

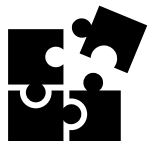
Python Review

Numpy ND Array

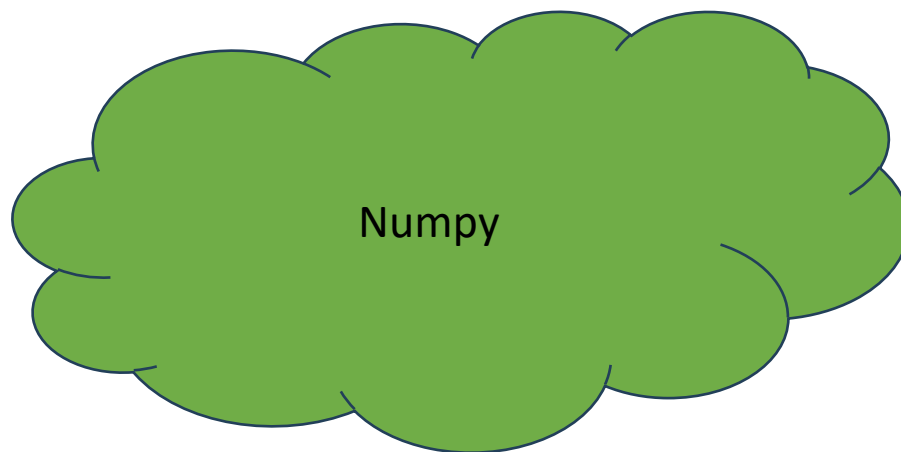
Outline

- **Review Numpy basic**
- **Creating Arrays**
- **Important Function**

Review-Numpy array



Game ôn tập kiến thức về numpy



1D array – Basic



Cả lớp thực hiện một bài tập đơn giản để ôn tập lại kiến thức!

Cho ma trận NumPy matrix có kích thước (4, 3) như sau:

```
np.array([[1, 5, 3],  
          [9, 2, 6],  
          [4, 8, 7],  
          [0, 3, 2]])
```

Hãy thực hiện các yêu cầu sau:

- Tính tổng của tất cả các phần tử trong ma trận và lưu kết quả vào biến `sum_of_matrix`.
- Tìm giá trị lớn nhất trong ma trận và lưu kết quả vào biến `max_value`.
- Tìm giá trị nhỏ nhất trong ma trận và lưu kết quả vào biến `min_value`.

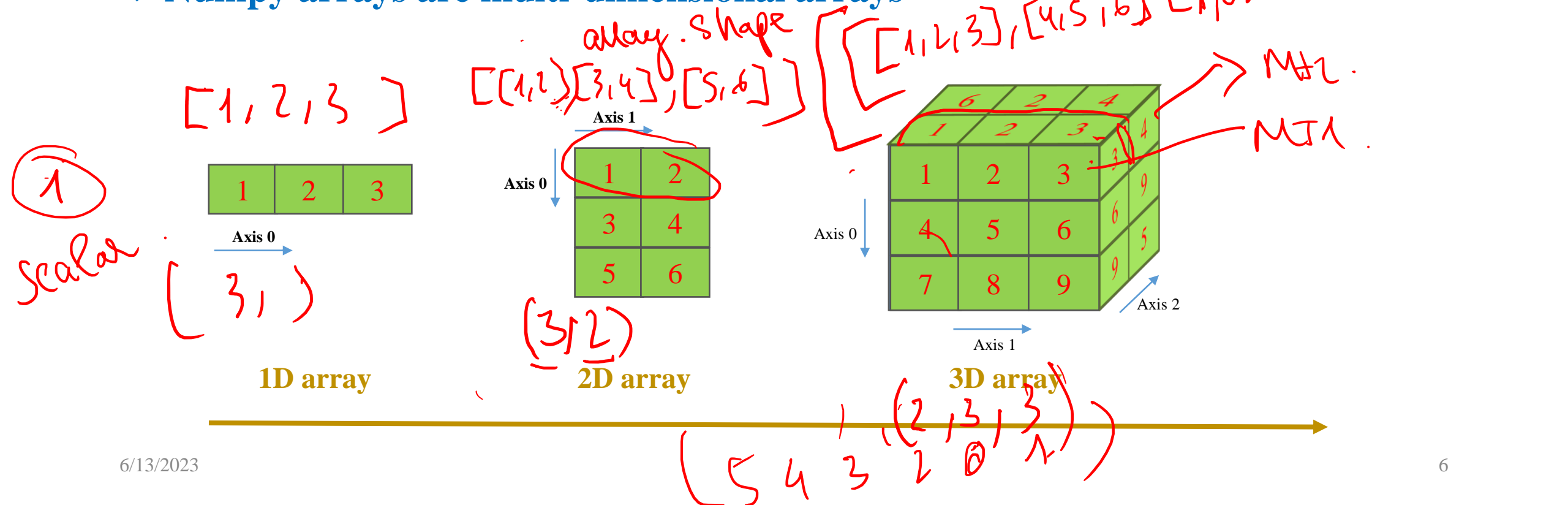
Outline

- **Review Numpy basic**
- **Creating Arrays**
- **Important Function**

Creating Arrays

- ❖ Numpy is a Python library
- ❖ For scientific computations
- ❖ Numpy array ↔ Tensor in Tensorflow and Pytorch
- ❖ Numpy arrays are multi-dimensional arrays

pip install numpy=python base



Creating Arrays

❖ Create Numpy array

❖ From List

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

data	
0	1
1	2
2	3

data[0]

1

data[1]

2

```
1 # aivietnam.ai
2 # tạo ndarray từ list
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
12 print(data)
13 print(data[0])
14 print(data[1])
```

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

Creating Arrays

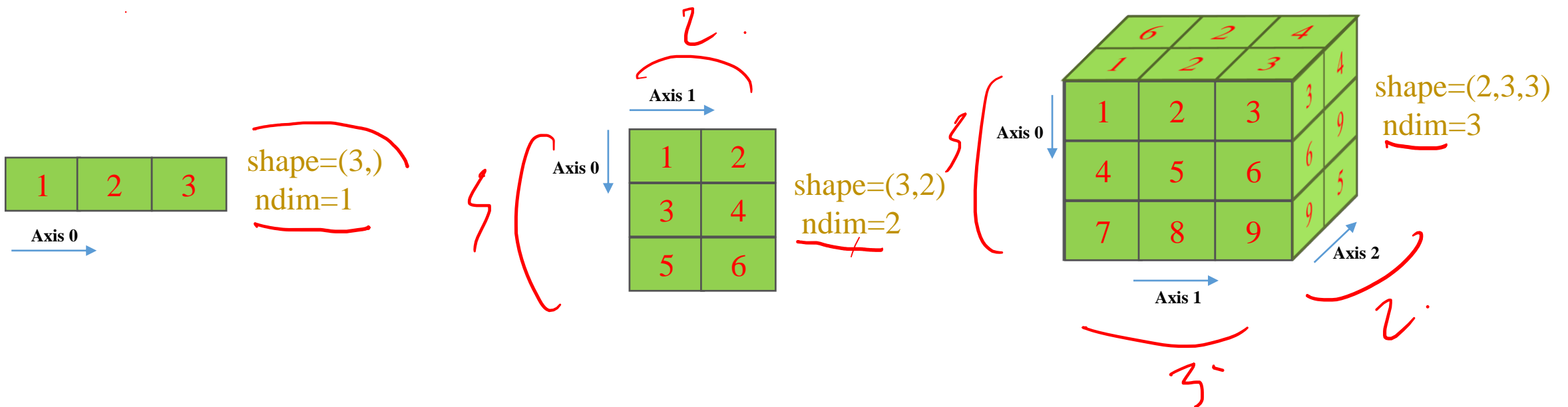
Bài tập 1: Bạn hãy xây dựng chương trình thực hiện:

1. Tạo một mảng 1D từ danh sách [1, 2, 3, 4, 5].
2. Tạo một mảng 2D kích thước 2x3 từ danh sách [[1, 2, 3], [4, 5, 6]].
3. Tạo một mảng 3D kích thước 2x2x3 từ danh sách [[[1, 2, 3], [4, 5, 6]],
[[7, 8, 9], [10, 11, 12]]].

Creating Arrays

❖ Common attributes

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



Creating Arrays

❖ Common attributes

❖ **dtype**: data type

❖ **shape**: return a tuple of #elements in each dimension

❖ **ndim**: return #dimensions

dtype example

```
1  # aivietnam.ai
2  # tạo ndarray từ list
3
4  import numpy as np
5
6  # tạo ndarray
7  data1 = np.array([1,2,3])
8  print(data1.dtype)
9
10 data2 = np.array([1.,2.,3.])
11 print(data2.dtype)
12
13 data3 = np.array([1,2,3], dtype=np.int64)
14 print(data3.dtype)
```

```
int32
float64
int64
```

Creating Arrays

❖ Viết chương trình thực hiện cập nhật 1 element trong array?

	index 0	1	2
data =	1	2	3

data[0] = 8

data =	8	2	3
--------	---	---	---

Creating Arrays

❖ Create Numpy array

`zeros((2, 3))`

2 (

	0	1	2
0	0	0	0
1	0	0	0

)

3.

`ones((2, 3))`

	0	1	2
0	1	1	1
1	1	1	1

`full((2, 3), 9)`

	0	1	2
0	9	9	9
1	9	9	9

Creating Arrays

❖ Create Numpy array

an 1D.

arange(start=0, stop=5, step=1)

arr1 =

	0	1	2	3	4
	0	1	2	3	4

(3,3)-

eye(3)

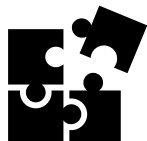
	0	1	2
0	1	0	0
1	0	1	0
2	0	0	1

Shape ←

random.random((2, 3))

	0	1	2
0	0.574	0.682	0.704
1	0.806	0.844	0.799

Creating Arrays



Game Initial Placeholders

Score:



Team 1
24

Tien +1. +2) 9
Duc +5

Admin +3.

Le Anh +2 +1.) 9

Quang +3.

Khôi +3

Deet +3

Bety +3

Score:



Team 2
12

Kien +1. +1. +1.

+3

Ngoc +3.

+3

Huong

ND array-Creating Arrays

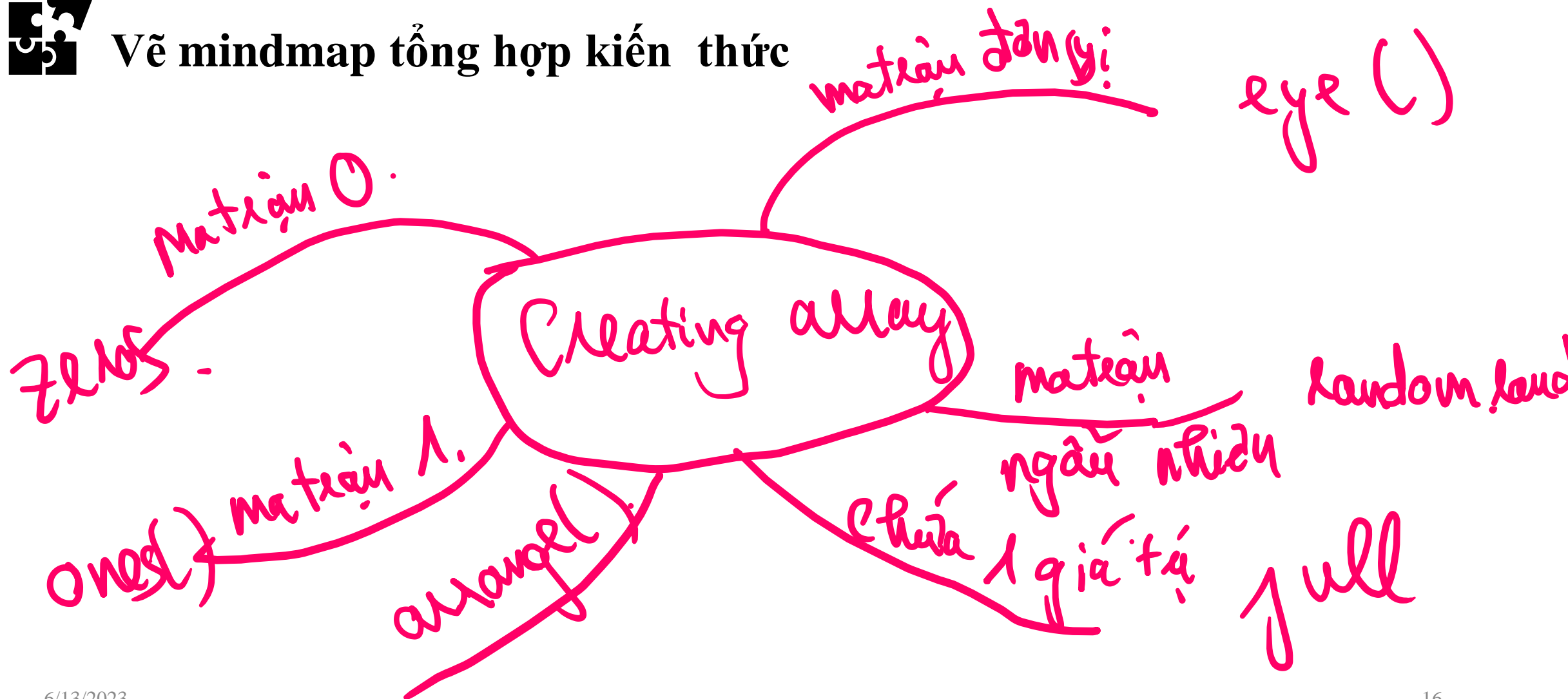
Bài tập 2: Initial Placeholders

1. Khởi tạo một mảng 2D kích thước 3x4 với tất cả các phần tử là 0 bằng phương thức **zeros()**
2. Khởi tạo một mảng 2D kích thước 3x4 với tất cả các phần tử là 1 bằng phương thức **ones()**
3. Khởi tạo một mảng 1D có giá trị từ 1 đến 10 bằng phương thức **arange()**
4. Khởi tạo một mảng 3D kích thước 2x3x2 với tất cả các phần tử là 7 bằng phương thức **full()**
5. Khởi tạo một ma trận đơn vị (identity matrix) kích thước 4x4 bằng phương thức **eye()**
6. Khởi tạo một mảng 2D kích thước 3x3 với các giá trị ngẫu nhiên trong khoảng từ 0 đến 1 bằng phương thức **random.random()**

ND array-Creating Arrays



Vẽ mindmap tổng hợp kiến thức



Outline

- **Review Numpy basic**
- **Creating Arrays**
- **Important Function**

Important Functions

❖ 1. clip()



➡ Hàm clip cắt giới hạn các giá trị trong một mảng vào một khoảng giá trị nhất định. /

Important Functions

Bài 1: Cho một mảng NumPy `arr = np.array([1, 5, 8, 10, 3, 6])`. Hãy cắt giới hạn các giá trị trong mảng vào khoảng từ 3 đến 8.

~~arr~~

`[1, 5, 8, 10, 3, 6]`

`[3, 5, 8, 8, 3, 6]`

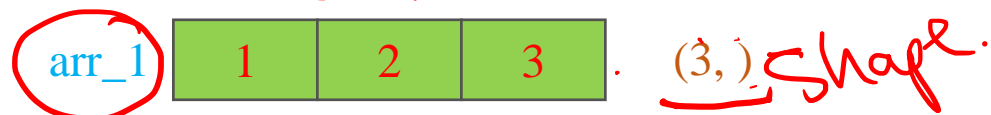
`np.clip(arr, a_min = 3, a_max = 8)`

Introduction

❖ 2. Concatenate

`numpy.concatenate()`

axis=0

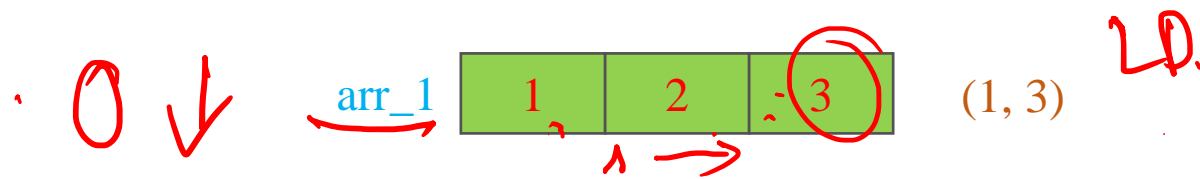


`concatenate((arr_1, arr_2), axis=0)`

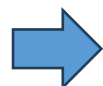
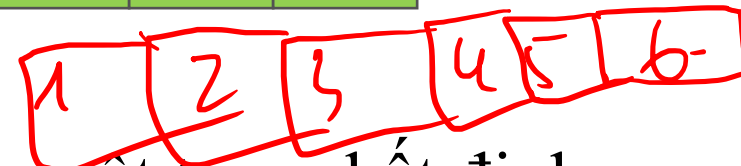
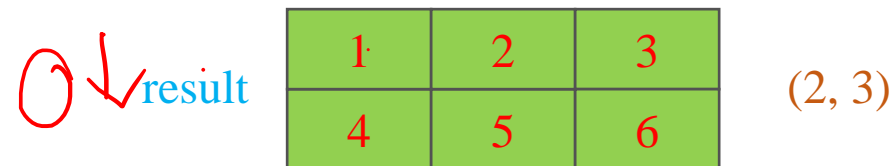


axis 0 ↓ 1 → 2 ↑

`numpy.concatenate()`



`concatenate((arr_1, arr_2), axis=0)`



Hàm concatenate nối các mảng lại với nhau theo một trục nhất định.

Introduction

❖ 2. Concatenate

numpy.concatenate()

arr_1

1	2	3
---	---	---

 (3,)


arr_2

4	5	6
---	---	---

 (3,)

axis = 0

`concatenate((arr_1, arr_2), axis=1)`

 **Error**

numpy.concatenate()

arr_1

1	2	3 ...
---	---	-------

 (1, 3)

arr_2

4	5	6
---	---	---

 (1, 3)

(1 x 6)

`concatenate((arr_1, arr_2), axis=1)`

result

1	2	3	4	5	6
---	---	---	---	---	---

 (1, 6)

➡ Hàm concatenate nối các mảng lại với nhau theo một trục nhất định.

Important Functions

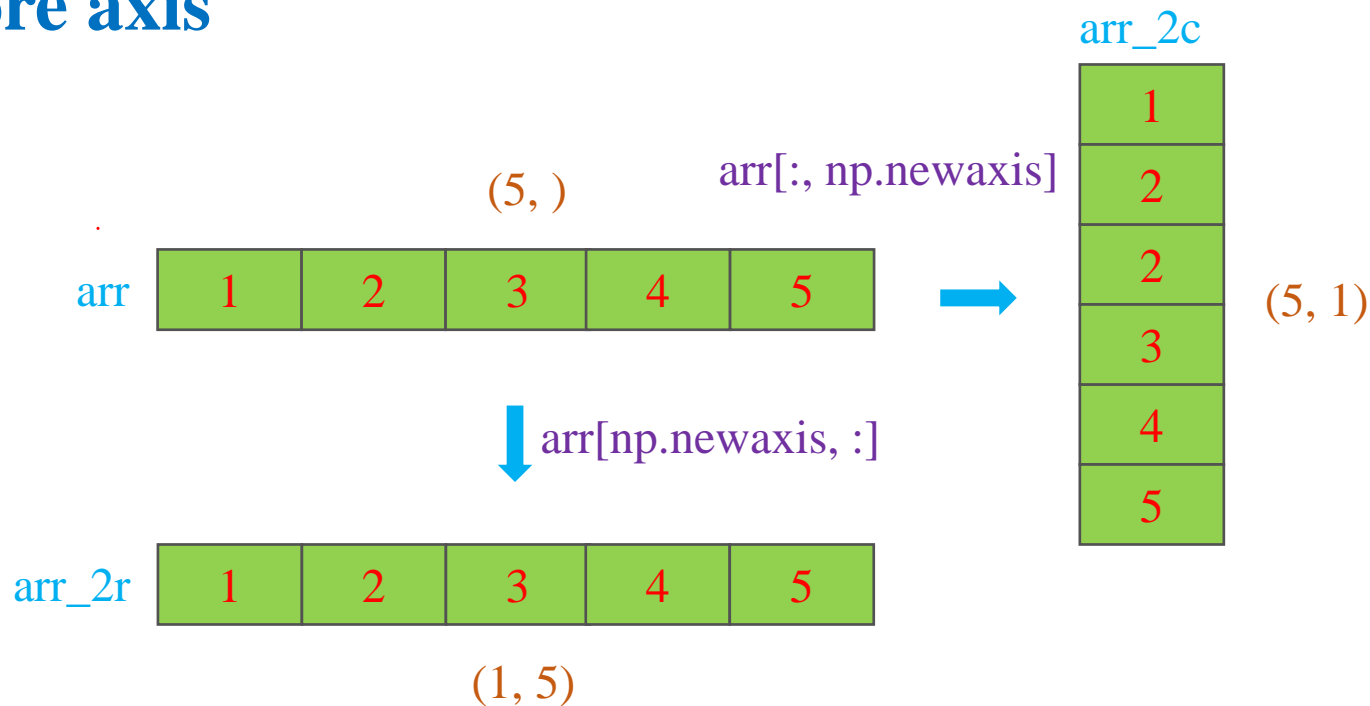
Bài 2: Cho hai mảng NumPy `arr1 = np.array([1, 2, 3])` và `arr2 = np.array([4, 5, 6])`. Hãy nối các mảng lại thành một mảng mới theo chiều ngang.

→
0

$arr1 = [1, 2, 3]$ $(3, 1)$
 $arr2 = [4, 5, 6]$ $(3, 1)$
 $[1, 2, 3, 4, 5, 6]$ $(6, 1)$

Introduction

❖ 3. Add more axis



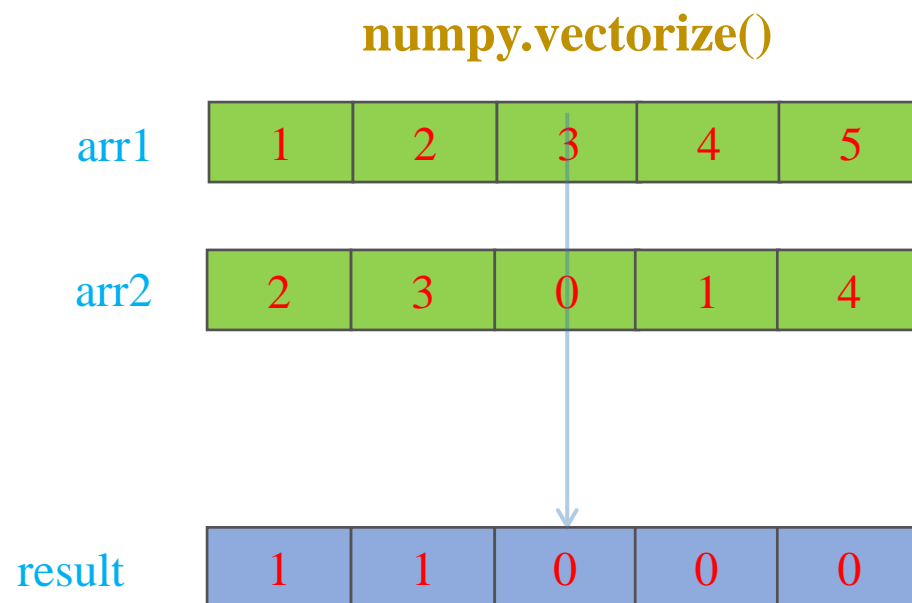
➡ `np.newaxis` được sử dụng để thêm một trục mới vào mảng, mở rộng số chiều của mảng.

Important Functions

Bài 3: Cho một mảng NumPy `arr = np.array([1, 2, 3, 4, 5])`. Hãy thêm một trục mới vào mảng.

Introduction

❖ 4. Vectorize()



arr1 < arr2 → 1.
arr1 > 2 → 0



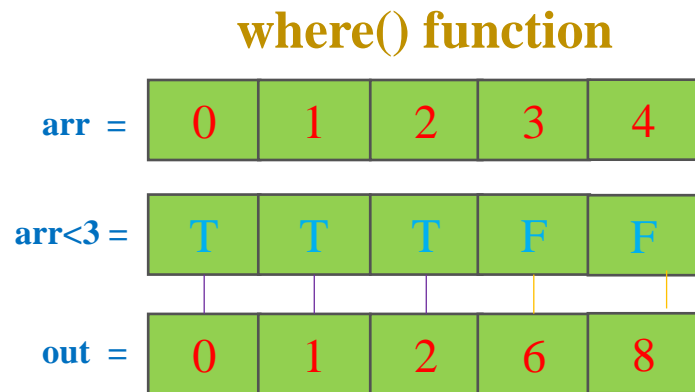
Hàm `vectorize` chuyển đổi một hàm không phải vector thành một hàm có thể áp dụng cho các mảng NumPy.

Important Functions

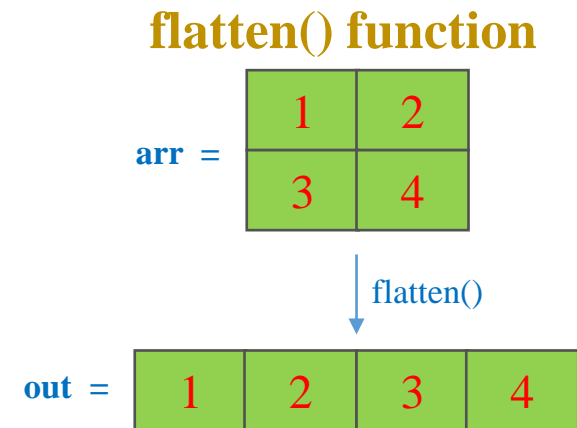
Bài 4: Viết một hàm `calculate_circle_area(radius)` nhận đầu vào là một mảng `radius` và trả về một mảng chứa diện tích các hình tròn tương ứng. Sử dụng hàm `vectorize` để tạo một phiên bản `vectorized` của hàm này và áp dụng nó cho một mảng `radius_values` chứa các bán kính `[1, 2, 3, 4]`.

Introduction

❖ 5. Where()



❖ 6. Flatten()



➡ Hàm `where` xác định vị trí của các phần tử trong mảng thỏa mãn một điều kiện nhất định.

Important Functions

Bài 5: Cho một mảng NumPy `arr = np.array([1, 2, 3, 4, 5])`. Sử dụng hàm `where` để tạo một mảng mới `new_arr`, trong đó các phần tử lớn hơn 3 được thay bằng 0 và các phần tử nhỏ hơn hoặc bằng 3 giữ nguyên giá trị.

Important Functions

Bài 6: Cho một mảng hai chiều `arr = np.array([[1, 2, 3], [4, 5, 6]])`. Sử dụng hàm `flatten` để biến đổi mảng này thành một mảng một chiều.

Introduction

❖ 7. reshape()

reshape() function

data			data_rs	
1	2	3	1	2
4	5	6	3	4
			5	6



Hàm reshape() thay đổi hình dạng của một mảng mà không làm thay đổi dữ liệu trong mảng đó

Important Functions

Bài 7: Cho một mảng một chiều `arr = np.array([1, 2, 3, 4, 5, 6])`. Sử dụng hàm `reshape` để biến đổi mảng này thành một mảng hai chiều có hình dạng `(2, 3)`.

Important Functions



Vẽ mindmap tổng hợp kiến thức

