AI VIET NAM
@aivietnam.edu.vn

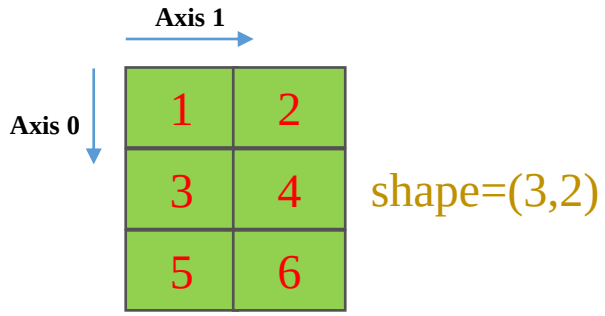
Introduction to Numpy

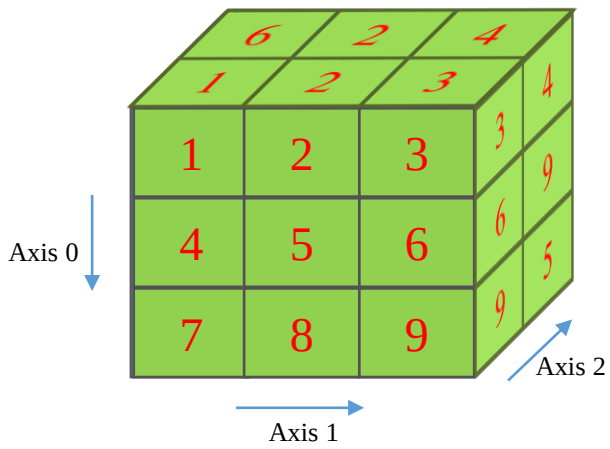
Quang-Vinh Dinh
PhD in Computer Science

Year 2024

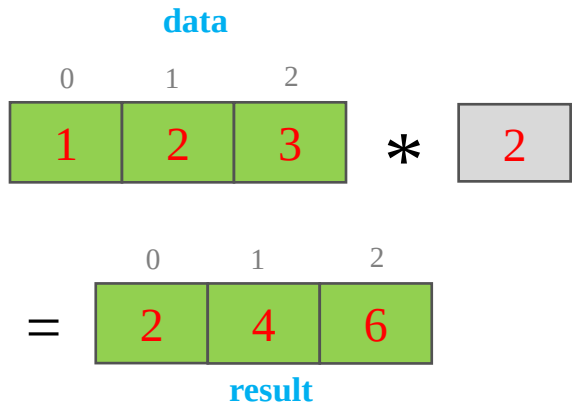
Objectives

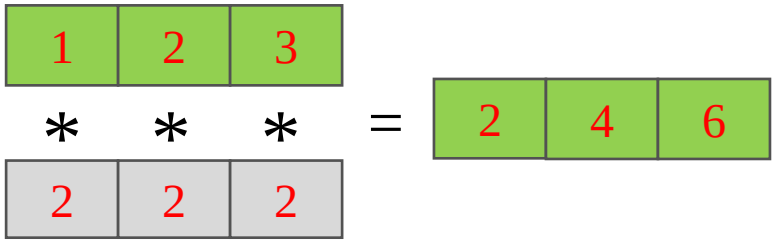
Indexing





Broadcasting





Examples

Feature	Label
Petal_Length	Category
1.4	0
1	0
1.5	0
3	1
3.8	1
4.1	1

One-hot encoding for label

$$y = 0 \rightarrow \boldsymbol{y} = [1 \quad 0]$$

$$y = 1 \rightarrow \boldsymbol{y} = [0 \quad 1]$$

scalar vector

Outline

SECTION 1

Introduction

SECTION 2

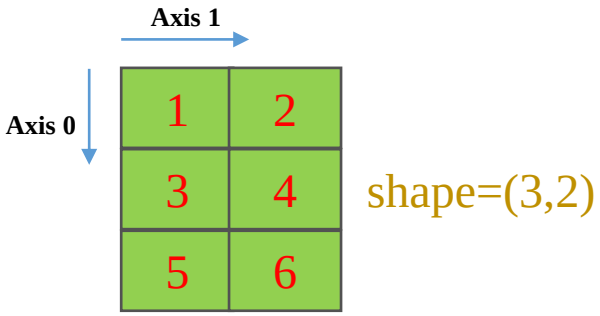
Common Functions

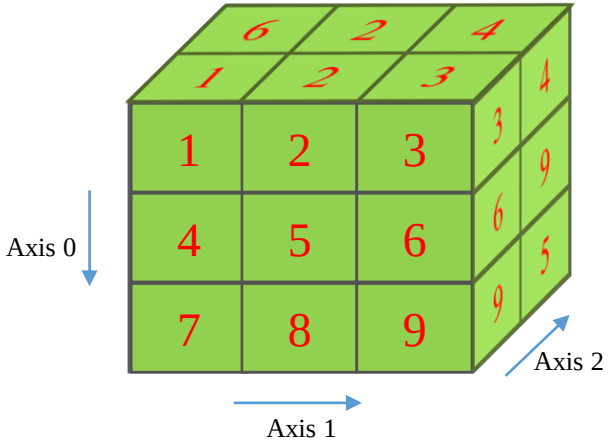
SECTION 3

Indexing and Broadcasting

SECTION 4

Examples



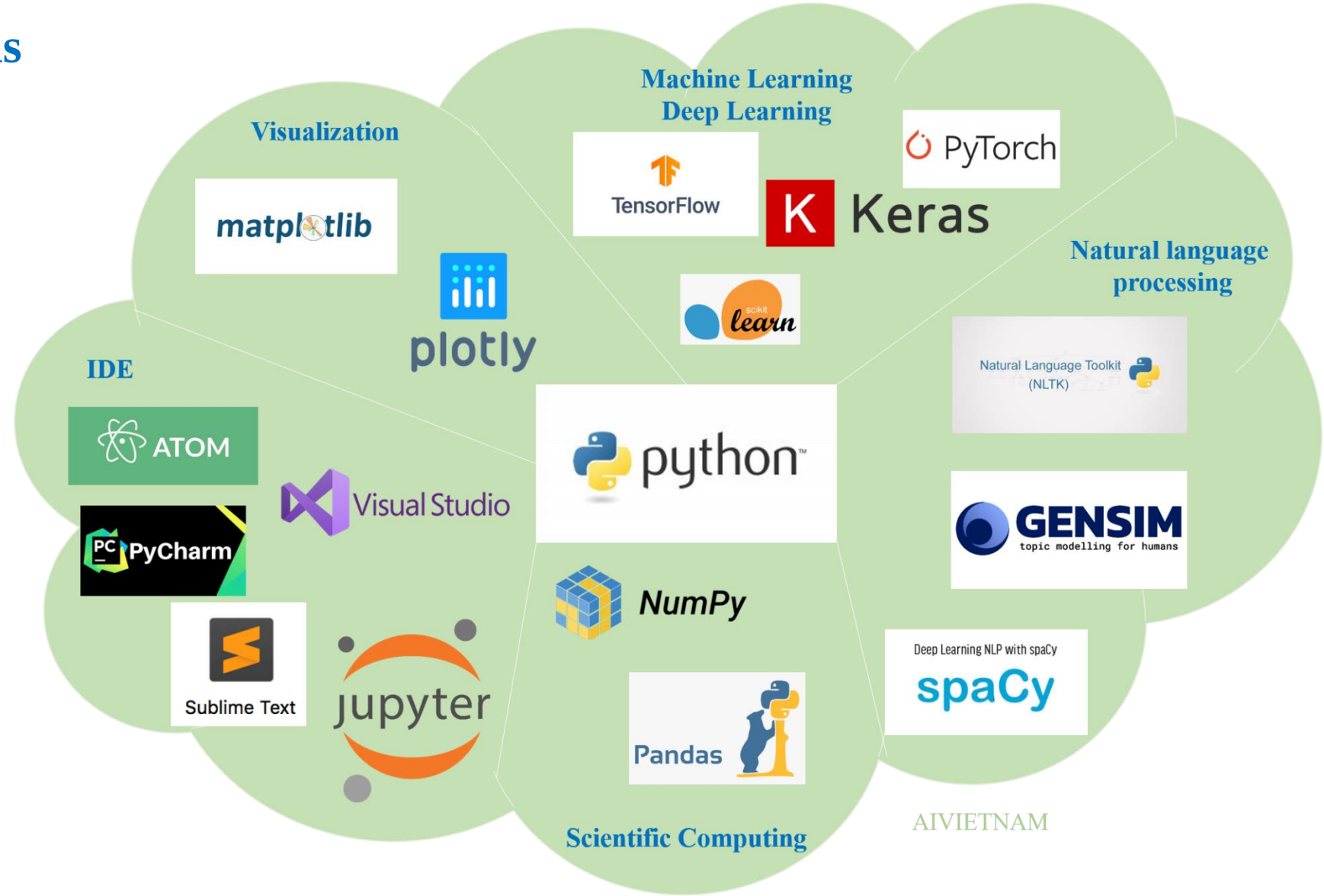


Introduction

- Numpy is a Python library
- For scientific computations

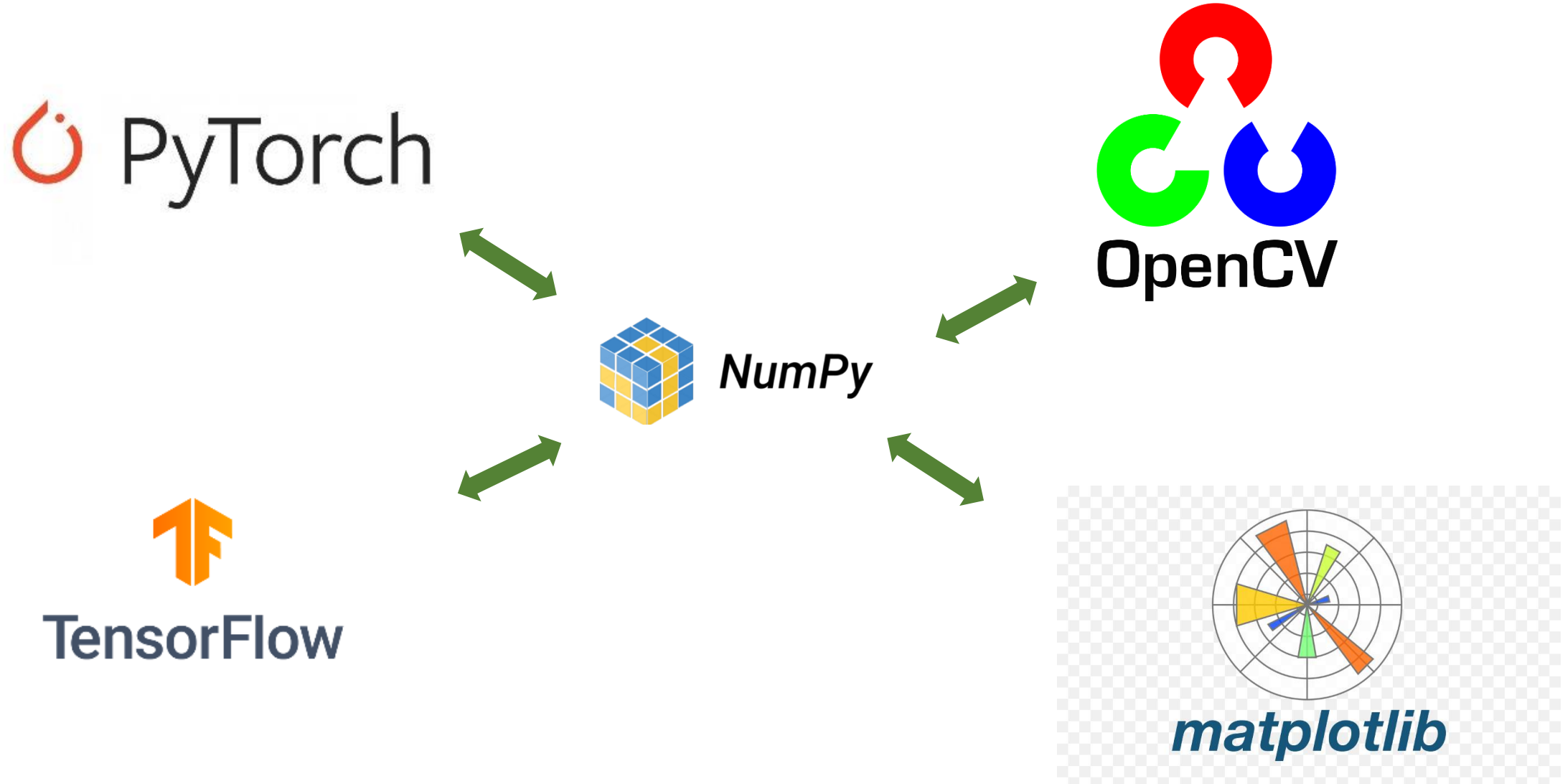
NumPy (pronounced /ˈnʌmpaɪ/ *NUM-py*) is a library for the Python programming language

Original author(s)	Travis Oliphant
Developer(s)	Community project
Initial release	As Numeric, 1995; as NumPy, 2006
Stable release	2.0.0[1] / 16 June 2024; 12 days ago
Repository	github.com/numpy/numpy
Written in	Python, C
Operating system	Cross-platform
Type	Numerical analysis
License	BSD[2]
Website	numpy.org



Introduction

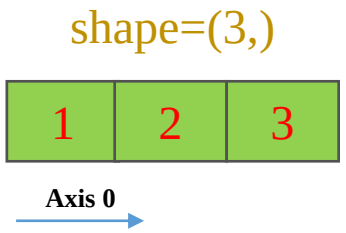
❖ Data type for common libraries



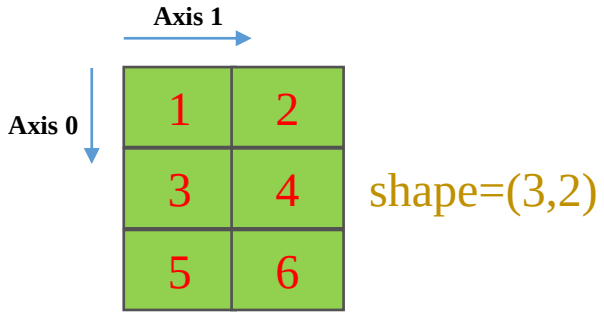
Introduction

arr_np = np.array(python_list)

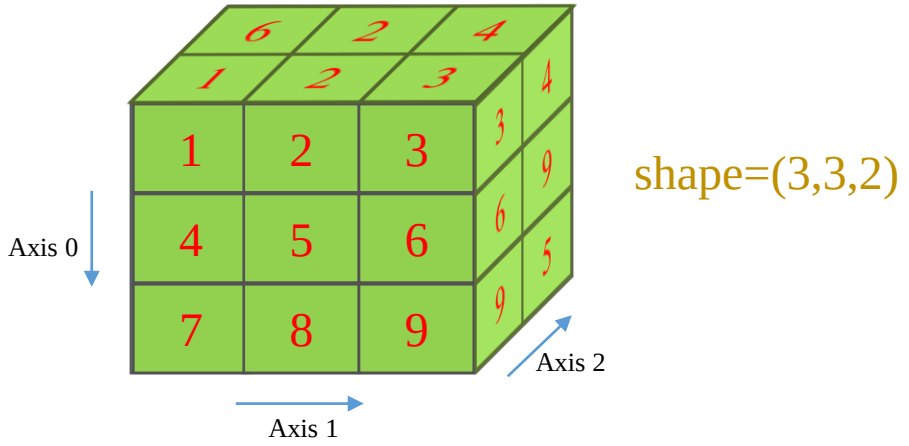
❖ Numpy arrays are multi-dimensional arrays



1D array



2D array



3D array

```python
import numpy as np

# tạo list
list1D = [1,2,3]

# tạo ndarray
data = np.array(list1D)

print(data)
print(data.shape)
```

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

```python
import numpy as np

# tạo list
list2D = [[1,2],[3,4],[5,6]]

# tạo ndarray
data = np.array(list2D)

print(data)
print(data.shape)
```

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

```python
import numpy as np

# tạo list
list3D = [[[1,6], [2,2], [3,4]],
          [[4,7], [5,2], [6,9]],
          [[7,7], [8,2], [9,5]]]

# tạo ndarray
data = np.array(list3D)

#print(data)
print(data.shape)
```

```
(3, 3, 2)
```

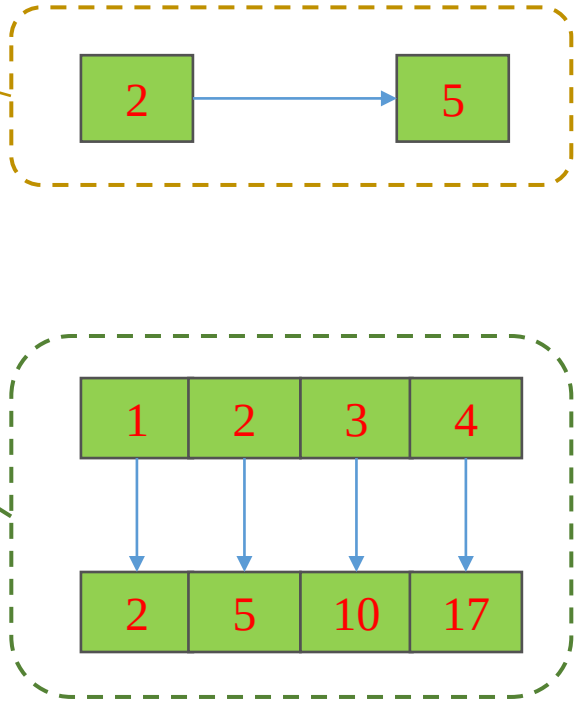
Motivation

❖ Why do I need Numpy?

```python
def f_function(number):
    result = number**2 + 1
    return result

def f_function_list(numbers):
    result = []
    for n in numbers:
        result.append(f_function(n))
    return result

# test
numbers = [1, 2, 3, 4]
print(f_function_list(numbers))
```

✓ 0.0s

```
[2, 5, 10, 17]
```



Motivation

❖ Why do I need Numpy?

```python
def f_function(number):
    result = number**2 + 1
    return result

def f_function_list(numbers):
    result = []
    for n in numbers:
        result.append(f_function(n))
    return result

# test
numbers = [1, 2, 3, 4]
print(f_function_list(numbers))
```

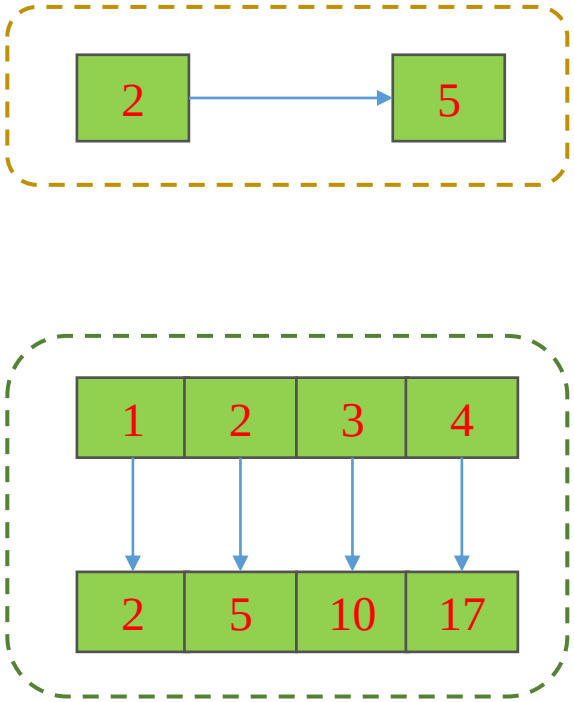
✓ 0.0s

```
[2, 5, 10, 17]
```

```python
import numpy as np

def f_function(number):
    result = number**2 + 1
    return result

factorial_np = np.vectorize(f_function)

# test
numbers = [1, 2, 3, 4]
numbers_array = np.array(numbers)
print(factorial_np(numbers_array))
```

✓ 0.0s

```
[ 2  5 10 17]
```



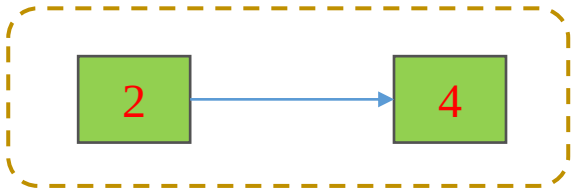
Motivation

❖ Why do I need Numpy?

```python
def f_function(number):
    result = number**2 + 1
    return result

def f_function_list(numbers):
    result = []
    for n in numbers:
        result.append(f_function(n))
    return result

# test
numbers = [1, 2, 3, 4]
print(f_function_list(numbers))
```

✓ 0.0s

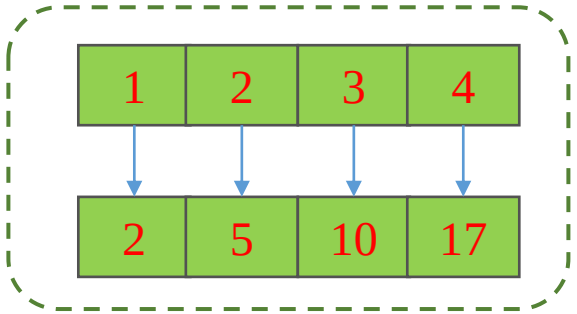
```
[2, 5, 10, 17]
```




```python
import numpy as np

def f_function(number):
    result = number**2 + 1
    return result

factorial_np = np.vectorize(f_function)

# test
numbers = [1, 2, 3, 4]
numbers_array = np.array(numbers)
print(factorial_np(numbers_array))
```

✓ 0.0s

```
[ 2  5 10 17]
```




```python
import numpy as np

# data
numbers = [1, 2, 3, 4]
numbers_np = np.array(numbers)

# compute
numbers_np = numbers_np**2 + 1
print(numbers_np)
```

✓ 0.0s

```
[ 2  5 10 17]
```

Implementation view
implement for each element

Execution view
run for all the elements

Outline

SECTION 1

Introduction

SECTION 2

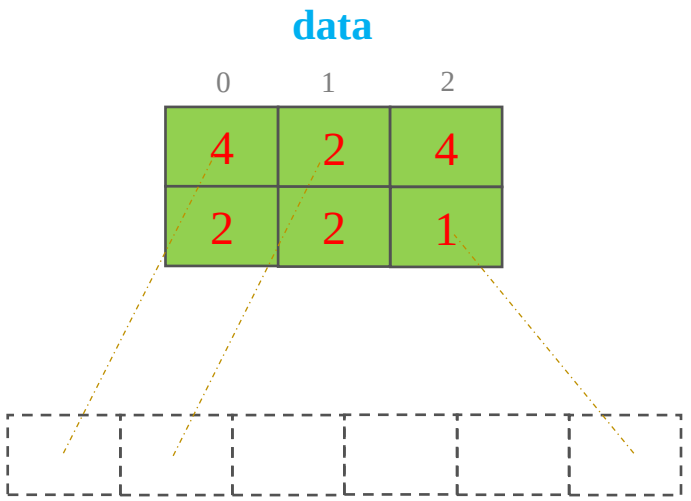
Common Functions

SECTION 3

Indexing and Broadcasting

SECTION 4

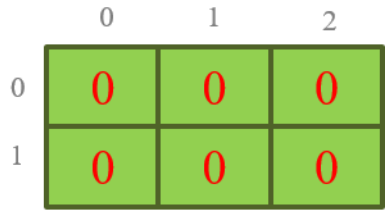
Examples



Some Common Functions

❖ Create Numpy arrays

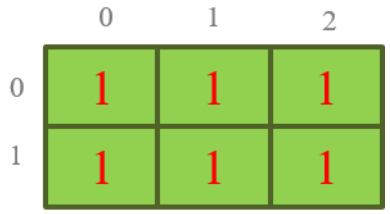
zeros() function




```python
# aivietnam.ai
# Tạo một numpy array
# với tất cả phần tử là 0

import numpy as np

# shape: 2 dòng, 3 cột
arr = np.zeros((2,3))
print(arr)
```

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

ones() function




```python
# aivietnam.ai
# Tạo một numpy array với
# tất cả phần tử là 1

import numpy as np

# numpy.ones(shape)
# shape: 2 dòng, 3 cột
arr = np.ones((2,3))
print(arr)
```

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

full() function


```python
# aivietnam.ai
# Tạo một numpy array với tất
# cả phần tử là hằng số fill_value

import numpy as np

# numpy.full(shape, fill_value)
# shape: 2 dòng, 3 cột
arr = np.full((2,3), 9)
print(arr)
```

```
[[9 9 9]
 [9 9 9]]
```

Some Common Functions

❖ Create Numpy arrays

arange(start, end, step)

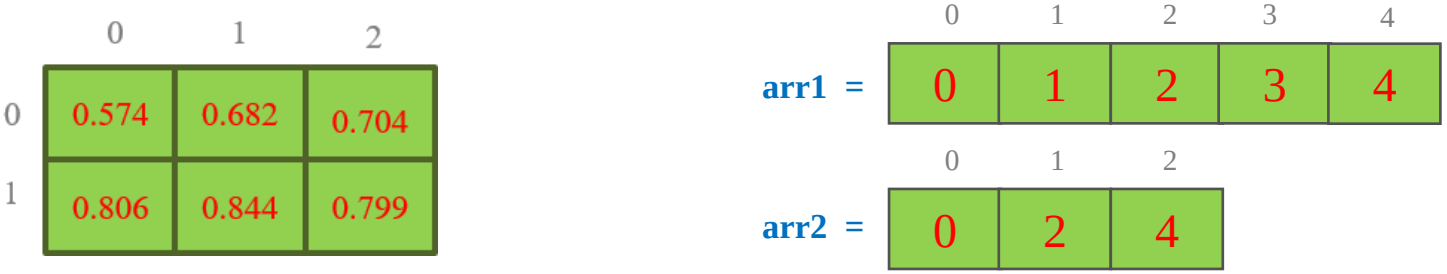
eye() function

	0	1	2
0	1	0	0
1	0	1	0
2	0	0	1

random() function

	0	1	2
0	0.574	0.682	0.704
1	0.806	0.844	0.799

arange() function




```python
# aivietnam.ai
# Tạo một numpy array với đường chéo là số 1
# số 0 được điền vào những ô phần tử còn lại

import numpy as np

# numpy.eye(N)
# shape: 3 dòng, 3 cột
arr = np.eye(3)
print(arr)
```

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

```python
# aivietnam.ai
# Tạo một numpy array với
# giá trị ngẫu nhiên

import numpy as np

# np.random.random(size)
# shape: 2 dòng, 3 cột; với
# phần tử có giá trị ngẫu nhiên
arr = np.random.random((2,3))
print(arr)
```

```
[[0.57488062 0.68266312 0.70438569]
 [0.80661973 0.84413356 0.79905247]]
```

```python
# aivietnam.ai
import numpy as np

# np.arange(start=0, stop, step=1)
arr1 = np.arange(5)
print(arr1)

arr2 = np.arange(0, 5, 2)
print(arr2)
```

```
[0 1 2 3 4]
[0 2 4]
```

Some Common Functions

❖ reshape function

reshape() function

data

1	2	3
4	5	6

data_rs

1	2
3	4
5	6

```python
# aivietnam.ai
import numpy as np

# tạo list
l = [[1,2,3],
     [4,5,6]]

# tạo ndarray
data = np.array(l)
print('data\n', data)
print('data shape\n', data.shape)

# reshape
data_rs = np.reshape(data, (3,2))
print('data_rs\n', data_rs)
print('data_rs shape\n', data_rs.shape)
```

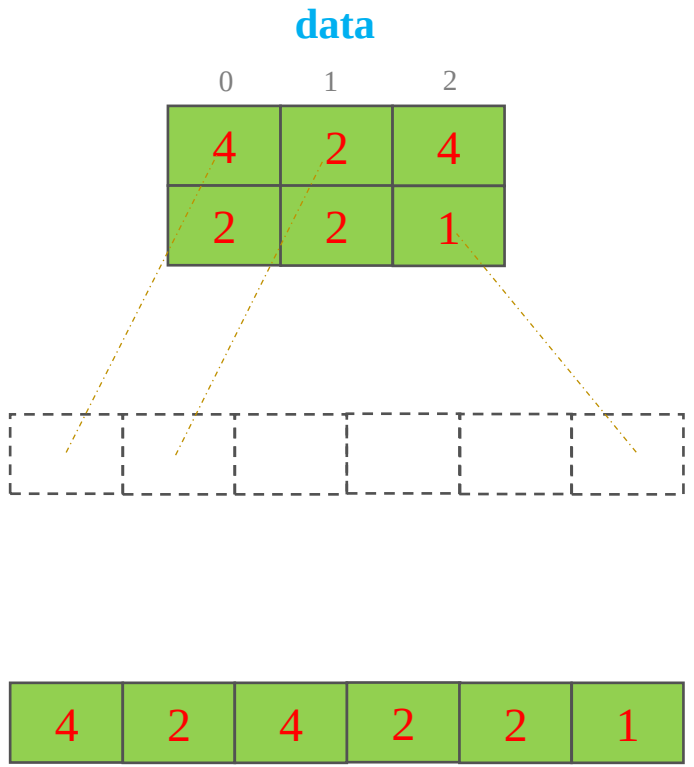
```
data
 [[1 2 3]
 [4 5 6]]
data shape
 (2, 3)
data_rs
 [[1 2]
 [3 4]
 [5 6]]
data_rs shape
 (3, 2)
```

Some Common Functions

❖ reshape and flatten functions

```python
import numpy as np

# generate random integers
data = np.random.randint(1, 5, (2, 3))
print('data: \n', data)

vector1 = data.flatten()
print('vector1: \n', vector1)

vector2 = data.reshape(-1)
print('vector2: \n', vector2)
```

```
data:
 [[4 2 4]
 [2 2 1]]
vector1:
 [4 2 4 2 2 1]
vector2:
 [4 2 4 2 2 1]
```



Some Common Functions

❖ reshape and flatten functions

```python
import numpy as np

# generate random integers
data = np.random.randint(1, 9, (2, 2))
print('data: \n', data)
```

```
data:
 [[4 3]
 [7 1]]
```

```python
# repeat
data_repeat = np.repeat(data, 2, axis=0)
print(data_repeat)
```

```
[[4 3]
 [4 3]
 [7 1]
 [7 1]]
```

```python
# repeat
data_repeat = np.repeat(data, 2, axis=1)
print(data_repeat)
```

```
[[4 4 3 3]
 [7 7 1 1]]
```

repeat(data, repeats, axis)

data

4	3
7	1

repeat_axis_0

4	3
4	3
7	1
7	1

repeat_axis_1

4	4	3	3
7	7	1	1

Outline

SECTION 1

Introduction

SECTION 2

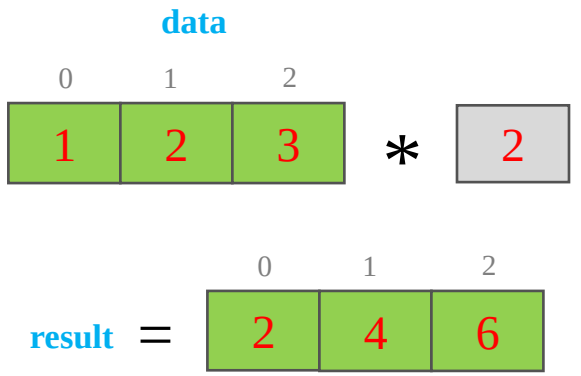
Common Functions

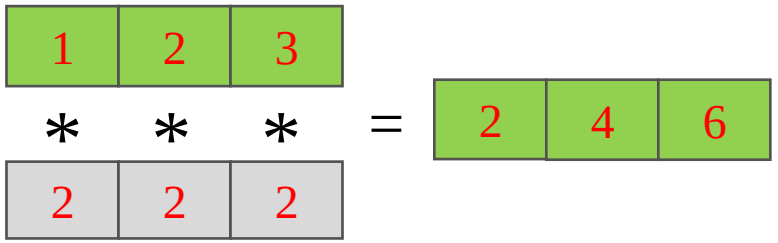
SECTION 3

Indexing and Broadcasting

SECTION 4

Examples





Array Indexing

❖ Slicing

arr[for_axis_0, for_axis_1, …]

‘**:**’: get all the elements
‘**a:b**’: get the elements from a^{th} to $(b^{th}-1)$

data

	0	1
0	1	2
1	3	4
2	5	6

data[0, 1]

	0	1
0	1	2
1	3	4
2	5	6

data[1: 3]

	0	1
0	1	2
1	3	4
2	5	6

data[0: 1 , 0]

	0	1
0	1	2
1	3	4
2	5	6

data[: , :]

	0	1
0	1	2
1	3	4
2	5	6

```python
# aivietnam.ai
import numpy as np

# Khởi tạo numpy array a_arr
a_arr = np.array([[1,2,3],
                  [5,6,7]])

# Sử dụng slicing để tạo mảng b_arr
# bằng cách lấy tất cả các dòng và cột 1,2
b_arr = a_arr[:, 1:3]

print(a_arr)
print(b_arr)
```

```
[[1 2 3]
 [5 6 7]]
[[2 3]
 [6 7]]
```

a_arr =

	0	1	2
0	1	2	3
1	5	6	7

b_arr = a_arr[:, 1:3]

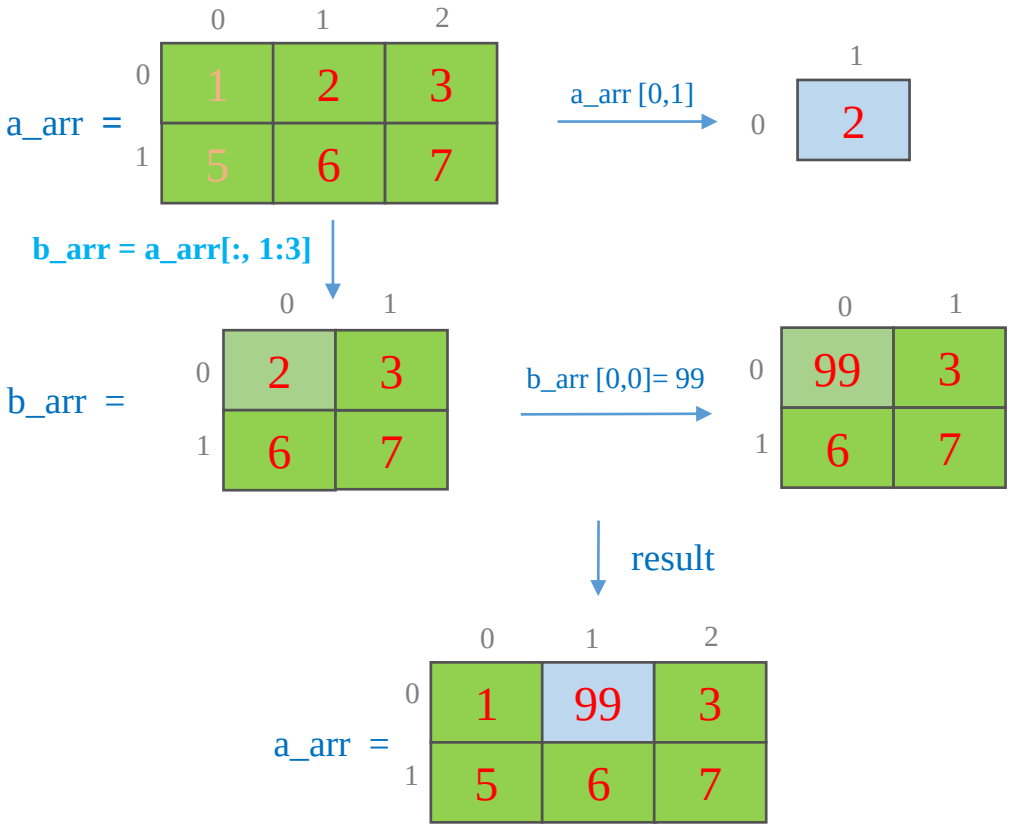
b_arr =

	0	1
0	2	3
1	6	7

Array Indexing

❖ Slicing

❖ Mutable

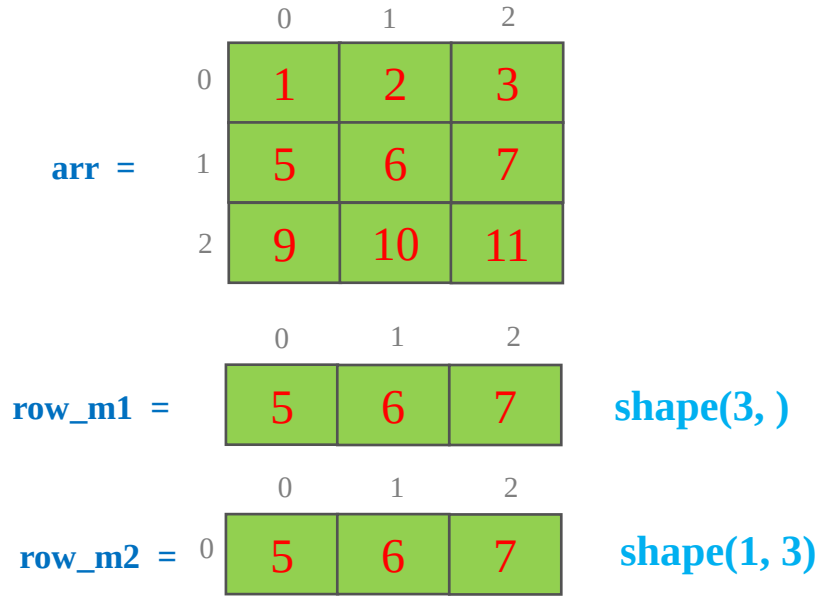



```python
# aivietnam.ai
import numpy as np

# Khởi tạo numpy array a_arr
a_arr = np.array([[1,2,3],
                  [5,6,7]])
print('a_arr \n', a_arr)

# Sử dụng slicing để tạo mảng b_arr
b_arr = a_arr[:, 1:3]
print('b_arr \n', b_arr)

print('before changing \n', a_arr[0, 1])
b_arr[0, 0] = 99
print('after changing \n', a_arr[0, 1])
```

```
a_arr
 [[1 2 3]
 [5 6 7]]
b_arr
 [[2 3]
 [6 7]]
before changing
 2
after changing
 99
```

Array Indexing

❖ Get a row

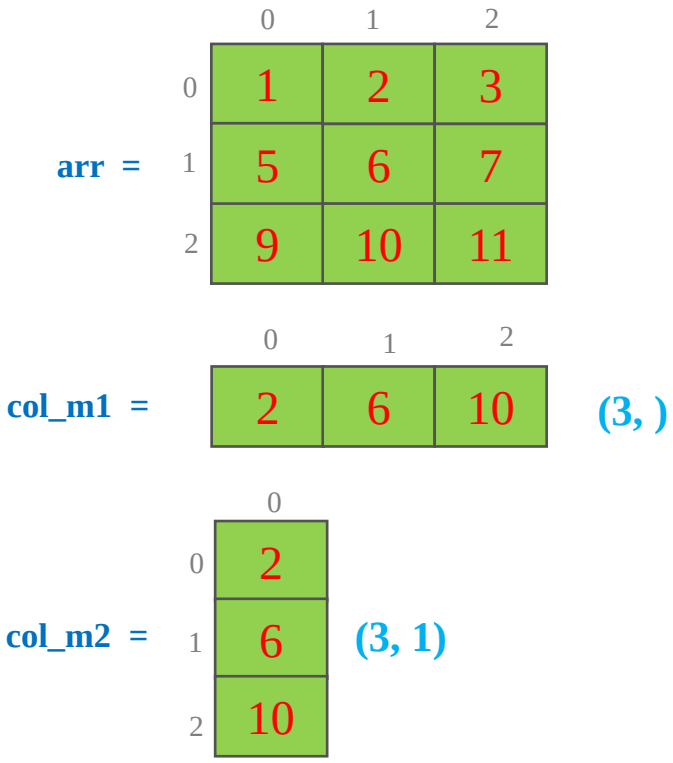



```python
# aivietnam.ai
import numpy as np

# Tạo một numpy array có shape (3, 3) với giá trị
# [[ 1  2  3]
#  [ 5  6  7]
#  [ 9 10 11]]
arr = np.array([[1, 2, 3],
                [5, 6, 7],
                [9,10,11]])

# Hai cách truy cập dữ liệu ở dòng index=1
# cách 1: số chiều giảm
row_m1 = arr[1, :]

# cách 2: số chiều được giữa nguyên
row_m2 = arr[1:2, :]

print(row_m1, row_m1.shape)
print(row_m1, row_m2.shape)
```

```
[5 6 7] (3,)
[5 6 7] (1, 3)
```

Array Indexing

❖ Get a column

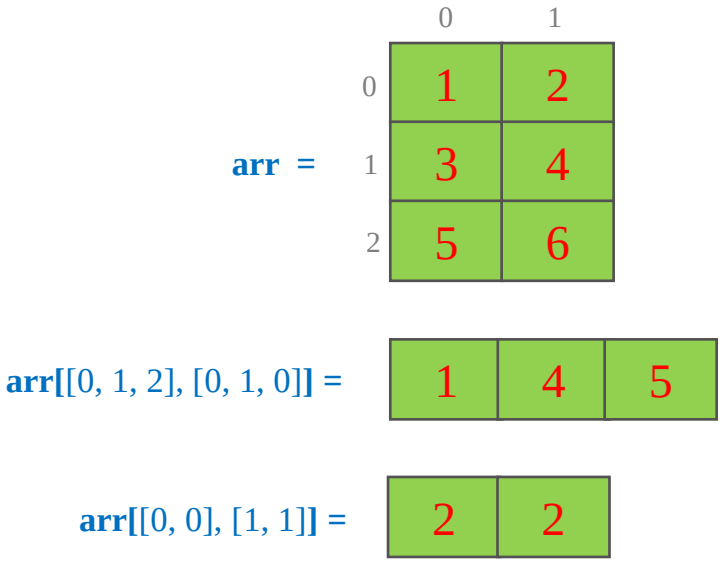



```python
# aivietnam.ai
import numpy as np

# Tạo một numpy array có shape (3, 3) với giá trị
arr = np.array([[1,2,3],
                [5,6,7],
                [9,10,11]])

# Hai cách truy cập dữ liệu ở cột index=1 của mảng
# cách 1: số chiều giảm
col_m1 = arr[:, 1]

# cách 2: số chiều được giữa nguyên
col_m2 = arr[:, 1:2]

print(col_m1, col_m1.shape)
print(col_m2, col_m2.shape)
```

```
[ 2  6 10] (3,)
[[ 2]
 [ 6]
 [10]] (3, 1)
```

Array Indexing

❖ Using Lists as indices

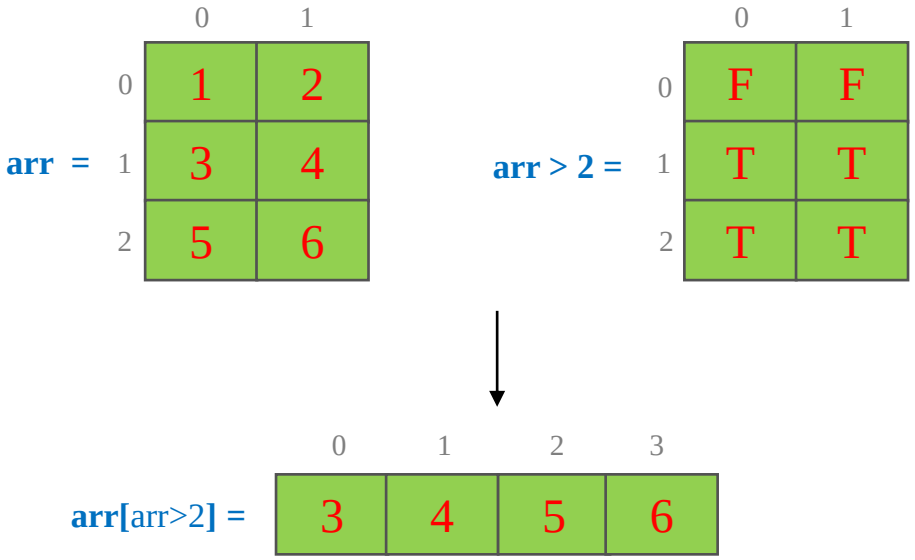



```python
# aivietnam.ai
import numpy as np

# tạo arr
arr = np.array([[1,2],
                [3, 4],
                [5, 6]])

# lấy giá trị vị trí (0,0), (1,1) và (2,0)
out1 = arr[[0, 1, 2], [0, 1, 0]]
print('out1:\n', out1)

# Có thể truy xuất tới 1 phần tử nhiều hơn 1 lần
out2 = arr[[0, 0], [1, 1]]
print('out2:\n', out2)
```

```
out1:
 [1 4 5]
out2:
 [2 2]
```

Array Indexing

❖ Boolean indices

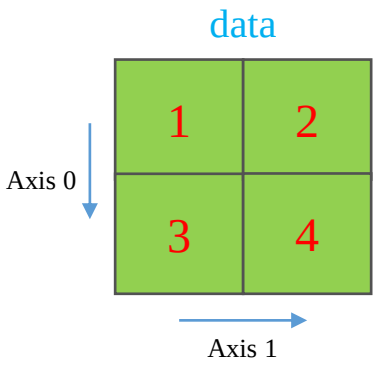



```python
# aivietnam.ai
import numpy as np

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

# tìm các phần tử lớn hơn 2
bool_idx = (arr > 2)

# get satisfying elements
result = arr[bool_idx]
print(result)
```

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

Numpy Array Operations

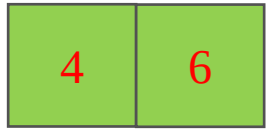
❖ Summation



sum(data)

10

sum(data, axis=0)



sum(data, axis=1)


```python
# aivietnam.ai
import numpy as np

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

# Tổng các phần tử của mảng
print(np.sum(data))

# Tính tổng theo từng cột
print(np.sum(data, axis=0))

# Tính tổng theo từng dòng
print(np.sum(data, axis=1))
```

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

Numpy Array Operations

❖ Max and min

data

1
2
3

.max( ) = 3

data

1
2
3

.min( ) = 1

```
1  # aivietnam.ai
2  import numpy as np
3
4  data = np.array([1, 2, 3])
5
6  print(data.max())
7  print(data.min())
```

```
3
1
```

1	2
3	4
5	6

.max(axis=0) = | 5 | 6 |

1	2
3	4
5	6

.min(axis=0) = | 1 | 2 |

1	2
3	4
5	6

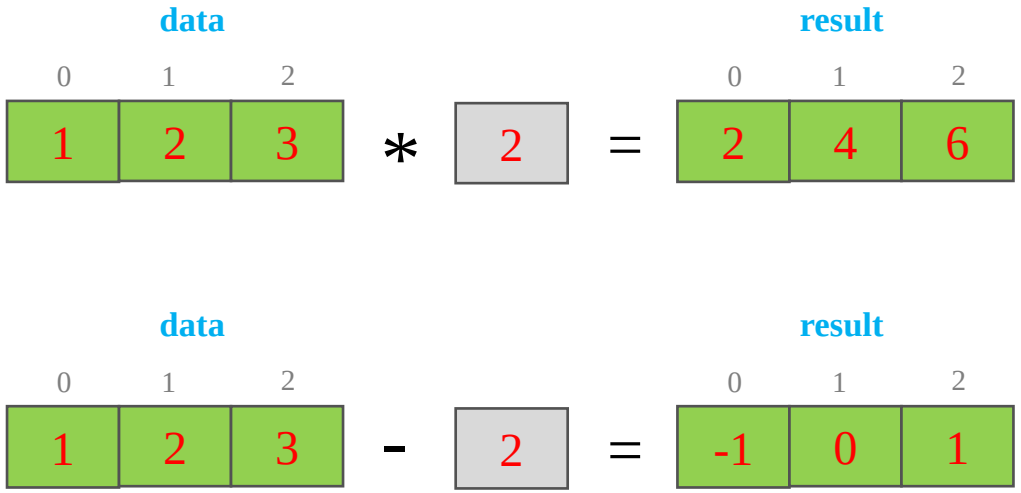
.max(axis=1) = | 2 | 4 | 6 |

1	2
3	4
5	6

.min(axis=1) = | 1 | 3 | 5 |

Broadcasting

❖ Vector and a scalar

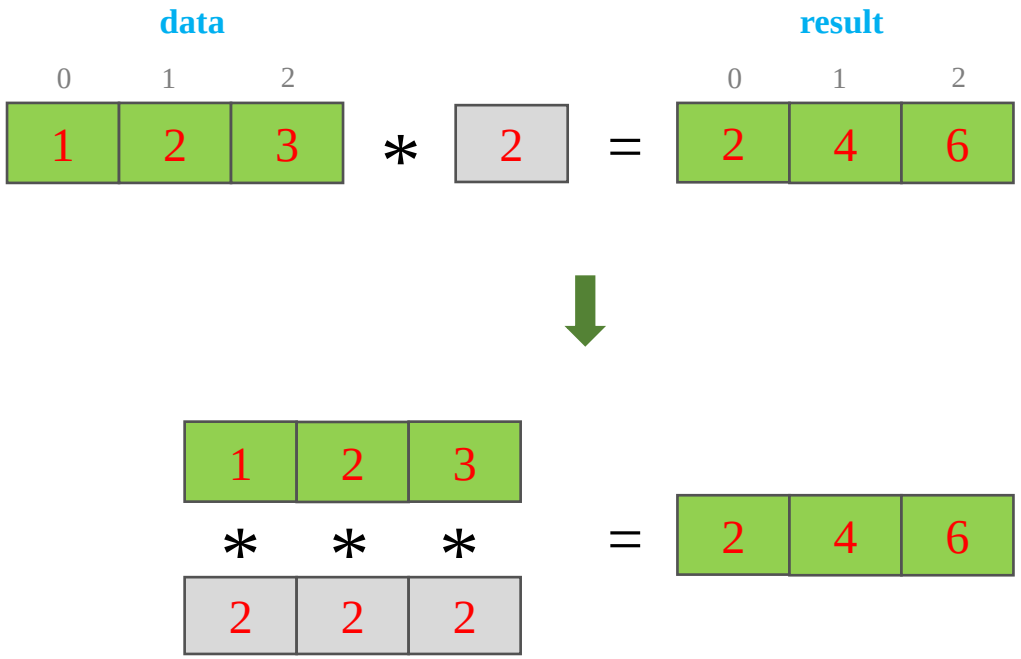



```python
# aivietnam.ai
import numpy as np

# create data
data = np.array([1,2,3])
factor = 2

# broadcasting
result_multiplication = data*factor
result_minus = data - factor

print(data)
print(result_multiplication)
print(result_minus)
```

```
[1 2 3]
[2 4 6]
[-1  0  1]
```

Broadcasting

❖ Vector and a scalar

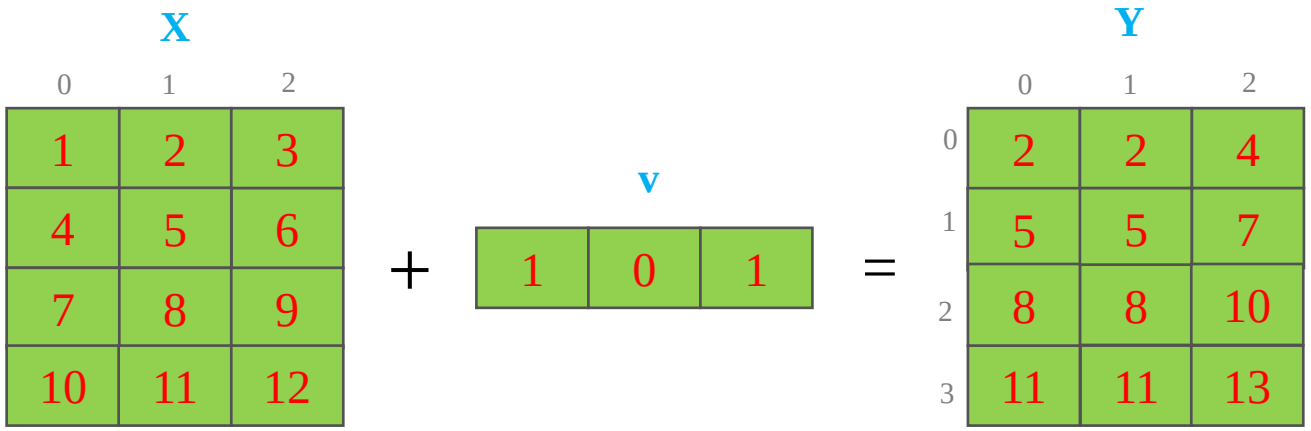



```python
# aivietnam.ai
import numpy as np

# create data
data = np.array([1,2,3])
factor = 2

# broadcasting
result_multiplication = data*factor
result_minus = data - factor

print(data)
print(result_multiplication)
print(result_minus)
```

```
[1 2 3]
[2 4 6]
[-1  0  1]
```

Broadcasting

❖ Matrix and vector



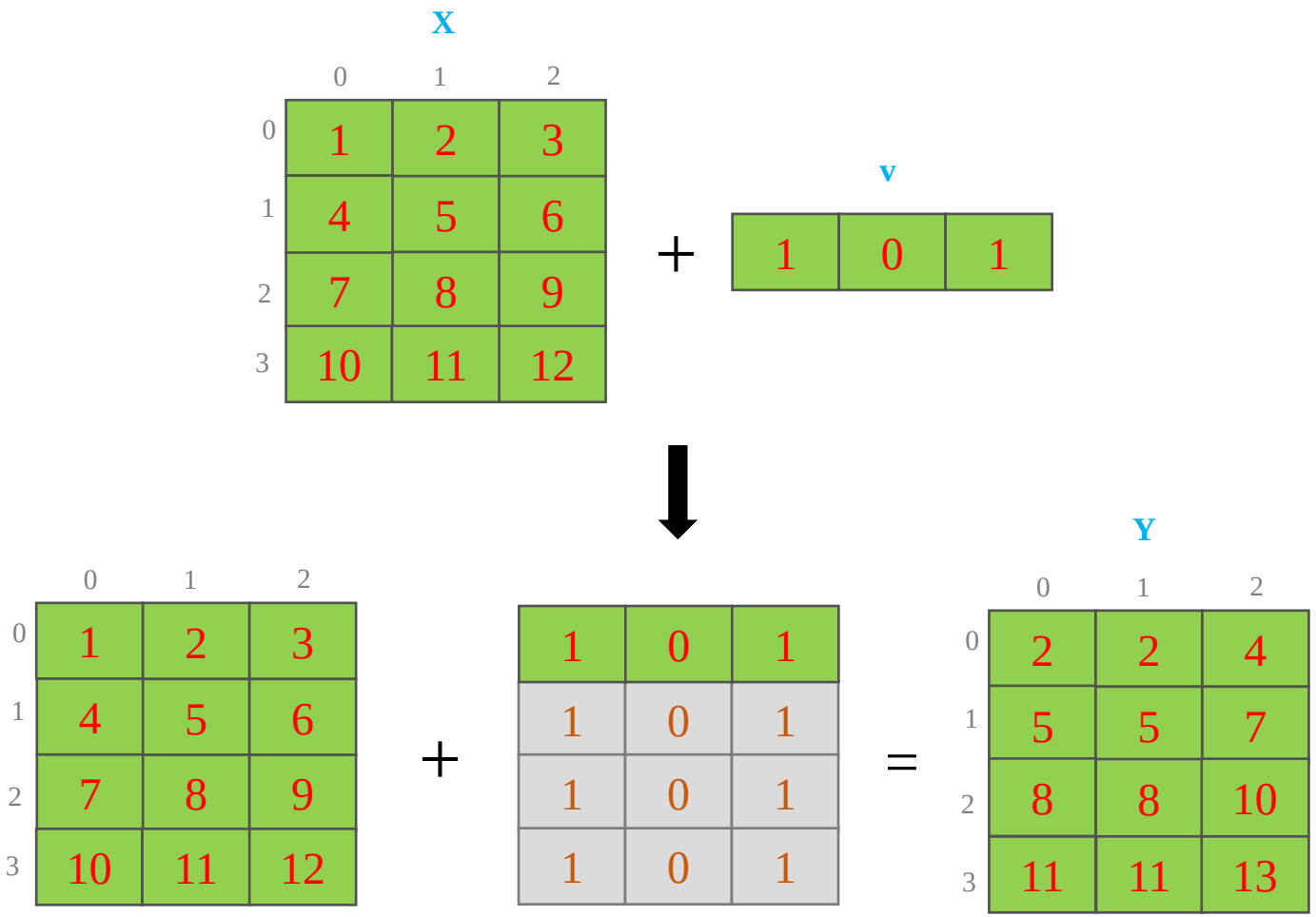

```
0
1
2
3
```

```
# aivietnam.ai
import numpy as np

X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9],
              [10, 11, 12]])
v = np.array([1, 0, 1])

Y = X + v
print(Y)
```

```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

Broadcasting

❖ Matrix and vector




```python
# aivietnam.ai
import numpy as np

X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9],
              [10, 11, 12]])
v = np.array([1, 0, 1])

Y = X + v
print(Y)
```

```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

Example 1: Softmax function

❖ Chuyển các giá trị của một vector thành các giá trị xác suất

(Stable) Formula

$$m = \max(\boldsymbol{x})$$

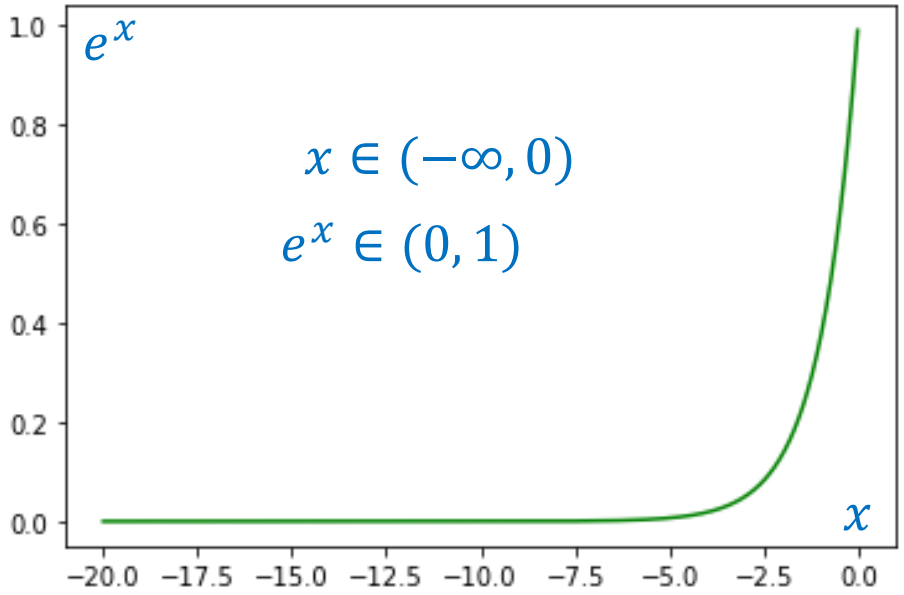
$$f(x_i) = \frac{e^{(x_i - m)}}{\sum_j e^{(x_j - m)}}$$

X	X-m	Probability
$x_1 = 1.0$	$x_1 = -2.0$	$f(x_1) = 0.09$
$x_2 = 2.0$ →	$x_2 = -1.0$ →	Softmax → $f(x_2) = 0.24$
$x_3 = 3.0$	$x_3 = 0$	$f(x_3) = 0.67$

```python
import numpy as np

def stable_softmax(X):
    exps = np.exp(X-np.max(X))
    return exps / np.sum(exps)
```

```python
X = np.array([1.0, 2.0, 3.0])
f = stable_softmax(X)
print(f)
```

```
[0.09003057 0.24472847 0.66524096]
```




QUIZ TIME

Outline

SECTION 1

Introduction

SECTION 2

Common Functions

SECTION 3

Indexing and Broadcasting

SECTION 4

Examples

Feature	Label
Petal_Length	Category
1.4	0
1	0
1.5	0
3	1
3.8	1
4.1	1

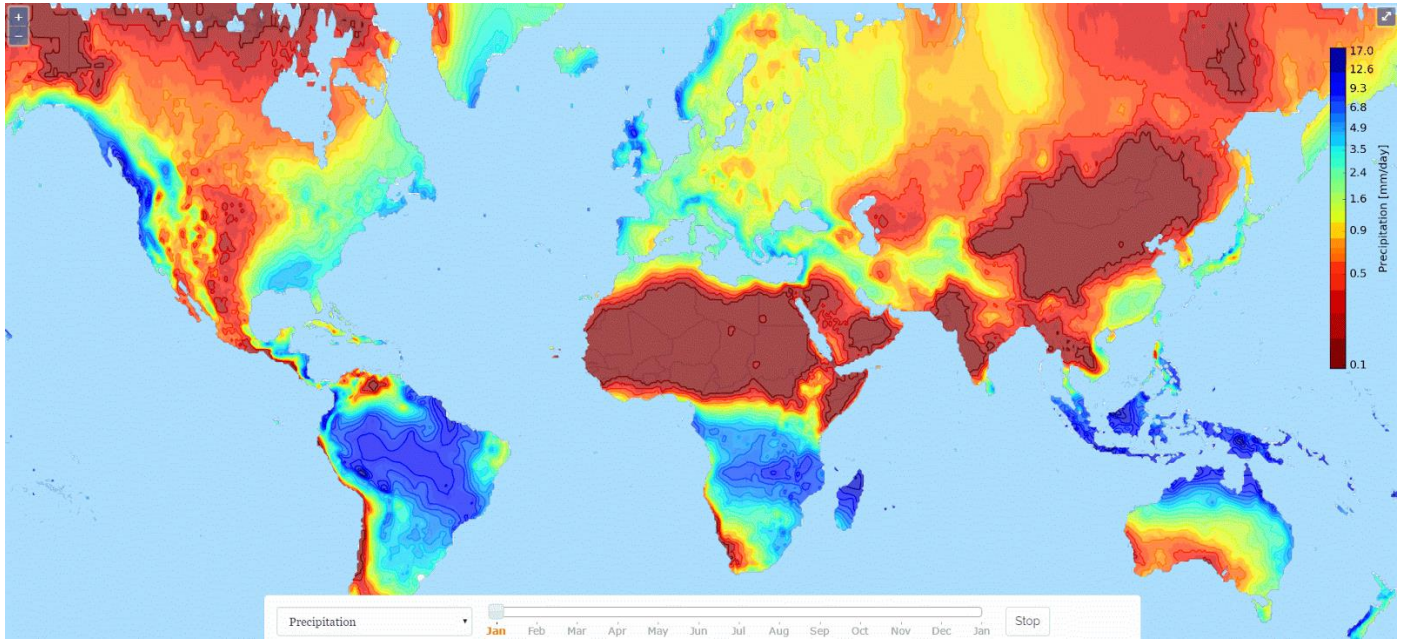
One-hot encoding for label

$$y = 0 \rightarrow \boldsymbol{y} = [1 \quad 0]$$
$$y = 1 \rightarrow \boldsymbol{y} = [0 \quad 1]$$

scalar vector

Example 2

❖ Weather forecasting



Predict future temperature in weather forecasting

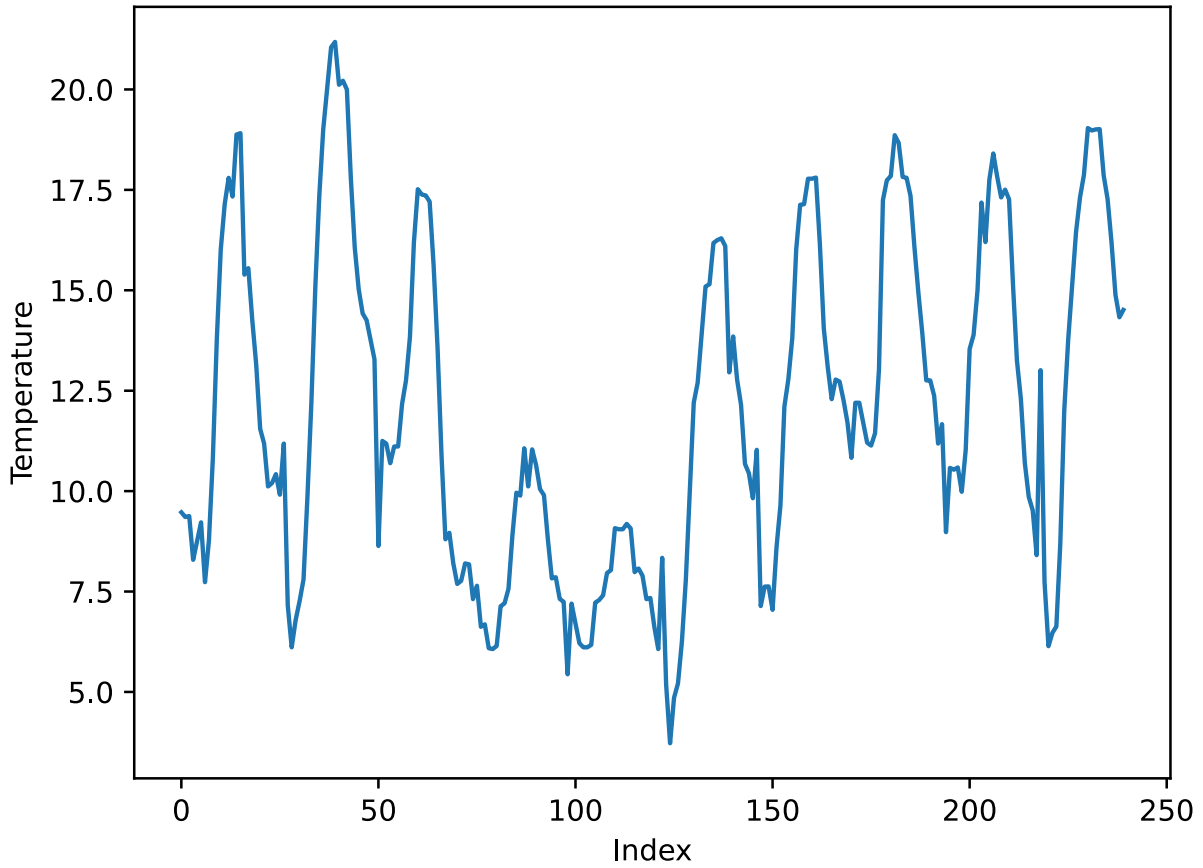
Example 2

❖ Weather forecasting

Date	Temperature (C)
2006-04-01 00	9.472222222
2006-04-01 01	9.355555556
2006-04-01 02	9.377777778
2006-04-01 03	8.288888889
2006-04-01 04	8.755555556
2006-04-01 05	9.222222222
2006-04-01 06	7.733333333
2006-04-01 07	8.772222222
2006-04-01 08	10.82222222
2006-04-01 09	13.77222222
2006-04-01 10	16.01666667
2006-04-01 11	17.14444444
2006-04-01 12	17.8
2006-04-01 13	17.33333333
2006-04-01 14	18.87777778
2006-04-01 15	18.91111111
2006-04-01 16	15.38888889
2006-04-01 17	15.55
2006-04-01 18	14.25555556
2006-04-01 19	13.14444444
2006-04-01 20	11.55
2006-04-01 21	11.18333333
2006-04-01 22	10.11666667
2006-04-01 23	10.2

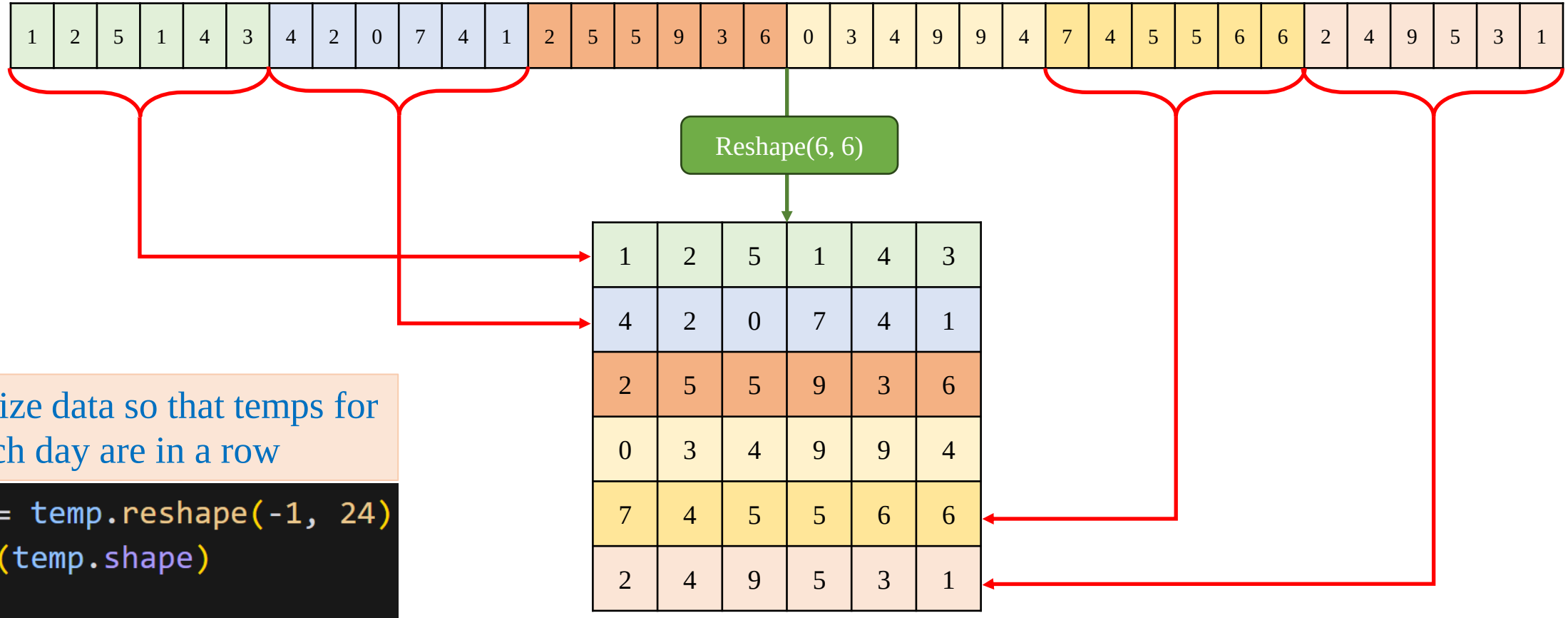
No.	Temperature (C)
0	9.472222222
1	9.355555556
2	9.377777778
3	8.288888889
4	8.755555556
5	9.222222222
6	7.733333333
7	8.772222222
8	10.82222222
9	13.77222222
10	16.01666667
11	17.14444444
12	17.8
13	17.33333333
14	18.87777778
15	18.91111111
16	15.38888889
17	15.55
18	14.25555556
19	13.14444444
20	11.55
21	11.18333333
22	10.11666667
23	10.2

```python
import pandas as pd
import numpy as np

data = pd.read_csv('temperature-1d.csv').to_numpy()
temp = data[:, 1]
```



Example 2

❖ Daily average temperatures



Re-organize data so that temps for each day are in a row

```
temp = temp.reshape(-1, 24)
print(temp.shape)
```

✓ 0.0s

```
(10, 24)
```

Reshape an array

Example 2

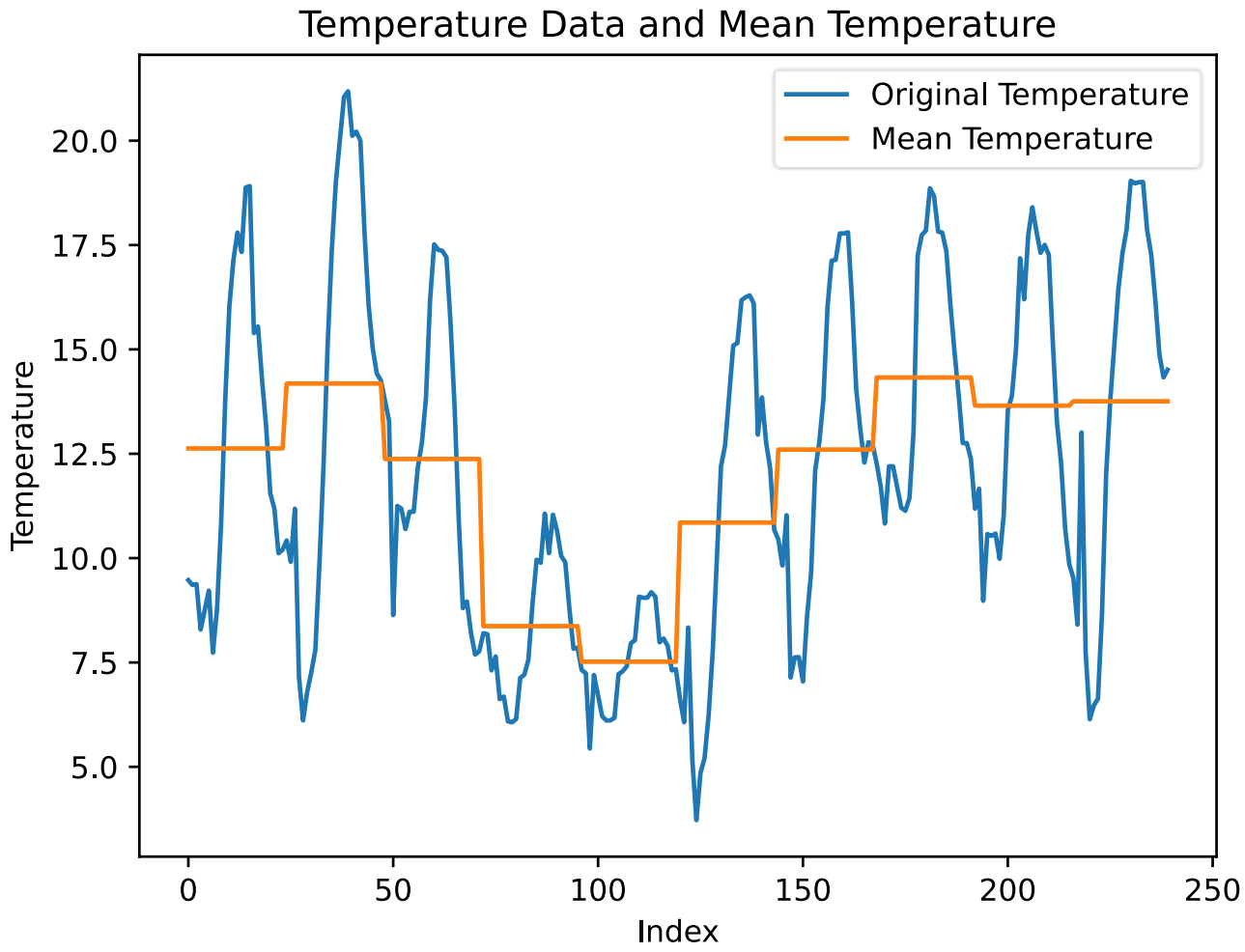
❖ Daily average temperatures

```python
import pandas as pd
import numpy as np

data = pd.read_csv('temperature-1d.csv').to_numpy()
temp = data[:, 1]
```

```python
# Compute mean for each 24 data points
mean_24 = np.mean(temp.reshape(-1, 24), axis=1)
print(result.shape)

mean_24_repeat = np.repeat(mean_24, 24)
print(mean_24_repeat.shape)
```

✓ 0.0s

```
(240,)
(240,)
```



Example 3

❖ One-hot Encoding

Feature	Label
Petal_Length	Category
1.4	0
1	0
1.5	0
3	1
3.8	1
4.1	1

One-hot encoding for label

$$y = 0 \rightarrow \boldsymbol{y} = [1 \quad 0]$$
$$y = 1 \rightarrow \boldsymbol{y} = [0 \quad 1]$$

scalar vector

Feature		Label
Petal_Length	Petal_Width	Label
1.5	0.2	1
1.4	0.2	1
1.6	0.2	1
4.7	1.6	2
3.3	1.1	2
4.6	1.3	2
5.6	2.2	3
5.1	1.5	3
5.6	1.4	3

One-hot encoding for label

$$y = 0 \rightarrow \boldsymbol{y} = [1 \quad 0 \quad 0]$$
$$y = 1 \rightarrow \boldsymbol{y} = [0 \quad 1 \quad 0]$$
$$y = 2 \rightarrow \boldsymbol{y} = [0 \quad 0 \quad 1]$$

One-hot Encoding

Feature		Label
Petal_Length	Petal_Width	Label
1.4	0.2	0
1.4	0.2	0
1.3	0.2	0
4.5	1.5	1
4.9	1.5	1
4	1.3	1
4.5	1.7	2
6.3	1.8	2
5.8	1.8	2

#classes=3

#samples=9

One-hot encoding for label

$$y = 0 \rightarrow \boldsymbol{y} = [1 \quad 0 \quad 0]$$
$$y = 1 \rightarrow \boldsymbol{y} = [0 \quad 1 \quad 0]$$
$$y = 2 \rightarrow \boldsymbol{y} = [0 \quad 0 \quad 1]$$

```python
import numpy as np

#num_labels
K = 3

# num_samples
N = 9

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

```
[0 0 0 1 1 1 2 2 2]
```

```python
# create an NxK array
labels_onehot = np.zeros((N, K))

# one-hot encoding
labels_onehot[np.arange(N), labels] = 1
print(labels_onehot)
```

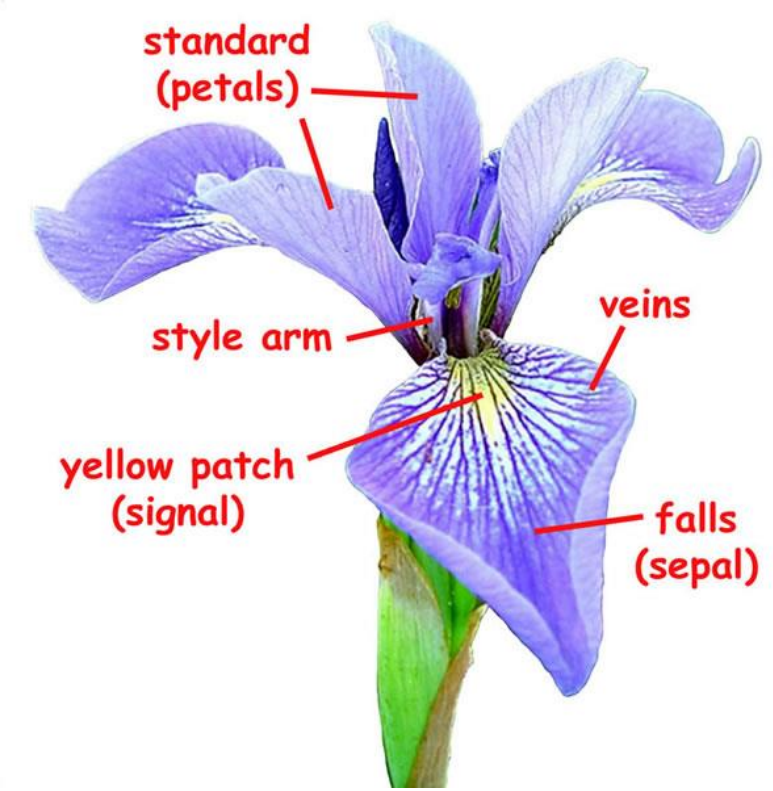
```
[[1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]]
```

Example 4

❖ Text data

❖ IRIS data

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa



Example 4

❖ Some functions

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa

```
np.genfromtxt(fname='…',
    dtype=float, delimiter=',',
    skip_header=1,
    usecols=[0, 1, 2, 3])
```

```
np.genfromtxt(fname='…',
    dtype=float, delimiter=',',
    skip_header=1,
    usecols=4)
```

Example 4

❖ Some functions

Find the unique elements

```
np.unique(data)
```

1	2	3	2	1

data

np.unique(data) =

1	2	3

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

result = np.unique(data)
print(result)
```

✓ 0.0s

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

'class-1'	'class-2'	'class-3'	'class-2'

data

result =

'class-1'	'class-2'	'class-3'

```
data = np.array(['class-1', 'class-2',
                 'class-3', 'class-2'])
print(data)

result = np.unique(data)
print(result)
```

✓ 0.0s

```
['class-1' 'class-2' 'class-3' 'class-2']
['class-1' 'class-2' 'class-3']
```

Example 4

❖ Some functions

Cast to a specified type

```
np.ndarray.astype(dtype=float)
```

For each element in **data**, replace **old** by **new**

```
np.char.replace(data,
                old,
                new)
```

data = | ‘class-1’ | ‘class-2’ | ‘class-3’ |

data = | ‘0’ | ‘1’ | ‘2’ |

data = | 0 | 1 | 2 |

```
data = np.array(['class-1', 'class-2', 'class-3'])
data = np.char.replace(data, 'class-1', '0')
data = np.char.replace(data, 'class-2', '1')
data = np.char.replace(data, 'class-3', '2')
print(data)

data = data.astype(float)
print(data)
```

✓ 0.0s

```
['0' '1' '2']
[0. 1. 2.]
```

Example 4

❖ Text data

❖ IRIS data

col 0	col 1	col 2	col 3	col 4
sepal_length	sepal_width	petal_length	petal_width	species

```python
# aivietnam.ai
# Đọc file IRIS.csv

import numpy as np
import numpy.core.defchararray as np_f

# lấy các đặc trưng và lưu vào biến X
X = np.genfromtxt('IRIS.csv', delimiter=',',
                  dtype='float', usecols=[0,1,2,3],
                  skip_header=1)
print(X.shape)

# lấy species và lưu vào biến y
y = np.genfromtxt('IRIS.csv', delimiter=',',
                  dtype='str', usecols=4, skip_header=1)

# thay chuỗi bằng số
categories = np.unique(y)
for i in range(categories.size):
    y = np_f.replace(y, categories[i], str(i))

# đưa về kiểu float
y = y.astype('float')
print(y)
```

```
(150, 4)
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2.]
```

AI VIET NAM
@aivietnam.edu.vn

Numpy for 2D and 3D Data

Quang-Vinh Dinh
PhD in Computer Science

Year 2024

Objectives

OpenCV and Numpy

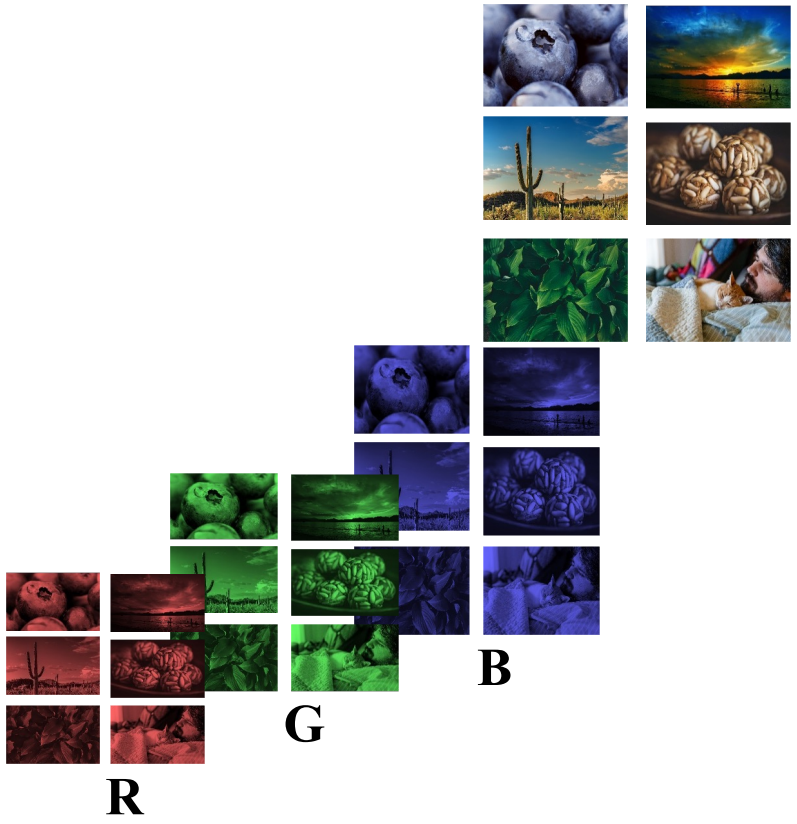
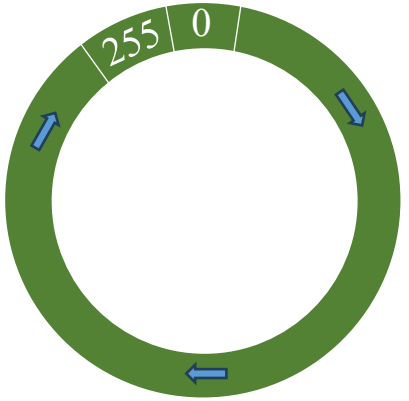


Image Loading

Other Examples



in Numpy

Outline

SECTION 1

OpenCV and NumPy

SECTION 2

Image Loading

SECTION 3

Brightness Changes

SECTION 4

Color Conversion

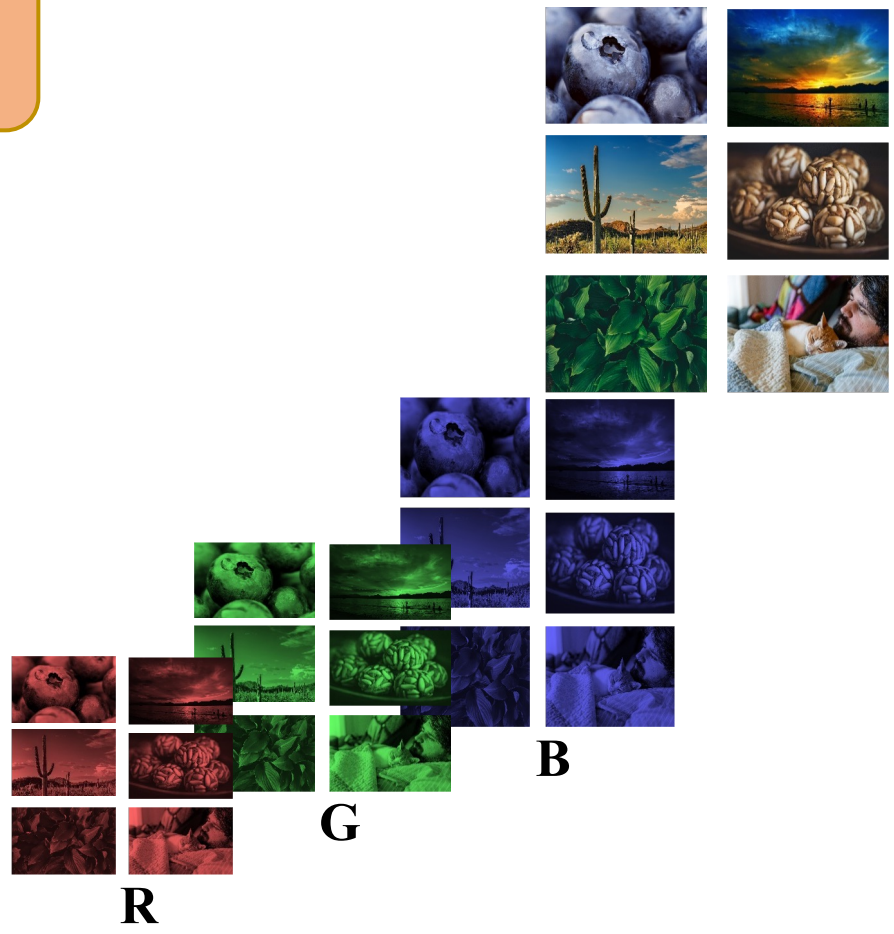
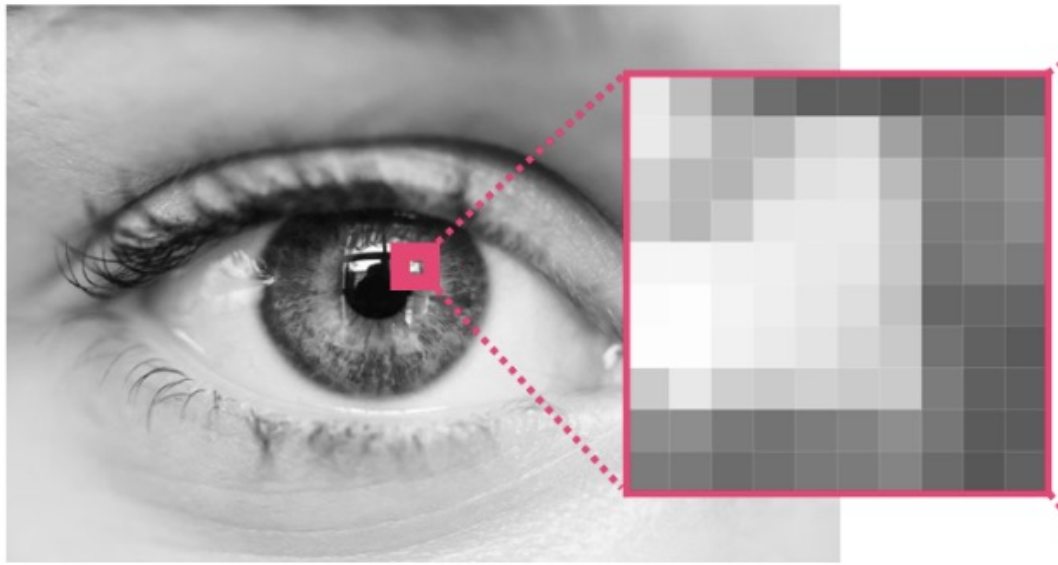


Image Data

❖ Grayscale images

Height



Width

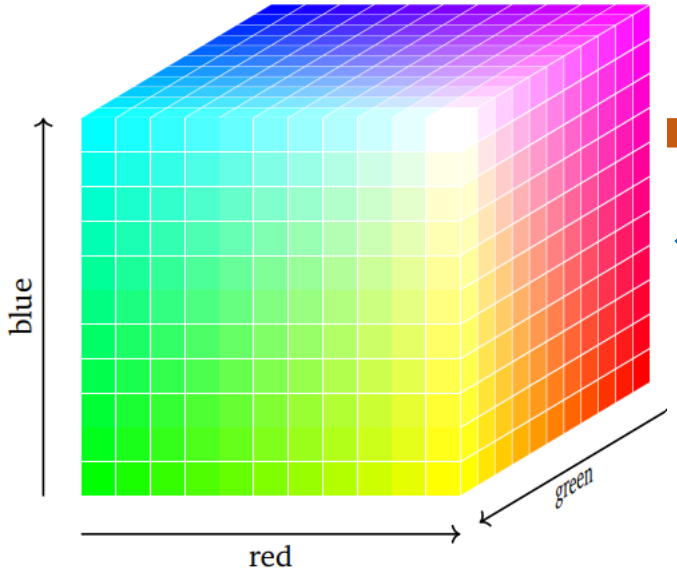
230	194	147	108	90	98	84	96	91	101
237	206	188	195	207	213	163	123	116	128
210	183	180	205	224	234	188	122	134	147
198	189	201	227	229	232	200	125	127	135
249	241	237	244	232	226	202	116	125	126
251	254	241	239	230	217	196	102	103	99
243	255	240	231	227	214	203	116	95	91
204	231	208	200	207	201	200	121	95	95
144	140	120	115	125	127	143	118	92	91
121	121	108	109	122	121	134	106	86	97

Pixel p = scalar

$0 \leq p \leq 255$

Resolution: #pixels
Resolution = HeightxWidth

Image Data

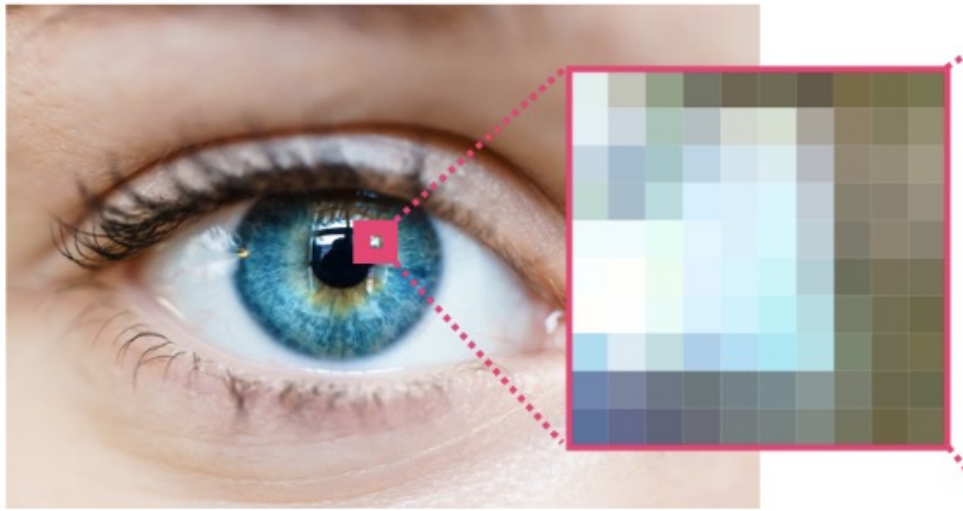


❖ Color images

RGB color image

Pixel p= $\begin{bmatrix} r \\ g \\ b \end{bmatrix}$

$0 \leq r,g,b \leq 255$



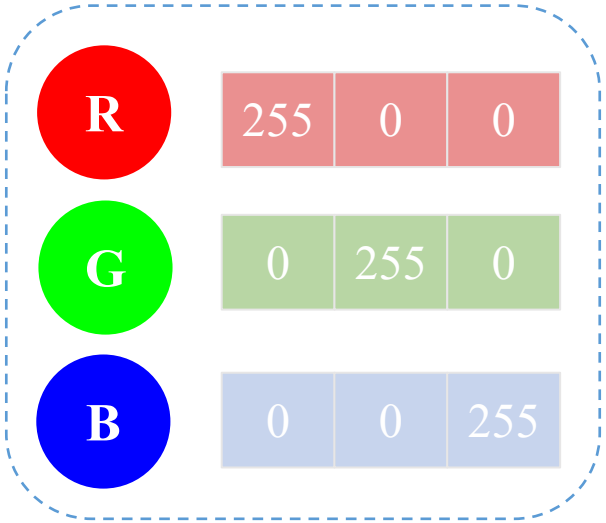
(Height, Width, channel)

	233	188	137	96	90	95	63	73	73	82
237	202	159	120	105	110	88	107	112	121	109
226	191	147	110	101	112	98	123	110	119	142	131
221	191	176	182	203	214	169	144	133	145	155	122
185	160	161	184	205	223	186	137	147	161	140	115
181	174	189	207	206	215	194	136	142	151	133	87
246	237	237	231	208	206	192	122	143	144	111	74
254	254	241	224	199	192	181	99	122	117	107	74
239	248	232	207	187	182	184	110	114	110	113	74
193	215	193	167	158	164	181	114	112	111	105	82
113	119	110	111	113	123	135	120	108	106	113	
93	97	91	103	107	111	122	112	104	114		

Resolution: #pixels
Resolution = HeightxWidth

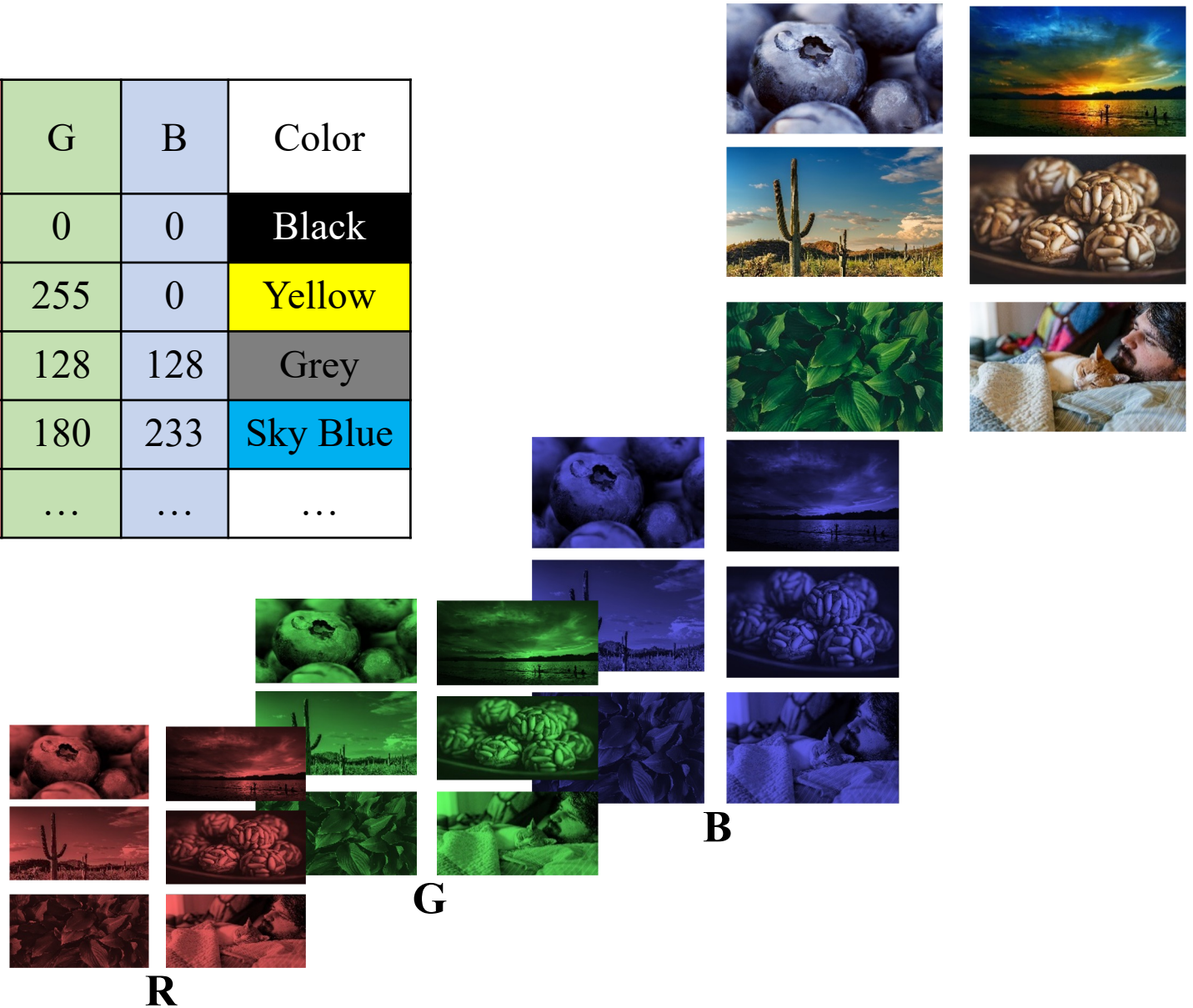
https://textbooks.math.gatech.edu/ila/ila.pdf

- RGB: Đây là cách phổ biến nhất để biểu diễn màu sắc trong hình ảnh. Mỗi hình ảnh được biểu diễn bằng ba kênh màu - đỏ (Red), xanh lục (Green), và xanh lam (Blue). Mỗi kênh này có giá trị từ 0 đến 255.



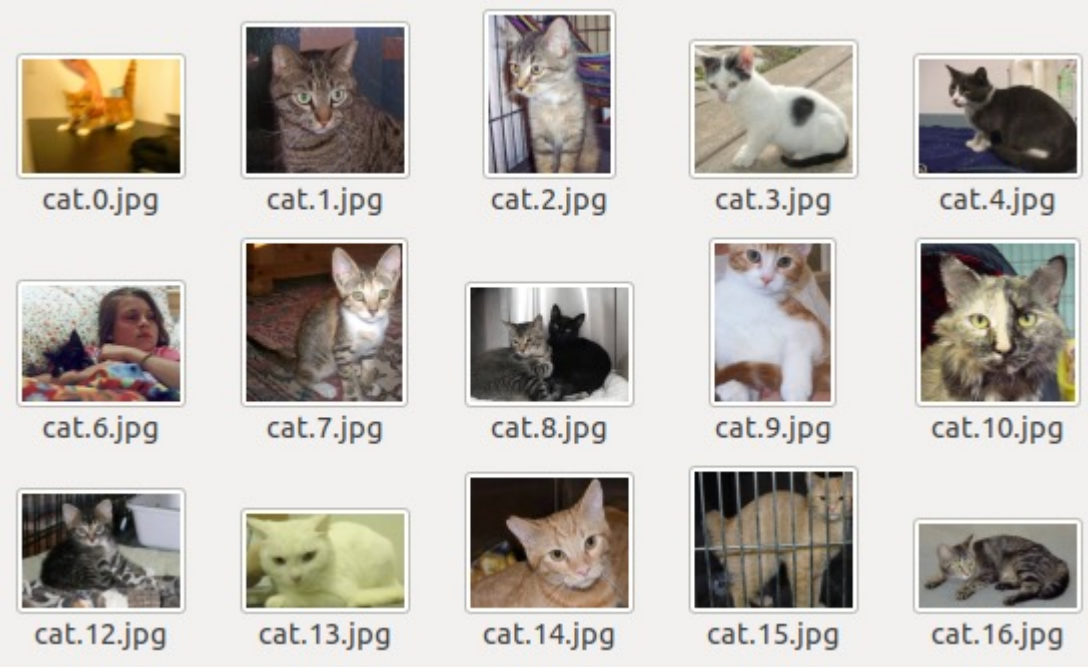
R	G	B	Color
0	0	0	Black
255	255	0	Yellow
128	128	128	Grey
86	180	233	Sky Blue
…	…	…	…

- Hình ảnh RGB thông thường



Data Processing

❖ Images in files




```
output = cv2.imread(path, mode)
```

mode=0: read images in grayscale

mode=1: read images in color

```
output = cv2.resize(input, (height, width))
```

width

height

OpenCV and Numpy

❖ Read an image

image1.png

```
import cv2

# Read the image
image = cv2.imread('image1.png', 1)
print(image.shape)
```

✓ 0.0s

```
(162, 311, 3)
```

image2.png

```
import cv2

# Read the image
image = cv2.imread('image2.png', 0)
print(image.shape)
```

✓ 0.0s

```
(162, 311)
```

OpenCV and Numpy

❖ Read an image

image2.png

```python
import cv2
import matplotlib.pyplot as plt

# Read the image
image = cv2.imread('image2.png', 0)

# Display the image
plt.imshow(image, cmap='gray')
```

✓ 0.0s

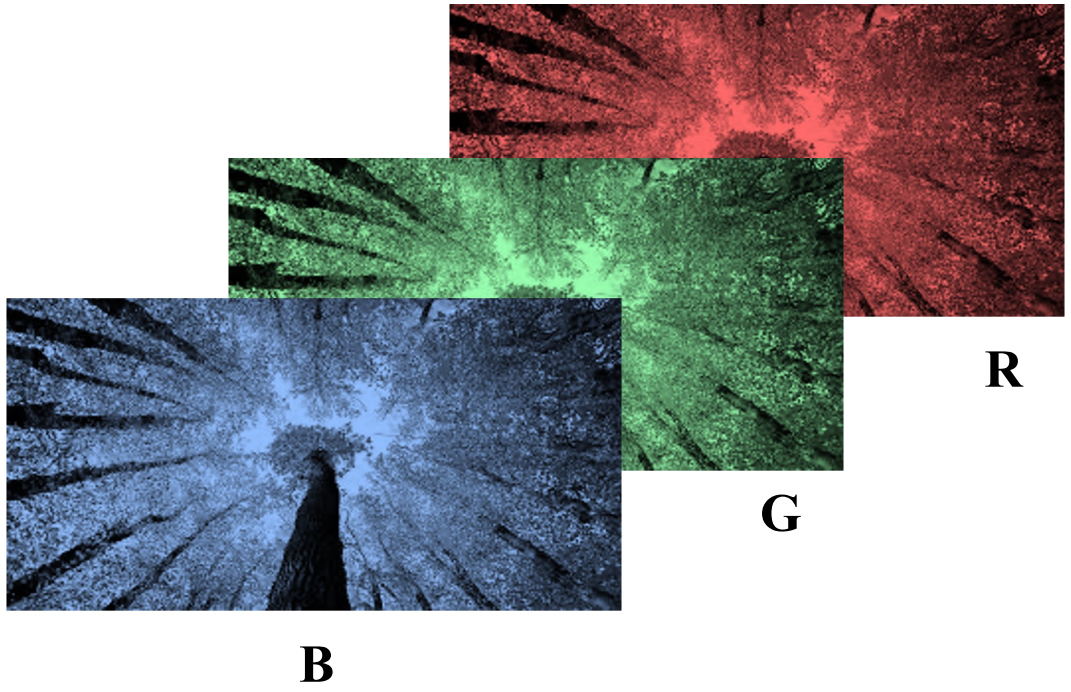
OpenCV and Numpy

❖ Read an image

```
import cv2

# Read the image
image = cv2.imread('image1.png', 1)
print(image.shape)
```

✓ 0.0s

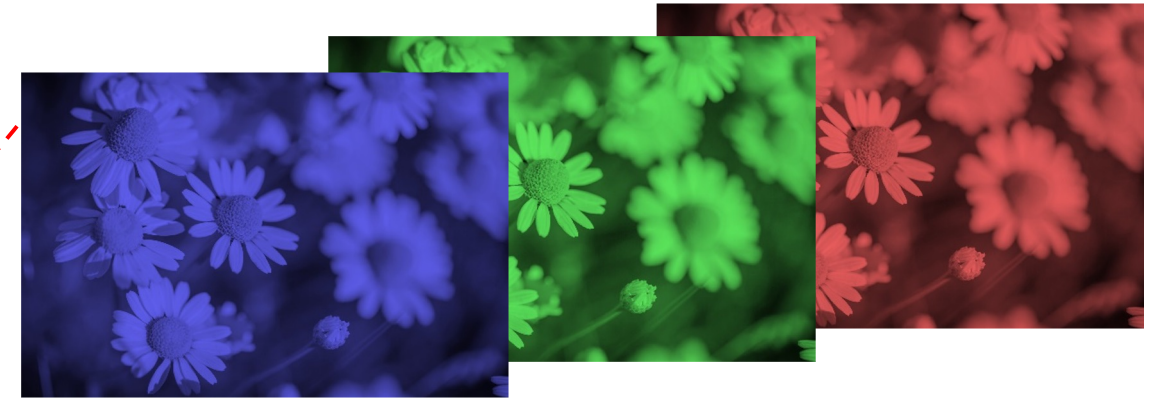
```
(162, 311, 3)
```



❑ Đọc và hiển thị ảnh với OpenCV

```python
import cv2

# Đọc hình ảnh
img = cv2.imread("flowers.jpg")
cv2.imshow("Original", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Not runnable on Colab



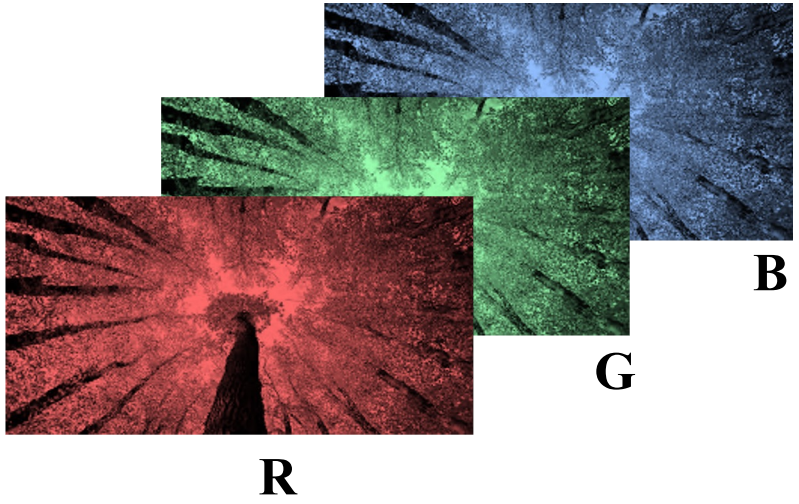
B **G** **R**

Thư viện OpenCV sử dụng kênh màu BGR khi đọc ảnh.

Đọc ảnh với OpenCV và hiển thị bằng Matplotlib

```python
import cv2
import matplotlib.pyplot as plt

# Đọc hình ảnh
img = cv2.imread("flowers.jpg")
plt.imshow(img)
```

Input

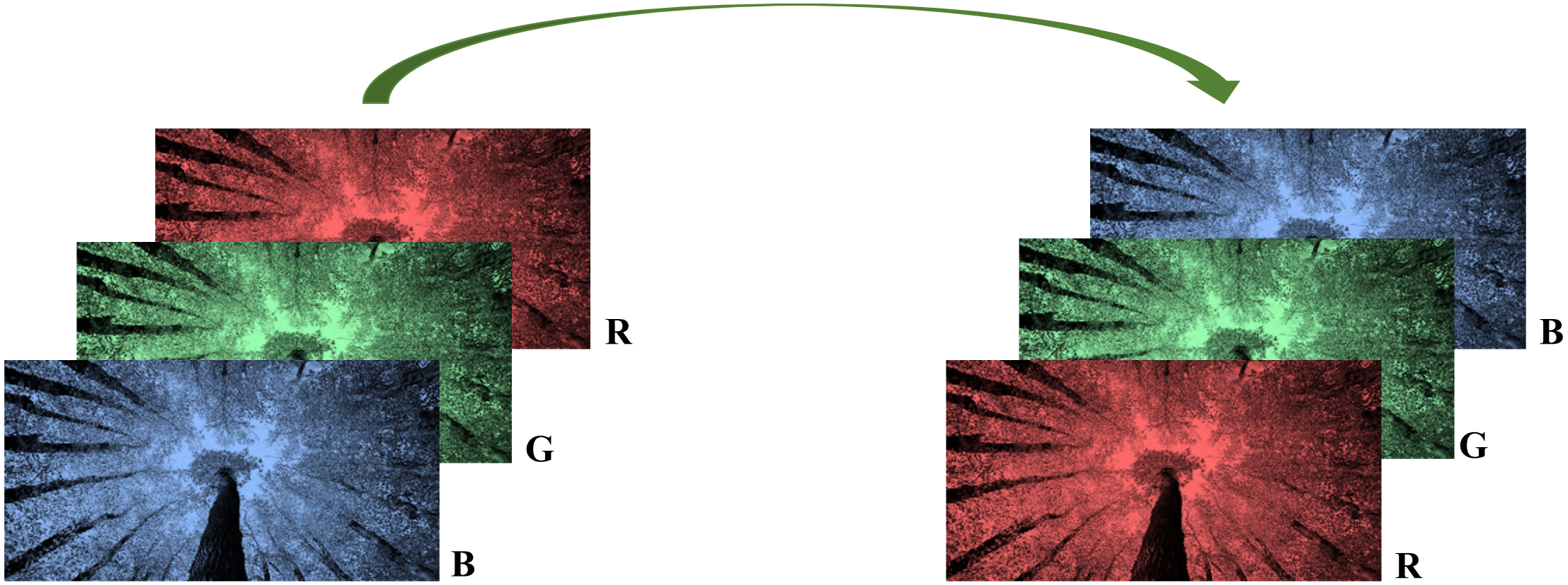


Output

Màu sắc hình ảnh sau khi hiển thị bằng thư viện Matplotlib không đúng, tại sao?

OpenCV and Numpy

❖ how?




```
image_rgb = image[:, :, [2, 1, 0]]
image_rgb = image[:, :, ::-1]
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

OpenCV and Numpy

❖ Read an image

image1.png

```python
import cv2
import matplotlib.pyplot as plt

# Read the image
image = cv2.imread('image1.png')

# Convert the image from BGR to RGB
# image_rgb = image[:, :, ::-1]
image_rgb = image[:, :, [2, 1, 0]]

# Display the image
plt.imshow(image_rgb)
```

✓ 0.0s

Đọc ảnh với OpenCV và hiển thị bằng Matplotlib

```python
import cv2
import matplotlib.pyplot as plt

# Đọc hình ảnh
img = cv2.imread("flowers.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.axis("off")
plt.show()
```

Input

Output

Thư viện matplotlib sử dụng kênh màu RGB, nên sau khi đọc hình ảnh bằng **cv2.imread()**, chúng ta cần chuyển đổi kênh màu sang dạng RGB.

Outline

SECTION 1

OpenCV and NumPy

SECTION 2

Image Loading

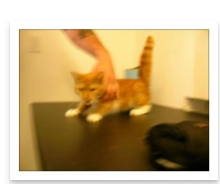
SECTION 3

Brightness Changes

SECTION 4

Color Conversion

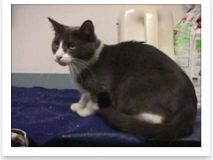
samples


cat.0.jpg


cat.1.jpg

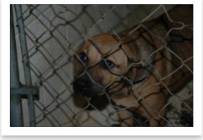

cat.2.jpg


cat.3.jpg


cat.4.jpg


cat.5.jpg

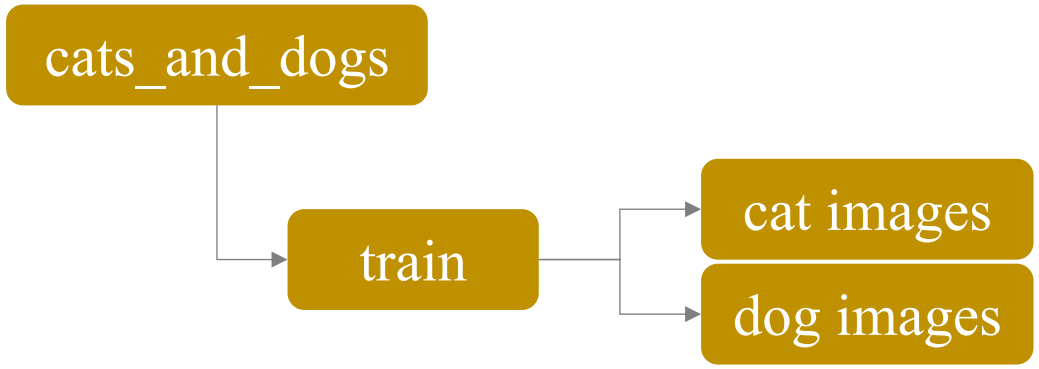

dog.7864.jpg

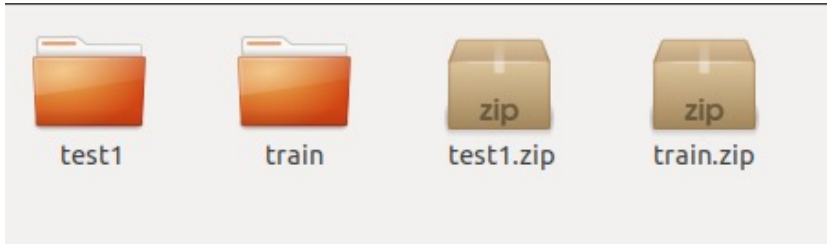

dog.7865.jpg


dog.7866.jpg

Image File Loading

❖ Cat-Dog dataset





Load all the images and save in a Numpy array

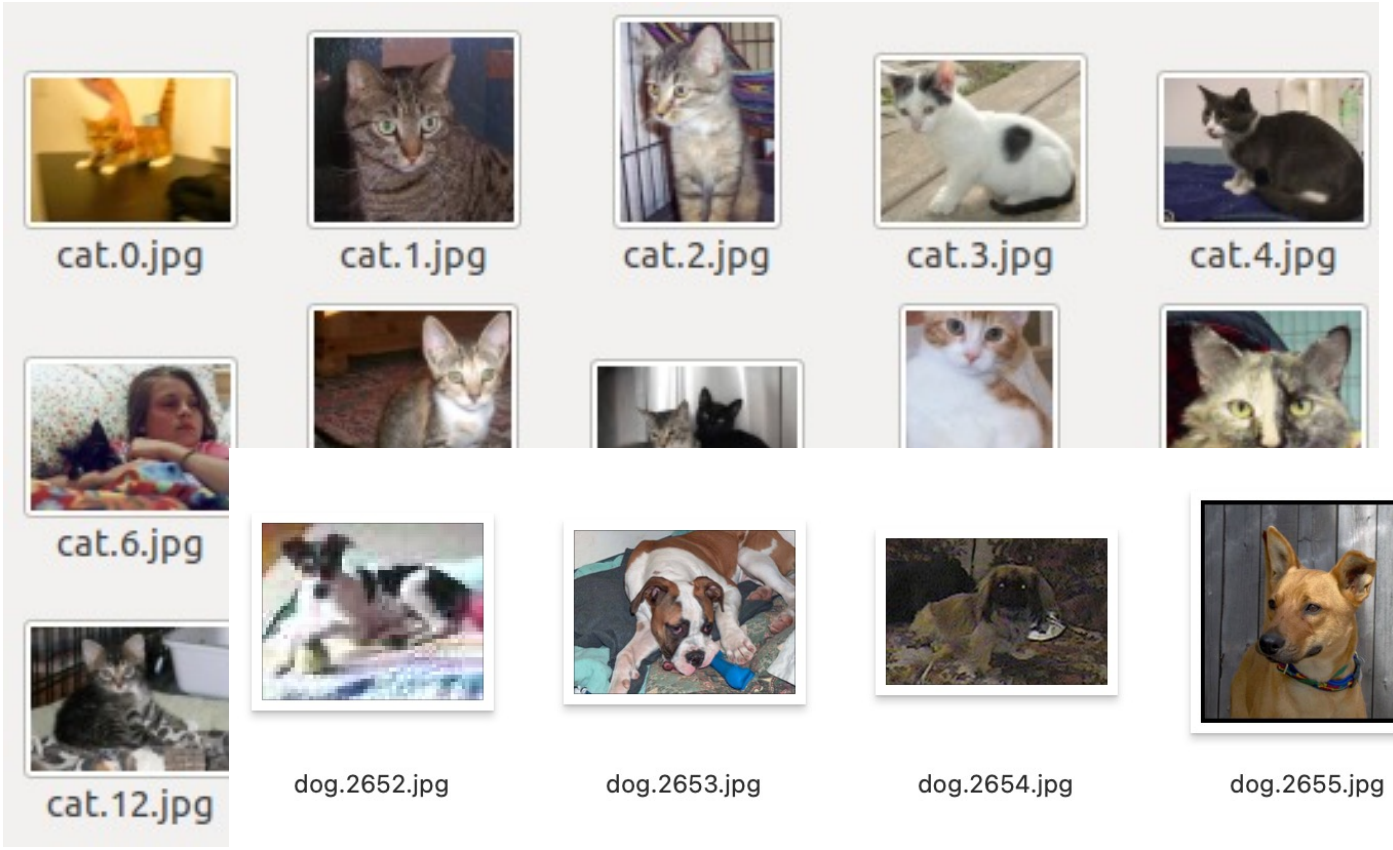


Image File Loading

❖ Get file paths

```
import glob

file_paths = glob.glob('samples/*jpg')

# print paths
for path in file_paths:
    print(path)
```

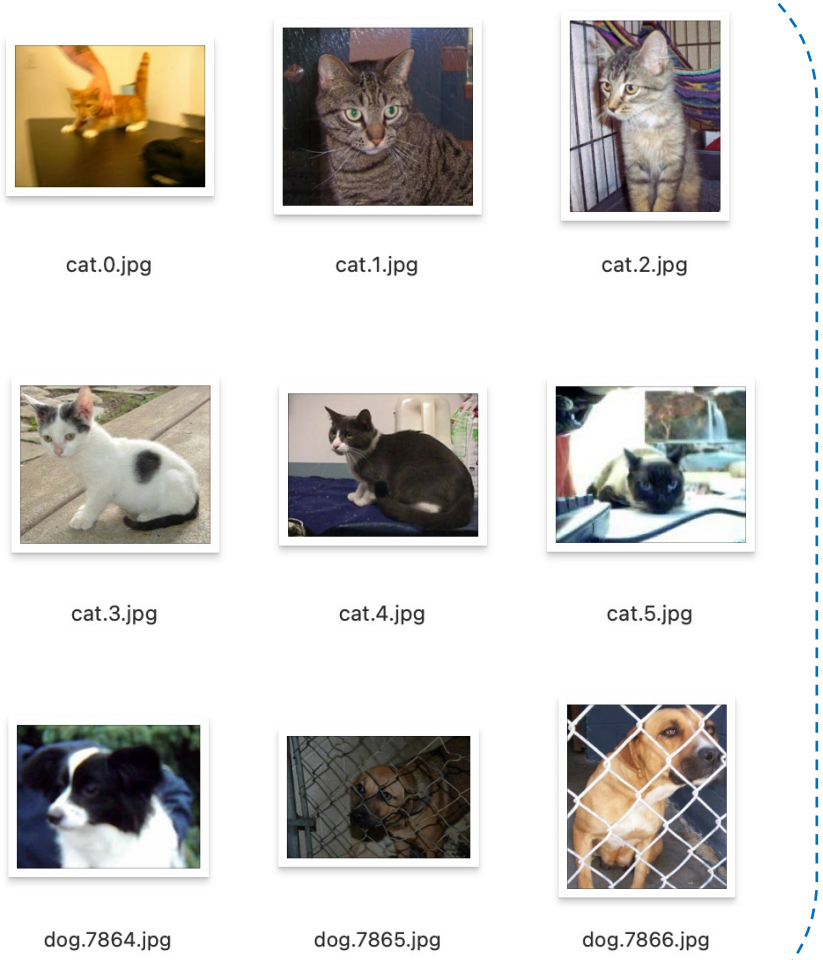
✓ 0.0s

```
samples/cat.5.jpg
samples/cat.4.jpg
samples/cat.0.jpg
samples/dog.7866.jpg
samples/cat.1.jpg
samples/cat.3.jpg
samples/dog.7864.jpg
samples/dog.7865.jpg
samples/cat.2.jpg
```

Image File Loading

❖ Load images

samples



cv2.imread(path, mode)

```
import glob
import cv2

file_paths = glob.glob('samples/*jpg')

# read and resize images
for path in file_paths:
    img = cv2.imread(path, 1)
    print(path, img.shape)
```

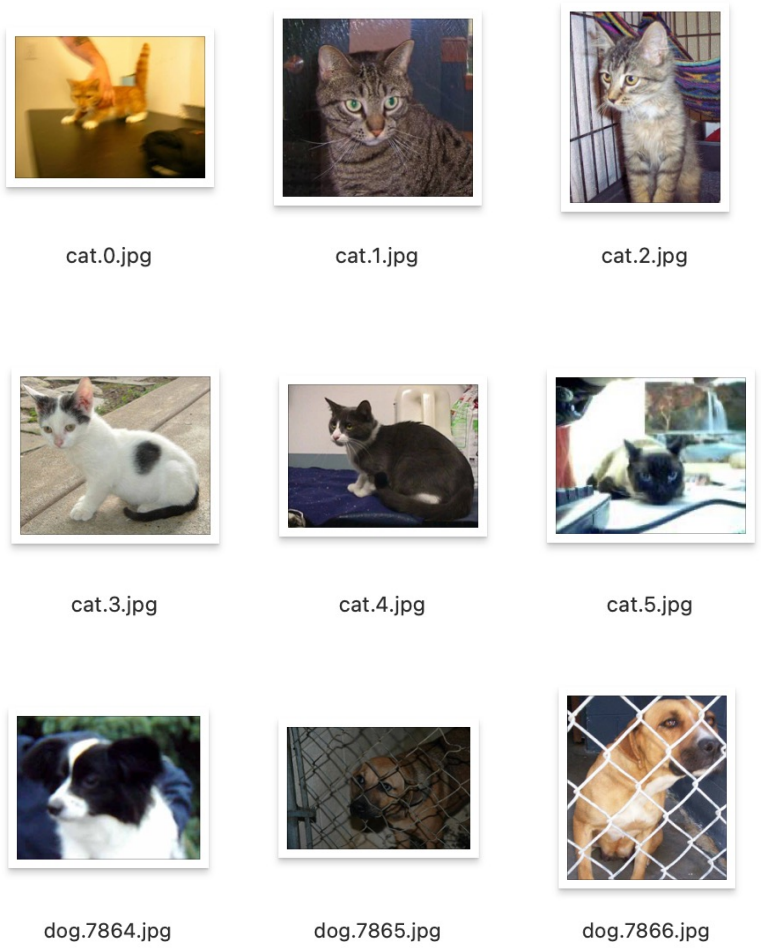
✓ 0.0s

```
samples/cat.5.jpg (144, 175, 3)
samples/cat.4.jpg (375, 499, 3)
samples/cat.0.jpg (374, 500, 3)
samples/dog.7866.jpg (500, 435, 3)
samples/cat.1.jpg (280, 300, 3)
samples/cat.3.jpg (414, 500, 3)
samples/dog.7864.jpg (281, 359, 3)
samples/dog.7865.jpg (332, 499, 3)
samples/cat.2.jpg (396, 312, 3)
```

Image File Loading

❖ Load images

samples



cv2.resize(image, shape)

```
import glob
import cv2

file_paths = glob.glob('samples/*jpg')

# read and resize images
for path in file_paths:
    img = cv2.imread(path, 1)
    img = cv2.resize(img, (50, 50))
    print(path, img.shape)
```

✓ 0.0s

```
samples/cat.5.jpg (50, 50, 3)
samples/cat.4.jpg (50, 50, 3)
samples/cat.0.jpg (50, 50, 3)
samples/dog.7866.jpg (50, 50, 3)
samples/cat.1.jpg (50, 50, 3)
samples/cat.3.jpg (50, 50, 3)
samples/dog.7864.jpg (50, 50, 3)
samples/dog.7865.jpg (50, 50, 3)
samples/cat.2.jpg (50, 50, 3)
```

Image File Loading

❖ Load images


```python
import glob
import cv2

file_paths = glob.glob('samples/*jpg')

# read and resize images
images = []
for path in file_paths:
    img = cv2.imread(path, 1)
    img = cv2.resize(img, (50, 50))

    # add to list
    images.append(img)

# test (images is a list)
print(len(images))
```

✓ 0.0s

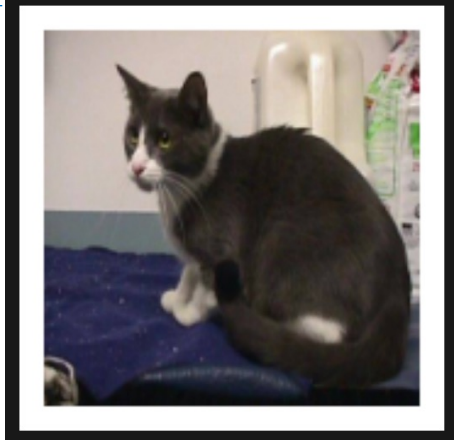
```
9
```

Image File Loading

❖ Load images


```
import matplotlib.pyplot as plt

an_image = images[1]
an_image = cv2.cvtColor(an_image,
                        cv2.COLOR_BGR2RGB)
plt.imshow(an_image)
```

✓ 0.0s



```
import glob
import cv2
import numpy as np

file_paths = glob.glob('samples/*jpg')

# read and resize images
images = []
for path in file_paths:
    img = cv2.imread(path, 1)
    img = cv2.resize(img, (50, 50))

    # add to list
    images.append(img)

# convert list to numpy array
images = np.array(images)

# test (images is an array)
print(images.shape)
```

✓ 0.0s

```
(9, 50, 50, 3)
```

Image File Loading

❖ matplotlib

```python
import glob
import cv2
import numpy as np

file_paths = glob.glob('train/*jpg')

# read and resize images
images = []
for path in file_paths:
    img = cv2.imread(path, 1)
    img = cv2.resize(img, (50, 50))

    # add to list
    images.append(img)

# convert list to numpy array
images = np.array(images)

# test (images is an array)
print(images.shape)
```

✓ 9.7s

```
(22000, 50, 50, 3)
```

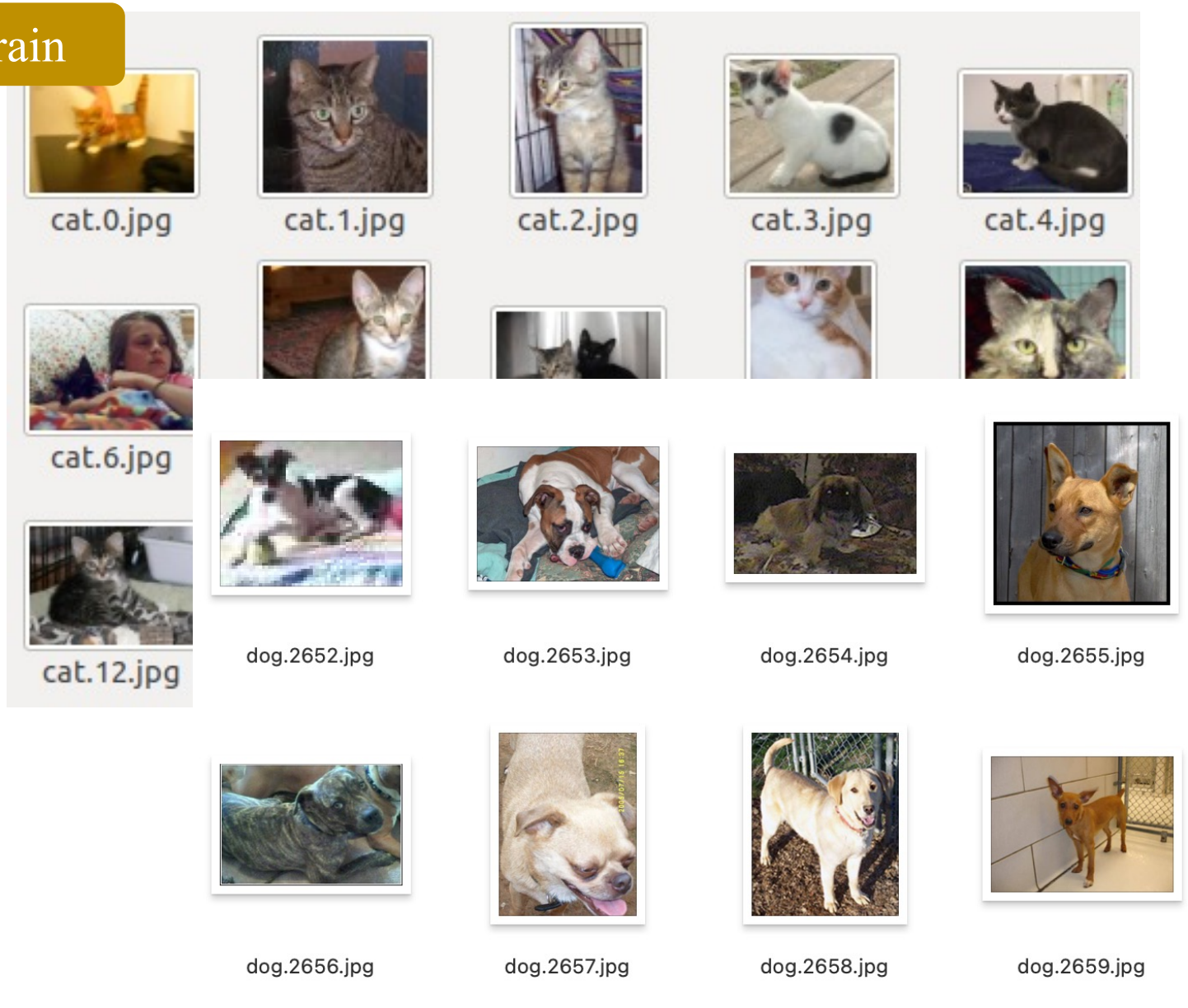
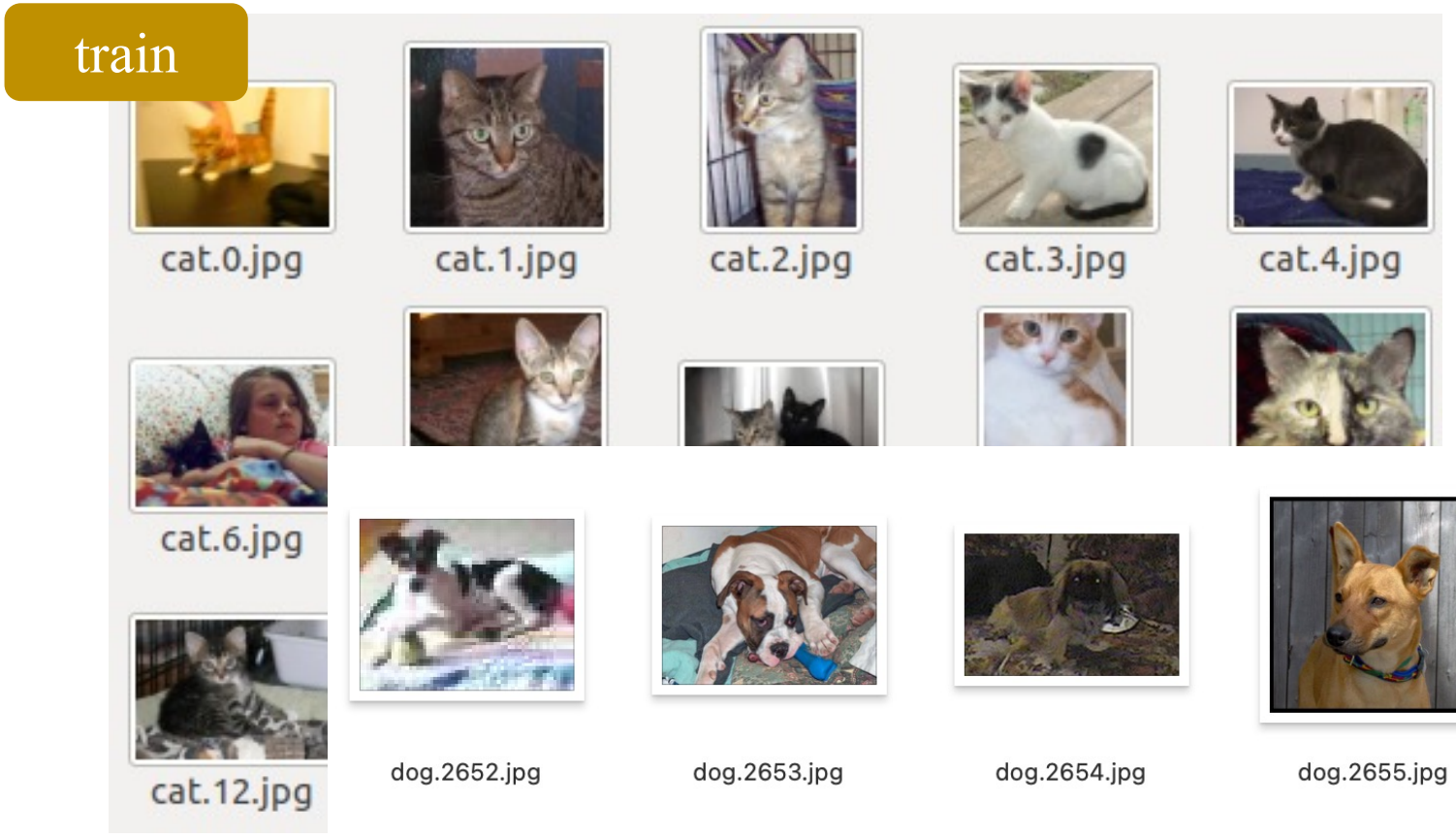


Image File Loading

❖ matplotlib

```
import glob
import cv2
import numpy as np
from tqdm import tqdm

file_paths = glob.glob('train/*jpg')

# read and resize images
images = []
for path in tqdm(file_paths):
    img = cv2.imread(path, 1)
    img = cv2.resize(img, (50, 50))

    # add to list
    images.append(img)

# convert list to numpy array
images = np.array(images)

# test (images is an array)
print(images.shape)
```

```
100%|██████████| 22000/22000 [00:09<00:00,
(22000, 50, 50, 3)
```



Outline

SECTION 1

OpenCV and NumPy

SECTION 2

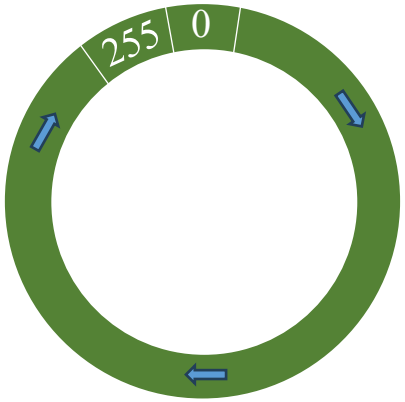
Image Loading

SECTION 3

Brightness Changes

SECTION 4

Color Conversion



in Numpy

What we think

Brightness Changes

❖ Data type

image1.png

uint8 data type

- Unsigned: All values are positive
- 8 bits: $A \in [0,255]$
- Integer: All values are whole number

```
import cv2

# Read the image
image = cv2.imread('image1.png', 1)
print(image.shape)
print(image.dtype)
```

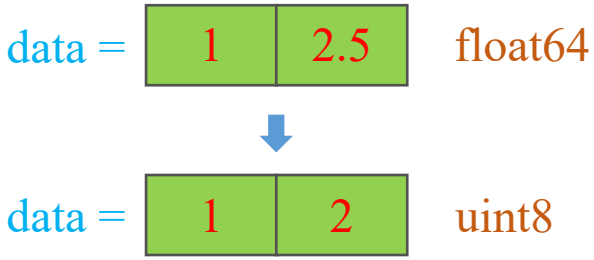
```
(162, 311, 3)
uint8
```

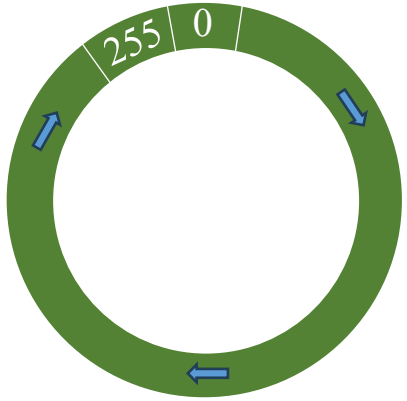
Cast to a specified type

np.ndarray.astype(dtype = np.uint8)

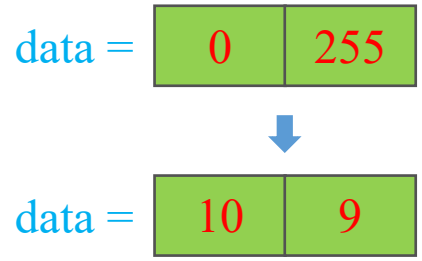
Brightness Changes

❖ Problem of out of range





in Numpy




```
import numpy as np

data = np.array([1, 2.5])
print(data.dtype)

data = data.astype(np.uint8)
print(data.dtype)
```

✓ 0.0s

```
float64
uint8
```

What we think

```
import numpy as np

data = np.array([0, 255])
data = data.astype(np.uint8)
print(data)

data = data + 10
print(data)
```

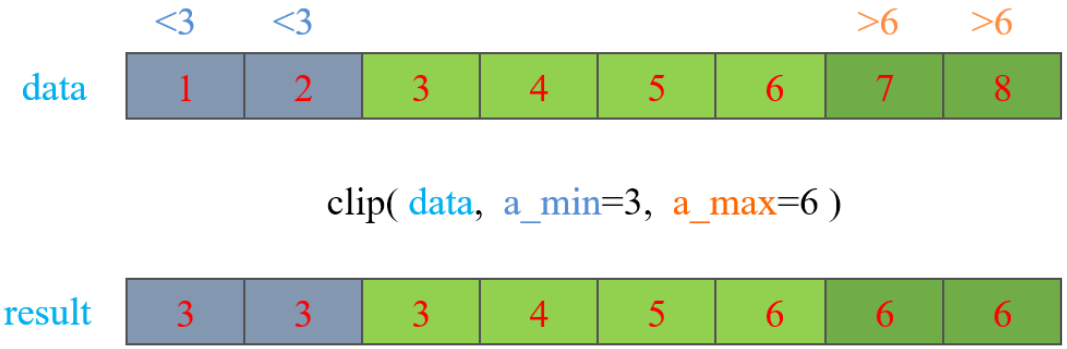
✓ 0.0s

```
[  0 255]
[10   9]
```

Brightness Changes

❖ clip() and where() functions

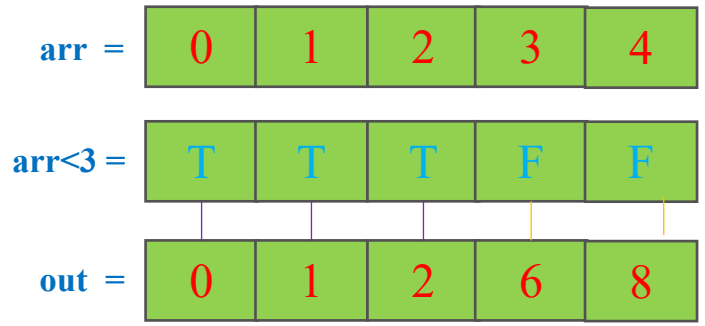
numpy.clip()




```python
import numpy as np

data = np.array([1, 2, 3, 4, 5, 6, 7, 8])
print("data: ", data)

# element < 3 sẽ gán bằng 3
# element > 6 sẽ gán bằng 6
result = np.clip(data, a_min=3, a_max=6)
print("result: ", result)
```

```
data:  [1 2 3 4 5 6 7 8]
result:  [3 3 3 4 5 6 6 6]
```

where() function




```python
# create an array
arr = np.arange(5)
print(arr)

# condition
condition = arr < 3
out = np.where(condition, arr, arr*2)

print(condition)
print(out)
```

```
[0 1 2 3 4]
[ True  True  True False False]
[0 1 2 6 8]
```

Brightness Changes

❖ **Given two images**

Grayscale Image

(Height, Width)

Color Image

(Height, Width, Channel)

Brightness Changes

❖ Load an image

```python
import cv2

# read a grayscale image
img = cv2.imread('nature.jpg', 0)

# save the image
cv2.imwrite('processed_image.jpg', img)
```

(Height, Width)

```python
# get image info

import numpy as np
import cv2

# read a grayscale image
img = cv2.imread('nature.jpg', 0)

shape = img.shape
print(shape)
```

(500, 1200)

img = cv2.imread(path, 0)

cv2.imwrite(path, img)

Brightness Changes

❖ Load an image

```python
import cv2

# read a color image
img = cv2.imread('nature.jpg', 1)

# save the image
cv2.imwrite('processed_image.jpg', img)
```

(Height, Width, Channel)

```python
# get image info

import numpy as np
import cv2

# read a grayscale image
img = cv2.imread('nature.jpg', 1)

shape = img.shape
print(shape)
```

(500, 1200, 3)

img = cv2.imread(path, 1)

cv2.imwrite(path, img)

Brightness Changes

❖ **Increase the brightness of a grayscale image**

Idea
For each pixel
    Increase pixel value
    by a value v

Increase brightness

$I = I + v$
$I = clip(I)$

Brightness Changes

❖ **Decrease the brightness of a grayscale image**

Idea
For each pixel in each channel
    Decrease pixel value
    by a value **v**

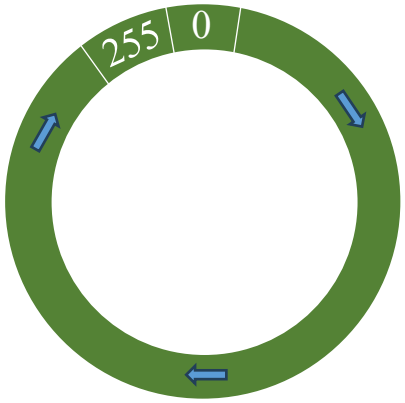
Decrease brightness

$I = I - v$
$I = clip(I)$

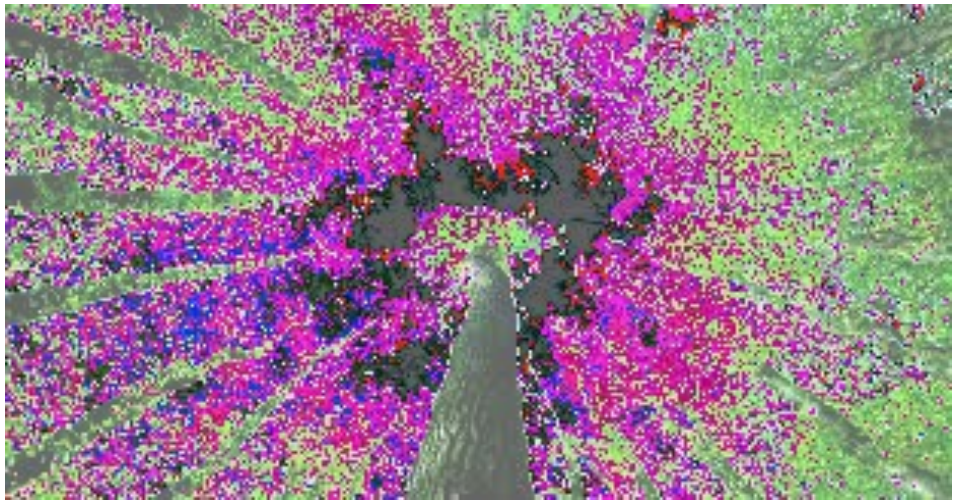
Brightness Changes

❖ Why?

in Numpy



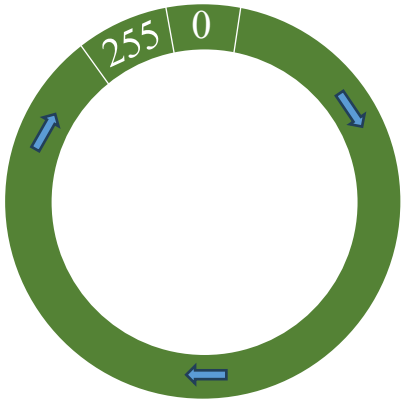

```python
# problem

import cv2
import matplotlib.pyplot as plt

# Read the image
image = cv2.imread('image1.png', 1)
image = image + 100

# Convert and save
cv2.imwrite('output1.png', image)
```



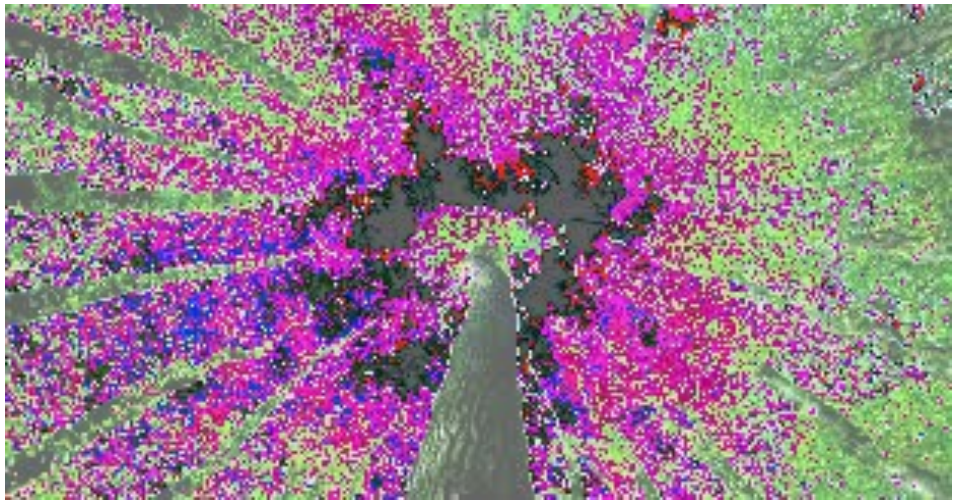
Brightness Changes

❖ Why?

in Numpy




```python
# problem

import cv2
import numpy as np

# Read the image
image = cv2.imread('image1.png', 1)
image = np.where(image + 100 > 255,
                 255, image + 100)

# Convert and save
cv2.imwrite('output2.png', image)
```



Brightness Changes

❖ Solution

200 + 100 = 44
uint8 uint8

200 + 100 = 300
float64 float64

```
# solution

import cv2
import numpy as np

# Read the image
image = cv2.imread('image1.png', 1)
image = np.where(image.astype(float) + 100 > 255,
                 255, image + 100)

# Convert and save
cv2.imwrite('output3.png', image)
```

Brightness Changes

❖ Increase brightness - Implementation - 1

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('nature.jpg', 1)
img = img.astype(float)

img = img + 50
img = np.clip(img, 0, 255)

img = img.astype(np.uint8)
cv2.imwrite('output.jpg', img)
```

Brightness Changes

❖ Increase brightness - Implementation - 2

```
1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  img = cv2.imread('nature.jpg', 1)
6  img = img.astype(float)
7
8  img = img + 50
9  img = np.where(img<0., 0., img)
10 img = np.where(img>255., 255., img)
11
12 img = img.astype(np.uint8)
13 cv2.imwrite('increase50_where.jpg', img)
```

Brightness Changes

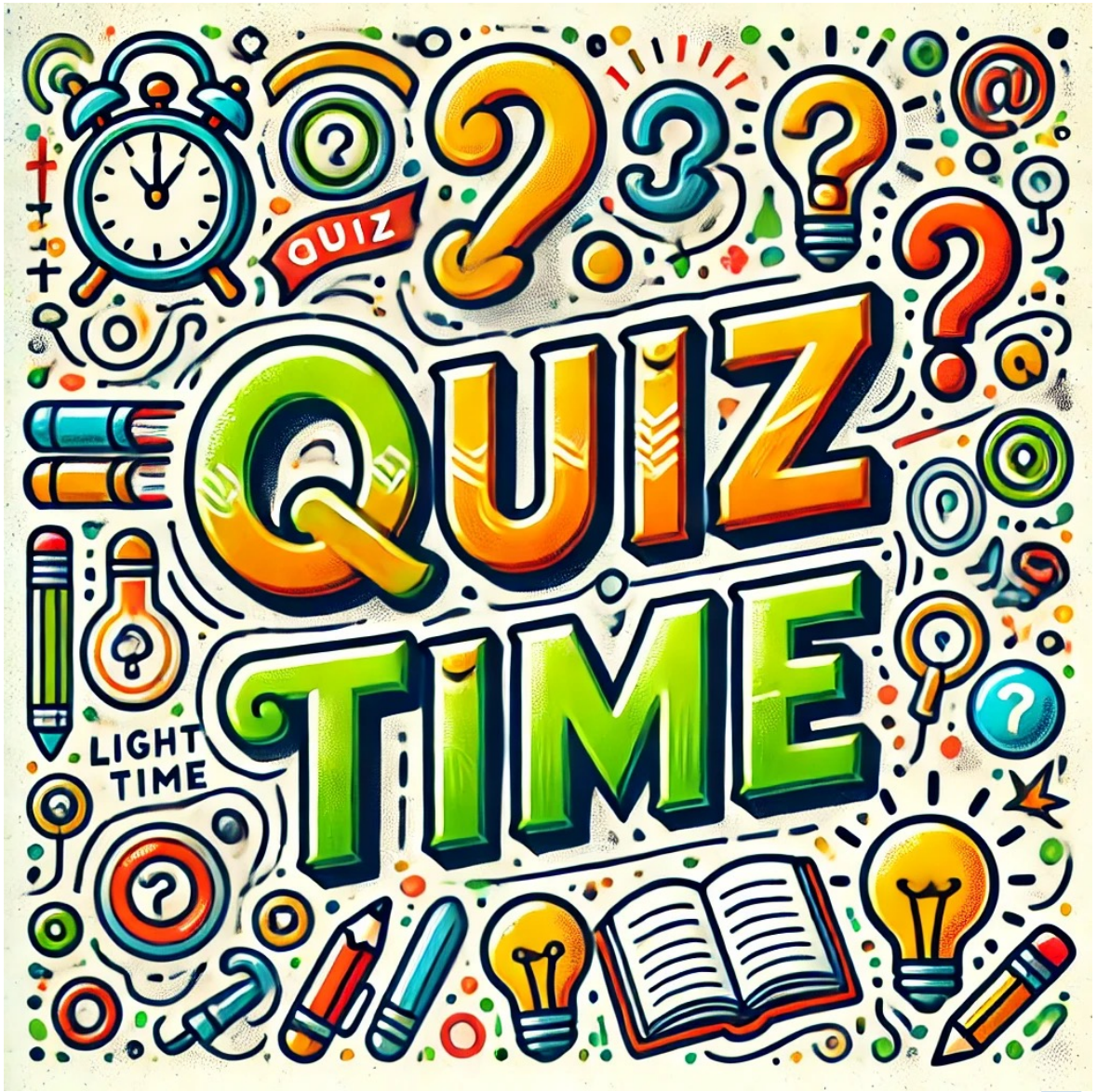
❖ Decrease brightness - Implementation - 1

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('nature.jpg', 1)
img = img.astype(float)

img = img - 80
img = np.clip(img, 0, 255)

img = img.astype(np.uint8)
cv2.imwrite('output.jpg', img)
```

Brightness Changes

❖ Decrease brightness - Implementation - 2

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('nature.jpg', 1)
img = img.astype(float)

img = img - 80
img = np.where(img<0., 0., img)
img = np.where(img>255., 255., img)

img = img.astype(np.uint8)
cv2.imwrite('decrease80_where.jpg', img)
```


QUIZ
QUIZ
TIME
LIGHT
TIME

Outline

SECTION 1

OpenCV and NumPy

SECTION 2

Image Loading

SECTION 3

Brightness Changes

SECTION 4

Color Conversion

Color to Grayscale Conversion

❖ apply_along_axis() function

data

2	6	0	1
3	0	8	3
4	4	5	0
2	0	4	6

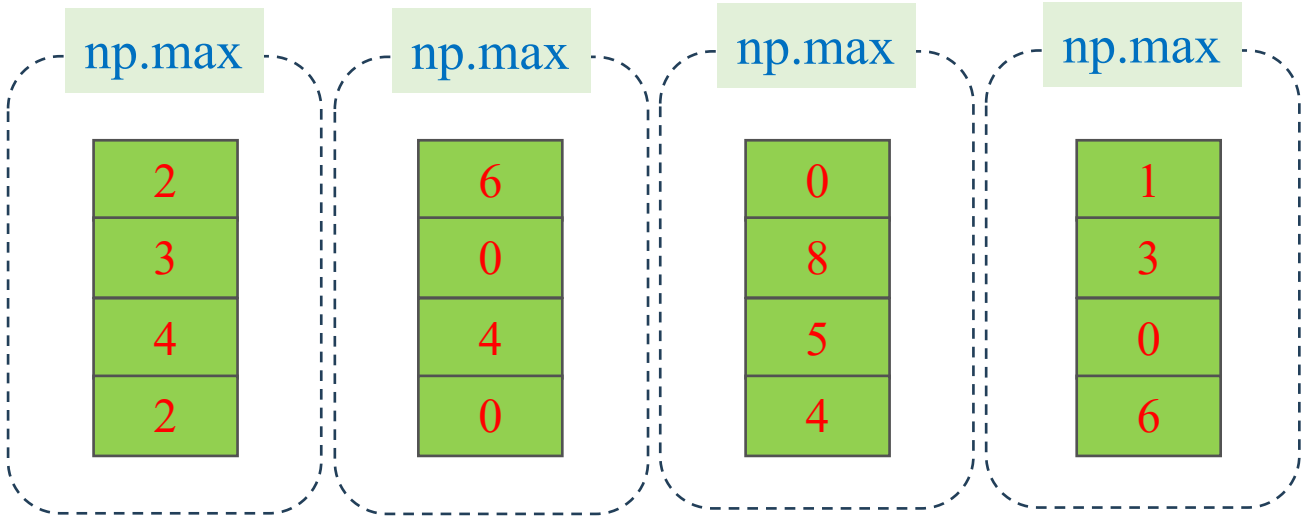
data

2	6	0	1
3	0	8	3
4	4	5	0
2	0	4	6

np.apply_along_axis(np.max,
axis=0,
arr=data)

result0

4	6	8	6




```
import numpy as np

data = np.array([[2, 6, 0, 1],
                 [3, 0, 8, 3],
                 [4, 4, 5, 0],
                 [2, 0, 4, 6]])
print("data: \n", data)

# Sử dụng function np.max với axis=0
result0 = np.apply_along_axis(np.max, axis=0, arr=data)
print("result0: \n", result0)
```

Color to Grayscale Conversion

❖ apply_along_axis() function

data

2	6	0	1
3	0	8	3
4	4	5	0
2	0	4	6

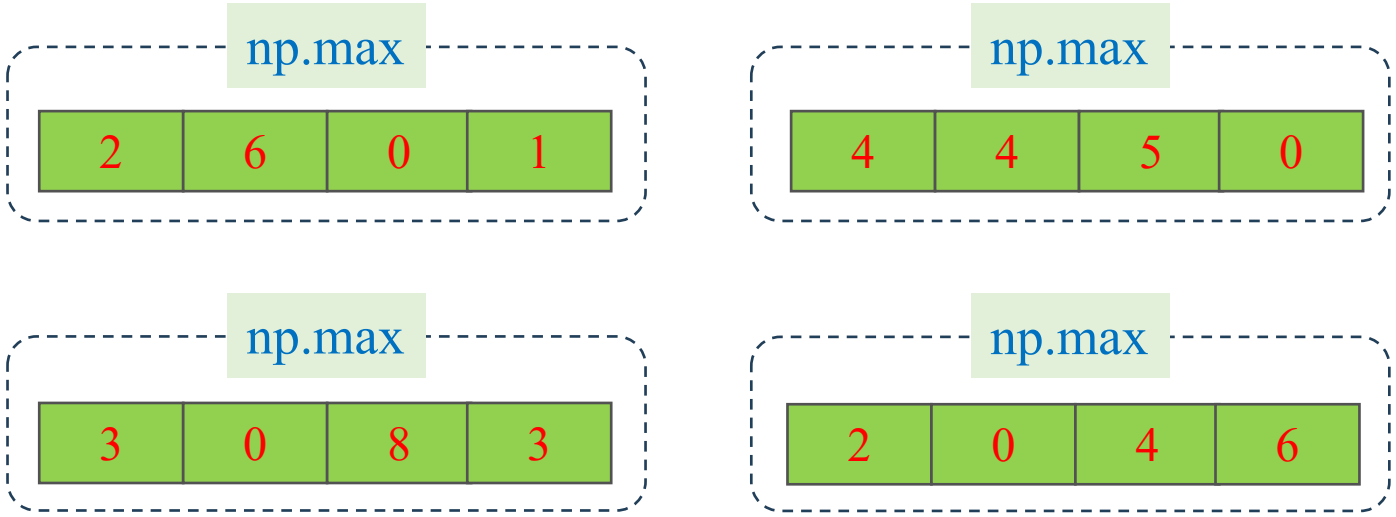
data

2	6	0	1
3	0	8	3
4	4	5	0
2	0	4	6

result1

6
8
5
6

np.apply_along_axis(np.max,
axis=1,
arr=data)



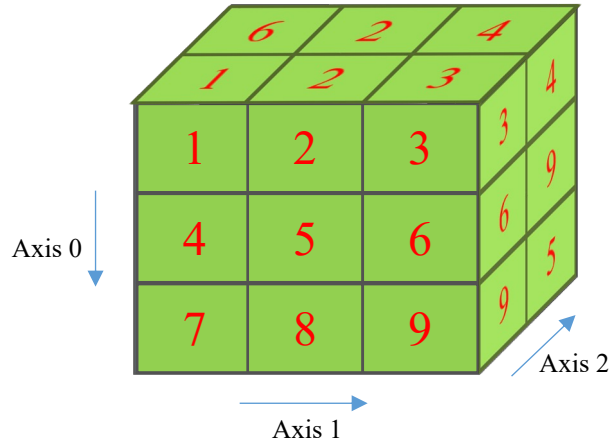

```
import numpy as np

data = np.array([[2, 6, 0, 1],
                 [3, 0, 8, 3],
                 [4, 4, 5, 0],
                 [2, 0, 4, 6]])
print("data: \n", data)

# Sử dụng function np.max với axis=1
result1 = np.apply_along_axis(np.max,
                              axis=1,
                              arr=data)
```

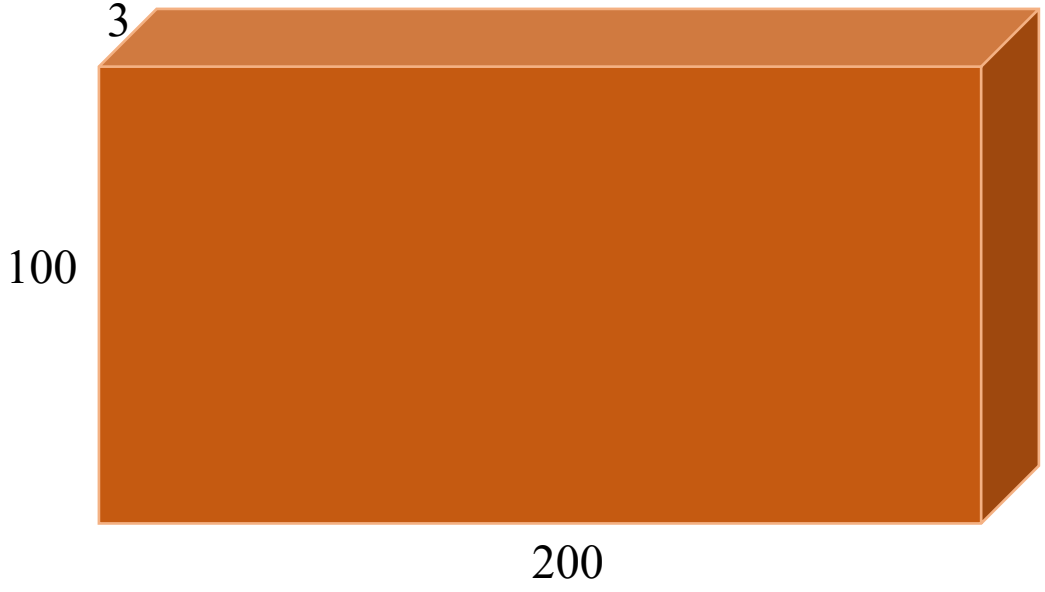
Color to Grayscale Conversion

❖ Apply with images

200

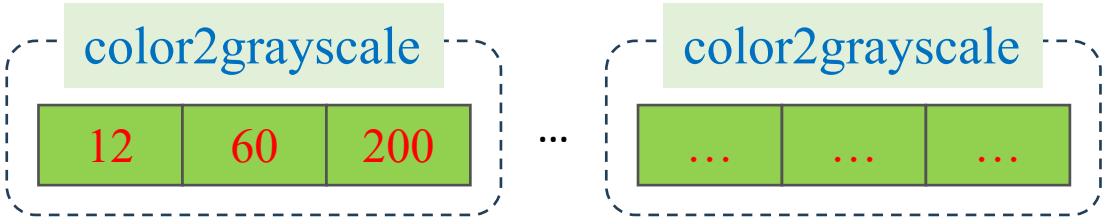
100





Color to Grayscale Conversion

❖ Apply with images




```python
import cv2
import numpy as np

# Read the image
image = cv2.imread('image_color.png', 1)

def color2grayscale(vector):
    result = vector.mean()
    result = result.astype(np.uint8)
    return result

# Convert
grayscale_image = np.apply_along_axis(color2grayscale,
                                      axis=2,
                                      arr=image)

# Save the grayscale image
cv2.imwrite('output3.png', grayscale_image)
```

Cropping a Region

❖ Slicing

arr[for_axis_0, for_axis_1, …]

‘**:**’: get all the elements
‘**a:b**’: get the elements from a^{th} to $(b^{th}-1)$

data

	0	1
0	1	2
1	3	4
2	5	6

data[0, 1]

	0	1
0	1	2
1	3	4
2	5	6

data[1: 3]

	0	1
0	1	2
1	3	4
2	5	6

data[0: 1 , 0]

	0	1
0	1	2
1	3	4
2	5	6

data[: , :]

	0	1
0	1	2
1	3	4
2	5	6

```python
# aivietnam.ai
import numpy as np

# Khởi tạo numpy array a_arr
a_arr = np.array([[1,2,3],
                  [5,6,7]])

# Sử dụng slicing để tạo mảng b_arr
# bằng cách lấy tất cả các dòng và cột 1,2
b_arr = a_arr[:, 1:3]

print(a_arr)
print(b_arr)
```

```
[[1 2 3]
 [5 6 7]]
[[2 3]
 [6 7]]
```

a_arr =

	0	1	2
0	1	2	3
1	5	6	7

b_arr = a_arr[:, 1:3]

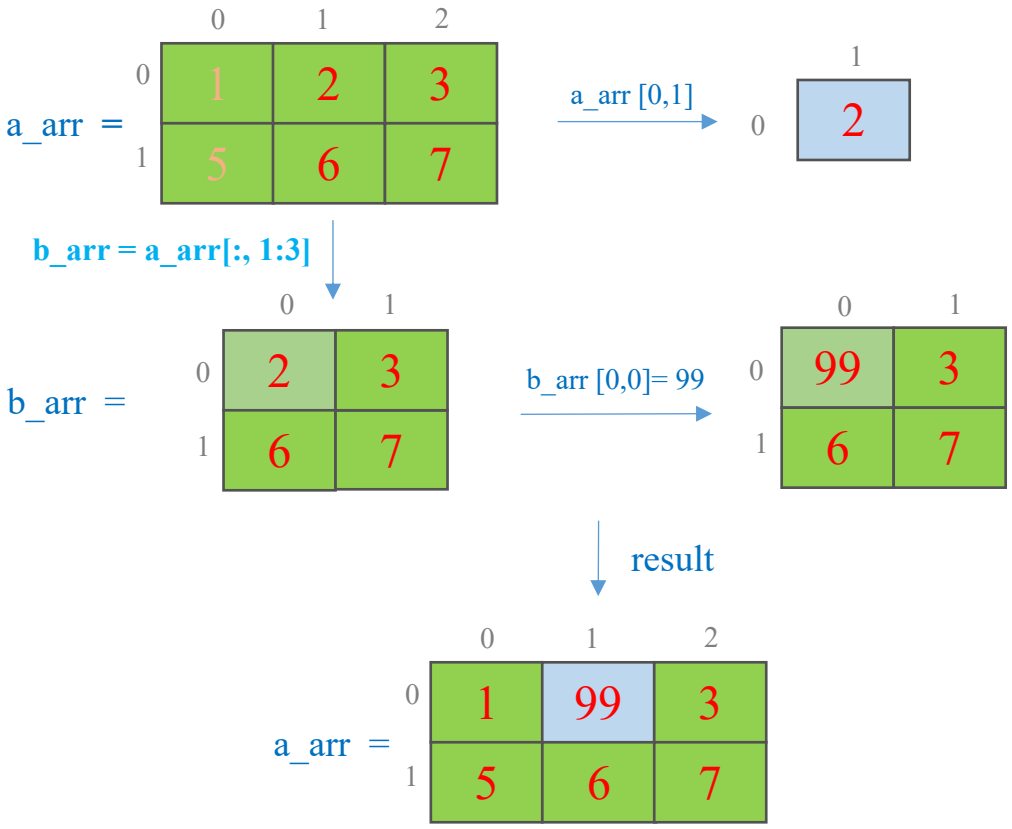
b_arr =

	0	1
0	2	3
1	6	7

Cropping a Region

❖ Slicing

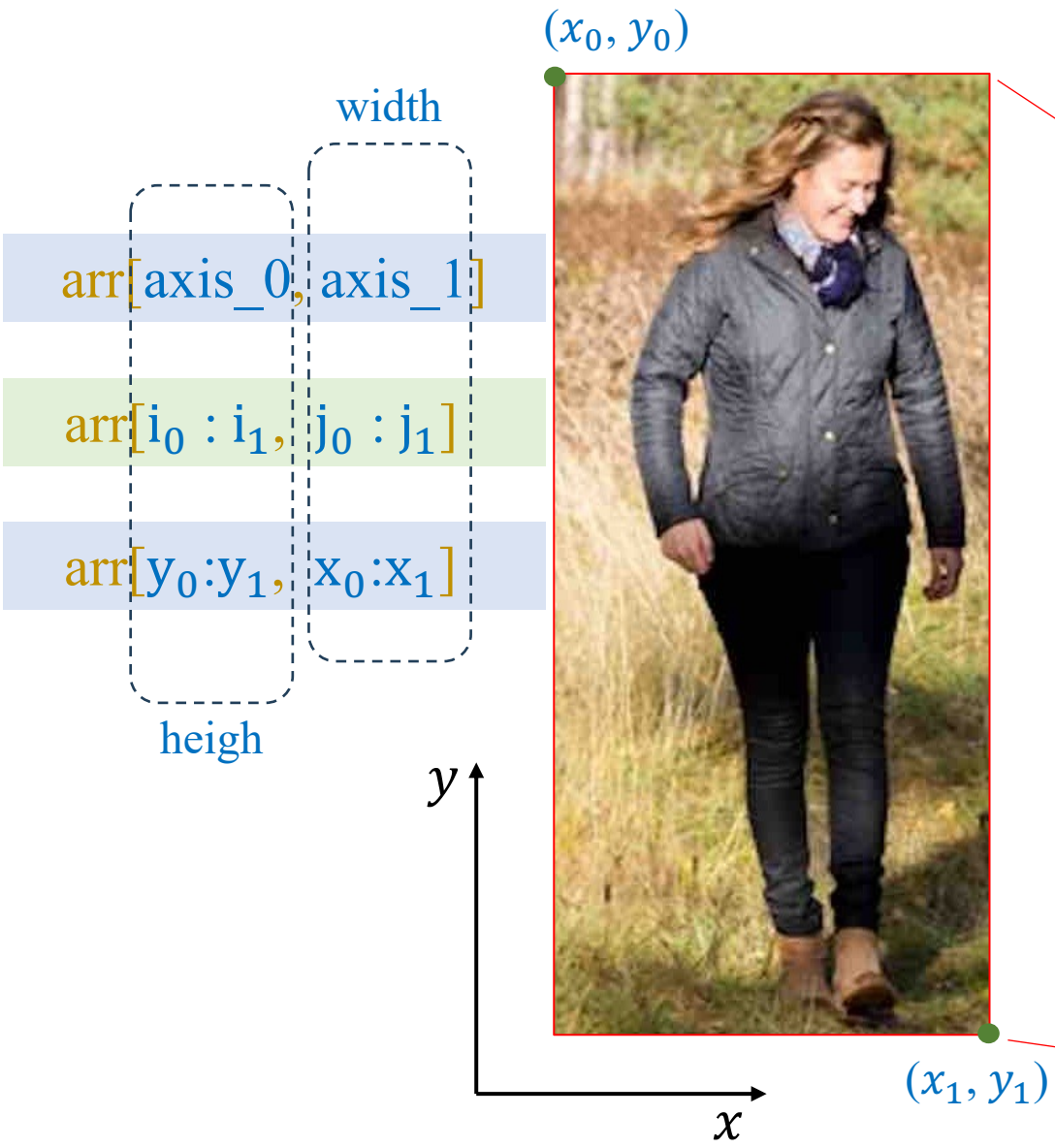
❖ Mutable



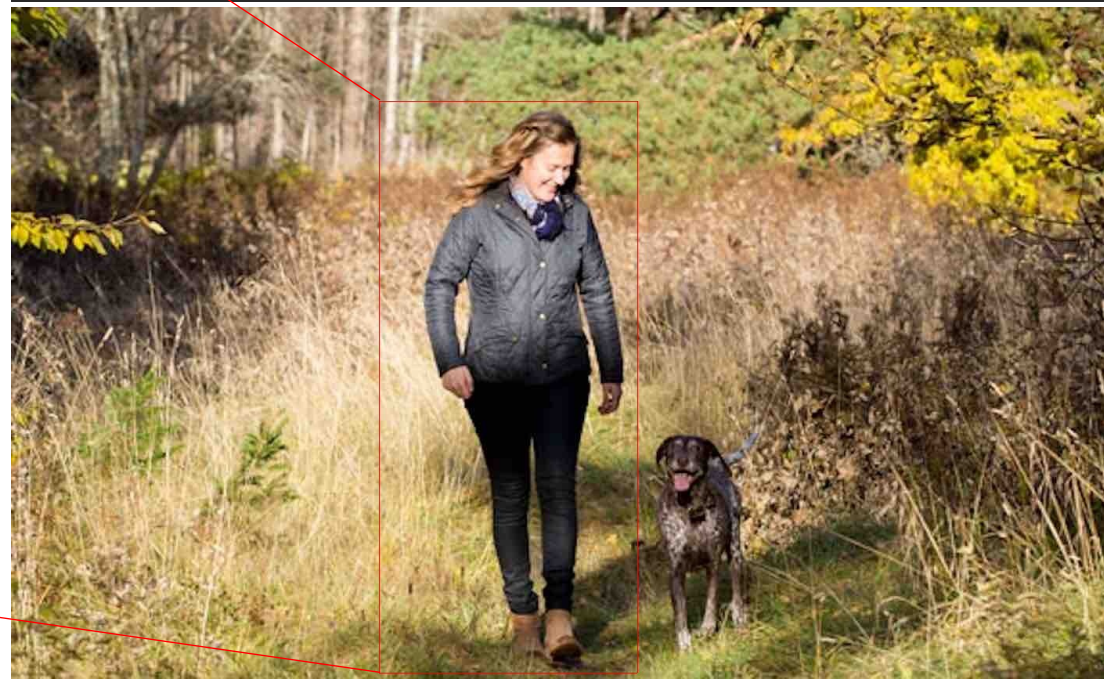

```python
# aivietnam.ai
import numpy as np

# Khởi tạo numpy array a_arr
a_arr = np.array([[1,2,3],
                  [5,6,7]])
print('a_arr \n', a_arr)

# Sử dụng slicing để tạo mảng b_arr
b_arr = a_arr[:, 1:3]
print('b_arr \n', b_arr)

print('before changing \n', a_arr[0, 1])
b_arr[0, 0] = 99
print('after changing \n', a_arr[0, 1])
```

```
a_arr
 [[1 2 3]
 [5 6 7]]
b_arr
 [[2 3]
 [6 7]]
before changing
 2
after changing
 99
```

Cropping a Region




```python
import cv2
import numpy as np

img = cv2.imread('image.jpg', 1)
height, width, channel = img.shape

x_0, y_0 = 470, 120
x_1, y_1 = 800, 850

region = img[y_0:y_1, x_0:x_1, :]
cv2.imwrite('output_region.jpg', region)
```



Drawing a Bounding Box


```python
import cv2
import numpy as np

img = cv2.imread('image.jpg', 1)
height, width, channel = img.shape

x_0, y_0 = 470, 120
x_1, y_1 = 800, 850
color = np.array([0, 0, 255])

img[y_0, x_0:x_1, :] = color
img[y_1, x_0:x_1, :] = color
img[y_0:y_1, x_0, :] = color
img[y_0:y_1, x_1, :] = color

cv2.imwrite('output_bb.jpg', img)
```