ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

CS116 - LẬP TRÌNH PYTHON CHO MÁY HỌC

LẬP TRÌNH VỚI THƯ VIỆN NUMPY

TS. Nguyễn Vinh Tiệp

Cài đặt thư viện NumPy

```
conda install numpy
```

or

```
pip install numpy
```

```
import numpy as np
```

Tại sao cần Đại số tuyến tính

- Biểu diễn dữ liệu gốc bằng các khái niệm tensor:
  - Tensor 0 chiều – Scalar
  - Tensor 1 chiều – Vector
  - Tensor 2 chiều – Ma trận
  - Tensor nhiều chiều

- Biến đổi dữ liệu bằng các phép toán trên tensor

Khái niệm tensor

- Tensor 0 chiều – Scalar
- Tensor 1 chiều – Vector
- Tensor 2 chiều – Ma trận
- Tensor nhiều chiều

Scalar – Tensor 0 chiều

0
1
2
-2
-5
3

Số nguyên

0.1
1.5
2.1
-2.6
3.2

Số thực

Scalar – Định nghĩa

- Ký hiệu: $a \in \mathbb{R}$ là một scalar vô hướng (số chiều bằng 0) chứa duy nhất 1 số thực
- Scalar được dùng để biểu diễn **tốc độ, khối lượng, chiều dài**
- Cách gọi khác: *scalar tensor, 0-dimensional tensor, 0D tensor*
- Trong Numpy, các tensor có 2 thông số quan trọng là ‘*ndim*’ và ‘*shape*’:

```
x = np.array(12)
print('x =', x)
print('x.ndim =', x.ndim)
print('x.shape =', x.shape)
```

```
x = 12
x.ndim = 0
x.shape = ()
```

Vector – Tensor 1 chiều

Tập hợp thông tin các nhà trong một khu phố

Diện tích (m^2)	Số phòng ngủ	Giá (K$)
2104	3	400
1600	3	330
2400	3	369
1416	2	232

Vector chứa thông tin của 1 căn nhà
(2104, 3, 400)

Vector – Khai báo

Vector được khai vector trong NumPy qua hàm array:

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

or:

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

Xuất phần tử đầu tiên

```
>>> print(a[0])
[1 2 3 4]
```

Vector – Định nghĩa

- Ký hiệu: $\mathbf{x} \in \mathbb{R}^n$ là một vector với 1 chiều không gian, gồm n phần tử
- Vector gồm tập hợp các scalar, viết dưới dạng cột: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
- Python:

```python
x = np.array([12, 3, 6, 14])
print('x =', x)
print('x.ndim =', x.ndim)
print('x.shape =', x.shape)
```

```
x = [12  3  6 14]
x.ndim = 1
x.shape = (4,)
```

Trực quan hóa

Vector – Hàm khai báo phổ biến

```
>>> np.zeros(2)
array([0., 0.])
```

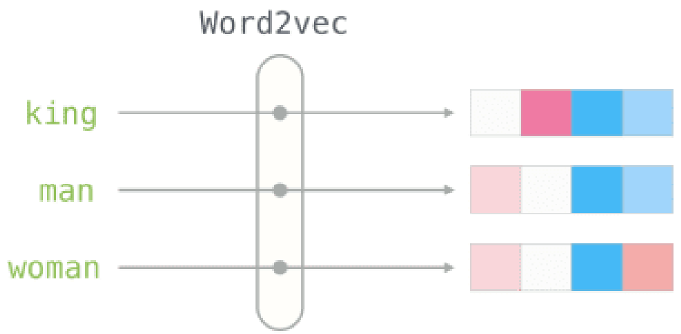
```
>>> np.ones(2)
array([1., 1.])
```

```
>>> # Create an empty array with 2 elements
>>> np.empty(2)
array([3.14, 42.  ])  # may vary
```

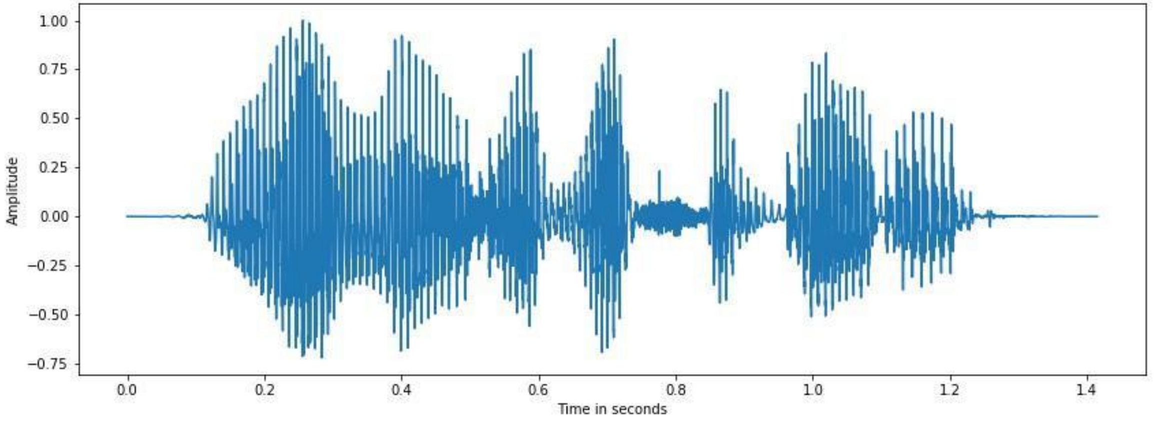
```
>>> np.arange(4)
array([0, 1, 2, 3])
```

Vector - Biểu diễn dữ liệu

- Các từ được biểu diễn dưới dạng vector trước khi tính toán



- Tín hiệu âm thanh được biểu diễn dưới dạng vector



Ma trận – Giới thiệu

Tập hợp thông tin các căn nhà trong một khu phố

Diện tích (m^2)	Số phòng ngủ	Giá (K$)
2104	3	400
1600	3	330
2400	3	369
1416	2	232

Ma trận chứa thông tin của tập hợp các ngôi nhà

Ma trận – Định nghĩa

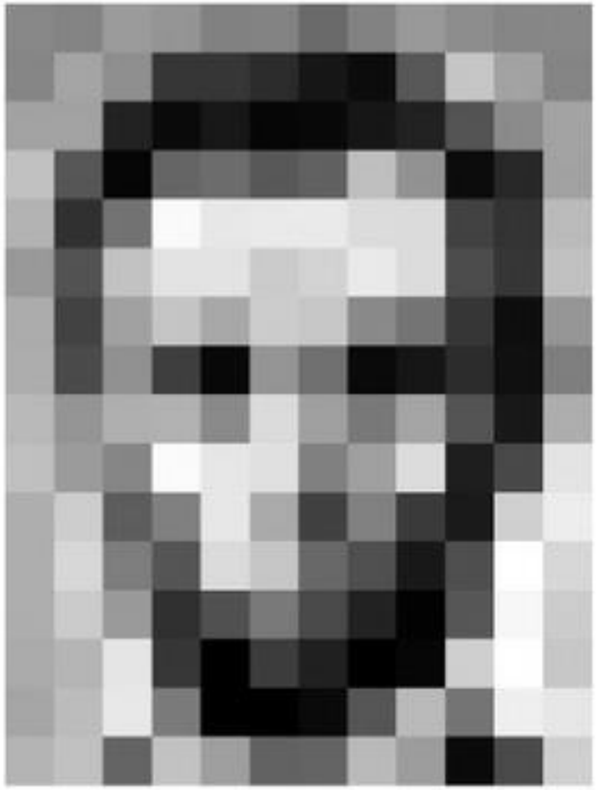
- Ký hiệu: $A \in \mathbb{R}^{m \times n}$ là một ma trận với 2 chiều không gian, gồm m dòng và n cột
- Ma trận gồm tập hợp các vector **cùng kích thước**: $A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & & A_{2n} \\ \vdots & & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}$
- Python:

```
x = np.array(
    [[1, 2, 3, 4],
     [5, 6, 7, 8],
     [9, 1, 2, 3]]
)
print('x =', x)
print('x.ndim =', x.ndim)
print('x.shape =', x.shape)
```

```
x = [[1 2 3 4]
 [5 6 7 8]
 [9 1 2 3]]
x.ndim = 2
x.shape = (3, 4)
```

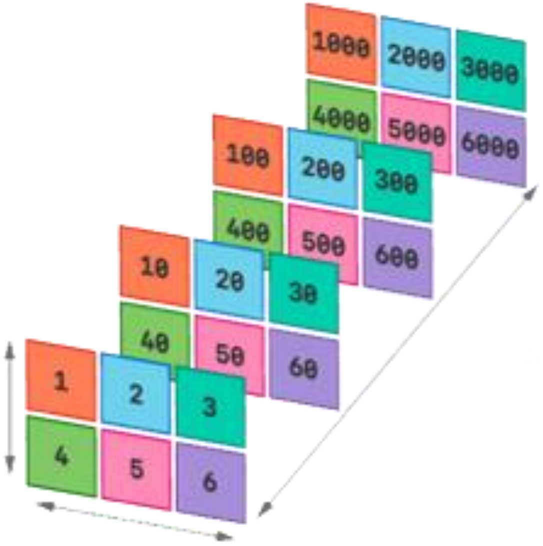
Ma trận - Biểu diễn dữ liệu

- Một ảnh xám được biểu diễn thô dưới dạng ma trận



3D Tensor

- Ký hiệu: $\mathbf{X} \in \mathbb{R}^{m \times n \times p}$ là một 3D tensor với 3 chiều không gian, có kích thước từng chiều là: m, n và p
- 3D Tensor bao gồm tập hợp các ma trận **cùng kích thước**
- Python:




```
x = np.array(
    [[[1, 2, 3],
      [4, 5, 6]],
     [[10, 20, 30],
      [40, 50, 60]],
     [[100, 200, 300],
      [400, 500, 600]],
     [[1000, 2000, 3000],
      [4000, 5000, 6000]]]
)
print('x.ndim =', x.ndim)
print('x.shape =', x.shape)
```

```
x.ndim = 3
x.shape = (4, 2, 3)
```

3D Tensor - Biểu diễn dữ liệu

- Một bức ảnh màu RGB có thể lưu dưới dạng 3D Tensor

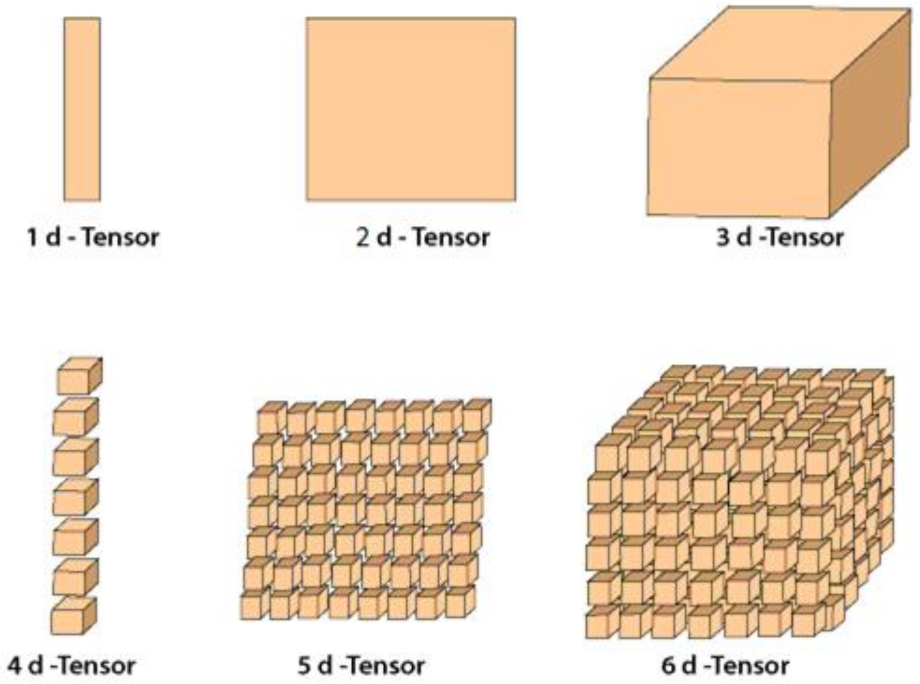
5 10 55 0 6

7 20 5 12 7

0	120	3	0	6
100	0	43	2	128
2	1	255	0	1
34	0	4	5	7
1	1	56	0	1

Các Tensor khác

- Nếu **ghép** các 3D Tensor **cùng kích thước**, chúng ta thu được 4D Tensor
- Tương tự như vậy, ta sẽ thu được các Tensor với chiều lớn hơn 4



Các Tensor khác – Biểu diễn dữ liệu

3D: bao gồm 3 kênh màu

Video là một tensor 4D

Một số phép toán – Phép chuyển vị

- Phép chuyển vị (transpose): là một toán tử biến các vector cột của một ma trận thành dòng hoặc ngược lại

$$A = \begin{bmatrix} | & \boxed{\begin{matrix} | \\ a_2 \\ | \end{matrix}} & & | \\ a_1 & & \cdots & a_n \\ | & & & | \end{bmatrix}$$

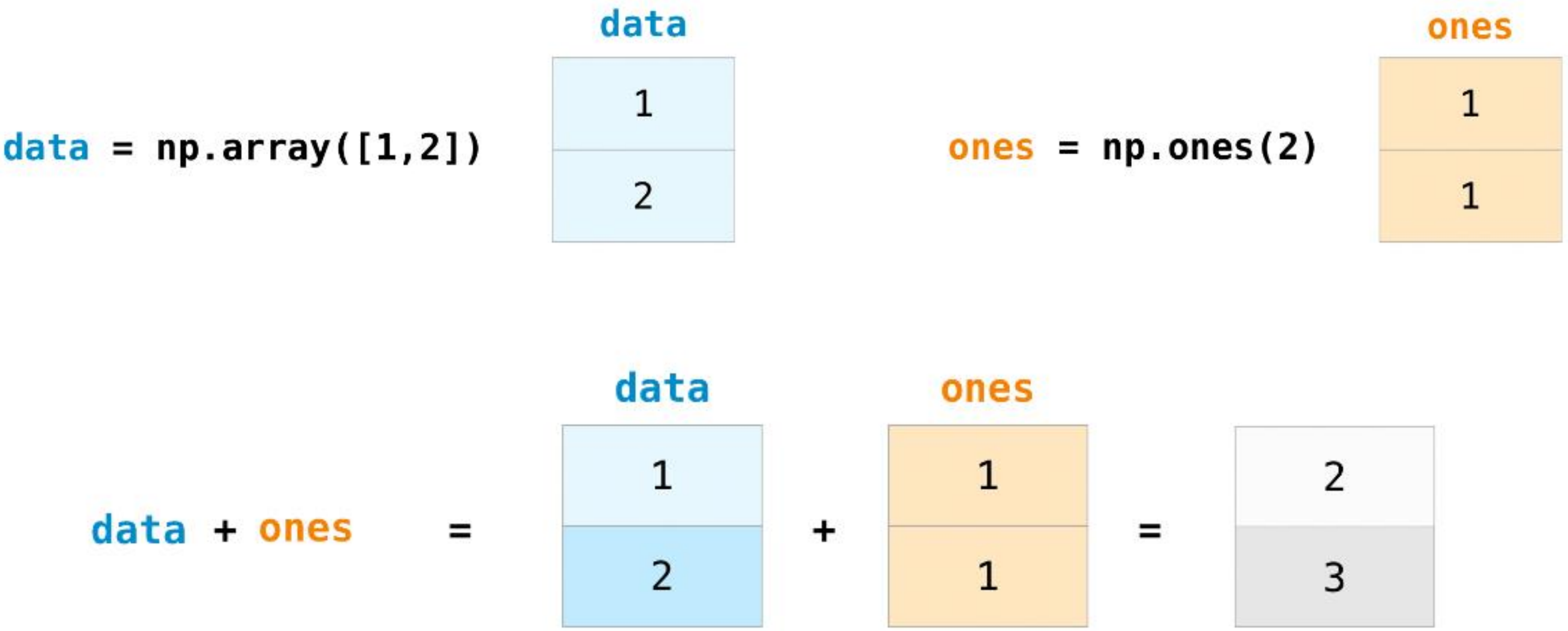
$$A^\top = \begin{bmatrix} — & a_1^\top & — \\ \hline — & a_2^\top & — \\ \hline & \vdots & \\ — & a_n^\top & — \end{bmatrix}$$

$$\begin{array}{|cccc|} \hline 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ \hline \end{array} \quad A$$

Phép toán cộng vector

- Tổng hai vector $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$: $\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}$

Ví dụ phép toán cộng vector



Phép cộng ma trận

- Tổng hai ma trận $A, B \in \mathbb{R}^{m\times n}$:

$$A + B = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & & A_{2n} \\ \vdots & & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1n} \\ B_{21} & B_{22} & & B_{2n} \\ \vdots & & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} & \cdots & A_{1n} + B_{1n} \\ A_{21} + B_{21} & A_{22} + B_{22} & & A_{2n} + B_{2n} \\ \vdots & & \ddots & \vdots \\ A_{m1} + B_{m1} & A_{m2} + B_{m2} & \cdots & A_{mn} + B_{mn} \end{bmatrix}$$

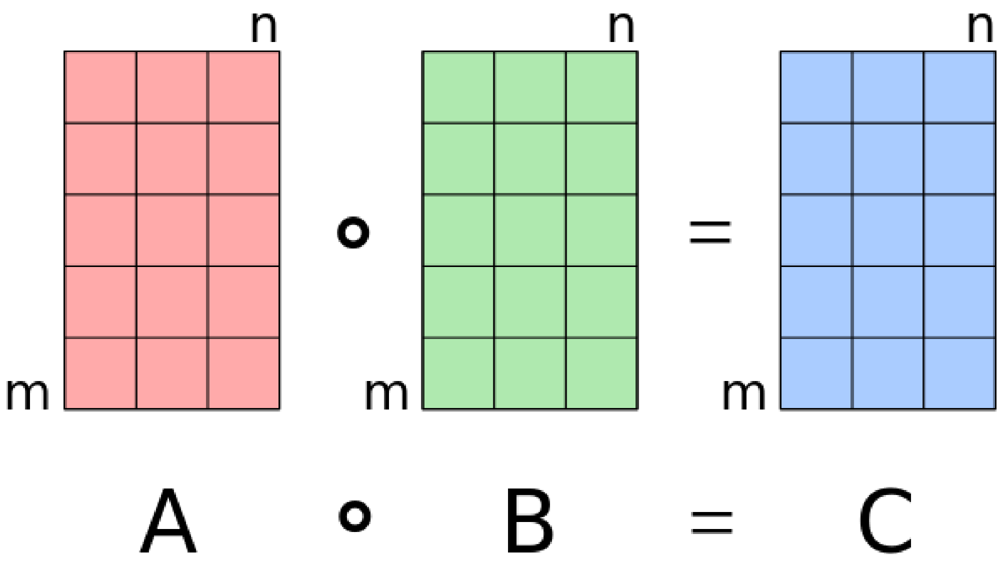
Phép cộng ma trận – Ví dụ

- Để chuyển đổi một bức ảnh màu RGB sang một bức ảnh xám ta có thể sử dụng phép cộng trung bình

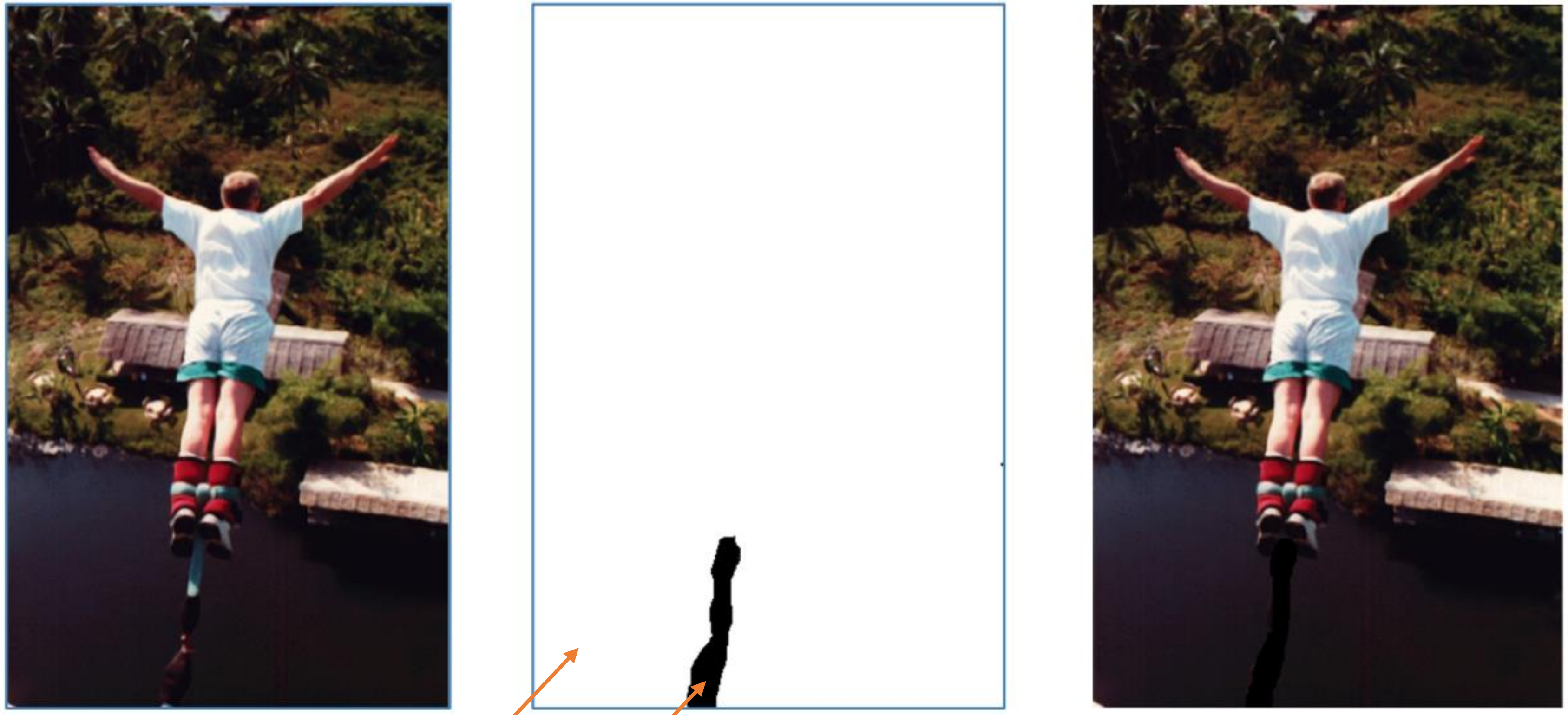
Tích Hadamard

- Tích Hadamard hai ma trận $A \in \mathbb{R}^{m \times n}$ và $B \in \mathbb{R}^{m \times n}$: $C = A \circ B \in \mathbb{R}^{m \times n}$

$$C_{ij} = A_{ij} B_{ij}$$



Tích Hadamard: Ví dụ



Img

Mask

Kết quả = Img $\circ$ Mask

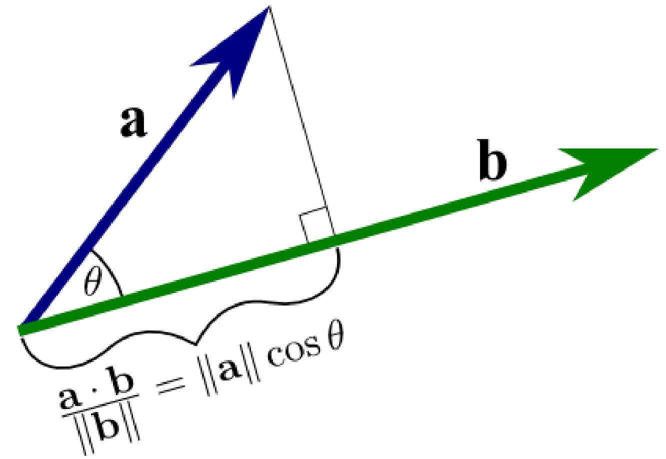
1

0

Phép nhân vector - vector

- Tích vô hướng hai vector $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$:

$$\mathbf{a}^{\mathrm{T}}\mathbf{b} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \sum_{i=1}^{n} a_i b_i$$



- Chú ý: $\mathbf{a}^{\mathrm{T}}\mathbf{b} = \mathbf{b}^{\mathrm{T}}\mathbf{a}$

Phép nhân vector – vector: Ví dụ

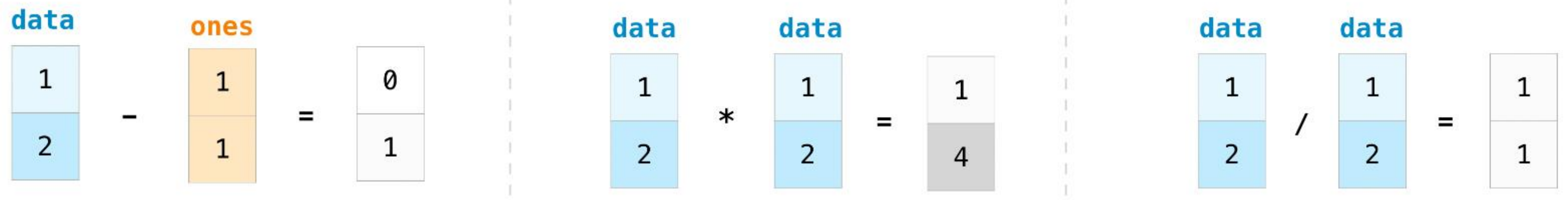
Thành phần	Cà rốt	Bắp cải	Dưa leo
Vitamin A (mg/kg)	35	0.5	0.5
Vitamin C (mg/kg)	60	300	10
Chất xơ (g/kg)	30	20	10

- Gọi x, y, z là số kg "Cà rốt", "Bắp cải", "Dưa leo" mua được. Khi đó lượng vitamin A của bữa ăn là:

$$\begin{bmatrix} 35 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Ví dụ các toán tử vector

```
>>> data - ones
array([0, 1])
>>> data * data
array([1, 4])
>>> data / data
array([1., 1.])
```



Broadcasting

- Kết quả 2 dòng lệnh dưới:
  1. Chạy được
  2. Báo lỗi

```
>>> data = np.array([1.0, 2.0])
>>> data * 1.6
array([1.6, 3.2])
```

Broadcasting

```
>>> data = np.array([1.0, 2.0])
>>> data * 1.6
array([1.6, 3.2])
```

Phép nhân ma trận - vector

- Cho ma trận $A \in \mathbb{R}^{m \times n}$ và vector $\mathbf{x} \in \mathbb{R}^{n}$, tích $y = A\mathbf{x} \in \mathbb{R}^{m}$ là vector:

$$y = A\mathbf{x} = \begin{bmatrix} - \; a_1 \; - \\ \boxed{- \; a_2 \; -} \\ \vdots \\ - \; a_n \; - \end{bmatrix} \boxed{\mathbf{x}} = \begin{bmatrix} a_1^{\mathrm{T}}\mathbf{x} \\ \boxed{a_2^{\mathrm{T}}\mathbf{x}} \\ \vdots \\ a_n^{\mathrm{T}}\mathbf{x} \end{bmatrix}$$

- Lưu ý:
    - Kết quả của phép tính luôn là một vector
    - Một ma trận $m \times n$ nhân với một vector $n \times 1$ sẽ có tích là một vector $m \times 1$

Phép nhân ma trận – vector: Ví dụ 1

Thành phần	Cà rốt	Bắp cải	Dưa leo
Vitamin A (mg/kg)	35	0.5	0.5
Vitamin C (mg/kg)	60	300	10
Chất xơ (g/kg)	30	20	10

- Gọi x, y, z là số kg "Cà rốt", "Bắp cải", "Dưa leo" mua được. Khi đó lượng vitamin A của bữa ăn là:

$$\begin{bmatrix} 35 & 0.5 & 0.5 \\ 60 & 300 & 10 \\ 30 & 20 & 10 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Phép nhân ma trận – vector: Ví dụ 2

- Ta có thể biểu diễn một hệ phương trình dưới dạng phép nhân ma trận – vector

$$\begin{aligned} a_1x + b_1y + c_1z &= C_1 \\ a_2x + b_2y + c_2z &= C_2 \\ a_3x + b_3y + c_3z &= C_3 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix}$$

Phép nhân ma trận – ma trận

- Tích hai ma trận $A \in \mathbb{R}^{m\times n}$ và $B \in \mathbb{R}^{n\times p}$: $C = A.B \in \mathbb{R}^{m\times p}$

$$C_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj}$$

$$C = AB = \begin{bmatrix} - & a_1^\top & - \\ - & a_2^\top & - \\ & \vdots & \\ - & a_m^\top & - \end{bmatrix} \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} a_1^\top b_1 & a_1^\top b_2 & \cdots & a_1^\top b_p \\ a_2^\top b_1 & a_2^\top b_2 & \cdots & a_2^\top b_p \\ \vdots & \vdots & \ddots & \vdots \\ a_m^\top b_1 & a_m^\top b_2 & \cdots & a_2^\top b_p \end{bmatrix}$$

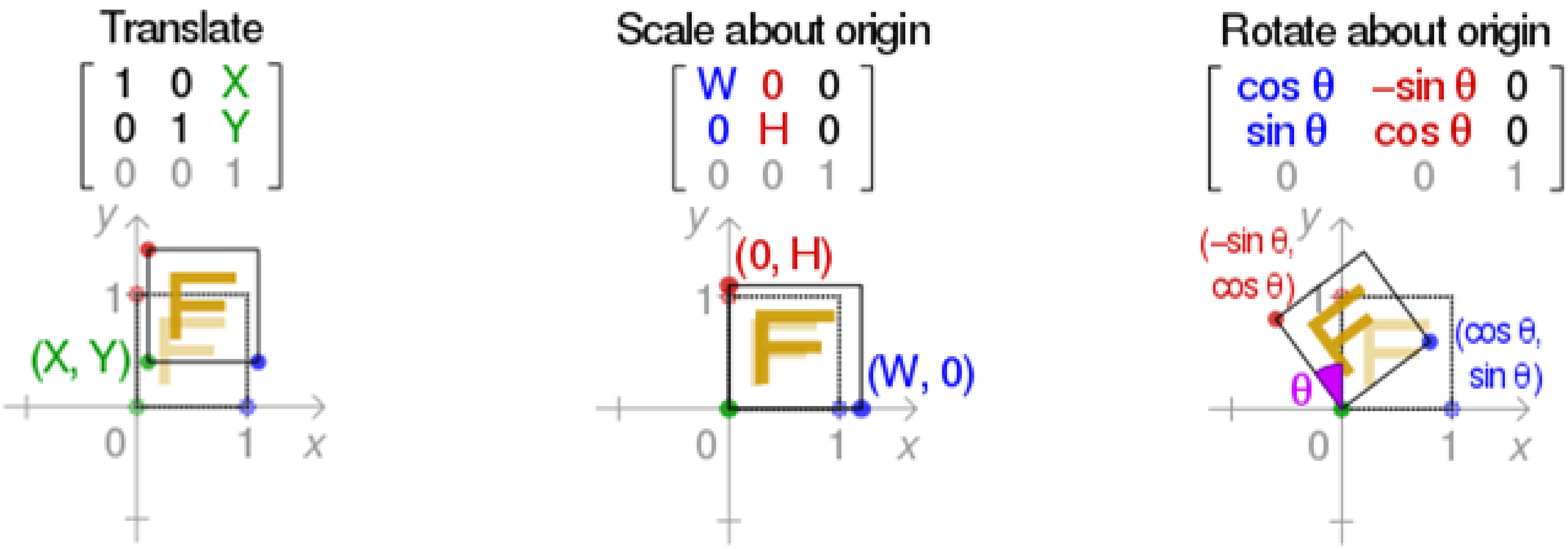
- Tính chất:
  - Phép nhân ma trận không có tính giao hoán: $AB \neq BA$
  - Tính kết hợp: $(AB)C = A(BC)$
  - Tính kết hợp: $A(B + C) = AB + AC$
  - Kết hợp với chuyển vị: $(A^{\mathbf{T}})^{\mathbf{T}} = A$

    $(AB)^{\mathbf{T}} = B^{\mathbf{T}}A^{\mathbf{T}}$

    $(A + B)^{\mathbf{T}} = A^{\mathbf{T}} + B^{\mathbf{T}}$

Phép nhân ma trận – Ví dụ

- Chúng ta có thể sử dụng các phép tính ma trận cho các biến đổi hình học



Một số ma trận đặc biệt

- **Ma trận đơn vị**: ký hiệu là $I_n \in \mathbb{R}^{n \times n}$ là ma trận vuông với tất cả phần tử trên đường chéo chính bằng 1, các phần tử còn lại bằng 0

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

- Tính chất: Với mọi A ta có $AI_n = A$
- Trong Python, ta dùng hàm '*eye*' để tạo ma trận đơn vị

Một số ma trận đặc biệt

- **Ma trận nghịch đảo**: của ma trận $A \in \mathbb{R}^{n \times n}$ ký hiệu là ma trận $A^{-1} \in \mathbb{R}^{n \times n}$, là ma trận thỏa mãn:

$$AA^{-1} = A^{-1}A = I_n$$

Tài liệu tham khảo

- numpy.org/doc/stable/

BÀI QUIZ VÀ HỎI ĐÁP