trị
- Có thể chứa các loại dữ liệu khác nhau
- Có thể thay đổi, thêm, xóa
- Đầy đủ chức năng tính toán
- Rất nhiều thư viện được xây dựng sẵn trong Numpy
- Cần cho Data Science/Data analytics
  - Thực hiện công việc tính toán trên các bộ dữ liệu (data set)
  - Tốc độ cao

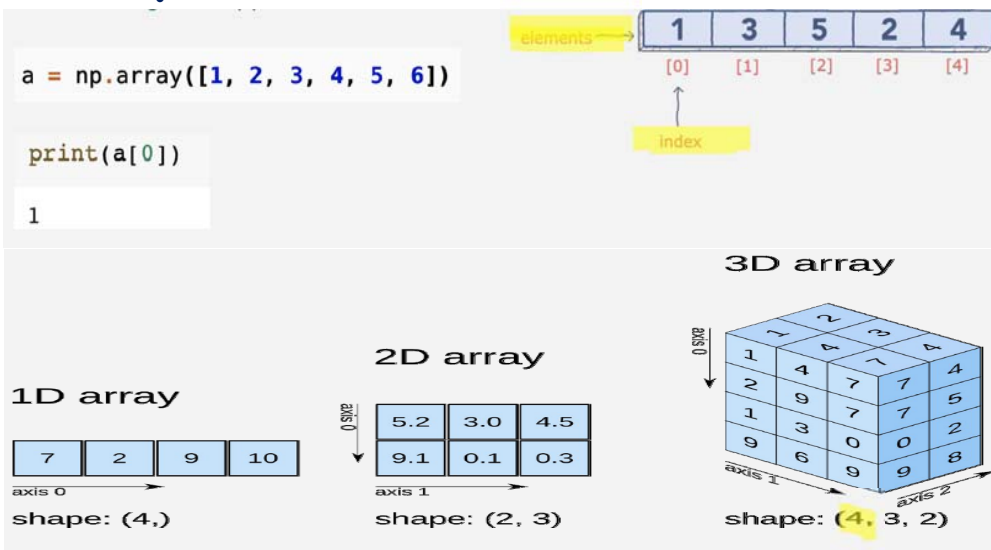


## NumPy - Nddarray Object

- Đối tượng quan trọng nhất trong NumPy là kiểu mảng **N-dimensional** được gọi là **ndarray**. Nó mô tả một bộ các **item cùng kiểu**. Item trong mảng có thể truy xuất bằng cách sử dụng **zero-based index**.
- Mỗi item trong ndarray có cùng kích thước block trong bộ nhớ và là một đối tượng data-type gọi là **dtype**.



## Ví dụ



## Mảng một chiều



### ❑ Tạo mảng một chiều

- Từ object có định dạng array: sử dụng `array(object, dtype=None)`

```
import numpy as np
# Tạo mảng một chiều
array_1=np.array([1,2,3])
print(array_1.shape) # (3,)
print(array_1.ndim)  # 1
print(type(array_1)) # <class 'numpy.ndarray'>
print(array_1.dtype) # int32
print(array_1.size)  # 3
```

9

## Mảng một chiều



### ❑ Tạo mảng một chiều

- Từ object có định dạng array: sử dụng `array(object, dtype=None)`

```
In [12]: import numpy as np
# Tạo mảng một chiều
array_2=np.array([1,2,3],dtype=float)
array_2.dtype
```

```
Out[12]: dtype('float64')
```

10

## Mảng một chiều



- Từ list, tuple: sử dụng `asarray(input, dtype=None)`

```
import numpy as np
x = [1,2,3]
a = np.asarray(x)
print(a)
```

```
[1 2 3]
```

```
# Tạo mảng từ Tuple
x = (1, 2, 3)
a = np.asarray(x)
print(a)
```

```
[1 2 3]
```

```
import numpy as np
x=[1,2,3]
a=np.array(x)
print(a)
```

```
[1 2 3]
```

```
import numpy as np
x=(1,2,3)
a=np.array(x)
print(a)
```

```
[1 2 3]
```

11

## Mảng một chiều



- Có giá trị mặc định là 0,1: dùng `zeros(shape, dtype=float)`, `ones(shape, dtype=None)`

```
# Tạo mảng các phần tử có GT ban đầu Là 0
arr_0=np.zeros(5,dtype=int)
print(arr_0)
```

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

```
# Tạo mảng các phần tử có GT ban đầu Là 1
arr_1=np.ones(5)
print(arr_1)
```

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

12

## Mảng một chiều



- Từ range: sử dụng numpy.**arange** (start, stop, step, dtype)  

```
mang_2 = np.arange(10,20,2)
print(mang_2)
```

```
[10 12 14 16 18]
```
- Từ range có khoảng cách đều nhau: sử dụng numpy.**linspace** (start, stop, num, endpoint, dtype)

```
mang_3 = np.linspace(10,20, 5)
print(mang_3)
mang_4 = np.linspace(10,20, 5, endpoint = False)
print(mang_4)
```

```
[10.  12.5 15.  17.5 20. ]
[10. 12. 14. 16. 18.]
```

13

## Mảng một chiều



- numpy.random.rand(d0, d1,...,dn)  

```
np.random.rand(3,2)
```

```
array([[0.48662287, 0.37849263],
       [0.64750646, 0.55548139],
       [0.84158438, 0.01563752]])
```
- numpy.random.randint(low, high, size=None, dtype=int)  

```
np.random.randint(low=1,high=5,size=(2,3))
```

```
array([[4, 2, 4],
       [4, 2, 4]])
```
- numpy.random.normal(mean, var, size)

```
b=np.random.normal(6,0.2,1000)
```

14

## Mảng một chiều



### ❑ Tạo mảng

- np.array
- np.asarray
- np.ones, np.zeros
- np.arange
- np.linspace
- np.random.rand
- np.random.randint
- np.random.normal

**Questions: list và array của numpy?**

15

## Mảng một chiều



### ❑ Thao tác trên mảng

- Số phần tử của mảng

```
print(array_1)
print(array_1.shape[0])
```

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

- Cập nhật phần tử trong mảng

```
array_1[1] = 250
```

```
print(array_1)
```

```
[ 1 250  3]
```

16



## Mảng một chiều



### ❑ Thao tác trên mảng

- Truy xuất phần tử trên mảng theo chỉ số index

Index:	0	1	2	3	4
Value:	88	19	46	74	94

Index:	0	1	2	3	-1
Value:	88	19	46	74	94

17

## Mảng một chiều



### ❑ Ví dụ:

```
# Truy xuất phần tử trong mảng
mang_4 = np.array([1, 3, 2, 4, 5, 7, 9])
print(mang_4)
print(mang_4[3])
print(mang_4[3:])
print(mang_4[3:5])
print(mang_4[mang_4 < 5]) # có kèm điều kiện
```

[1 3 2 4 5 7 9]  
4  
[4 5 7 9]  
[4 5]  
[1 3 2 4]

```
arr=np.array([2,3,5,6,7,10])
print(arr[1::2])
```

[ 3 6 10]

18

## Mảng một chiều



### ❑ Thao tác trên mảng

- Truy xuất phần tử trên mảng: theo slice (start, stop, step)

```
# Truy xuất các phần tử trong mảng dùng slice
mang_5 = np.arange(10)
print(mang_5)
s = slice(3,8,2)
print(mang_5[s])
print(mang_5[s][1])
```

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

19

## Mảng một chiều



- ❑ Thao tác trên mảng: Thêm phần tử vào cuối mảng: sử dụng `append(arr, values)`

```
# Thêm các phần tử vào cuối mảng
mang_5 = np.arange(10)
mang_5 = np.append(mang_5, [25, 30])
print(mang_5)
```

[ 0 1 2 3 4 5 6 7 8 9 25 30]

- Thêm phần tử mới: sử dụng `insert(arr, index, values)`

```
# Thêm phần tử vào index = 3, giá trị 100
print(mang_5)
mang_5 = np.insert(mang_5, 3, 100)
print(mang_5)
```

[ 0 1 2 3 4 5 6 7 8 9 25 30]  
[ 0 1 2 100 3 4 5 6 7 8 9 25 30]

20

## Mảng một chiều



### ❑ Thao tác trên mảng

- Xóa phần tử tại index trong mảng: sử dụng `delete(arr, index)`.

```
# Xóa phần tử tại vị trí index = 11
print(mang_5)
mang_5 = np.delete(mang_5, 11)
print(mang_5)
```

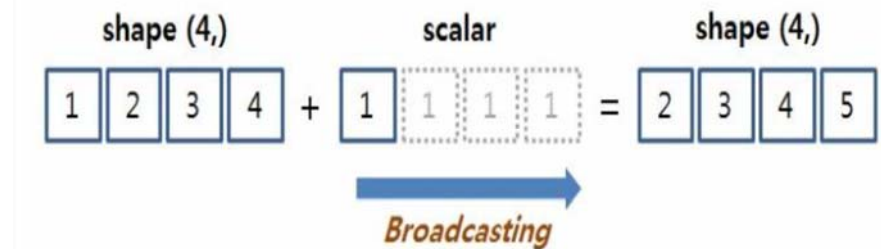
```
[ 0  1  2 100  3  4  5  6  7  8  9 25 30]
[ 0  1  2 100  3  4  5  6  7  8  9 30]
```

21

## Mảng một chiều



### ❑ Broadcasting: thực hiện các phép toán số học trên mảng



22

## Mảng một chiều



### ❑ Statistical function

- Tìm max, min: sử dụng `max()` hoặc `amax()`, min hoặc `amin()`, `mean()`, `std()`, `var()`, `argmax()`...

```
# max, min
d = np.array([3, 10, 9, 12, 8, 5])
print(np.amax(d))
print(np.amin(d))
```

```
12
3
```

- Tính tổng giá trị: sử dụng `sum()`

```
arr = np.arange(10) # array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
arr.sum()
```

45

23

## Mảng một chiều



### ❑ Sort, where, extract

- Sắp xếp tăng dần trong mảng: sử dụng `sort()`

```
mang_5=np.array([3,1,2,4,7,3,8,9,5,6])
print(mang_5)
print(np.sort(mang_5))
print(-np.sort(-mang_5))
```

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

`np.sort` -> quicksort algorithm

24

## Mảng một chiều



### ❑ Sort, where, extract

- Tìm index của các phần tử trong mảng thỏa điều kiện: sử dụng hàm `where()`

```
indexes=np.where(mang_5>=8)
print(indexes)
print(mang_5[indexes])

(array([6, 7], dtype=int64),
 [8 9])
```

25

## Mảng một chiều



### ❑ Sort, where, extract

- Sử dụng hàm `where()` như hàm `if`

```
arr=np.arange(10)
print(arr)
np.where(arr<5, arr, 10*arr)
```

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

array([ 0, 1, 2, 3, 4, 50, 60, 70, 80, 90])

26

## Mảng một chiều



- Tìm các phần tử trong mảng thỏa điều kiện: sử dụng `extract` (điều kiện, mảng)

```
print(mang_5)
print(np.extract(np.mod(mang_5,2)==0, mang_5))
print(np.extract(mang_5 >= 5, mang_5))
```

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

27

## Mảng một chiều



### ❑ Thảo luận

- Broadcasting
- `np.where`
- `np.extract`

- `arr=np.array([3,4,5,6])`
- `arr>5`
- `arr[arr>5]`

- `index=arr>5`
- `arr[index]`

28

## Mảng một chiều – Bài tập thực hành



- Bài tập tại lớp – file yêu cầu dạng Jupyter: numpy-ex1-1D.ipynb, thời gian 25 phút
- Nộp lên hệ thống elearning, đặt tên file để nộp: **STT**-Họ và tên-Lớp.ipynb. Ví dụ: 20-Le Van A-46K29-1.ipynb

29

## Mảng nhiều chiều



- ❑ **Tạo mảng nhiều chiều:** từ object có định dạng array: sử dụng hàm array()

```
import numpy as np
# Tạo mảng
array_2 = np.array([[1, 2, 3], [4, 5, 6]])
print(array_2)
print(type(array_2))
```

```
[[1 2 3]
 [4 5 6]]
<class 'numpy.ndarray'>
```

30

## Mảng nhiều chiều



- ❑ **Tạo mảng nhiều chiều:** có giá trị mặc định là 0,1: dùng zeros(), ones()

```
# Tạo mảng hai chiều với giá trị 0 hoặc 1
arr_zero=np.zeros([2,2],dtype=float)
print(arr_zero)
print("")
arr_one=np.ones([2,3])
print(arr_one)
```

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

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

31

## Mảng nhiều chiều



- ❑ **Thao tác trên mảng**

- Cập nhật phần tử trong mảng

```
#Cập nhật phần tử trong mảng
array_2[0,1]=200
print(array_2)
```

```
[[ 1 200  3]
 [ 4  5  6]]
```

32



## Mảng nhiều chiều



- Truy xuất phần tử trên mảng: theo ma trận, dòng, cột (index=0 -> chiều dài -1)

```
array_2=np.array([[1,2,3],[4,5,6]])  
print(array_2)  
print(array_2[1,2])  
print(array_2[1,])
```

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

33

## Mảng nhiều chiều



- Truy xuất phần tử trên mảng: theo ma trận, dòng, cột (index=0 -> chiều dài -1)

```
arr_2=np.zeros([3,2,2])  
print(arr_2)  
arr_2[0,1,1]=100  
print("")  
print(arr_2[0])
```

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

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

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

```
[[ 0.  0.]  
 [ 0. 100.]]
```

34

## Mảng nhiều chiều



### Thao tác trên mảng

- Truy xuất phần tử trên mảng: <tên mảng>[start:stop:step]

```
print(arr[0,3:5])  
print("")  
print(arr[4:,4:])  
print("")  
print(arr[:,2])  
print("")  
print(arr[2::2,:2])
```

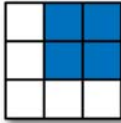
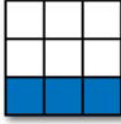
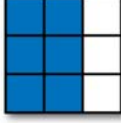

```
[3 4]  
  
[[44 45]  
 [54 55]]  
  
[ 2 12 22 32 42 52]  
  
[[20 22 24]  
 [40 42 44]]
```

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

35

## Mảng nhiều chiều



Expression	Shape
 arr[:2, 1:]	(2, 2)
 arr[2] arr[2, :] arr[2:, :]	(3,) (3,) (1, 3)
 arr[:, :2]	(3, 2)
 arr[1, :2] arr[1:2, :2]	(2,) (1, 2)

36

## Mảng nhiều chiều

- Thay đổi kích cỡ mảng: dùng shape, reshape (số\_lượng\_ma\_trận, số\_dòng, số\_cột)

```
# thay đổi kích cỡ mảng
print(array_2)
print(array_2.shape)
array_2.shape = (3,2)
print(array_2)

# thay đổi kích cỡ mảng dùng reshape
a = np.arange(24)
print(a)
b = a.reshape(2, 3, 4) # ma trận, dòng, cột
print(b)
print(b[0][1][2])

[[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
[[[ 0  1  2  3]
   [ 4  5  6  7]
   [ 8  9 10 11]]

[[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]]
6
```

37

## Mảng nhiều chiều

### Tạo mảng con

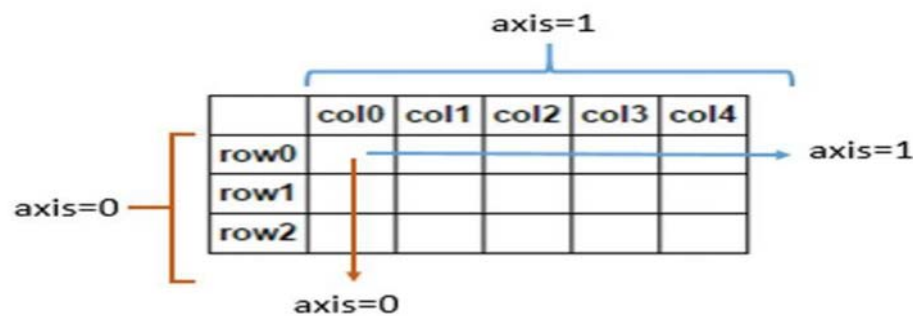
```
print(ar)
M1_all_row_Col2=ar[1,...,2]
print('M1_all_row_Col2:',M1_all_row_Col2)
M1_row1_all=ar[1,1,...]
print('M1_row1_all:',M1_row1_all)

[[[ 0  1  2  3]
   [ 4  5  6  7]
   [ 8  9 10 11]]

[[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]]
M1_all_row_Col2: [14 18 22]
M1_row1_all: [16 17 18 19]
```

38

## Mảng nhiều chiều



```
print(arr)
arr1=np.sum(arr,axis=0)
print('Ket qua:',arr1)
```

```
[[3 4 5]
 [4 8 9]
 [4 9 7]]
Ket qua: [11 21 21]
```

39

## Mảng nhiều chiều

### Broadcasting: thực hiện các phép toán số học trên mảng

$\text{np.arange}(3)+5$

$\text{np.ones}((3,3))+\text{np.arange}(3)$

$\text{np.arange}(3).\text{reshape}((3,1))+\text{np.arange}(3)$

40

## Mảng nhiều chiều - Broadcasting



### □ Steps when implementing broadcasting in numpy.

- Step 1: Make the dimensions of the 2 arrays equal. Array with smaller dimension will add dimension "1" to the left of that array until it equal dimension of the larger array.
- Step 2: Consider each pair of dimension from right to left of 2 arrays. If all dimensional pairs satisfy one of the following two conditions, broadcasting can be used. Two condition are as flows: (i) they are equal or (ii) 1 in 2 equals 1

41

## Mảng nhiều chiều - Broadcasting



### □ Questions: Check whether these matrices can be broadcasted, using the broadcasting rule.

- Q1: A matrix of shape (2,3) + a vector of shape (2,)
- Q2: A matrix of shape (3,2) + a vector of shape (3,)
- Q3: A matrix of shape (2,3) + a 3D tensor of shape (4,2,4)

42

## Mảng nhiều chiều



### □ Broadcasting: thực hiện các phép toán số học trên mảng

```
a=np.array([[1,2,3,4],[1,1,1,1],[2,2,2,2]])
b=np.array([2,4,6,8])
print(a)
print("b")
print(b)
print("a+b")
print(a+b)
```

```
[[1 2 3 4]
 [1 1 1 1]
 [2 2 2 2]]
b
[2 4 6 8]
a+b
[[ 3  6  9 12]
 [ 3  5  7  9]
 [ 4  6  8 10]]
```

Tương tự, a-b, a\*b, a/b

43

## Mảng nhiều chiều



### □ Broadcasting: thực hiện các phép toán số học trên mảng

#### Operator Equivalent ufunc Description

+	np.add	Addition (e.g., 1 + 1 = 2)
-	np.subtract	Subtraction (e.g., 3 - 2 = 1)
-	np.negative	Unary negation (e.g., -2)
*	np.multiply	Multiplication (e.g., 2 * 3 = 6)
/	np.divide	Division (e.g., 3 / 2 = 1.5)
//	np.floor_divide	Floor division (e.g., 3 // 2 = 1)
**	np.power	Exponentiation (e.g., 2 ** 3 = 8)
%	np.mod	Modulus/remainder (e.g., 9 % 4 = 1)

```
x = np.arange(4)
print("x =", x)
print("x + 5 =", x + 5)
print("x - 5 =", x - 5)
print("x * 2 =", x * 2)
print("x / 2 =", x / 2)
print("x // 2 =", x // 2) # floor division
```

```
x = [0 1 2 3]
x + 5 = [5 6 7 8]
x - 5 = [-5 -4 -3 -2]
x * 2 = [0 2 4 6]
x / 2 = [0.  0.5  1.  1.5]
x // 2 = [0 0 1 1]
```

```
print("-x =", -x)
print("x ** 2 =", x ** 2)
print("x % 2 =", x % 2)
```

```
-x = [ 0 -1 -2 -3]
x ** 2 = [0 1 4 9]
x % 2 = [0 1 0 1]
```



### Statistical function

- Giá trị trung vị: sử dụng median() dựa trên mảng đã sắp xếp, lấy trung bình phần tử ở giữa mảng.

```
print(b)
print('Median b:')
print(np.median(b))
print('Median b[1]:')
print(np.median(b[1]))
```

```
[[[3 7]
  [8 4]]
```

```
  [[2 4]
   [1 5]]]
Median b:
4.0
Median b[1]:
3.0
```

45



- Tính toán độ lệch chuẩn theo trục được chỉ định: sử dụng hàm std()

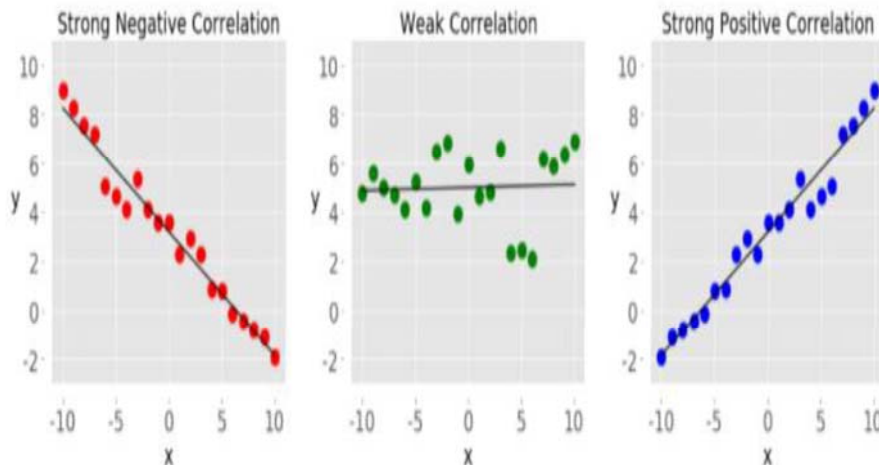
```
# Tính toán độ lệch chuẩn standard deviation
a = np.array([[1, 2], [3, 4]])
print(a)
print(np.std(a))
print(np.std(a, axis = 0))
print(np.std(a, axis = 1))
```

```
[[1 2]
 [3 4]]
1.118033988749895
[1. 1.]
[0.5 0.5]
```

46



- Hệ số tương quan giữa 2 biến x, y



47



- Hệ số tương quan giữa 2 biến x, y

```
x = np.array([-2, -1, 0, 1, 2])
y = np.array([4, 1, 3, 2, 0])
my_rho = np.corrcoef(x, y)

print(my_rho)
```

```
[[ 1. -0.7]
 [-0.7 1. ]]
```

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

- +1 - Complete positive correlation
- +0.8 - Strong positive correlation
- +0.6 - Moderate positive correlation
- 0 - no correlation whatsoever
- 0.6 - Moderate negative correlation
- 0.8 - Strong negative correlation
- 1 - Complete negative correlation

48





## Aggregation

Function Name	NaN-safe Version	Description
np.sum	np.nansum	Compute sum of elements
np.prod	np.nanprod	Compute product of elements
np.mean	np.nanmean	Compute mean of elements
np.std	np.nanstd	Compute standard deviation
np.var	np.nanvar	Compute variance
np.min	np.nanmin	Find minimum value
np.max	np.nanmax	Find maximum value
np.argmin	np.nanargmin	Find index of minimum value
np.argmax	np.nanargmax	Find index of maximum value
np.median	np.nanmedian	Compute median of elements
np.percentile	np.nanpercentile	Compute rank-based statistics of elements
np.any	N/A	Evaluate whether any elements are true
np.all	N/A	Evaluate whether all elements are true

49



## Sort, where, extract

- Sắp xếp tăng dần trong mảng: sử dụng sort()

```
c = np.array([[[9,7],[8,4]],[[2,4],[6,5]]])
print(c)
print("Sort c")
print(np.sort(c))
```

```
[[[9 7]
  [8 4]]
```

```
[[[2 4]
  [6 5]]]
```

```
Sort c
[[[7 9]
  [4 8]]]
```

```
[[[2 4]
  [5 6]]]
```

```
print('Sort c[0] axis 0 (theo cột):')
print(np.sort(c[0], axis = 0))
```

```
Sort c[0] axis 0:
[[8 4]
 [9 7]]
```

```
print('Sort c[0] axis 1 (theo dòng):')
print(np.sort(c[0], axis = 1))
```

```
Sort c[0] axis 1 (theo dòng):
[[7 9]
 [4 8]]
```

50



- Tìm index của các phần tử trong mảng thỏa điều kiện: sử dụng where() `np.random.seed(1)`

```
# Tạo một mảng hai chiều ngẫu nhiên
arr = np.random.randint(0, 10, size=(3, 3))
print(arr)
# Tìm các chỉ mục của các phần tử trong mảng mà có giá trị lớn hơn 5
indices_of_elements_gt_5 = np.where(arr > 5)
print(indices_of_elements_gt_5)
elements_gt_5 = arr[indices_of_elements_gt_5]
print(elements_gt_5)

[[3 5 7]
 [8 6 6]
 [8 4 9]]
(array([0, 1, 1, 1, 2, 2], dtype=int64), array([2, 0, 1, 2, 0, 2], dtype=int64))
[7 8 6 6 8 9]
```

51



- Tìm index của các phần tử trong mảng thỏa điều kiện: sử dụng extract(điều kiện, mảng)

```
print(c)
print("Trên c:", np.extract(c >= 5, c))
print("Trên c[0]:", np.extract(c[0] >= 5, c[0]))
```

```
[[[9 7]
  [8 4]]]
```

```
[[[2 4]
  [6 5]]]
```

```
Trên c: [9 7 8 6 5]
```

```
Trên c[0]: [9 7 8]
```

52



## □ Numpy – Data Type Objects (dtype)

- Dtype mô tả:
  - Kiểu dữ liệu (int, float,...)
  - Kích thước dữ liệu
  - Thứ tự byte (byte order)

53



## ▪ Numpy cho phép tạo mảng cấu trúc

- Cú pháp: `numpy.dtype(object, [align, copy])`
- Trong đó:
  - Object: để được chuyển đổi thành data type object
  - Align: nếu = True, các trường được căn chỉnh để tối ưu bộ nhớ.
  - Copy: nếu = True, tạo một bản sao mới của dtype object. Nếu = False, không tạo bản sao của dữ liệu khi tạo một mảng mới với kiểu dữ liệu này.

54



## ▪ Ví dụ:

```
dt = np.dtype(np.int32)
print(dt)
```

`int32`

```
dt = np.dtype([('age', 'i8')])
a = np.array([(10,), (20,), (30,)], dtype = dt)
print(a['age'])
```

`[10 20 30]`

55



## ▪ Ví dụ:

```
name char[10]
age int
weight double
```

Brad	Jane	John	Fred
33	25	47	54
135.0	105.0	225.0	140.0

```
emp=np.dtype([('name', 'S10'),('age', 'int'),('weight', 'float')])
```

```
arr_emp = np.empty([0],dtype=emp)
```

```
arr_emp=np.append(arr_emp,np.array([('Brad', 33, 135.0)],dtype=emp))
```

```
arr_emp=np.append(arr_emp,np.array([('Jane', 25, 105.0)],dtype=emp))
```

```
arr_emp
```

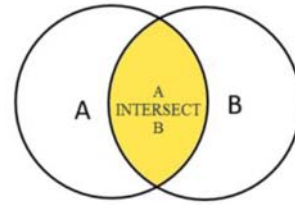
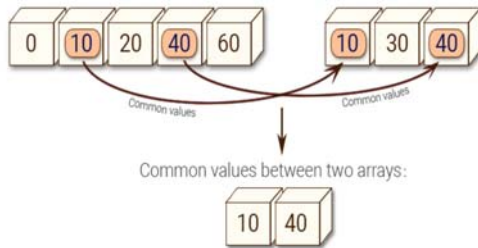
```
array([(b'Brad', 33, 135.), (b'Jane', 25, 105.)],
      dtype=[('name', 'S10'), ('age', '<i4'), ('weight', '<f8')])
```

56

## Mảng nhiều chiều



### ❑ Tìm phần tử chung: `numpy.intersect1d`



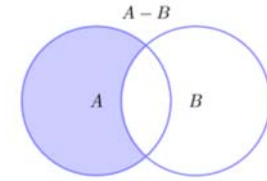
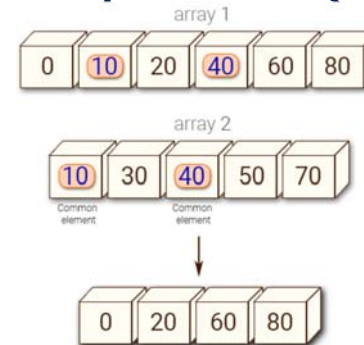
```
np.intersect1d([0, 10, 20, 40, 60], [10, 30, 40])  
array([10, 40])
```

57

## Mảng nhiều chiều



### ❑ Tìm phần tử “hiệu”: `numpy.setdiff1d`



```
np.setdiff1d([0, 10, 20, 40, 60, 80], [10, 30, 40, 50, 70])  
array([ 0, 20, 60, 80])
```

58

## Các phép toán cơ bản của ma trận



### ❖ Cộng hai ma trận:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} a+w & b+x \\ c+y & d+z \end{bmatrix}$$

### ❖ Trừ hai ma trận:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} a-w & b-x \\ c-y & d-z \end{bmatrix}$$

```
import numpy as np  
a=np.array([[1,2,3],[4,5,6]])  
b=np.array([[2,3,5],[7,9,1]])  
print('a+b:',a+b)  
print('-----')  
print('a-b:',a-b)
```

```
a+b: [[ 3  5  8]  
      [11 14  7]]  
-----  
a-b: [[-1 -1 -2]  
      [-3 -4  5]]
```

59

## Các phép toán cơ bản của ma trận



### ❖ Nhân hai ma trận:

**Matrix1.dot(matrix2)**  
**Matrix1@matrix2**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

$[2 \times 3] \quad [3 \times 2] \quad [2 \times 2]$

```
# Nhân ma trận  
import numpy as np  
a=np.array([[1,2,3],[4,5,6]])  
b=np.array([[7,8],[9,10],[11,12]])
```

```
np.dot(a,b) # a@b  
array([[ 58,  64],  
       [139, 154]])
```

60



## Các phép toán cơ bản của ma trận



- ❖ Ma trận đơn vị (identity matrix): Cấu trúc của ma trận đơn vị là 1 ma trận có số 1 trên đường chéo chính.

**np.eye(size)**

```
import numpy as np
b=np.eye(3,3)
print(b)
```

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

61

## Các phép toán cơ bản của ma trận



- ❖ Ma trận nghịch đảo (Inverse matrix):
  - Ma trận nghịch đảo ký hiệu:  $A^{-1}$
  - Tích của ma trận  $A * A^{-1} = 1$  (ma trận đơn vị)
  - Tính ma trận nghịch đảo ở Numpy:

**np.linalg.pinv(matrix)**

```
import numpy as np
A=np.array([[2,3],[4,5]])
A_inverse=np.linalg.pinv(A)
print(A_inverse)
```

```
[[ -2.5  1.5]
 [  2.  -1. ]]
```

62

## Các phép toán cơ bản của ma trận



- ❖ Ma trận chuyển vị (Transpose matrix):
  - Ma trận chuyển vị ký hiệu:  $A^T$
  - Là ma trận đảo hàng và cột so với ma trận gốc
  - Tính ma trận chuyển vị ở Numpy:

**np.transpose(matrix)**

```
import numpy as np
A=np.array([[2,3,4],[4,5,6]])
A_tran=np.transpose(A)
print(A_tran)
```

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

63

## Giải hệ phương trình



**Phương pháp giải hệ Cramer:**

EX: Giải hệ 
$$\begin{cases} 2x - y - z = 4 \\ 3x + 4y - 2z = 11 \\ 3x - 2y + 4z = 11 \end{cases}$$

Ta có  $A = \begin{pmatrix} 2 & -1 & -1 \\ 3 & 4 & -2 \\ 3 & -2 & 4 \end{pmatrix}; X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}; B = \begin{pmatrix} 4 \\ 11 \\ 11 \end{pmatrix}$

Dạng ma trận của hệ:  $AX = B$

Khi đó, nghiệm:  $X = A^{-1}B = \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}$

64





```
A=np.array([[2,-1,-1],[3,4,-2],[3,-2,4]])
B=np.array([4,11,11])
print(A)
print('----')
print(B)
# np.linalg.pinv: Ma trận nghịch đảo
# np.transpose: Ma trận chuyển vị
X=np.dot(np.linalg.pinv(A),np.transpose(B))
print('Ket qua')
print (X)
```

```
[[ 2 -1 -1]
 [ 3  4 -2]
 [ 3 -2  4]]
----
[ 4 11 11]
Ket qua
[3.  1.  1.]
```

Linear algebra/pseudo inverse

65



- Bài tập tại lớp – file yêu cầu dạng Jupyter: numpy-ex2-2D.ipynb, thời gian 30 phút
- Nộp lên hệ thống elearning, đặt tên file để nộp: STT-Họ và tên-Lớp.ipynb. Ví dụ: 20-Le Van A-46K29-1-numpy2.ipynb

66

## Hỏi & Đáp



Cám ơn các bạn đã lắng nghe.