

Học List và Numpy Qua Các Ví Dụ

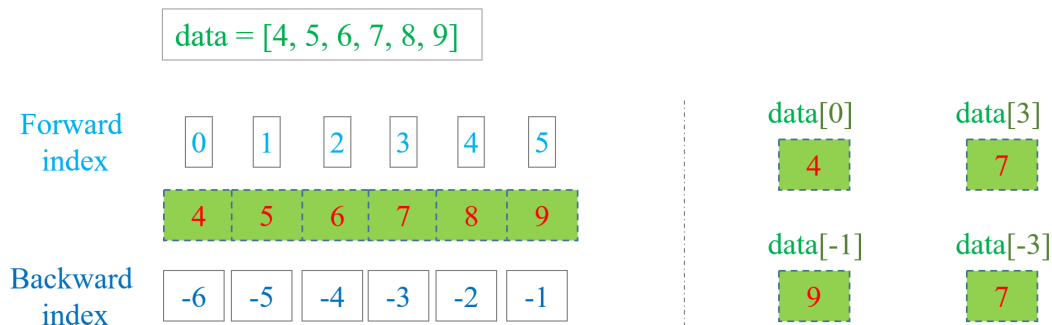
Đinh Quang Vinh

Ngày 9 tháng 7 năm 2024

List

1. Chỉ mục xuôi và chỉ mục ngược

Ví dụ cho một danh sách có tên là `data`, chứa sáu phần tử lần lượt là 4, 5, 6, 7, 8, và 9 như hình bên dưới. Mỗi phần tử trong danh sách được tiếp cận (lấy hay thay đổi giá trị) bằng thông tin chỉ mục (index) của nó. Thông tin chỉ mục bao gồm thông tin chỉ mục xuôi (từ trái sang phải) và thông tin chỉ mục ngược (từ phải sang trái). Khi từ "chỉ mục" được đề cập mà không nói rõ là chỉ mục xuôi hay chỉ mục ngược, chúng ta thường ngầm hiểu đó là chỉ mục xuôi.



Ví dụ 1: Chỉ mục xuôi và chỉ mục ngược của một danh sách.

Giá trị của chỉ mục xuôi là những số nguyên, bắt đầu từ 0 và tăng với đơn vị là 1. Với danh sách `data` có sáu phần tử, giá trị chỉ mục xuôi lần lượt là 0, 1, 2, 3, 4, và 5. Chúng ta có thể nhầm giá trị chỉ mục xuôi cho phần tử đầu tiên là 0 và phần tử cuối cùng của danh sách là $6 - 1 = 5$.

Thông tin chỉ mục ngược bao gồm những giá trị nguyên âm, từ phải sang trái, giá trị bắt đầu từ -1 và giảm dần với đơn vị là 1. Cụ thể ở danh sách `data`, chỉ mục ngược cho các phần tử từ phải sang trái là -1, -2, -3, -4, -5, và -6. Giá trị chỉ mục ngược -1 được sử dụng phổ biến nhất để lấy phần tử cuối cùng của danh sách.

Đoạn code bên dưới minh họa việc tạo một danh sách và lấy các phần tử của danh sách dùng giá trị chỉ mục xuôi và chỉ mục ngược.

```
1 # Python code
2 data = [4, 5, 6, 7, 8, 9]
3 print(data[0])
4 print(data[3])
5 print(data[-1])
6 print(data[-3])
```

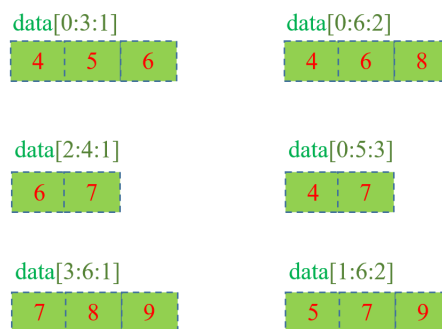
```
===== Output =====
4
7
9
7
=====
```

2. Kỹ thuật slicing

Kỹ thuật slicing cho phép thao tác với nhiều phần tử trong danh sách. Cú pháp của kỹ thuật slicing là tên-danh-sách[start:end:step], trong đó start, end, và step cách nhau bằng dấu hai chấm (:) và có giá trị mặc định lần lượt là 0, độ dài của danh sách len(.), và 1. Cú pháp trên có ý nghĩa là lấy các phần tử có chỉ mục bắt đầu từ start đến chỉ mục (end-1) với giá trị tăng lên là step. Khi Xét ví dụ 2 với một danh sách data chứa sáu phần tử và mỗi phần tử có một giá trị chỉ mục. Ba phần tử đầu tiên của danh sách được lấy theo kỹ thuật slicing là data[0:3:1], trong đó start=0, end=3, và step=1. data[0:3:1] sẽ lấy các phần tử theo thứ tự có chỉ mục là 0, 1, và 2. Ở đây, phần tử với chỉ mục là 3 sẽ không được lấy vì slicing chỉ lấy phần tử tới chỉ mục (end-1).

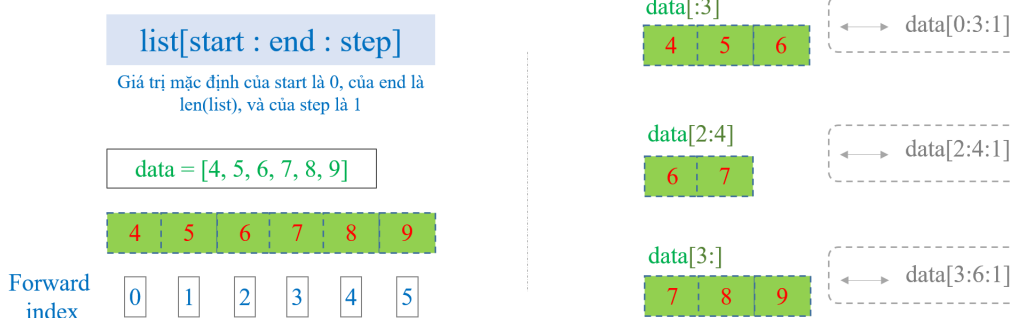
```
1 # Python code
2 data = [4, 5, 6, 7, 8, 9]
3 print(data[0:3:1])
4 print(data[2:4:1])
5 print(data[3:6:1])
6 print(data[0:6:2])
7 print(data[0:5:3])
8 print(data[1:6:2])
```

```
===== Output =====
[4, 5, 6]
[6, 7]
[7, 8, 9]
[4, 6, 8]
[4, 7]
[5, 7, 9]
=====
```



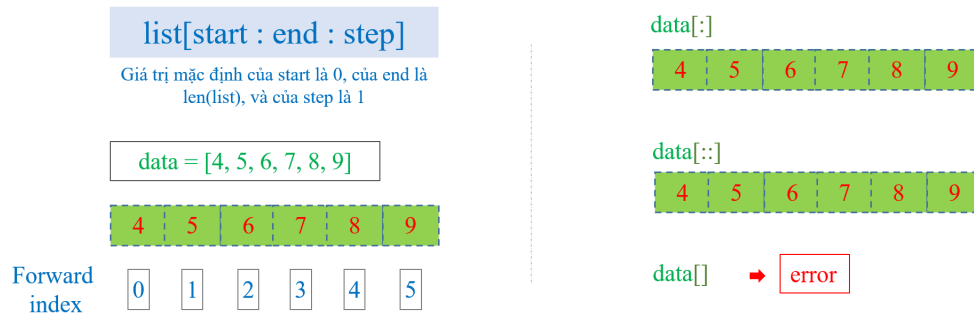
Ví dụ 2: Kỹ thuật slicing trong Python.

Mặc dù chúng ta thấy có ba thông tin start, end, và step ở kỹ thuật slicing, ba thông tin này có thể không cần khai báo như ở ví dụ 3. Khi chỉ có một dấu hai chấm (:) được dùng, lúc này Python hiểu là đây là dấu hai chấm đầu tiên trong cú pháp slicing. Cụ thể, data[2:4] nghĩa là start=2, end=4, và step nhận giá trị mặc định là 1. Một trường hợp khác, data[3:] nghĩa là start=2, và end=4 và step nhận giá trị mặc định là 6 và 1.



Ví dụ 3: Các giá trị start, end và step có thể không khai báo trong kỹ thuật slicing.

Ba giá trị start, end và step có thể không khai báo khi dùng slicing như ở ví dụ 4. Lúc này start, end và step sẽ nhận các giá trị mặc định là 0, len(list) và 1. Như đã thảo luận, chúng ta thậm chí có thể không khai báo đầy đủ hai dấu hai chấm như `data[:]` chỉ khai báo một dấu hai chấm, và bỏ đi start, end, stop, và một dấu hai chấm. Tuy nhiên, nếu chúng ta bỏ đi luôn cả hai dấu hai chấm như `data[]`, chương trình sẽ báo lỗi cú pháp không hợp lệ.



Ví dụ 4: Chỉ dùng dấu hai chấm và không khai báo start, end và step trong kỹ thuật slicing.

Trong thực tế, dù kỹ thuật slicing cho phép sử dụng một cách linh động. Chúng ta vẫn nên khai báo đầy đủ thông tin để code tường minh hơn và rõ ràng hơn. Hai đoạn code 3 và 4 trình bày code Python và kết quả của 2 chương trình.

```
1 # Python code 3
2 data = [4, 5, 6, 7, 8, 9]
3 print(data[:3])
4 print(data[2:4])
5 print(data[3:])
6 print(data[:])
7 print(data[::])
```

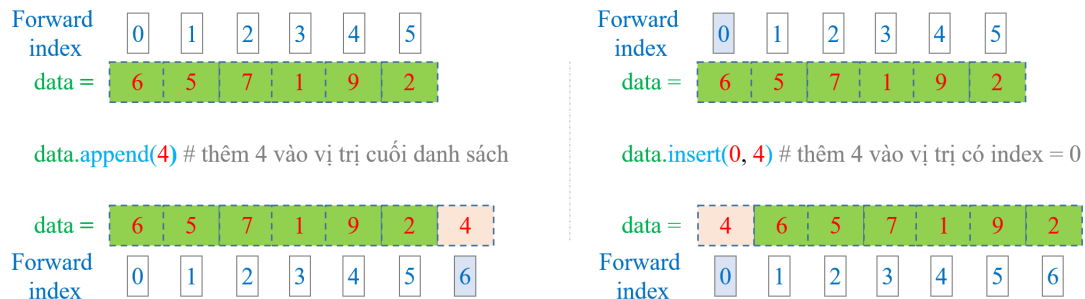
```
===== Output =====
[4, 5, 6]
[6, 7]
[7, 8, 9]
[4, 5, 6, 7, 8, 9]
[4, 5, 6, 7, 8, 9]
=====
```

```
1 # Python code 4
2 data = [4, 5, 6, 7, 8, 9]
3 print(data[])
4
5
6 #=====
```

```
===== Output =====
Cell In [.] , line 3
    print(data[])
           ^
SyntaxError: invalid syntax
=====
```

3. Thêm phần tử vào danh sách

Chúng ta có thể thêm một hoặc nhiều phần tử vào một danh sách. Hàm `append(element)` cho phép thêm phần tử `element` vào cuối danh sách. Hàm `insert(index, element)` chèn phần tử `element` vào vị trí có chỉ mục là `index`. Ví dụ 3.1 minh họa việc thêm một phần tử vào một danh sách dùng hàm `append(element)` và hàm `insert(index, element)`. Code 3.1 và 3.2 cài đặt việc thêm phần tử vào một danh sách.



Ví dụ 3.1: Thêm một phần tử vào một danh sách.

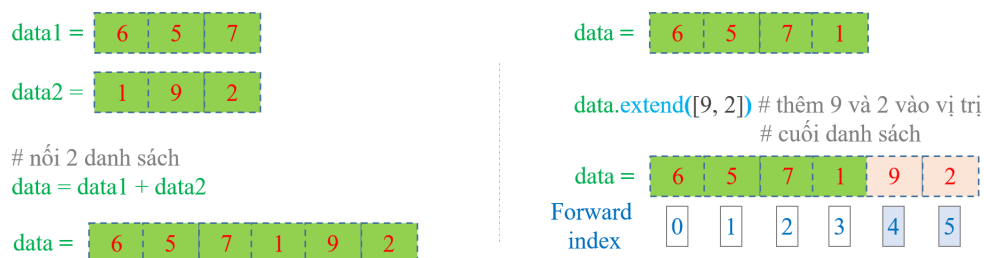
```
1 # Python code 3.1
2 data = [6, 5, 7, 1, 9, 2]
3 print(data)
4 data.append(4)
5 print(data)
```

```
===== Output =====
[6, 5, 7, 1, 9, 2]
[6, 5, 7, 1, 9, 2, 4]
=====
```

```
1 # Python code 3.2
2 data = [6, 5, 7, 1, 9, 2]
3 print(data)
4 data.insert(0, 4)
5 print(data)
```

```
===== Output =====
[6, 5, 7, 1, 9, 2]
[4, 6, 5, 7, 1, 9, 2]
=====
```

Ngoài ra, Python cho phép thêm nhiều phần tử vào một danh sách hay nối hai danh sách vào thành một. Hàm `extend(a_list)` thêm các phần tử trong `a_list` vào cuối một danh sách. Chúng ta dùng dấu cộng (+) theo cú pháp `list_1 + list_2` để nối hai danh sách `list_1` và `list_2`. Ví dụ 3.2 minh họa việc thêm nhiều phần tử vào một danh sách và việc nối hai danh sách vào thành một. Code 3.3 và 3.4 trình bày phần cài đặt cho ví dụ 3.2.



Ví dụ 3.2: Thêm nhiều phần tử vào một danh sách và nối hai danh sách.

```
1 # Python code 3.3
2 data = [6, 5, 7, 1]
3 print(data)
4 data.extend([9, 2])
5 print(data)
```

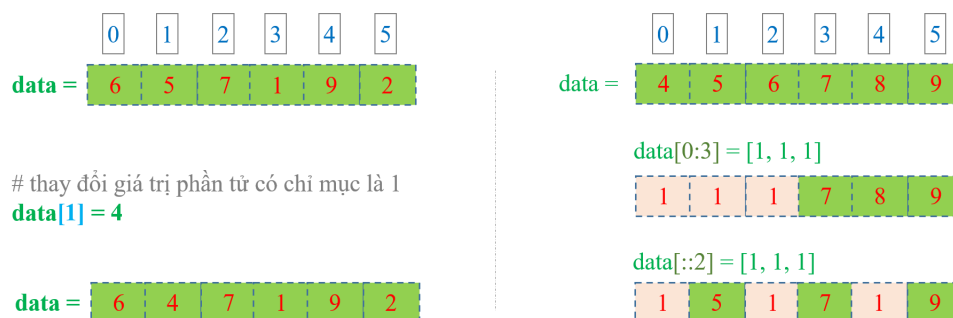
```
===== Output =====
[6, 5, 7, 1]
[6, 5, 7, 1, 9, 2]
=====
```

```
1 # Python code 3.4
2 data = [6, 5, 7] + [1, 9, 2]
3 print(data)
```

```
===== Output =====
[6, 5, 7, 1, 9, 2]
=====
```

4. Cập nhật phần tử trong danh sách

Một hay nhiều phần tử của một danh sách có thể được thay đổi giá trị thông qua giá trị chỉ mục, như ở ví dụ 4. `data[1] = 4` nghĩa là cập nhật phần tử ở chỉ mục là 1 với giá trị là 4. Kỹ thuật slicing có thể được sử dụng để cập nhật nhiều phần tử trong một danh sách. Ví dụ `data[::2]` tiếp cận ba phần tử ở chỉ mục 0, 2, và 4; do đó, `data[::2] = [1, 1, 1]` cập nhật giá trị 1 cho ba phần tử ở chỉ mục 0, 2, và 4. Code 4.1, 4.2 và 4.3 trình bày cài đặt Python cho việc cập nhật một và nhiều phần tử trong một danh sách.



Ví dụ 4: Thêm nhiều phần tử vào một danh sách và nối hai danh sách.

```
1 # Python code 4.1
2 data = [6, 5, 7, 1, 9, 2]
3 print(data)
4 data[1] = 4
5 print(data)
```

```
===== Output =====
[6, 5, 7, 1, 9, 2]
[6, 4, 7, 1, 9, 2]
=====
```

```
1 # Python code 4.2
2 data = [4, 5, 6, 7, 8, 9]
3 print(data)
4 data[0:3] = [1, 1, 1]
5 print(data)
```

```
===== Output =====
[4, 5, 6, 7, 8, 9]
[1, 1, 1, 7, 8, 9]
=====
```

```
1 # Python code 4.3
2 data = [4, 5, 6, 7, 8, 9]
3 print(data)
4 data[::2] = [1, 1, 1]
5 print(data)
```


```
===== Output =====
[4, 5, 6, 7, 8, 9]
[1, 5, 1, 7, 1, 9]
=====
```

5. Xóa phần tử khỏi danh sách

`data =` 

`data.pop(2)` # tại vị trí index = 2

`data =` 

`data =` 

`data.remove(5)` # xóa phần tử đầu tiên
có giá trị là 5

`data =` 

Ví dụ 5: Xóa phần tử khỏi danh sách dùng hàm `pop()` và `remove()`.

`data =` 

xóa phần tử tại chỉ mục 1 và 2
`del data[1:3]`

`data =` 

`data =` 

xóa tất cả phần tử trong danh sách
`data.clear()`

`data = []`

Ví dụ 6: Xóa phần tử khỏi danh sách dùng hàm `clear()` và từ khóa `del`.

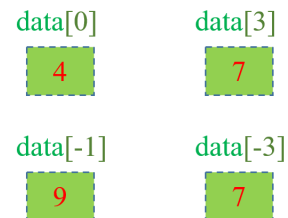
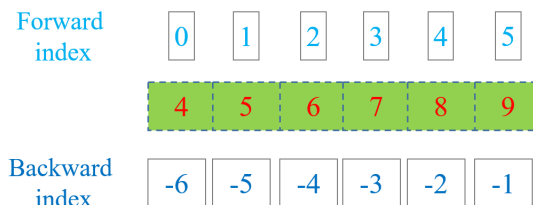
Numpy

Câu 1: Import gói Numpy và in ra phiên bản của nó

```
1 # Python code (aivietnam)
2 import numpy as np
3 print(np.version.version)
```

```
===== Output =====
1.25.0
=====
```

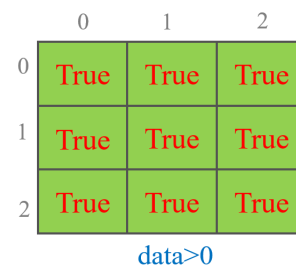
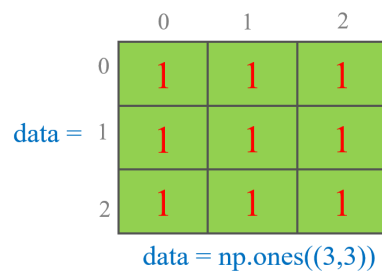
Câu 2: Tạo mảng một chiều từ 4 đến 9



```
1 # Python code (aivietnam)
2 import numpy as np
3
4 data = np.arange(4, 10)
5 print(data)
```

```
===== Output =====
[4 5 6 7 8 9]
=====
```

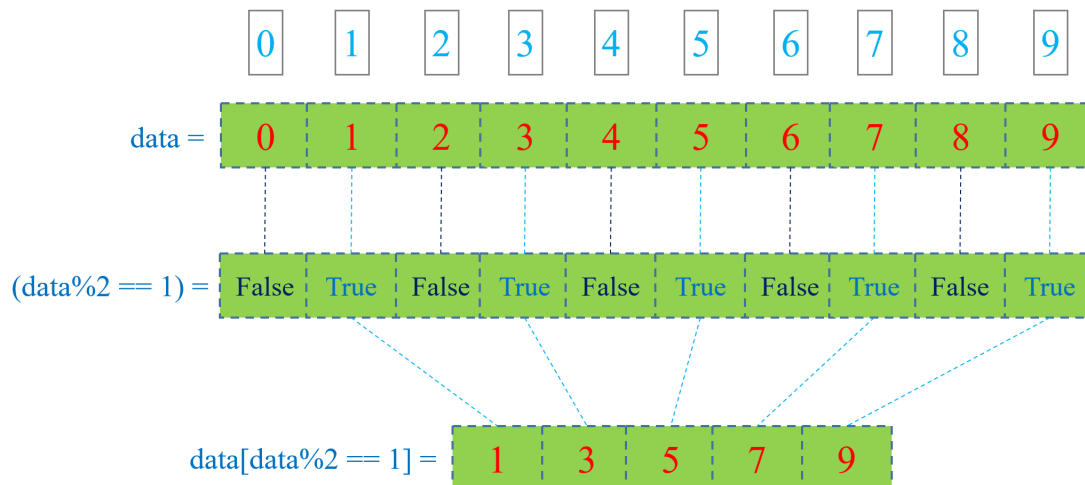
Câu 3: Tạo một mảng boolean 3x3 với tất cả giá trị là True



```
1 # Python code (aivietnam)
2 import numpy as np
3
4 # way 1
5 data1 = np.ones((3,3)) > 0
6 print(f'{data1} \n-----')
7
8 # way 2
9 data2 = np.ones((3,3), dtype=bool)
10 print(f'{data2} \n-----')
11
12 # way 3
13 data3 = np.full((3,3), True, dtype=bool)
14 print(f'{data3} \n-----')
```

```
===== Output =====
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]]
-----
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]]
-----
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]]
-----
=====
```


Câu 4: Lấy những phần tử mà thoả mãn một điều kiện cho trước của mảng một chiều



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # create an ndarray from 0 to 9
5 data = np.arange(0, 10)
6 print(data)
7
8 # Find odd numbers
9 data_odd = data[data%2 == 1]
10 print(data_odd)

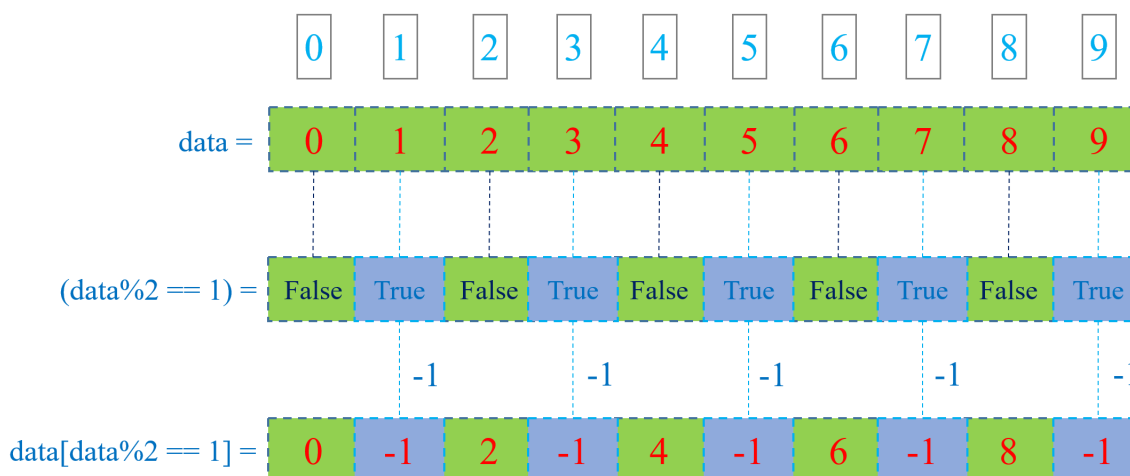
```

```

===== Output =====
[0 1 2 3 4 5 6 7 8 9]
[1 3 5 7 9]
=====

```

Câu 5: Thay thế phần tử thoả mãn điều kiện cho trước bằng một giá khác



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # create an ndarray from 0 to 9
5 data = np.arange(0, 10)
6 print(data)
7
8 # replace odd numbers by -1

```

```

9 data[data%2 == 1] = -1
10 print(data)

```

```

===== Output =====
[0 1 2 3 4 5 6 7 8 9]
[ 0 -1  2 -1  4 -1  6 -1  8 -1]
=====

```

```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # create an ndarray from 0 to 9
5 data = np.arange(0, 10)
6 print(data)
7
8 # replace odd numbers by -1
9 out = np.where(data%2 == 1, -1, arr)
10 print(out)

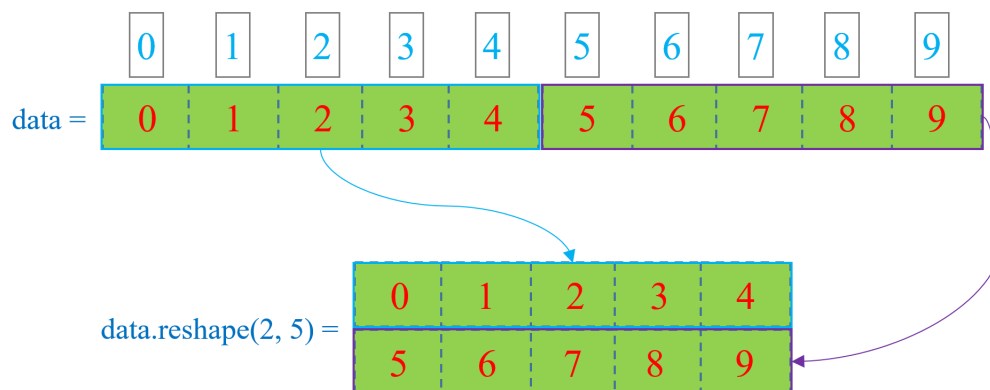
```

```

===== Output =====
[0 1 2 3 4 5 6 7 8 9]
[0 -1  2 -1  4 -1  6 -1  8 -1]
=====

```

Câu 6: Chuyển định dạng (shape) của một ndarray. Chuyển mảng một chiều thành mảng hai chiều



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # create a 1D ndarray from 0 to 9
5 data = np.arange(10)
6 print(data)
7
8 # reshape data to 2 rows and 5 columns
9 data_2d = data.reshape(2, 5)
10 print(data_2d)

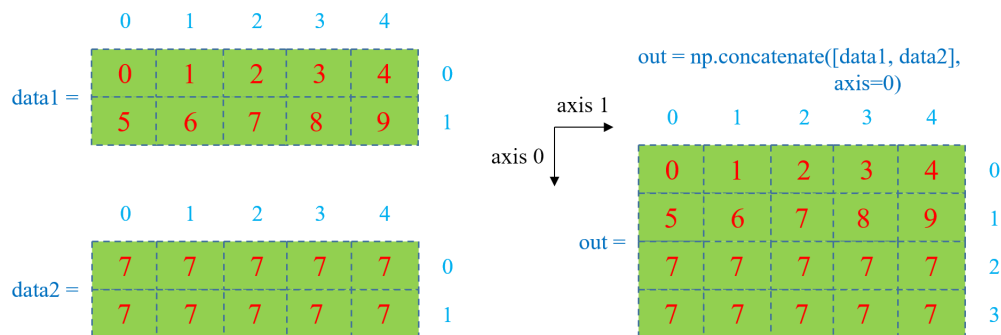
```

```

===== Output =====
[0 1 2 3 4 5 6 7 8 9]
[[0 1 2 3 4]
 [5 6 7 8 9]]
=====

```

Câu 7: Xếp chồng 2 mảng theo chiều dọc



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 data1 = np.arange(10).reshape(2, -1)
5 print(data1)
6
7 data2 = np.repeat(7, 10).reshape(2, -1)
8 print(data2)
9
10 # Way 1 :
11 out1 = np.concatenate([data1, data2],
12                        axis=0)
13 print(out1)
14
15 # Way 2 :
16 out2 = np.vstack([data1, data2])
17 print(out2)
18
19 # Way 3 :
20 out3 = np.r_[data1, data2]
21 print(out3)

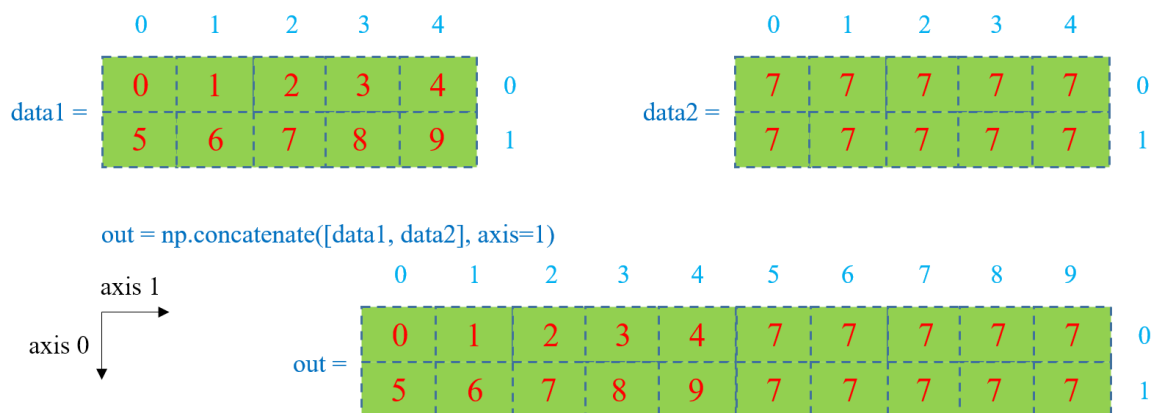
```

```

===== Output =====
[[0 1 2 3 4]
 [5 6 7 8 9]]
[[7 7 7 7 7]
 [7 7 7 7 7]]
[[0 1 2 3 4]
 [5 6 7 8 9]
 [7 7 7 7 7]
 [7 7 7 7 7]]
[[0 1 2 3 4]
 [5 6 7 8 9]
 [7 7 7 7 7]
 [7 7 7 7 7]]
[[0 1 2 3 4]
 [5 6 7 8 9]
 [7 7 7 7 7]
 [7 7 7 7 7]]
=====

```

Câu 8: Xếp chồng 2 mảng theo chiều ngang



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 data1 = np.arange(10).reshape(2, -1)
5 print(data1)
6
7 data2 = np.repeat(7, 10).reshape(2, -1)
8 print(data2)
9
10 # Way 1
11 out1 = np.concatenate([data1, data2],
12                        axis=1)
13 print(out1)
14
15 # Way 2
16 out2 = np.hstack([data1, data2])
17 print(out2)
18
19 # Way 3
20 out3 = np.c_[data1, data2]
21 print(out3)

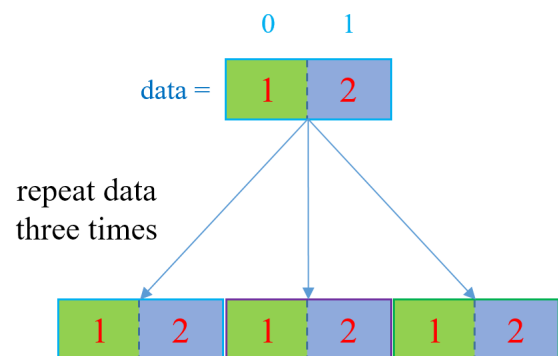
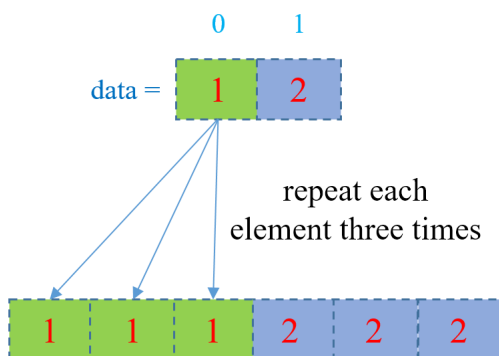
```

```

===== Output =====
[[0 1 2 3 4]
 [5 6 7 8 9]]
[[7 7 7 7 7]
 [7 7 7 7 7]]
[[0 1 2 3 4 7 7 7 7 7]
 [5 6 7 8 9 7 7 7 7 7]]
[[0 1 2 3 4 7 7 7 7 7]
 [5 6 7 8 9 7 7 7 7 7]]
[[0 1 2 3 4 7 7 7 7 7]
 [5 6 7 8 9 7 7 7 7 7]]
=====

```

Câu 9: Lặp data với repeat() và tile()



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 data = np.array([1, 2])
5 print(data)
6
7 # repeat each element three times
8 out1 = np.repeat(data, 3)
9 print(out1)
10
11 # repeat data three times
12 out2 = np.tile(data, 3)
13 print(out2)

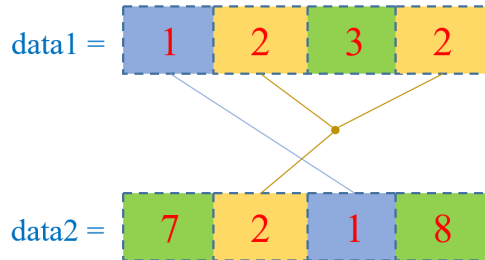
```

```

===== Output =====
[1 2]
[1 1 1 2 2 2]
[1 2 1 2 1 2]
=====

```

Câu 10: Lấy phần tử chung của 2 mảng



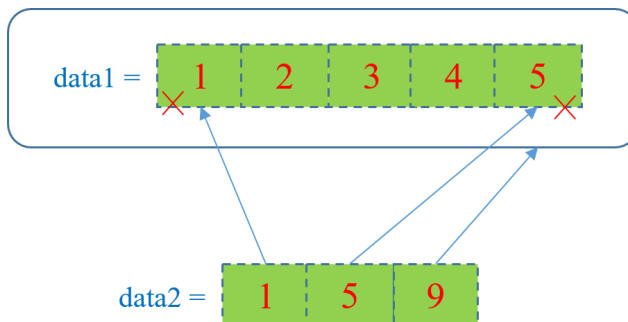
```
out = np.intersect1d(data1, data2)
```

```
out = [1, 2]
```

```
1 # Python code (aivietnam)
2 import numpy as np
3
4 data1 = np.array([1, 2, 3, 2])
5 data2 = np.array([7, 2, 1, 8])
6 print(data1)
7 print(data2)
8
9 out = np.intersect1d(data1, data2)
10 print(out)
```

```
===== Output =====
[1 2 3 2]
[7 2 1 8]
[1 2]
=====
```

Câu 11: Xóa phần tử từ một mảng mà tồn tại trong một mảng khác



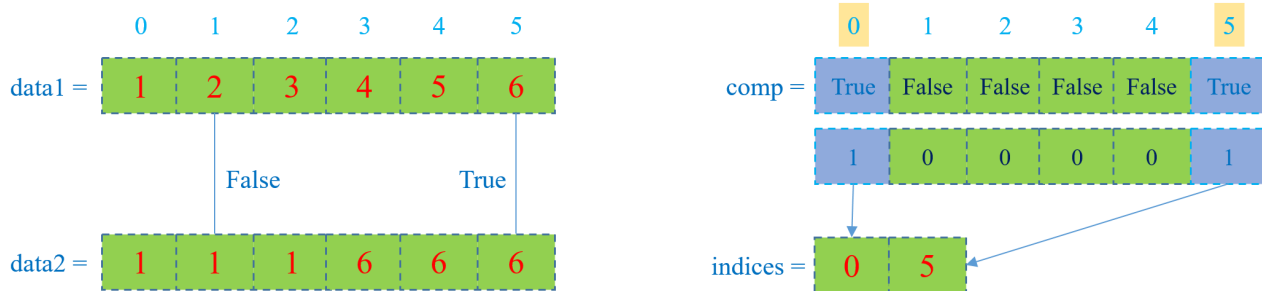
```
out = np.setdiff1d(data1, data2)
```

```
out = [2, 3, 4]
```

```
1 # Python code (aivietnam)
2 import numpy as np
3
4 data1 = np.array([1, 2, 3, 4, 5])
5 data2 = np.array([1, 5, 9])
6
7 out = np.setdiff1d(data1, data2)
8 print(out)
```

```
===== Output =====
[2 3 4]
=====
```

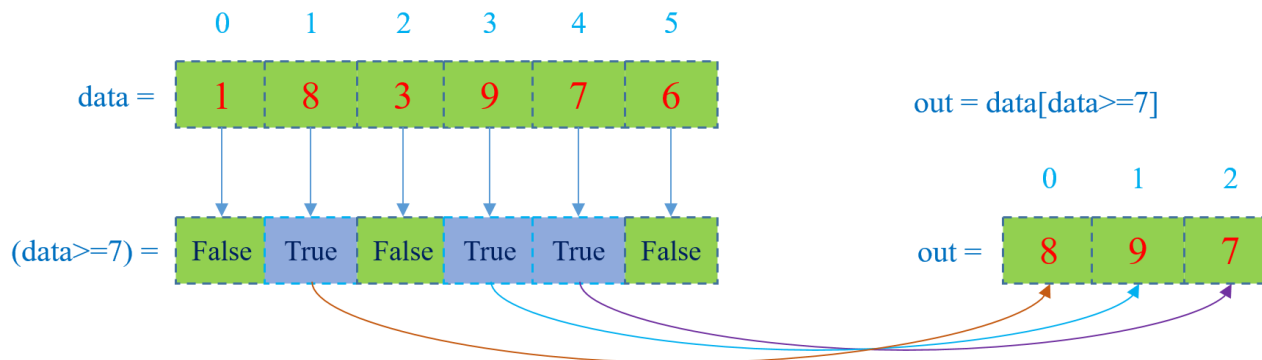
Câu 12: Lấy tất cả vị trí nơi giá trị các phần tử của hai mảng giống nhau



```
1 # Python code (aivietnam)
2 import numpy as np
3
4 # create data1 v data2
5 data1 = np.array([1, 2, 3, 4, 5, 6])
6 data2 = np.array([1, 1, 1, 6, 6, 6])
7
8 # compare the two array
9 comp = data1==data2
10
11 # get indices whose elements are not zero
12 indices = comp.nonzero()
13 print(indices)
```

```
===== Output =====
(array([0, 5], dtype=int64),)
=====
```

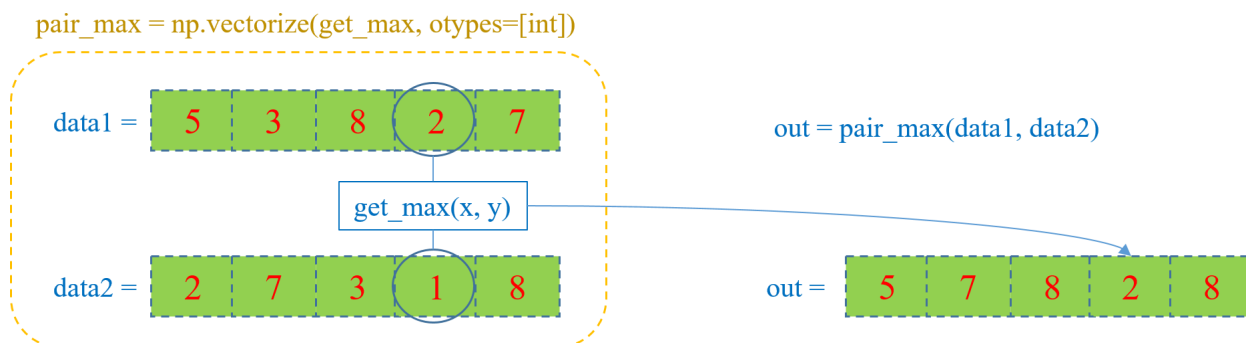
Câu 13: Lấy tất cả các giá trị trong một phạm vi cho trước



```
1 # Python code (aivietnam)
2 import numpy as np
3 data = np.array([1, 8, 3, 9, 7, 6])
4 print(data)
5
6 # Way 1
7 indices = np.where(data>=7)
8 out1 = data[indices]
9 print(out1)
10
11 # Way 2
12 out2 = data[data>=7]
13 print(out2)
```

```
===== Output =====
[1 8 3 9 7 6]
[8 9 7]
[8 9 7]
=====
```

Câu 14: Áp dụng một hàm user-defined cho ndarray dùng `np.vectorize()`. Áp dụng hàm `get_max()` cho hai mảng ndarray



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # get larger value
5 def get_max(x, y):
6     if x >= y :
7         return x
8     else:
9         return y
10
11 # vectorize the function
12 pair_max = np.vectorize(get_max,
13                          otypes=[int])
14
15 # create data1 and data2
16 data1 = np.array([5, 3, 8, 2, 7])
17 data2 = np.array([2, 7, 3, 1, 8])
18
19 # use pair_max as a function
20 out1 = pair_max(data1, data2)
21 print(out1)

```

```

===== Output =====
[5 7 8 2 8]

```

```

1 # Python code (aivietnam)
2 import numpy as np
3
4 data1 = np.array([5, 3, 8, 2, 7])
5 data2 = np.array([2, 7, 3, 1, 8])
6
7 # Way 2: Using the maximum() function
8 out2 = np.maximum(data1, data2)
9 print(out2)
10
11 # Way 3: Using the where() function
12 out3 = np.where(data1>data2, data1, data2)
13 print(out3)

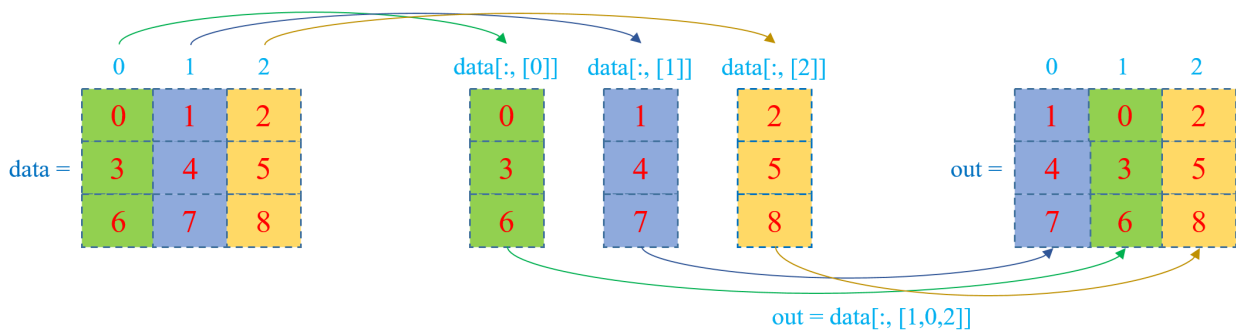
```

```

===== Output =====
[5 7 8 2 8]
[5 7 8 2 8]

```

Câu 15: Hoán đổi các cột trong mảng hai chiều



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # create a 3x3 matrix
5 data = np.arange(9).reshape(3,3)
6 print(data, '\n')
7
8 # A new matrix is constructed by the
9 # columns [1,0,2] from data
10 out = data[:, [1,0,2]]
11 print(out)

```

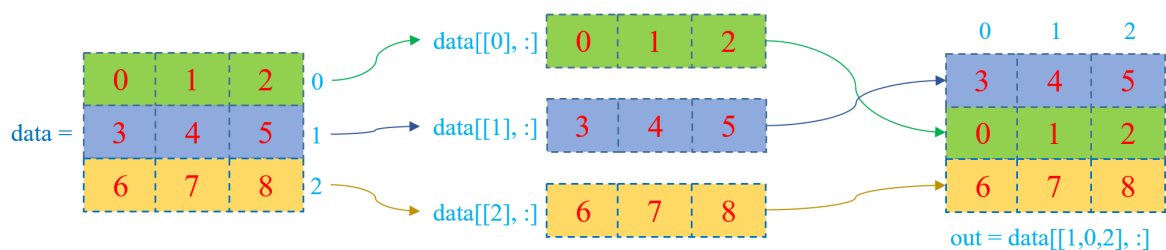
```

===== Output =====
[[0 1 2]
 [3 4 5]
 [6 7 8]]

[[1 0 2]
 [4 3 5]
 [7 6 8]]
=====

```

Câu 16: Hoán đổi các dòng trong mảng hai chiều



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # create a 3x3 matrix
5 data = np.arange(9).reshape(3,3)
6 print(data, '\n')
7
8 # A new matrix is constructed by the
9 # rows [1,0,2] from data
10 out = data[[1,0,2], :]
11 print(out)

```

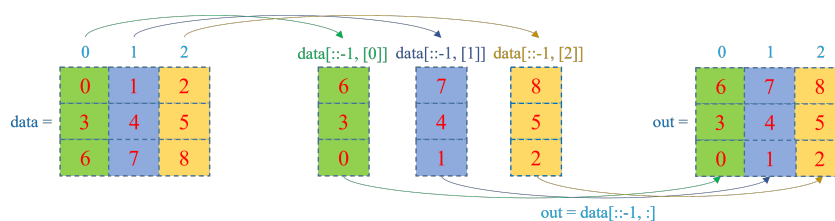
```

===== Output =====
[[0 1 2]
 [3 4 5]
 [6 7 8]]

[[3 4 5]
 [0 1 2]
 [6 7 8]]
=====

```


Câu 17: Đảo ngược các phần tử của các cột trong mảng hai chiều



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # create a 3x3 matrix
5 data = np.arange(9).reshape(3,3)
6 print(data, '\n')
7
8 # reverse each column
9 out = data[::-1, :]
10 print(out)

```

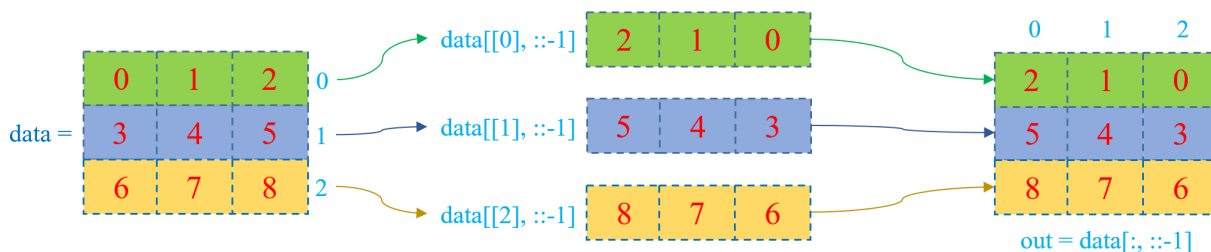
```

===== Output =====
[[0 1 2]
 [3 4 5]
 [6 7 8]]

[[6 7 8]
 [3 4 5]
 [0 1 2]]
=====

```

Câu 18: Đảo ngược các phần tử của các hàng trong mảng hai chiều



```

1 # Python code (aivietnam)
2 import numpy as np
3
4 # create a 3x3 matrix
5 data = np.arange(9).reshape(3,3)
6 print(data, '\n')
7
8 # reverse each row
9 out = data[:, ::-1]
10 print(out)

```

```

===== Output =====
[[0 1 2]
 [3 4 5]
 [6 7 8]]

[[2 1 0]
 [5 4 3]
 [8 7 6]]
=====

```

Câu 19: Tạo mảng hai chiều chứa số ngẫu nhiên (kiểu số lẻ)

data1 =

9.36	9.05	8.22
5.11	8.58	6.60

`np.random.uniform(5, 10, size=(2, 3))`

data2 =

0.71	0.48	0.96
0.27	0.54	0.37

`np.random.random([2, 3])`

```
1 # Python code (aivietnam)
2 import numpy as np
3
4 # create a 2x3 matrix (from 5 to 10)
5 data1 = np.random.uniform(5, 10,
6                             size=(2, 3))
7 print(data1, '\n')
8
9 # create a 2x3 matrix (from 0 to 1)
10 data2 = np.random.random([2, 3])
11 print(data2)
```

```
===== Output =====
[[9.36267749 9.05352382 8.22208754]
 [5.1182297  8.58622514 6.60564768]]

[[0.71150894 0.48885357 0.96791849]
 [0.2763089  0.54731615 0.37406768]]

=====
```

Câu 1: Tính độ dài của một vector

$$\|\vec{u}\| = \sqrt{u_1^2 + \dots + u_n^2}$$

$$\vec{u} = \begin{bmatrix} 1 & 2 & 4 & 2 \end{bmatrix}$$

$$\|\vec{u}\| = \sqrt{1^2 + 2^2 + 4^2 + 2^2} = 5$$

```
1 # Python code (aivietnam)
2 import numpy as np
3
4 u = np.array([1, 2, 4, 2])
5
6 # compute the length of u
7 print(np.linalg.norm(u))
```

```
===== Output =====
5.0
=====
```

Câu 2: Phép cộng và trừ giữa hai vector

data x		data y		result
1		5		6
2		6		8
3	+	7	=	10
4		8		12

data x		data y		result
5		1		4
6		2		4
7	-	3	=	4
8		4		4

```
1 # Python code (aivietnam)
2 import numpy as np
3
4 # create two vectors
5 x = np.array([1,2,3,4])
6 y = np.array([5,6,7,8])
7
8 print('data x \n', x)
9 print('data y \n', y)
10
11 # Addition between the two vectors
12 print('x + y = \n', x + y)
13
14 # Subtraction between the two vectors
15 print('x - y = \n', x - y)
```

```
===== Output =====
data x
[1 2 3 4]
data y
[5 6 7 8]
x + y =
[ 6  8 10 12]
x - y =
[-4 -4 -4 -4]
=====
```

Câu 3: Nhân (Hadamard) và chia giữa hai vector

```
1 # Python code (aivietnam)
2 import numpy as np
3
4 x = np.array([1,2,3,4])
5 y = np.array([5,6,7,8])
6
7 print('data x \n', x)
```

```
8 print('data y \n', y)
9
10 # Hadamard product between the two vectors
11 print('x * y = \n', x*y)
12
13 # Division between the two vectors
14 print('x / y = \n', x/y)
```

data x		data y		result
1		5		5
2		6		12
3	*	7	=	21
4		8		32

data x		data y		result
1		5		0.2
2		6		0.33
3	/	7	=	0.42
4		8		0.5

```

===== Output =====
data x
[1 2 3 4]
data y
[5 6 7 8]
x * y =
[5 12 21 32]

```

```

x / y =
[0.2 0.33333333 0.42857143 0.5]
=====

```

Câu 4: Tính tích vô hướng (dot product) giữa hai vector

$$\vec{v} = \begin{bmatrix} v_1 \\ \dots \\ v_n \end{bmatrix} \quad \vec{u} = \begin{bmatrix} u_1 \\ \dots \\ u_n \end{bmatrix}$$

$$\vec{v} \cdot \vec{u} = v_1 \times u_1 + \dots + v_n \times u_n$$

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \end{bmatrix} = 8$$

```

1 # Python code (aivietnam)
2 import numpy as np
3
4 v = np.array([1, 2])
5 w = np.array([2, 3])
6
7 # Compute inner product between v and w
8 print('dot product \n', v.dot(w))

```

```

===== Output =====
dot product
8
=====

```

Câu 5: Các phép tính cơ bản trên ma trận

$$\begin{bmatrix} 4 & 2 \\ 9 & 8 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ 12 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 \\ 9 & 8 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 6 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 \\ 9 & 8 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 27 & 32 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 \\ 9 & 8 \end{bmatrix} / \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 4. & 1. \\ 3. & 2. \end{bmatrix}$$

```

1 # Python code (aivietnam)
2 import numpy as np
3
4 A = np.array([[4, 2],
5               [9, 8]])
6 B = np.array([[1, 2],
7               [3, 4]])
8
9 # Addition between the two matrices
10 print('A + B = \n', A+B)
11
12 # Subtraction between the two matrices
13 print('A - B = \n', A-B)
14
15 # Hadamard product between the two
    matrices
16 print('A * B = \n', A*B)
17
18 # Division between the two matrices
19 print('A / B = \n', A/B)

```

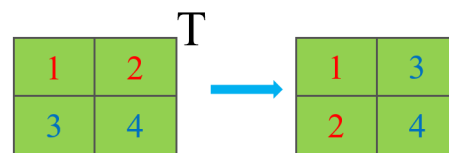
```

===== Output =====
A + B =
[[ 5  4]
 [12 12]]
A - B =
[[3 0]
 [6 4]]
A * B =
[[ 4  4]
 [27 32]]
A / B =
[[4.  1.]
 [3.  2.]]
=====

```

Câu 6: Chuyển vị một ma trận

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \quad A^T = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \dots & \dots & \dots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$$



```

1 # Python code (aivietnam)
2 X = np.array([[1, 2],
3               [3, 4]])
4 print('X: \n', X)
5
6 # transpose X
7 print('X.T: \n', X.T)

```

```

===== Output =====
X:
[[1 2]
 [3 4]]
X.T:
[[1 3]
 [2 4]]
=====

```

Câu 7: Phép nhân giữa ma trận và vector

```

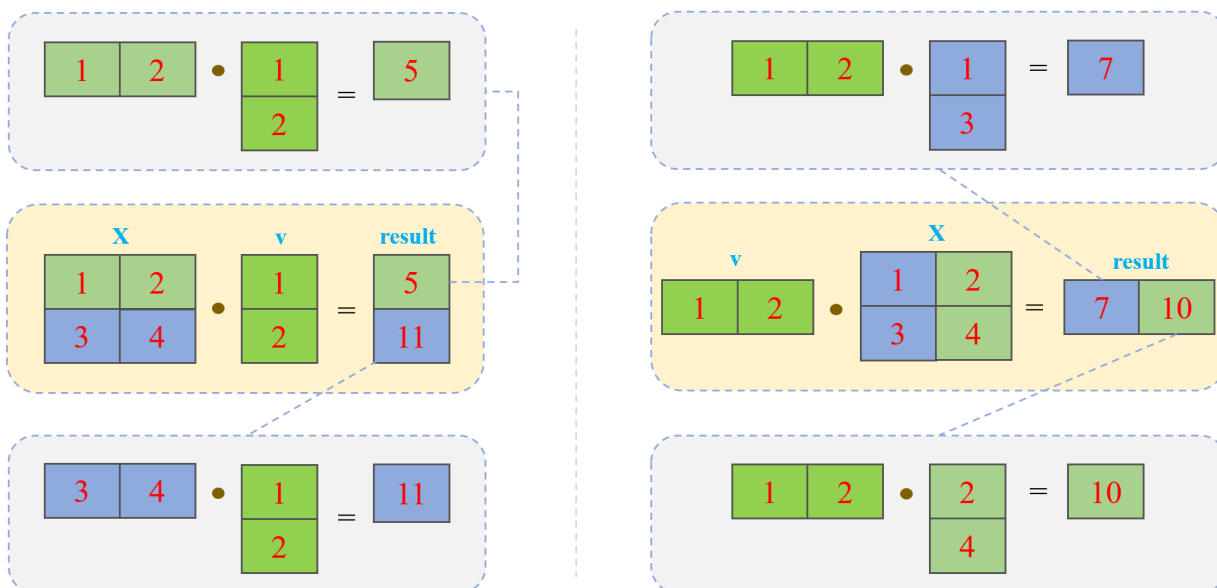
1 # Python code (aivietnam)
2 import numpy as np
3
4 X = np.array([[1, 2],
5               [3, 4]])
6 v = np.array([1, 2])
7
8 # Dot product between a matrix & a vector
9 print('X.dot(v) \n', X.dot(v))
10 print('v.dot(X) \n', v.dot(X))

```

```

===== Output =====
X.dot(v)
[ 5 11]
v.dot(X)
[ 7 10]
=====

```



Câu 8: Phép nhân giữa hai ma trận

$$\begin{matrix} X \\ \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \end{matrix} \cdot \begin{matrix} Y \\ \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \end{matrix} = \begin{matrix} \text{result} \\ \begin{bmatrix} 6 & 5 \\ 14 & 13 \end{bmatrix} \end{matrix}$$

$$\begin{matrix} Y \\ \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \end{matrix} \cdot \begin{matrix} X \\ \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \end{matrix} = \begin{matrix} \text{result} \\ \begin{bmatrix} 11 & 16 \\ 5 & 8 \end{bmatrix} \end{matrix}$$

```

1 # Python code (aivietnam)
2 import numpy as np
3
4 X = np.array([[1, 2],
5               [3, 4]])
6 Y = np.array([[2, 3],
7               [2, 1]])
8
9 # Dot product between the two matrices
10 print('X.dot(Y) \n', X.dot(Y))
11 print('Y.dot(X) \n', Y.dot(X))

```

```

===== Output =====
X.dot(Y)
[[ 6  5]
 [14 13]]
Y.dot(X)
[[11 16]
 [ 5  8]]
=====

```

Câu 9: Nhân vector/matrix với đại lượng vô hướng (scalar)

$$\begin{matrix} v \\ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \end{matrix} * \begin{matrix} 2 \end{matrix} = \begin{matrix} \text{result} \\ \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} \end{matrix}$$

$$\begin{matrix} X \\ \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \end{matrix} * \begin{matrix} 2 \end{matrix} = \begin{matrix} \text{result} \\ \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix} \end{matrix}$$

```
1 # Python code (aivietnam)
2 import numpy as np
3
4 v = np.array([1, 2, 3])
5 X = np.array([[1, 2],
6               [3, 4]])
7 scalar = 2
8
9 # Multiply with a scalar
10 print('v*2 \n', v*2)
11 print('X*2 \n', X*2)
```

```
===== Output =====
v*2
[2 4 6]
X*2
[[2 4]
 [6 8]]
=====
```