ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology
http://vku.udn.vn

Numpy

Python for AI

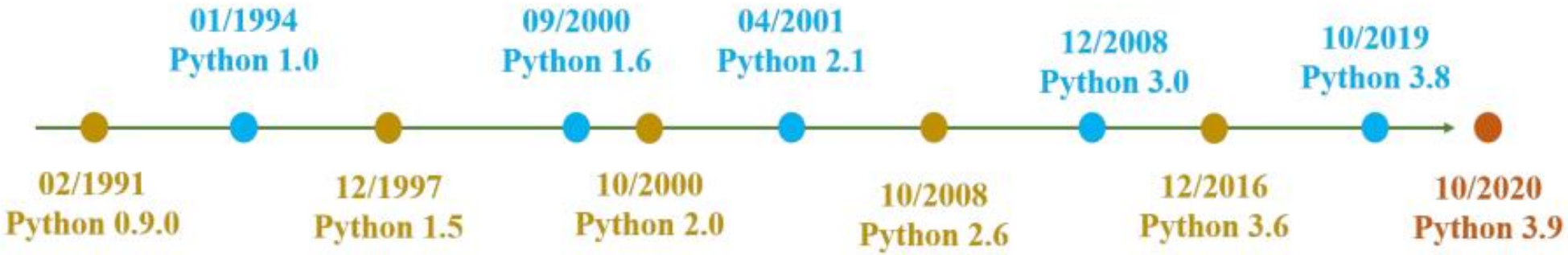
Ref from
AI VIETNAM

CONTENTS

- Introduction to Nump
- Numpy Array Indexing
- Numpy Array Operations
- Broadcasting
- Data Processing

CONTENTS

❖ Numpy is a Python library



Ý tưởng từ 1980s

Bắt đầu cài đặt
từ 12/1989

Guido van Rossum

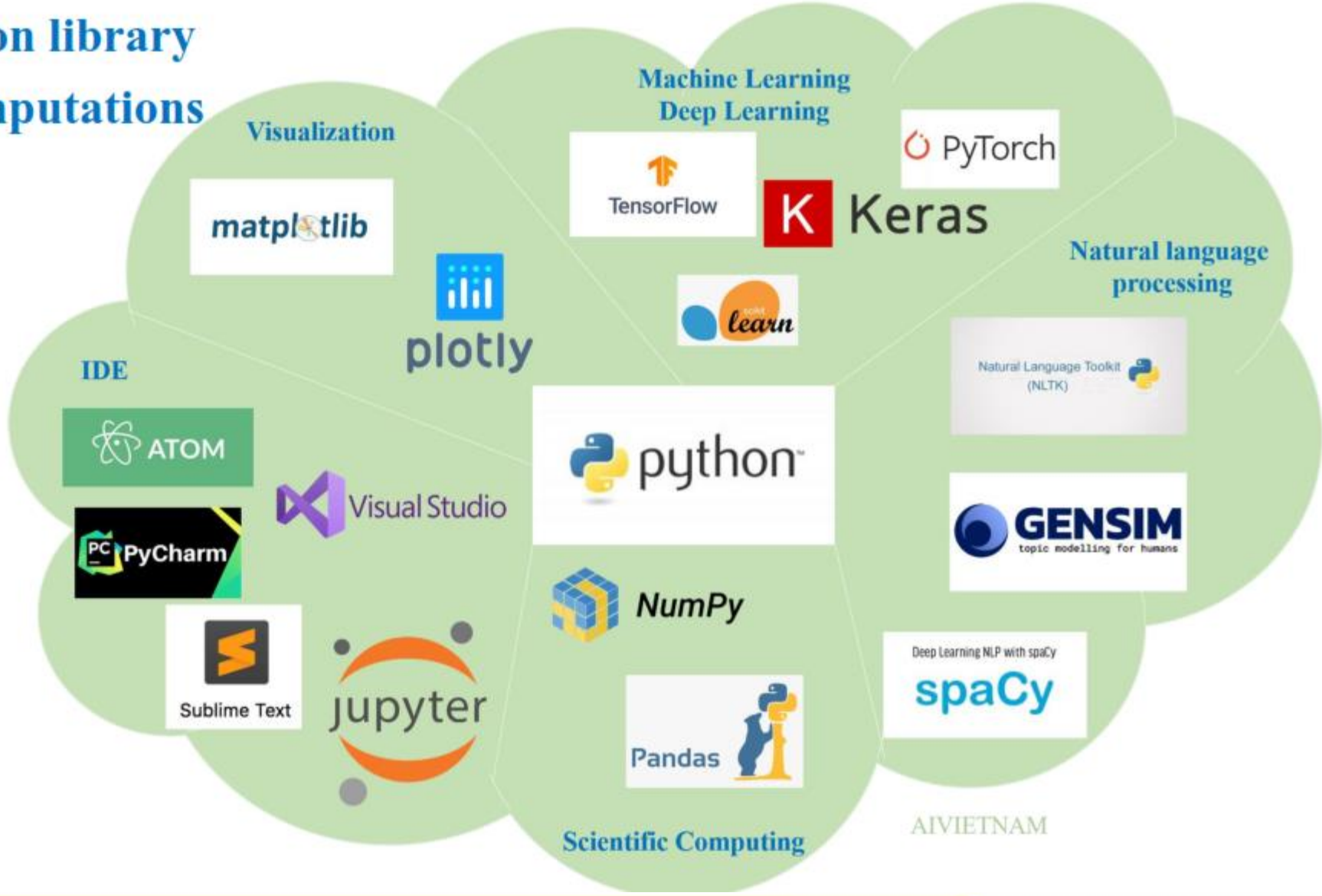
228,855 python packages
(PyPI)

Được đặt tên theo
nhóm hài Monty Python

Hỗ trợ rất mạnh cho
Data Science và Machine Learning

…Introduction

- Numpy is a Python library
- For scientific computations



…Introduction

❖ Numpy array ⟷ Tensor in Tensorflow and Pytorch

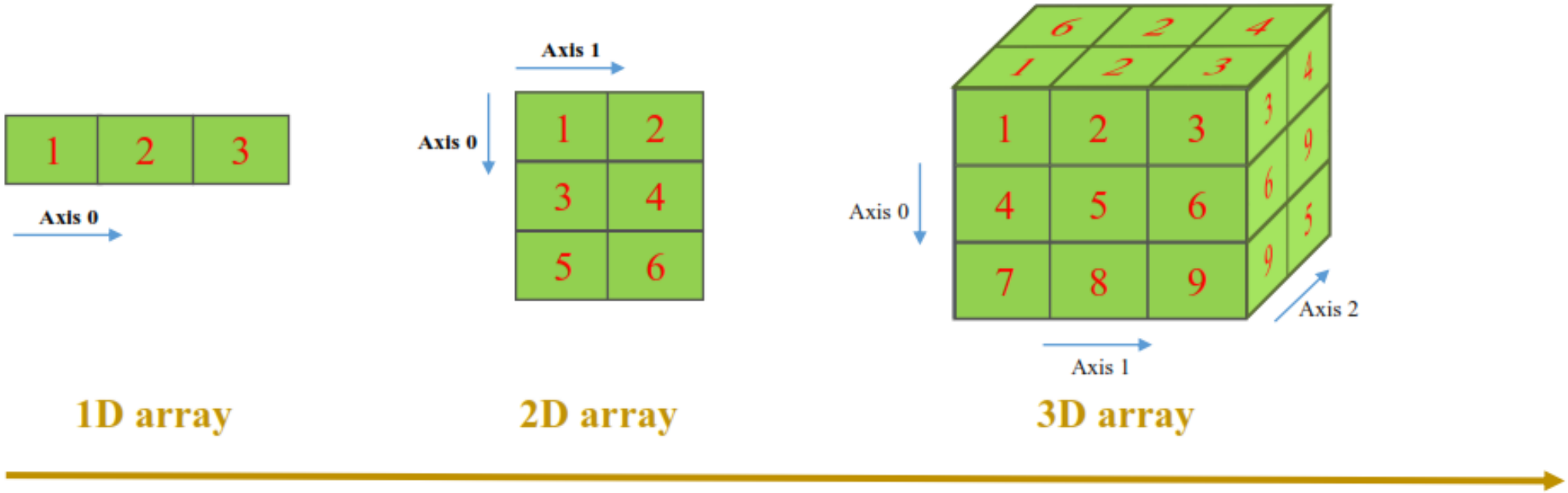
NumPy ⟷ PyTorch

TensorFlow

```python
import tensorflow as tf
import numpy as np

# create a numpy array
arr_np = np.arange(10)
print(arr_np)
print(arr_np.shape)
print('----------------------------')

# convert from numpy array to tensor
arr_tf = tf.convert_to_tensor(arr_np)
print(arr_tf)
print(arr_tf.shape)
print('----------------------------')

# convert from tensor to numpy array
arr_np_back = arr_tf.numpy()
print(arr_np_back)
print(arr_np_back.shape)
```

```
[0 1 2 3 4 5 6 7 8 9]
(10,)
----------------------------
tf.Tensor([0 1 2 3 4 5 6 7 8 9], shape=(10,), dtype=int64)
(10,)
----------------------------
[0 1 2 3 4 5 6 7 8 9]
(10,)
```

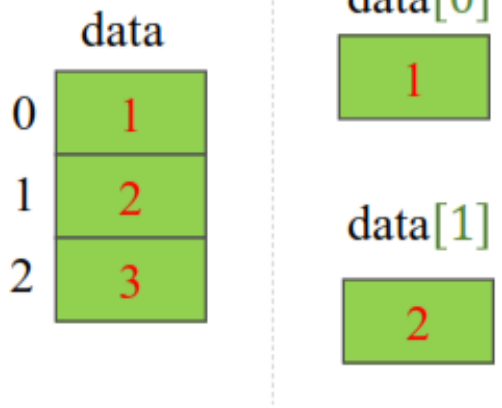
❖ Numpy arrays are multi-dimensional arrays



❖ Create Numpy array

❖ From List

```
arr_np = np.array(python_list)
```

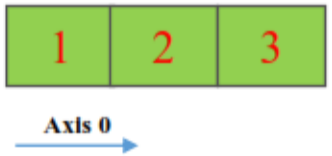



```python
# aivietnam.ai
# tạo ndarray từ list

import numpy as np

# tạo list
l = list(range(1, 4))

# tạo ndarray
data = np.array(l)

print(data)
print(data[0])
print(data[1])
```

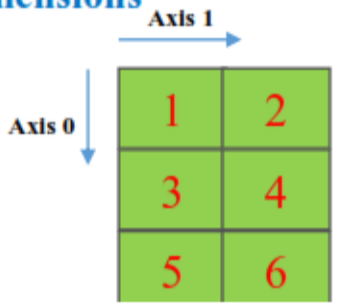
```
[1 2 3]
1
2
```

❖ Common attributes

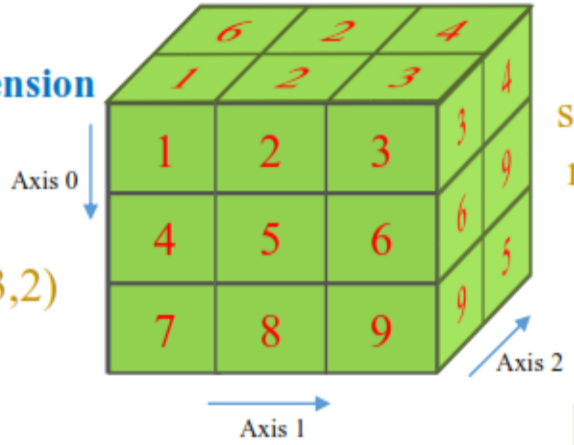
- ❖ dtype: data type
- ❖ shape: return a tuple of #elements in each dimension
- ❖ ndim: return #dimensions



shape=(3,)
ndim=1



shape=(3,2)
ndim=2



shape=(3,3,2)
ndim=2

```python
# aivietnam.ai
# tạo ndarray từ list

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
# aivietnam.ai
# tạo ndarray từ list

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
# aivietnam.ai
# tạo ndarray từ list

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)
```

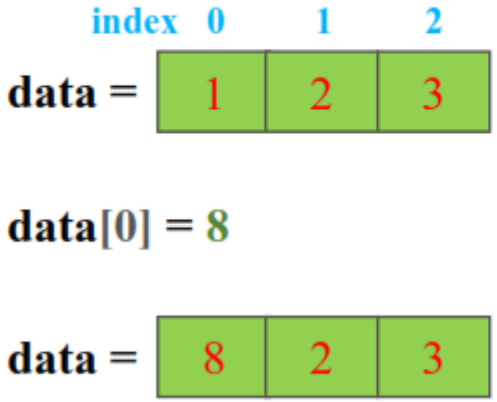
```
(3, 3, 2)
```

dtype example

```python
# aivietnam.ai
# tạo ndarray từ list

import numpy as np

# tạo ndarray
data1 = np.array([1,2,3])
print(data1.dtype)

data2 = np.array([1.,2.,3.])
print(data2.dtype)

data3 = np.array([1,2,3], dtype=np.int64)
print(data3.dtype)
```

```
int32
float64
int64
```

❖ Update an element

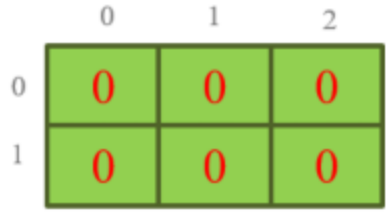



```
1  # aivietnam.ai
2  # thay đổi giá trị phần tử
3
4  import numpy as np
5
6  # tạo list
7  l = list(range(1, 4))
8
9  # tạo ndarray
10 data = np.array(l)
11 print(data)
12
13 data[0] = 8
14 print(data)
```

```
[1 2 3]
[8 2 3]
```

❖ Create Numpy array

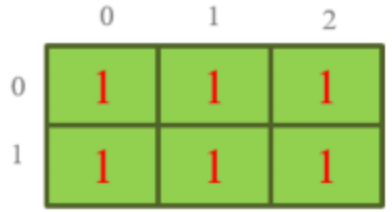
zeros() function




```
# 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.]]
```

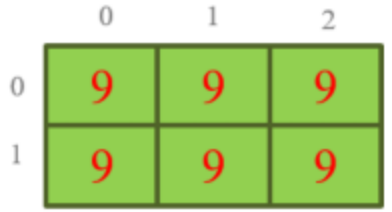
ones() function




```
# 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

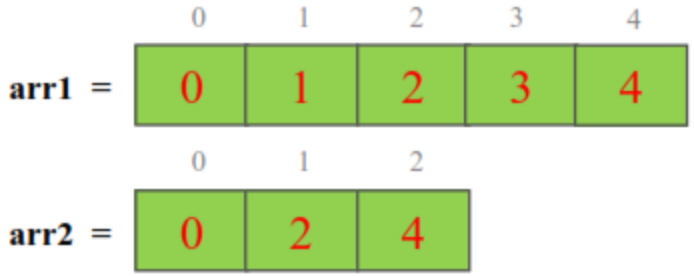



```
# aivietnam.ai
# Tạo một numpy array với tất
# cả phần tử là hằng số fill_value

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]]
```

❖ Create Numpy array

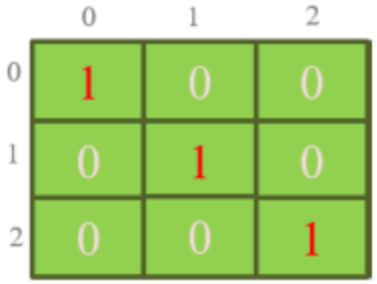
arange() function




```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]
```

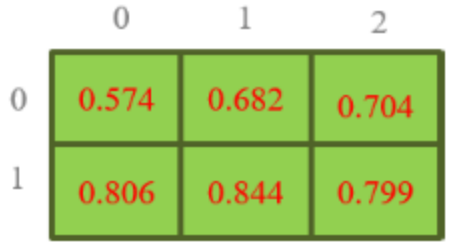
eye() function




```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.]]
```

random() function

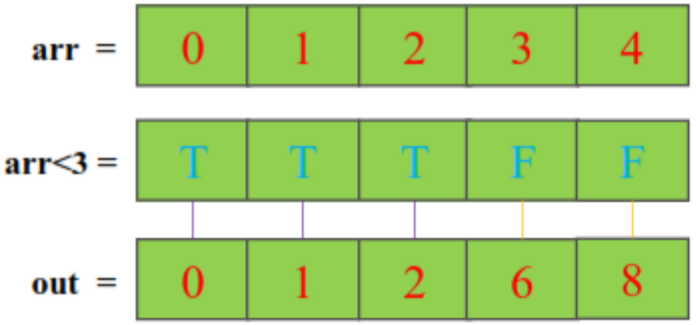



```python
# aivietnam.ai
# Tạo một numpy array với
# giá trị ngẫu nhiên

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]]
```

Some important functions

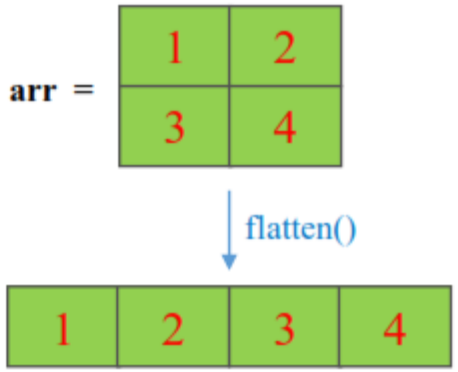
where() function




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

# 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]
```

flatten() function




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

arr = np.array([[1,2], [3,4]])
out = arr.flatten()

print(arr)
print(out)
```

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

Some important functions

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)
```

CONTENTS

- Introduction to Numpy
- **Numpy Array Indexing**
- Numpy Array Operations
- Broadcasting
- Data Processing

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}

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]

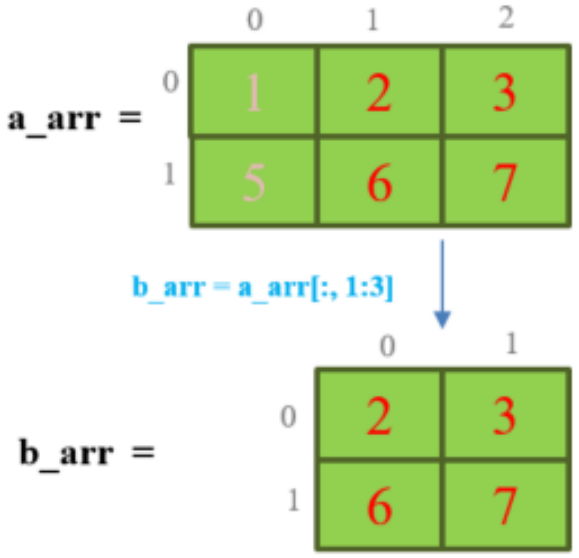
	0	1
0	1	2
1	3	4
2	5	6

data[: , :]

	0	1
0	1	2
1	3	4
2	5	6

…Array Indexing

❖ Slicing



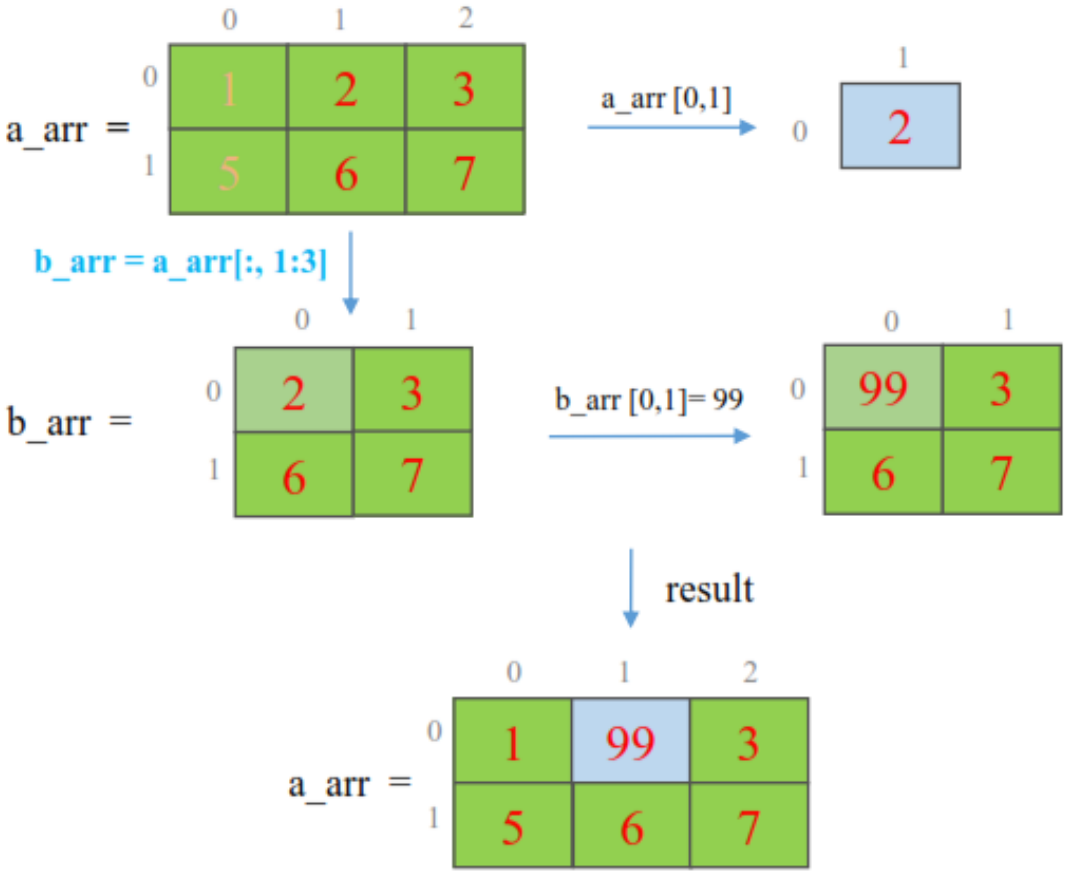

```python
# aivietnam.ai
import numpy as np

# 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]]
```

…Array Indexing

❖ Slicing

❖ Mutable

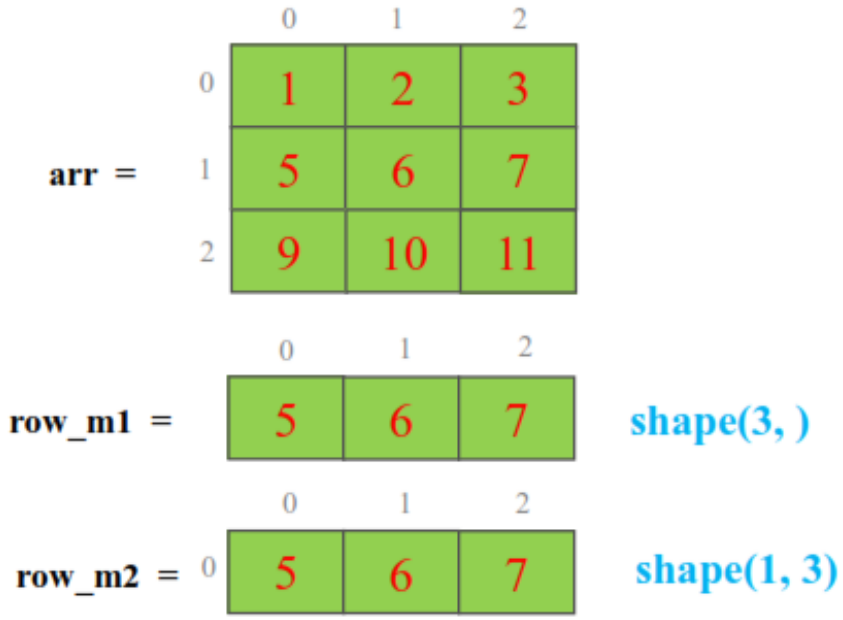



```python
# aivietnam.ai
import numpy as np

# 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

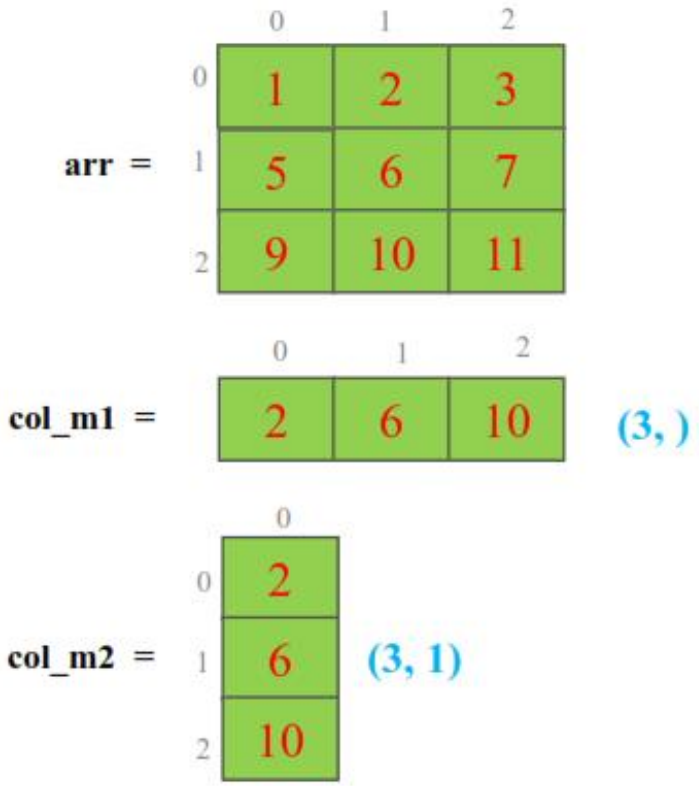



```python
# aivietnam.ai
import numpy as np

# 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

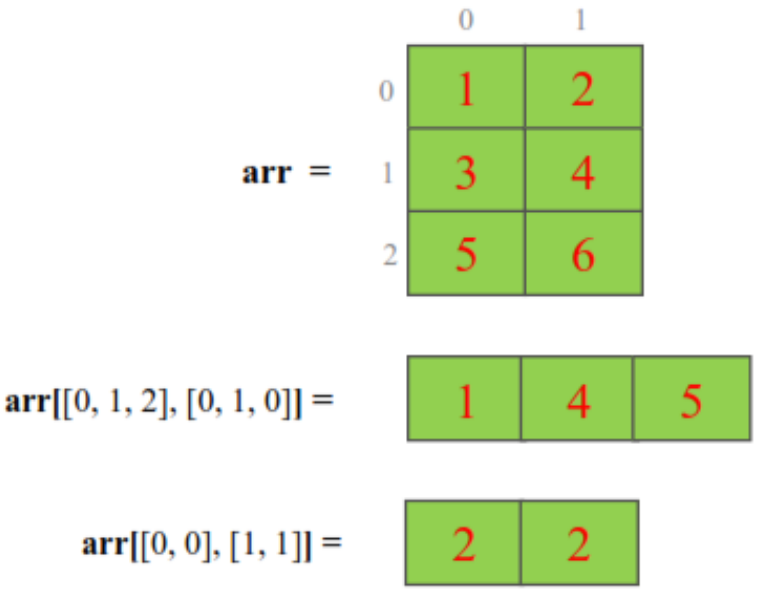



```python
# aivietnam.ai
import numpy as np

# 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

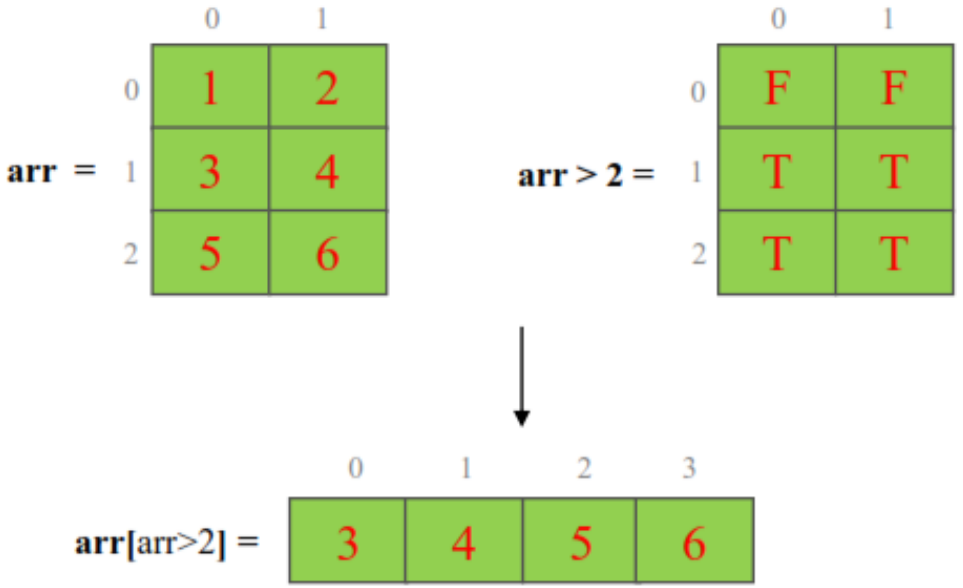



```python
# aivietnam.ai
import numpy as np

# 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]])
print(arr)

# Tìm các phần tử lớn hơn 2
bool_idx = (arr > 2)
print(bool_idx)
```

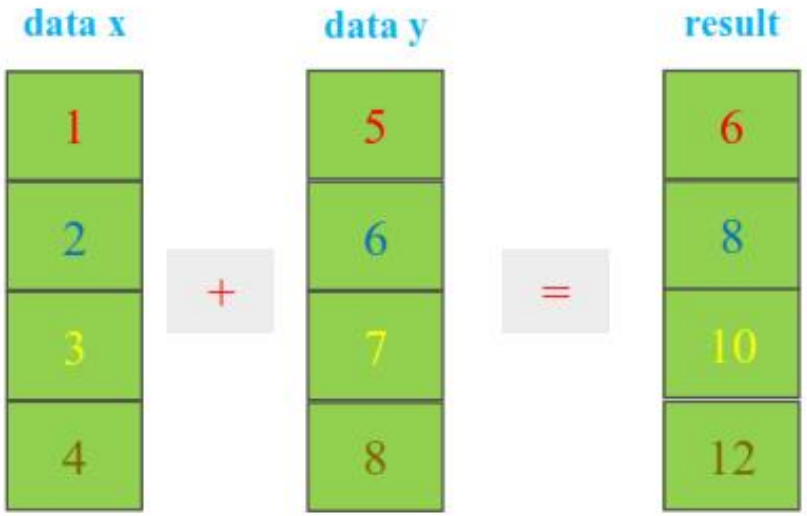
```
[[1 2]
 [3 4]
 [5 6]]
[[False False]
 [ True  True]
 [ True  True]]
```

CONTENTS

- Introduction to Numpy
- Numpy Array Indexing
- **Numpy Array Operations**
- Broadcasting
- Data Processing

Array Operations

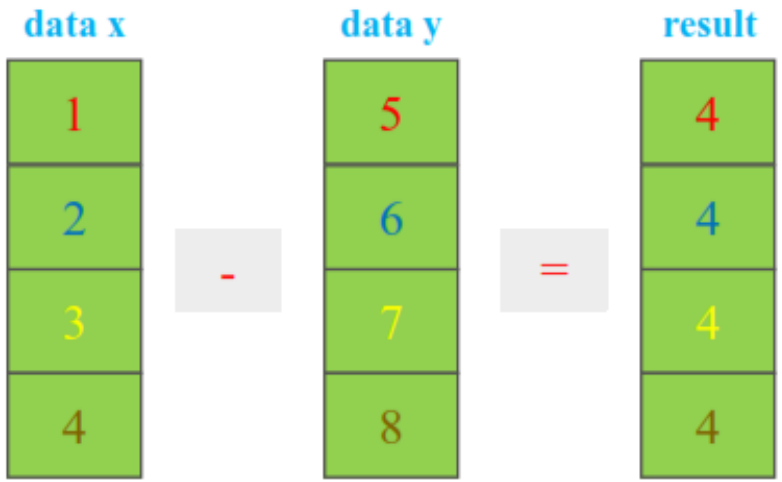
❖ Addition




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

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

print('data x \n', x)
print('data y \n', y)

# Tổng của 2 mảng
print('method 1 \n', x + y)
print('method 2 \n', np.add(x, y))
```

```
data x
 [1 2 3 4]
data y
 [5 6 7 8]
method 1
 [ 6  8 10 12]
method 2
 [ 6  8 10 12]
```

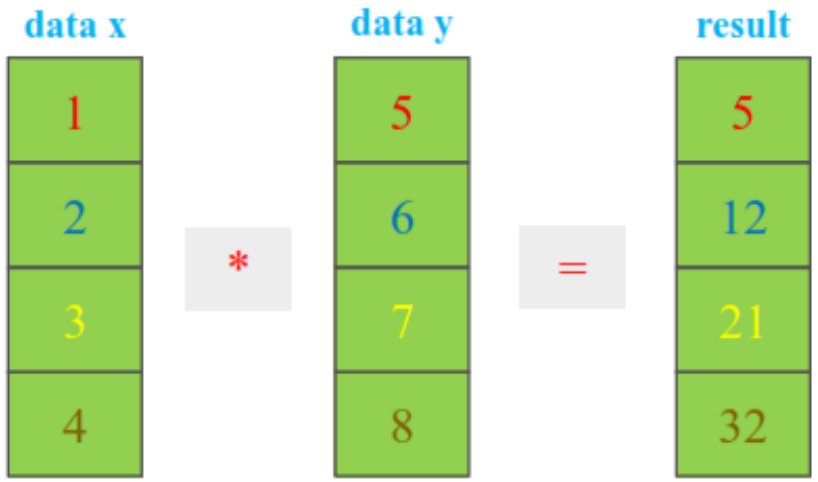
❖ Subtraction




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

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

print('data x \n', x)
print('data y \n', y)

# Hiệu 2 mảng
print('method 1 \n', x - y)
print('method 2 \n', np.subtract(x, y))
```

```
data x
 [5 6 7 8]
data y
 [1 2 3 4]
method 1
 [4 4 4 4]
method 2
 [4 4 4 4]
```

❖ Multiplication

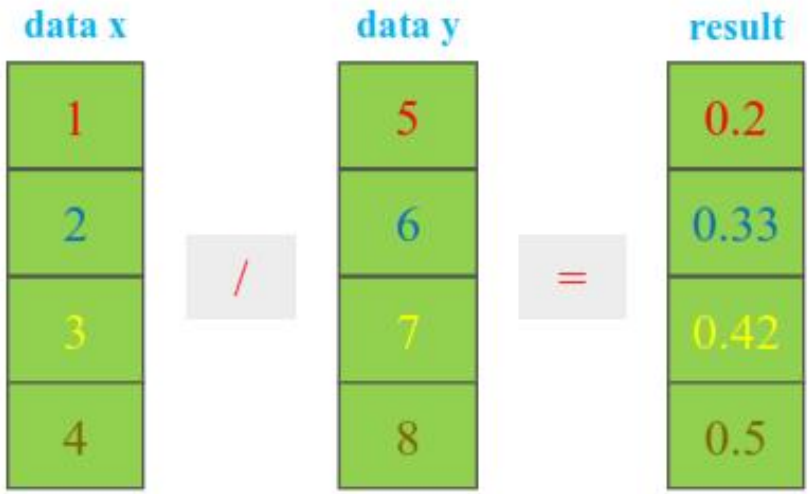



```python
# aivietnam.ai
import numpy as np

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

print('data x \n', x)
print('data y \n', y)

# Tích các phần tử tương ứng giữa x và y
print('method 1 \n', x*y)
print('method 2 \n', np.multiply(x, y))
```

```
data x
 [1 2 3 4]
data y
 [5 6 7 8]
method 1
 [ 5 12 21 32]
method 2
 [ 5 12 21 32]
```

…Array Operations

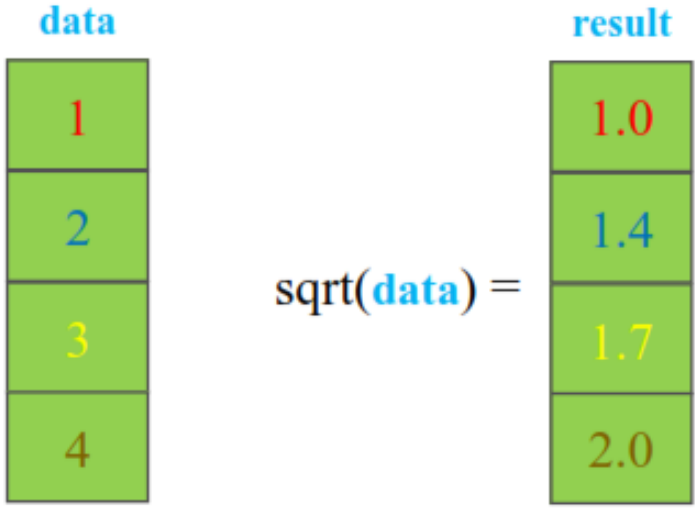
❖ Division




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

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

print('data x \n', x)
print('data y \n', y)

# Phép chia các từng phần tương ứng x cho y
print('method 1 \n', x / y)
print('method 2 \n', x // y)
print('method 3 \n', np.divide(x, y))
```

```
data x
 [5 6 7 8]
data y
 [1 2 3 4]
method 1
 [5.         3.         2.33333333 2.        ]
method 1
 [5 3 2 2]
method 2
 [5.         3.         2.33333333 2.        ]
```

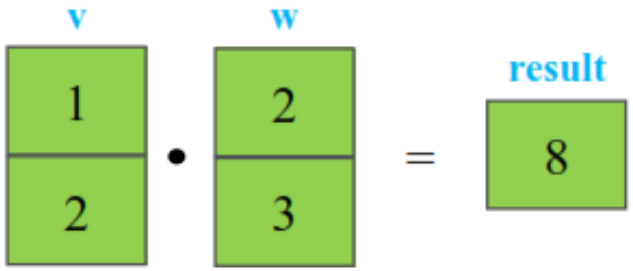
❖ Square root




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

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

print('data \n', data)

# Căn bậc 2 từng phần tử trong data
print('sqrt \n', np.sqrt(data))
```

```
data
 [1 2 3 4]
sqrt
 [1.         1.41421356 1.73205081 2.        ]
```

❖ Inner product

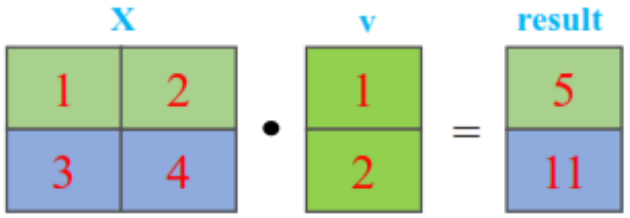


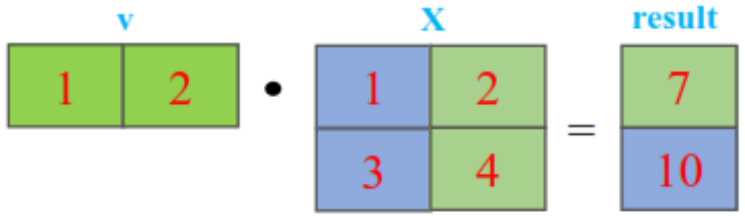

```python
# aivietnam.ai
import numpy as np

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

# Tính inner product giữa v và w
print('method 1 \n', v.dot(w))
print('method 2 \n', np.dot(v, w))
```

```
method 1
8

method 2
8
```

…Array Operations

❖ Vector-matrix multiplication

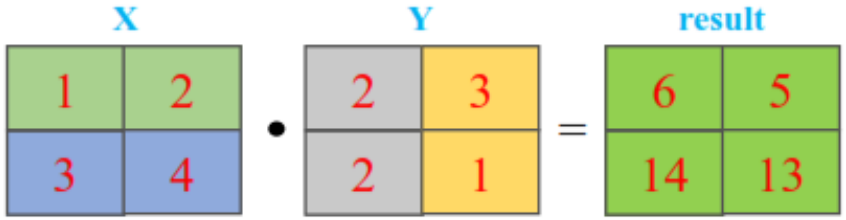


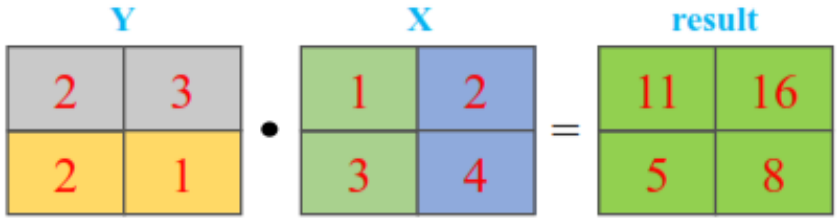



```python
# aivietnam.ai
import numpy as np

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

print('matrix X \n', X)
print('vector v \n', v)

# phép nhân giữa ma trận và vector
print('method 1: X.dot(v) \n', X.dot(v))
print('method 1: v.dot(X) \n', v.dot(X))
#print('\n method 2: X.dot(v) \n', np.dot(X, v))
#print('\n method 2: v.dot(X) \n', np.dot(v, X))
```

```
matrix X
 [[1 2]
 [3 4]]
vector v
 [1 2]
method 1: X.dot(v)
 [ 5 11]
method 1: v.dot(X)
 [ 7 10]
```

❖ Matrix-matrix multiplication

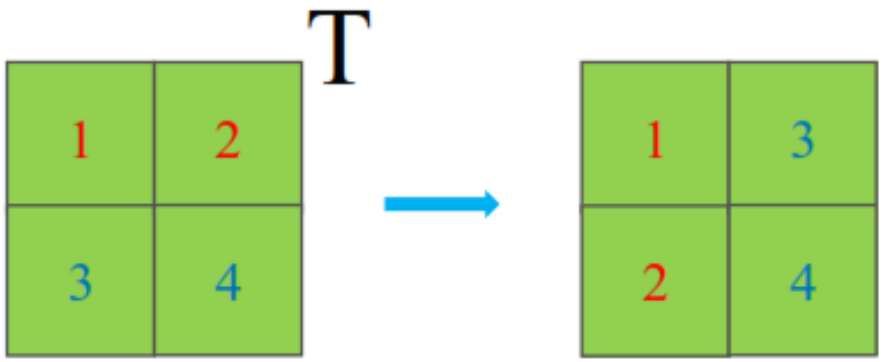





```python
# aivietnam.ai
import numpy as np

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

# Phép nhân giữa hai ma trận
print('method 1 \n', X.dot(Y))
print('method 1 \n', Y.dot(X))
#print('method 2 \n', np.dot(X, Y))
#print('method 2 \n', np.dot(Y, X))
```

```
method 1
 [[ 6  5]
 [14 13]]
method 1
 [[11 16]
 [ 5  8]]
```

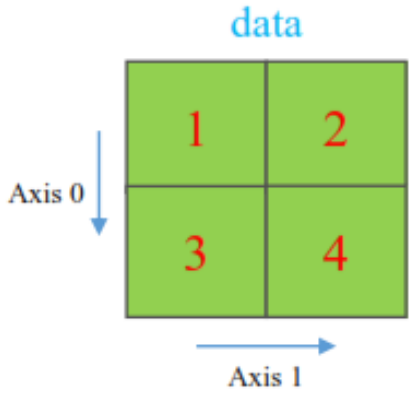
❖ Transpose




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

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

#chuyển vị
print(X.T)
```

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

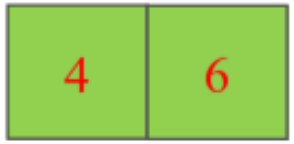
❖ Summation



sum(data)

10

sum(data, axis=0)



sum(data, axis=1)


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

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

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

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

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

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

…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

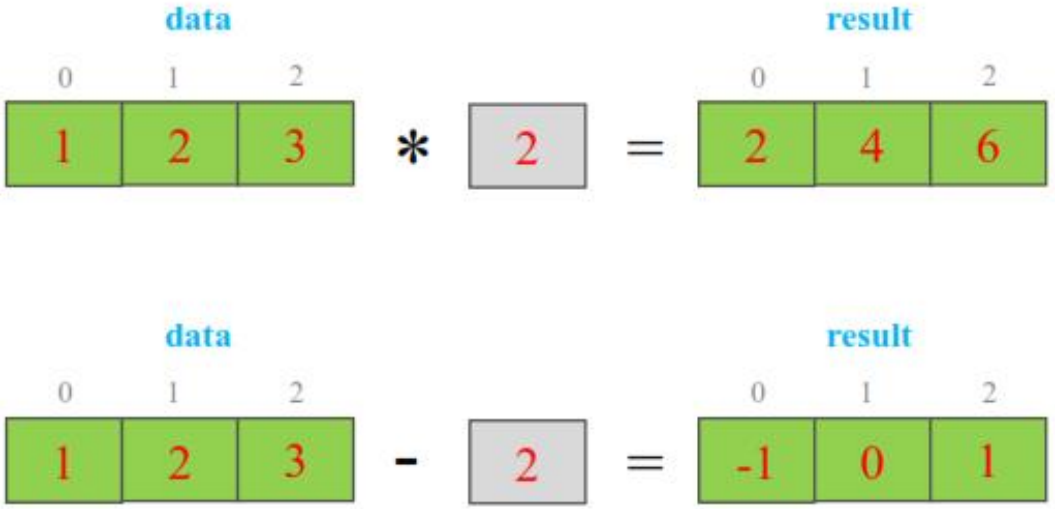
.min(axis=1) = | 1 | 3 | 5 |

CONTENTS

- Introduction to Numpy
- Numpy Array Indexing
- Numpy Array Operations
- **Broadcasting**
- Data Processing

Broadcasting

❖ Vector and a scalar

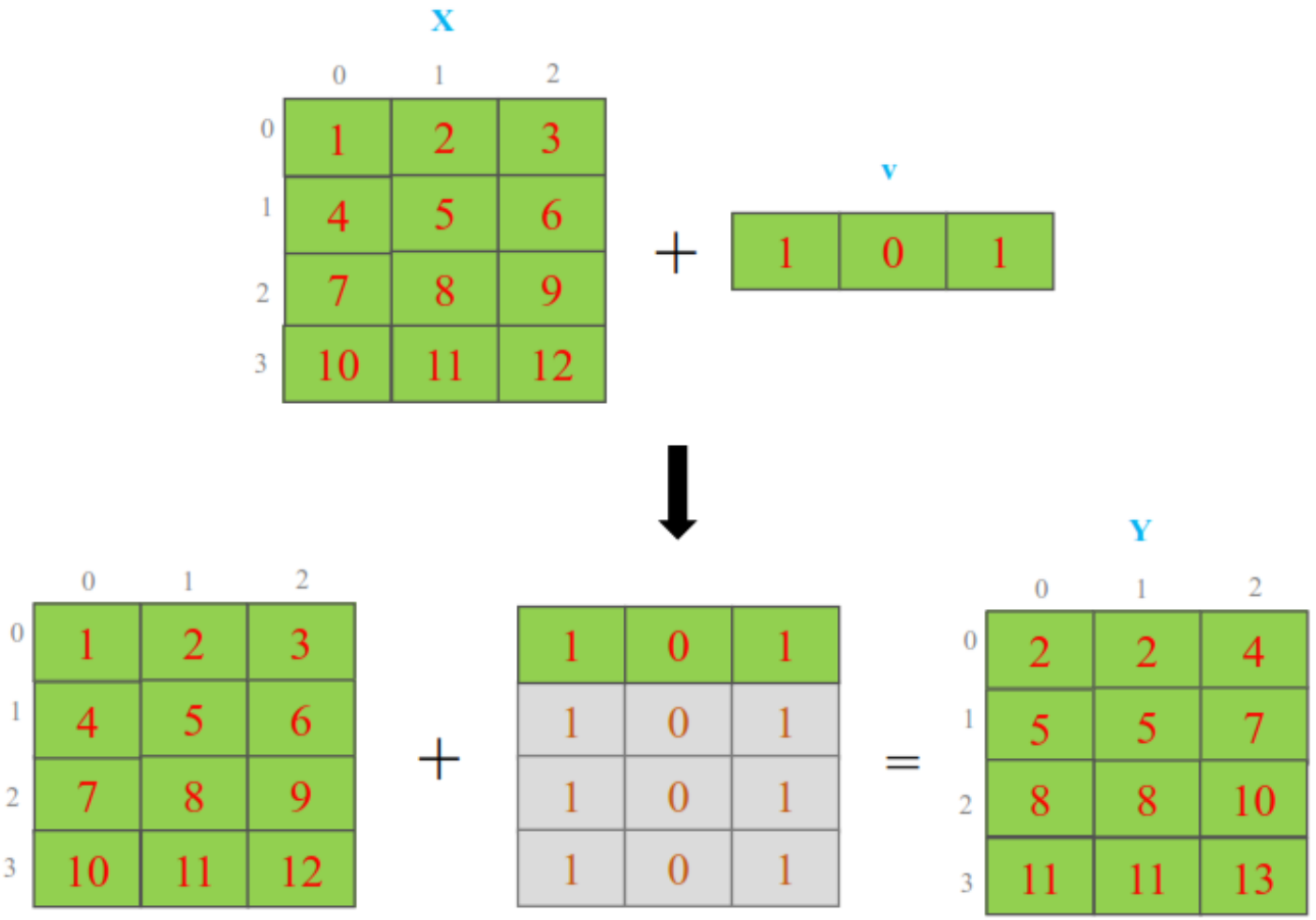



```python
# aivietnam.ai
import numpy as np

# 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




```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]]
```

CONTENTS

- Introduction to Numpy
- Numpy Array Indexing
- Numpy Array Operations
- Broadcasting
- **Data Processing**

Data Processing

- 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

…Data Processing

❖ Text data

❖ IRIS data

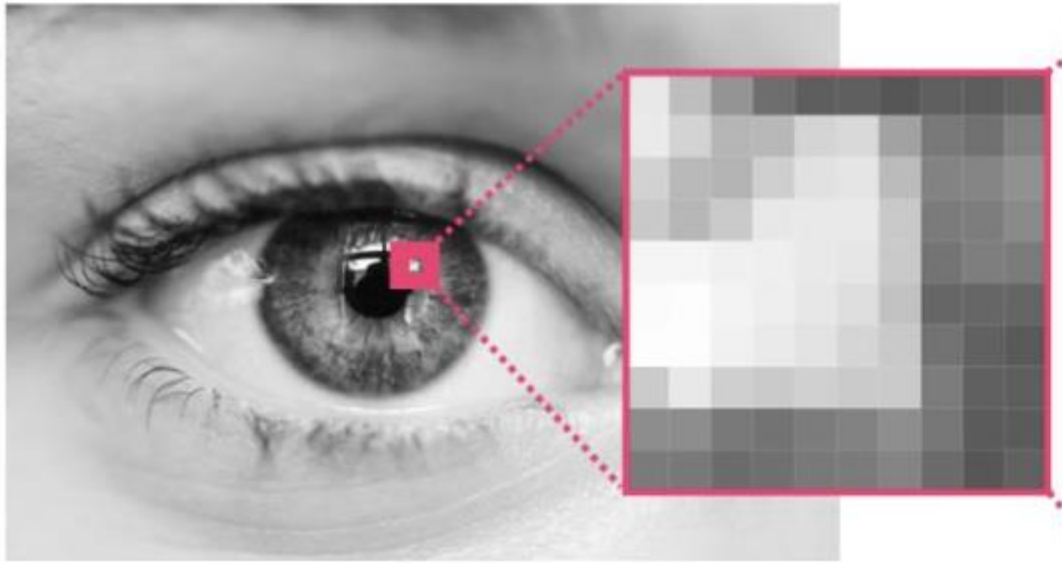
col 0	col 1	col 2	col 3	col 4
sepal_length	sepal_width	petal_length	petal_width	species

```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. 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.]
```

Image Data

❖ Grayscale images



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

(Height, Width)

Pixel p = scalar

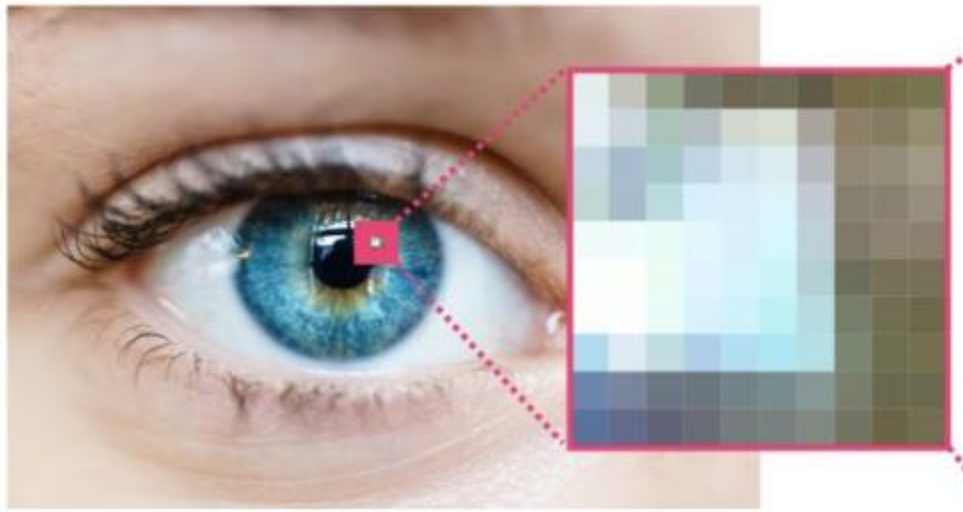
$0 \leq p \leq 255$

Resolution: #pixels

Resolution = HeightxWidth

Image Data

❖ Color images



	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		

(Height, Width, channel)

RGB color image

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

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

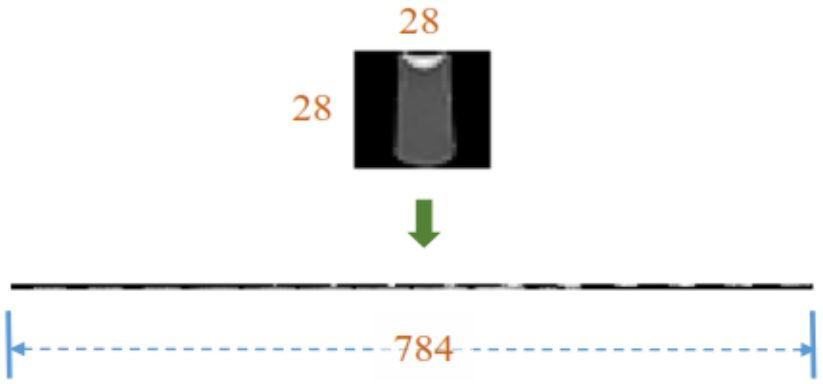
Resolution: #pixels
Resolution = HeightxWidth

http://jalammar.github.io/visual-numpy/

Image Classification

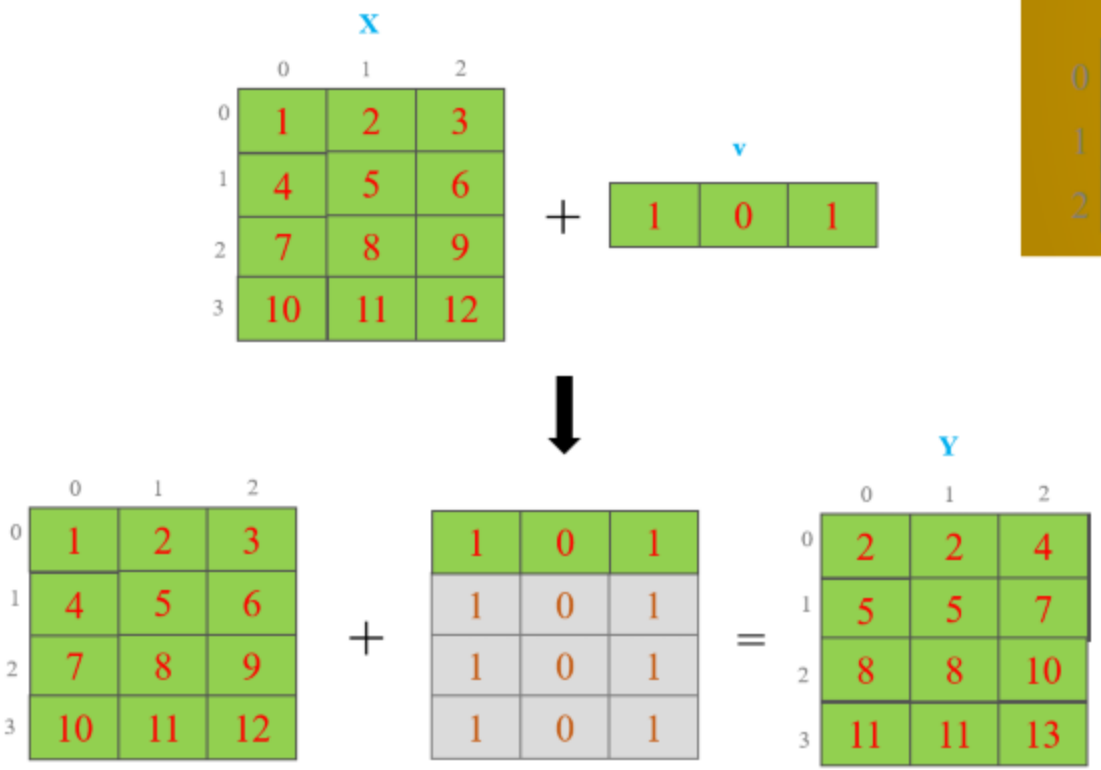
Fashion-MNIST dataset

- Grayscale images
- Resolution=28x28
- Training set: 60000 samples
- Testing set: 10000 samples

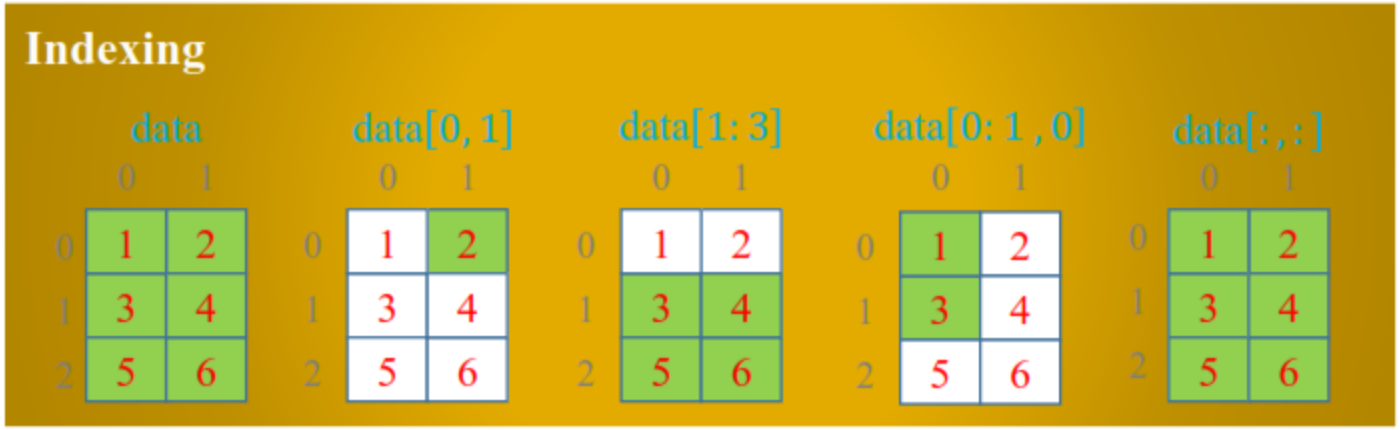


Summary

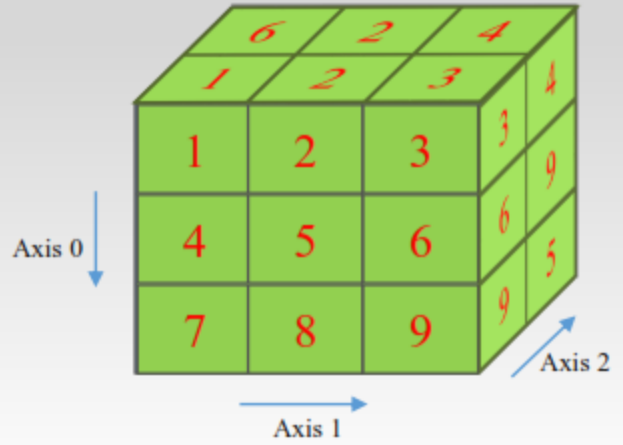
Broadcasting



Indexing

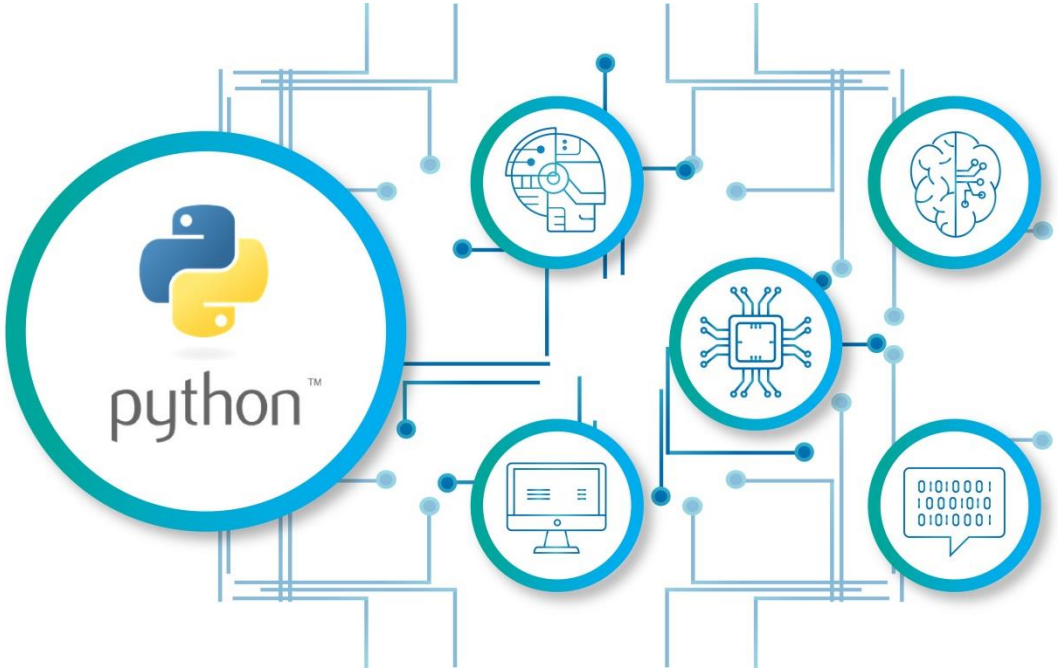


Multi-dimension Array



Numpy

Python for AI



Thank You…!