1. Giới thiệu về numpy

- Numpy là một thư viện toán học rất phổ biến và đẩy quyền năng của python. Nó cho phép làm việc hiệu quả với ma trận, tính toán các phép toán của ma trận nhanh chóng và hiệu quả.
- Đối tượng chính của Numpy là các dãy đa chiều đồng nhất (homogeneous multidimention array). Là một bảng tập hợp các phần tử cùng kiểu dữ liệu và được chỉ mục bằng các số nguyên dương.
- Khái niệm thứ nhất bạn cần nắm được đó làm numpy coi ma trận như là một không gian nhiều chiều.Số chiều của ma trận được numpy trả ra bởi hàm X.ndim:
- Ví dụ: X1= [1, 2, 3, 4] -> Đây là ma trận một chiều X1.ndim = 1
- X2 = [[0, 1], [2, 3]] -> Đây là ma trận 2 chiều X2.ndim = 2
- Khái niệm thứ hai cũng vẫn liên quan tới chiều của ma trận, ta đã biết hàm tính số chiều của ma trận nhưng ta chưa biết hàm tính size của từng chiều. Vậy numpy có hàm np.shape(X) hỗ trợ việc này:
- Ví dụ: X1 =[1, 2, 3, 4] -> np.shape(X1) = (4,)
- X2 = [[0, 1, 5], [2, 3, 9]] -> np.shape(X2) = (2,3)
  - Khái niệm cuối cùng bạn cần nắm và cũng là khái niệm rất quan trọng đó là axis trong numpy. Axis ám chỉ chiều của matrix. Một ma trận có số chiều là n thì sẽ có axis chạy từ 0 đến n-1. Sẽ có rất nhiều hàm có tham số axis và nó ám chỉ ta tính theo chiều nào của ma trận, trong khi số chiều gọi là rank.
- Ví dụ: Hàm sum: X có shape là (2, 3) thì hàm sum(axis = 0) thì sẽ cộng các số theo chiều thứ nhất và trả ra ma trận Y có shape là (1, 3). Tương tự sum(axis = 1) thì sẽ cộng các số theo chiều thứ hai và trả ra ma trận Y có shape là (2, 1).
- Numpy không copy mảng nếu không gọi hàm copy().

2. Một số mã

a. Khởi tạo mảng và list

```
from __future__ import print_function  #  Phiên bản 2 sử dụng như phiên bản 3
import numpy as np

data1 = [1, 2, 3, 4, 5]  # list
arr1 = np.array(data1)   # 1d array
data2 = [range(1, 5), range(5, 9)]  # list of lists
arr2 = np.array(data2)  # 2d array
ds = arr2.tolist()     # convert array back to list
```

```
print("List")
print(data1)
print("array")
print(arr1)
print("List cac doi tuong list")
print(data2)
print("array hai chieu")
print(arr2)
print("array to list")
print(ds)
```

```
List
[1, 2, 3, 4, 5]
array
[1 2 3 4 5]
List cac doi tuong list
[range(1, 5), range(5, 9)]
array hai chieu
[[1 2 3 4]
 [5 6 7 8]]
array to list
[[1, 2, 3, 4], [5, 6, 7, 8]]
```

b. Tạo vài mảng đặc biệt

```
print(np.zeros(10))
print(np.zeros((3, 6)))
print(np.ones(10))
print(np.linspace(0, 1, 5)) # 0 to 1 (inclusive) with 5 points
print(np.logspace(0, 3, 4)) # 10^0 to 10^3 (inclusive) with 4 points
```

```
[ 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.]
[ 0.    0.25  0.5   0.75  1.  ]
[    1.    10.   100.  1000.]
```

```
int_array = np.arange(5)
float_array = int_array.astype(float)
```

```
print(int_array)
print(float_array)
```

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

c. Khảo sát mảng

```
print(arr1.dtype) # float64
print(arr2.dtype) # int32
print(arr2.ndim) # 2
print(arr2.shape) # (2, 4) - axis 0 is rows, axis 1 is columns
print(arr2.size)  # 8 - total number of elements
print(len(arr2)) # 2 - size of first dimension (aka axis)
```

```
int32
int32
2
(2, 4)
8
2
```

```
arr = np.arange(10, dtype=float).reshape((2, 5))
print(arr)
print(arr.shape)
print(arr.reshape(5, 2))
```

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

d. Thêm 1 chiều

```
a = np.array([0, 1])
a_col = a[:, np.newaxis]
print(a)
```

```
print(a_col)
# or a_col = a[:, None]
```

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

e. Flatten: Luôn trả về danh sách copy dát mỏng 1 mảng

```
arr_flt = arr.flatten()
arr_flt[0] = 33
print(arr_flt)
print(arr)
```

```
[ 33.   1.   2.   3.   4.   5.   6.   7.   8.   9.]
[[ 0.  1.  2.  3.  4.]
 [ 5.  6.  7.  8.  9.]]
```

f. Ghép hai mảng

```
# Stack arrays
a = np.array([0, 1])
b = np.array([2, 3])
print(a)
print(b)
ab = np.stack((a, b))
cd = np.stack((a,b)).T # giống như ma trận chuyển vị
print(ab)
print(cd)
```

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

g. Truy cập các phần tử thuộc mảng

```
# Selection
arr = np.arange(10, dtype=float).reshape((2, 5))
```

```
print(arr)
print(arr[0]) # 0th element (slices like a list)
print(arr[1]) # 0th element (slices like a list)

print(arr[0, 3])          # row 0, column 3: returns 3
print(arr[0][3])          # alternative syntax
```

```
[[ 0.  1.  2.  3.  4.]
 [ 5.  6.  7.  8.  9.]]
[ 0.  1.  2.  3.  4.]
[ 5.  6.  7.  8.  9.]
3.0
3.0
```

```
arr = np.arange(10, dtype=float).reshape((2, 5))

print(arr[0, :]) # row 0: returns 1d array ([0, 1, 2, 3, 4])
print(arr[:, 0]) # column 0: returns 1d array ([0, 5])
print(arr[:, :2]) # columns strictly before index 2 (2 first columns)
print(arr[:, 2:]) # columns after index 2 included
arr2 = arr[:, 1:4] # columns between index 1 (included) and 4 (excluded)
print(arr2)
```

```
[ 0.  1.  2.  3.  4.]
[ 0.  5.]
[[ 0.  1.]
 [ 5.  6.]]
[[ 2.  3.  4.]
 [ 7.  8.  9.]]
[[ 1.  2.  3.]
 [ 6.  7.  8.]]
```

```
arr = np.arange(10, dtype=float).reshape((2, 5))
print(arr)
arr2 = arr
# returns a view (not a copy)
arr2[0, 0] = 33
print(arr2)
print(arr)
```

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

```
 [ 5.  6.  7.  8.  9.]]
[[ 33.   1.   2.   3.   4.]
 [  5.   6.   7.   8.   9.]]
[[ 33.   1.   2.   3.   4.]
 [  5.   6.   7.   8.   9.]]


arr = np.arange(10, dtype=float).reshape((2, 5))
print(arr)
arr2 = arr.copy() # Tạo một mảng copy
arr2[0, 0] = 33
print(arr2)
print(arr)

[[ 0.  1.  2.  3.  4.]
 [ 5.  6.  7.  8.  9.]]
[[ 33.   1.   2.   3.   4.]
 [  5.   6.   7.   8.   9.]]
[[ 0.  1.  2.  3.  4.]
 [ 5.  6.  7.  8.  9.]]

arr2 = arr[:, [1,2,3]] # return a copy
print(arr2)
arr2[0, 0] = 44
print(arr2)
print(arr)

[[ 1.  2.  3.]
 [ 6.  7.  8.]]
[[ 44.   2.   3.]
 [  6.   7.   8.]]
[[ 0.  1.  2.  3.  4.]
 [ 5.  6.  7.  8.  9.]]

print(arr)
arr2 = arr[arr > 5] # return a copy nhung phan tu > 5 -> mang 1 chieu
print(arr2)
arr2[0] = 44
print(arr2)
print(arr)

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

```
 [ 5.  6.  7.  8.  9.]]
[ 6.  7.  8.  9.]
[ 44.   7.   8.   9.]
[[ 0.  1.  2.  3.  4.]
 [ 5.  6.  7.  8.  9.]]
```

```
arr[arr > 5] = 0 # nhung phan tu > 5 thi gan bang 0
print(arr)
```

```
[[ 0.  1.  2.  3.  4.]
 [ 5.  0.  0.  0.  0.]]
```

h. Thao tác kiểu logic trên mảng

```
names = np.array(['Bob', 'Joe', 'Will', 'Bob'])
print(names == 'Bob') # returns a boolean array
print(names[names != 'Bob']) # logical selection
print((names == 'Bob') | (names == 'Will')) # keywords "and/or" don't work
with boolean ‹→arrays
names[names != 'Bob'] = 'Joe' # assign based on a logical selection
print(names)
a = np.unique(names) # set function
print(a)
```

```
[ True False False  True]
['Joe' 'Will']
[ True False  True  True]
['Bob' 'Joe' 'Joe' 'Bob']
['Bob' 'Joe']
```

i. Tính toán trên mảng

```
nums = np.arange(5)
print(nums)
print(nums * 10)              # multiply each element by 10
nums = np.sqrt(nums)    # square root of each element
print(nums)
print(np.ceil(nums))          # also floor, rint (round to nearest int)
print(np.floor(nums))         # Lam tron so nguyen san
print(np.isnan(nums))         # checks xem khong phai la so
print(nums + np.arange(5))     # Cong cac phan tu cua 2 mang
print(np.maximum(nums, np.array([1, -2, 3, -4, 5]))) # compare element-wise, lay
```

phan tu lớn nhất

```
# Compute Euclidean distance between 2 vectors
vec1 = np.random.randn(10)  # Tra ve 10 phan tu la phan phoi chuan trong 1 array
vec2 = np.random.randn(10)
dist = np.sqrt(np.sum((vec1 - vec2) ** 2))


# math and stats
rnd = np.random.randn(4, 2)  # random normals in 4x2 array co phan phoi chuan

print(rnd)
print(rnd.mean())         # Tinh mean
print(rnd.std())          # Tinh do lech chuan
print(rnd.argmin())               # index of minimum element
print(rnd.sum())
print(rnd.sum(axis=0))            # sum of columns
print(rnd.sum(axis=1))            # sum of rows
```

```
[0 1 2 3 4]
[ 0 10 20 30 40]
[ 0.         1.         1.41421356  1.73205081  2.        ]
[ 0.  1.  2.  2.  2.]
[ 0.  1.  1.  1.  2.]
[False False False False False]
[ 0.         2.         3.41421356  4.73205081  6.        ]
[ 1.         1.         3.         1.73205081  5.        ]
[[-1.01957824  0.81838943]
 [ 0.47882811  0.920258  ]
 [ 1.21961646 -1.44135102]
 [-0.1742606   0.95699866]]
0.219862600296
0.929375191586
5
1.75890080237
[ 0.50460573  1.25429507]
[-0.20118881  1.39908611 -0.22173455  0.78273806]
```

j. Các phương thức cho mảng Boolean

```
import numpy as np
```

```
rnd = np.random.randn(4, 2)
print(rnd)
print((rnd > 0).sum())   # counts number of positive values
print((rnd > 0).any())   # checks xem co gia tri nao > 0  -> true
print((rnd > 0).all())   # checks xem tat ca > 0 -> false

# random numbers
np.random.seed(12234) # Set the seed
print(np.random.rand(2, 3)) # 2 x 3 matrix in [0, 1]

print(np.random.randn(10)) # random normals (mean 0, sd 1)
print(np.random.randint(0, 3, 10)) # tao ra array co 10 thanh to, int, 0 hoac 1 hoac
2
```

```
[[ 1.18344381 -0.44590868]
 [ 0.35305993 -0.03219138]
 [ 0.78492624  0.3056839 ]
 [-0.05014605  0.02154613]]
5
True
False
[[ 0.00630595  0.20303476  0.76478993]
 [ 0.55513384  0.74358546  0.93777808]]
[ -2.79962074e-01   1.31281104e+00  -9.27155784e-01  -4.01302169e-01
  -2.31085929e+00  -2.08460156e+00   4.59241643e-01   1.62191344e+00
   1.94515120e-01  -2.08631547e-03]
[0 2 2 0 1 1 1 1 2 0]
```

k. Mở rộng mảng

```
# Broadcasting
a = np.array([[ 0, 0, 0],
[10, 10, 10],
[20, 20, 20],
[30, 30, 30]])
b = np.array([0, 1, 2])
print(a + b)
```

```
[[ 0  1  2]
 [10 11 12]
 [20 21 22]
```

[30 31 32]]

3. Bài Tập

Given the array:

```
X = np.random.randn(4, 2) # random normals in 4x2 array
```

- For each column find the row index of the minimum value.
- Write a function `standardize(X)` that return an array whose columns are centered and scaled (by std-dev).