



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

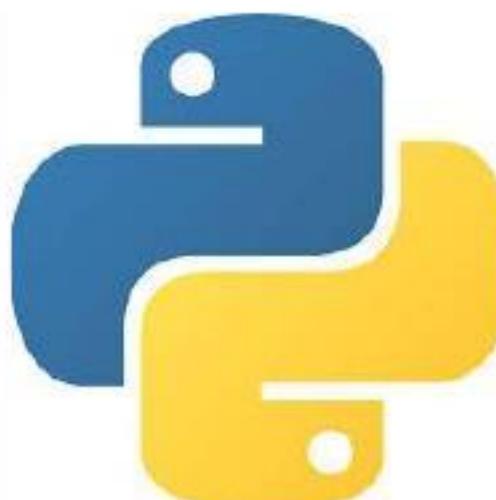
# PYTHON FOR MACHINE LEARNING, DATA SCIENCE & DATA VISUALIZATION

Bài 3: Numpy



Phòng LT & Mạng

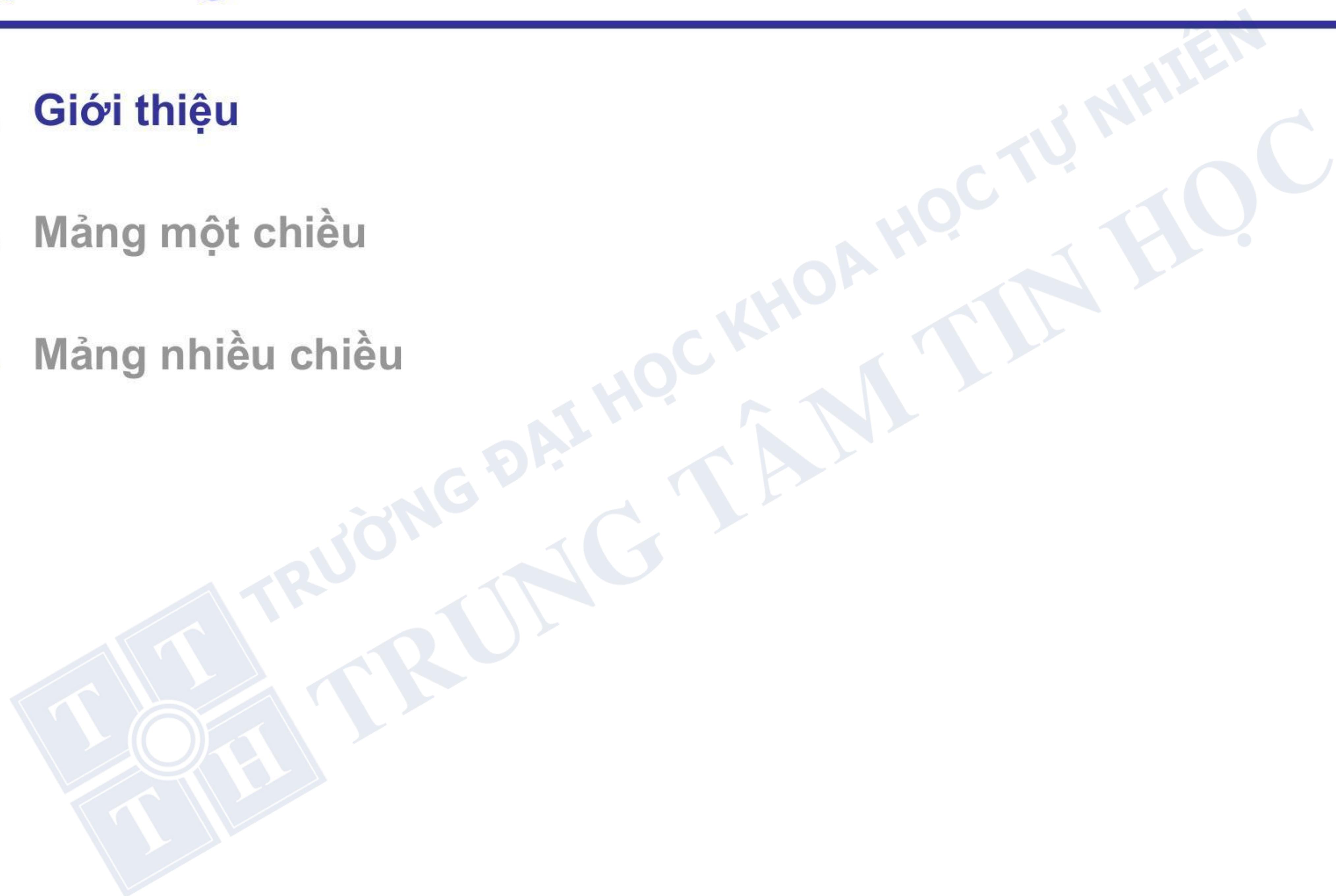
2020



# Nội dung

---

1. Giới thiệu
2. Mảng một chiều
3. Mảng nhiều chiều



# 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

## □ 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
  - Thực hiện công việc tính toán trên các bộ dữ liệu
  - Tốc độ cao

# Giới thiệu

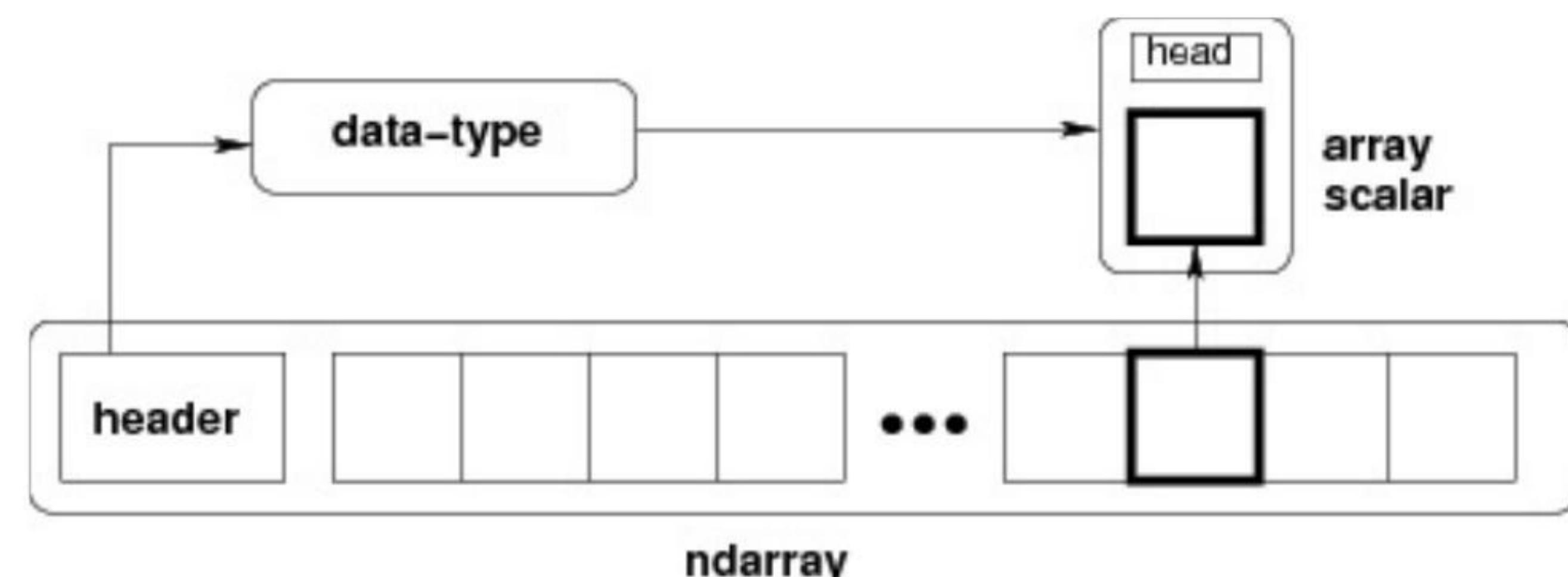
## ❑ NumPy - Ndarray Object

- 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**.

# Giới thiệu

## ❑ NumPy - Ndarray Object

- Bất kỳ item nào được trích xuất từ ndarray object bằng cách cắt ra được đại diện bởi một Python object của một kiểu mảng vô hướng (array scalar type)
- Mỗi quan hệ giữa ndarray, dtype và array scalar type:



# Giới thiệu

## ❑ NumPy - Ndarray Object

- Một thực thể của ndarray class có thể được tạo ra từ nhiều phương thức tạo mảng khác nhau.
- Ndarray cơ bản được tạo bằng cách sử dụng phương thức: numpy.array

# Giới thiệu

## □ NumPy - Ndarray Object

- Cú pháp:

```
numpy.array(object, dtype=None, copy=True, ndmin=0)
```

- Trong đó:

- Object: đối tượng có phương thức giao diện mảng
- Dtype: loại dữ liệu mong muốn của mảng (tùy chọn)
- Copy: mặc định = True đối tượng được copy (tùy chọn)
- ndmin: chỉ định kích thước tối thiểu của mảng kết quả

# Giới thiệu

- Ví dụ:

```
import numpy as np  
# Mảng một chiều  
np.array([1,2,3])  
print(a)
```

```
[1 2 3]
```

```
# Mảng hai chiều  
a = np.array([[1, 2, 3], [4, 5, 6]])  
print(a)  
  
[[1 2 3]  
 [4 5 6]]
```

```
# ndmin & dtype  
a = np.array([1, 2, 3, 4, 5], ndmin = 2, dtype = complex)  
print(a)
```

```
[[1.+0.j 2.+0.j 3.+0.j 4.+0.j 5.+0.j]]
```

# Giới thiệu

## ❑ Numpy – Data Type

- NumPy hỗ trợ rất nhiều kiểu dữ liệu khác nhau, như: bool\_, int\_, int8, int16, int32, int64, unit8 (số nguyên không dấu),, unit16, unit32, unit64, float\_, float16, float32, float64, complex\_, complex64, complex128  
=> dtype: numpy.bool\_, numpy.int8, ...

**Note: int8, int16, int32, int64 có thể thay thế bằng chuỗi 'i1', 'i2', 'i4', 'i8'**

# Nội dung

1. Giới thiệu

2. Mảng một chiều

7	2	9	10
---	---	---	----

3. Mảng nhiều chiều

# Mảng một chiều

## ☐ Tạo mảng một chiều

- Từ object có định dạng array: sử dụng `array()`

```
import numpy as np
# tao mang mot chieu
array_1 = np.array([1, 2, 3])
print(array_1)
print(type(array_1))
```

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

# Mảng một chiều

- Từ list, tuple: sử dụng asarray()

```
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]

- Có giá trị mặc định là 0, 1: dùng zeros(), ones()

```
# Mảng với các phần tử có giá trị ban đầu là 0
mang = np.zeros((5,), dtype = np.int)
print(mang)
```

[0 0 0 0 0]

```
# Mảng với các phần tử có giá trị ban đầu là 1
mang_1 = np.ones(5)
print(mang_1)
```

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

# 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.]
```

# 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]
```

# Mảng một chiều

## □ Thao tác trên mảng

- Truy xuất phần tử trên mảng theo index (từ 0 đến chiều dài mảng -1)

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

Index:	-1
Value:	88 19 46 74 94

# Mảng một chiều

## □ Thao tác trên mảng

- Truy xuất phần tử trên mảng theo index

```
# 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]
```

# 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
```

# Mảng một chiều

## □ Thao tác trên mảng

- Thêm phần tử vào cuối mảng: sử dụng append()

```
# 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 vào index trong mảng: sử dụng insert()

```
# 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]

# Mảng một chiều

## □ Thao tác trên mảng

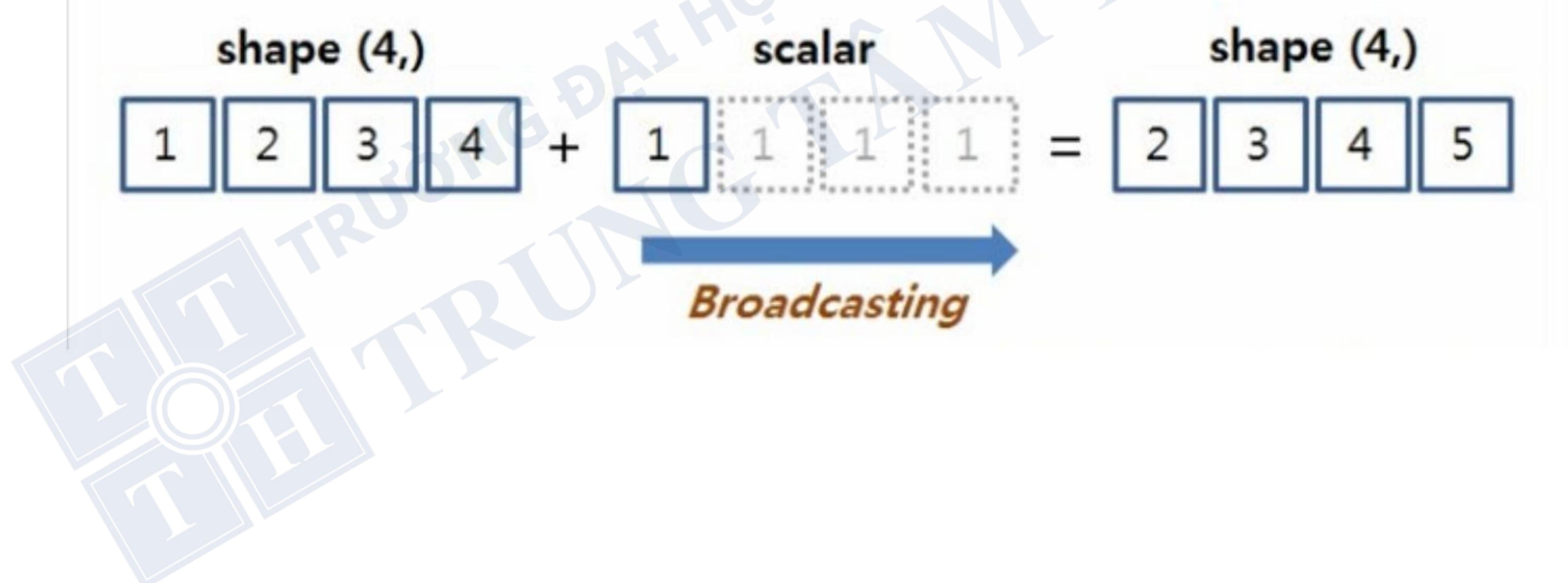
- Xóa phần tử tại index trong mảng: sử dụng `delete()`

```
# 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]
```

# Mảng một chiều

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



# Mảng một chiều

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

```
import numpy as np

a = np.array([1, 2, 3, 4])
b = np.array([5, 6, 7, 8])
mang_cong = a + b
print(mang_cong)
mang_tru = a - b
print(mang_tru)
mang_nhan = a * b
print(mang_nhan)
mang_chia = a / b
print(mang_chia)
```

```
[ 6  8 10 12]
[-4 -4 -4 -4]
[ 5 12 21 32]
[0.2          0.33333333 0.42857143 0.5      ]
```

# Mảng một chiều

## □ Statistical function

- Tìm max, min: sử dụng max() hoặc amax(), min hoặc amin()

```
# 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

# Mảng một chiều

## □ Statistical function

- Giá trị trung bình số học: sử dụng mean()

```
print(d)
print(np.mean(d))
```

```
[ 3 10  9 12  8  5]
7.833333333333333
```

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

```
print(d)
print(np.median(d)) # 3, 5, 8, 9, 10, 12 => trung bình giữa = (8+9)/2
```

```
[ 3 10  9 12  8  5]
8.5
```

# 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))
```

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

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

```
print(mang_5)
indexes = np.where(mang_5 >= 5)
print(indexes)
print(mang_5[indexes])
```

```
[3 1 2 4 7 3 8 9 5 6]
(array([4, 6, 7, 8, 9], dtype=int32),)
[7 8 9 5 6]
```

# Mảng một chiều

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

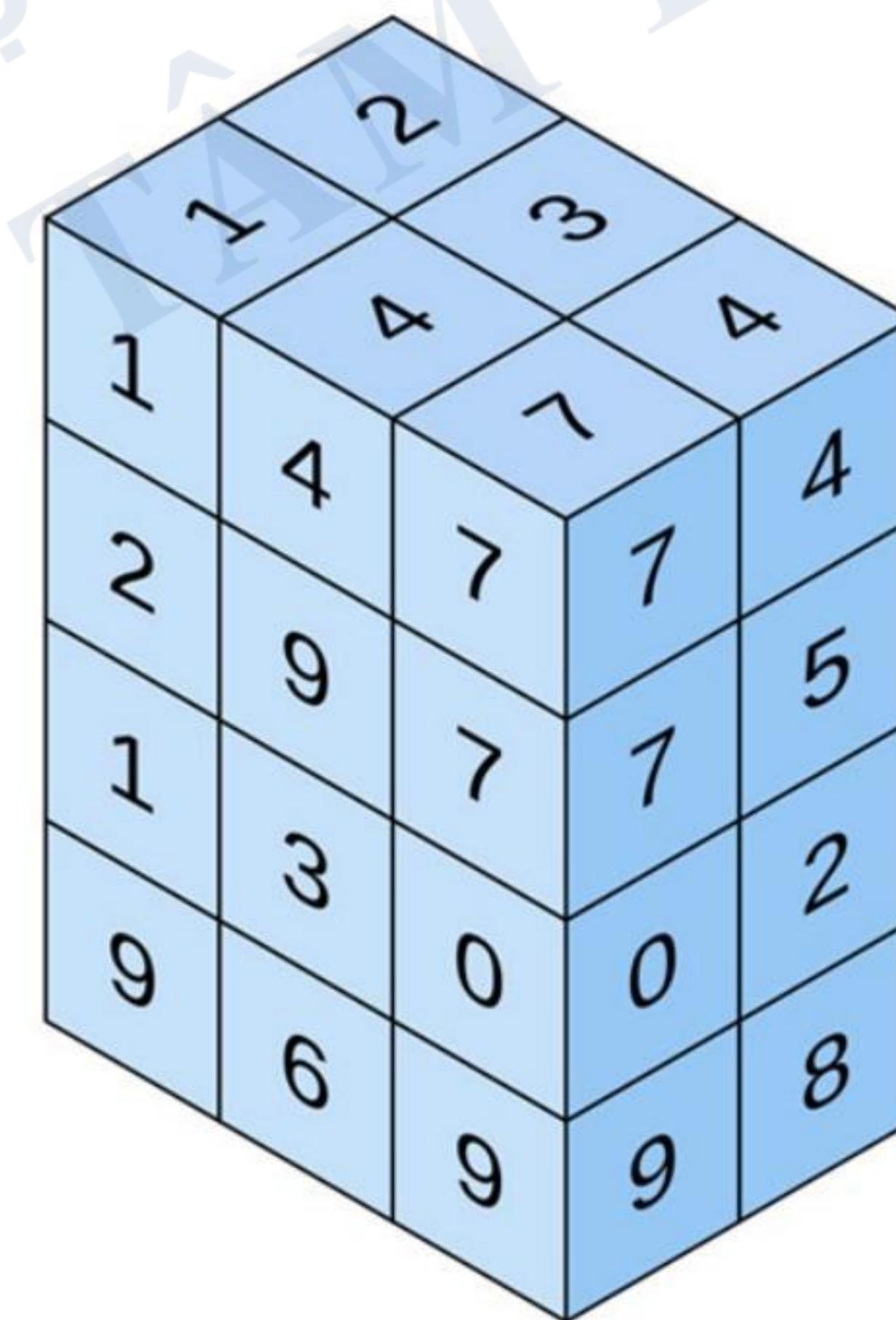
```
print(mang_5)
print(np.extract(np.mod(mang_5, 2)==0, mang_5))
print(np.extract(mang_5 >= 5, mang_5))
```

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

# Nội dung

1. Giới thiệu
2. Mảng một chiều
3. Mảng nhiều chiều

5.2	3.0	4.5
9.1	0.1	0.3



# Mảng nhiều chiều

## □ Tạo mảng nhiều chiều

- Từ object có định dạng array: sử dụng `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'>
```

# Mảng nhiều chiều

- Có giá trị mặc định là 0, 1: dùng zeros(), ones()

```
# tạo mảng với 0 hoặc 1
mang_zeros = np.zeros([2,2,2], dtype = int)
print("Mảng zeros")
print(mang_zeros)
print("Mảng ones")
mang_ones = np.ones([2,2], dtype = int)
print(mang_ones)
```

Mảng zeros

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

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

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

Mảng ones

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

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

# Mảng nhiều chiều

## □ Thao tác trên mảng

- Số dòng, cột trên mảng

```
print(array_2)
print(array_2.shape)
```

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

- Cập nhật phần tử trong mảng

```
# cập nhật
array_2[0][0] = 100
print(array_2)
```

```
[[100  2  3]
 [ 4  5  6]]
```

# Mảng nhiều chiều

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

```
# truy xuất phần tử trong mảng
print(array_2)
print(array_2[1,2])
print(array_2[1,:])
```

```
[[1 2 3]      print(b)
 [4 5 6]]    print("b[0][1][2] =", b[0][1][2]) # ma trận 0, dòng 1, cột 2
6
[4 5 6]      [[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
 [[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]]
b[0][1][2] = 6
```

# 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)
```

```
[[100  2  3]
 [ 4  5  6]]
(2, 3)
[[100  2]
 [ 3  4]
 [ 5  6]]
```

```
# 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]]]
```



# Mảng nhiều chiều

## ☐ Tạo mảng con

```
print(b)
b1 = b[1:]
print("b1:")
print(b1)
b1_x_2 = b[1,...,2]
print("b1_x_2:")
print(b1_x_2)
b11_x = b[1,1,...]
print("b11_x:")
print(b11_x)
```

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

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

b1:  
[[[12 13 14 15]  
 [16 17 18 19]  
 [20 21 22 23]]]

b1\_x\_2:  
[14 18 22]  
b11\_x:  
[16 17 18 19]

# Mảng nhiều chiều

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

```
import numpy as np
a = np.array([[1, 2, 3, 4], [1, 1, 1, 1], [1, 2, 3, 4], [2, 2, 2, 2]])
b = np.array([2, 4, 6, 8])
print(a)
print(b)
print("a + b")
print(a + b)
print("a - b")
print(a - b)
print("a * b")
print(a * b)
print("a / b")
print(a / b)
```

```
[[1 2 3 4]
 [1 1 1 1]
 [1 2 3 4]
 [2 2 2 2]]
[2 4 6 8]
a + b
[[ 3  6  9 12]
 [ 3  5  7  9]
 [ 3  6  9 12]
 [ 4  6  8 10]]
a - b
[[-1 -2 -3 -4]
 [-1 -3 -5 -7]
 [-1 -2 -3 -4]
 [ 0 -2 -4 -6]]
a * b
[[ 2  8 18 32]
 [ 2  4  6  8]
 [ 2  8 18 32]
 [ 4  8 12 16]]
a / b
[[0.5          0.5          0.5          0.5        ]
 [0.5          0.25         0.16666667 0.125      ]
 [0.5          0.5          0.5          0.5        ]
 [1.           0.5          0.33333333 0.25      ]]
```

# Mảng nhiều chiều

## □ Statistical function

- Tìm max, min: sử dụngamax(), amin()

```
print("Max:", np.amax(a))
print("Min:", np.amin(a))
print("Max - theo từng cột:", np.amax(a, 0))
print("Min - theo từng cột:", np.amin(a, 0))
print("Max - theo từng dòng:", np.amax(a, 1))
print("Min - theo từng dòng:", np.amin(a, 1))
```

```
Max: 9
Min: 2
Max - theo từng cột: [8 7 9]
Min - theo từng cột: [2 4 3]
Max - theo từng dòng: [7 8 9]
Min - theo từng dòng: [3 3 2]
```

# Mảng nhiều chiều

## □ Statistical function

- Giá trị trung bình số học: sử dụng mean()

```
b = np.array([[[3,7],[8,4]],[[2,4], [1, 5]]])
print(b)
print('Mean b: ')
print(np.mean(b))
print('Mean b[0]: ')
print(np.mean(b[0]))
```

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

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

```
Mean b:
```

```
4.25
```

```
Mean b[0]:
```

```
5.5
```

# Mảng nhiều chiều

## □ Statistical function

- Giá trị trung bình giữa: 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

# Mảng nhiều chiều

- Tính toán độ lệch chuẩn theo trực được chỉ định: sử dụng 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]
```

# Mảng nhiều chiều

- Tính toán hệ số tương quan theo thời điểm (Pearson Product)

```
a = np.array([[1,4,6], [1,2,3]])
print(a)
print("Coefficient")
print(np.corrcoef(a[0],a[1]))
```

```
[[1 4 6]
 [1 2 3]]
Coefficient
[[1.          0.99339927]
 [0.99339927 1.        ]]
```

# Mảng nhiều chiều

## ❑ 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]]]
```

# Mảng nhiều chiều

## ❑ Sort, where, extract

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

```
dt = np.dtype([('name', 'S10'), ('age', int)])
ds = np.array([('Tom', 21), ('Jenny', 25), ('Herry', 17), ('Matt', 27)], dtype = dt)
print(ds)

[(b'Tom', 21) (b'Jenny', 25) (b'Herry', 17) (b'Matt', 27)]
```

```
print('Order by name:')
print(np.sort(ds, order = 'name'))
print('Order by age:')
print(np.sort(ds, order = 'age'))
```

```
Order by name:
[(b'Herry', 17) (b'Jenny', 25) (b'Matt', 27) (b'Tom', 21)]
Order by age:
[(b'Herry', 17) (b'Tom', 21) (b'Jenny', 25) (b'Matt', 27)]
```

# Mảng nhiều chiều

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

```
print(c)
cond = np.where(c>5)
print(cond)
print(c[cond])
```

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

```
[[2 4]
 [6 5]]]
(array([0, 0, 0, 1], dtype=int32), array([0, 0, 1, 1], dtype=int32), array([0, 1, 0, 0], dtype=int32))
[9 7 8 6]
```

# Mảng nhiều chiều

- Tìm 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]
```

# Mảng dữ liệu tự định nghĩa

## ❑ Numpy – Data Type Objects (dtype)

- Dtype mô tả:

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

# Mảng dữ liệu tự định nghĩa

- 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, thêm phần đệm (padding) vào để làm cho nó tương tự như C-struct
  - Copy: nếu = True, tạo một bản sao mới của dtype object

# Mảng dữ liệu tự định nghĩa

## • 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]

# Mảng dữ liệu tự định nghĩa

- 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')])
```

