ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA

THƯ VIỆN NUMPY & MACHINE LEARNING

Khoa Công nghệ thông tin
ThS. Nguyễn Thị Lệ Quyên

Nội dung môn học

- Cơ bản ngôn ngữ lập trình Python
- **Numpy và Machine Learning**
- Tương tác Cơ sở dữ liệu
- Cơ bản HTML, CSS, JavaScript
- Flask framework
- Kiến thức bổ sung

NUMPY

Nội dung

- Mảng đa chiều
- Hàm tiện ích trong NumPy
- Array-Oriented Programming with Arrays
- Xuất nhập File
- Đại số tuyến tính
- Số ngẫu nhiên

Numpy là gì?

- NumPy (Numerical Python), là một thư viện tính toán quan trọng nhất trong Python.
- NumPy thiết kế đối tương mảng (array objects) cho việc trao đổi dữ liệu giữa các hàm một cách thuận tiện. Vì vậy NumPy được sử dụng hầu hết tại các gói (packages) tính toán trong lĩnh vực khoa học dữ liệu.
- NumPy cung cấp C API nên giúp cho Python trở thành ngôn ngữ năng động và có khả năng kế thừa được các thư viện có sẵn như C/C++ hay Fortran

Tính năng quan trọng của NumPy

- Tăng tốc toán tử tính toán với vector như làm sạch dữ liệu, tập con hóa, lọc, phép biến đổi,…
- Cung cấp các thuật toán phổ biến như sắp xếp, hợp nhất mảng, toán tử tập hợp (set operations)
- Thống kê và tổng hợp dữ liệu
- Hợp nhất các dữ liệu không đồng nhất
- Biểu diễn logic có điều kiện dưới dạng mảng thay vì sử dụng vòng lặp và lệnh điều kiện
- Thao tác dữ liệu theo nhóm (tập hợp, biến đổi, hàm ứng dụng)

Tốc độ xử lý của NumPy

- Gấp 10 đến 100 lần so với Python thuần túy.
- Ví dụ:

```
my_list = list(range(1000000))
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

✓ 0.7s Python

```
Wall time: 723 ms
```

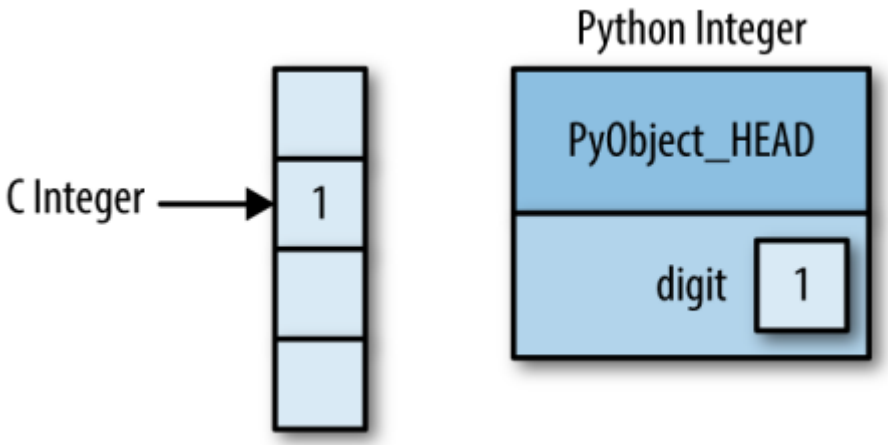
```
import numpy as np
my_arr = np.arange(1000000)
%time for _ in range(10): my_arr2 = my_arr * 2
```

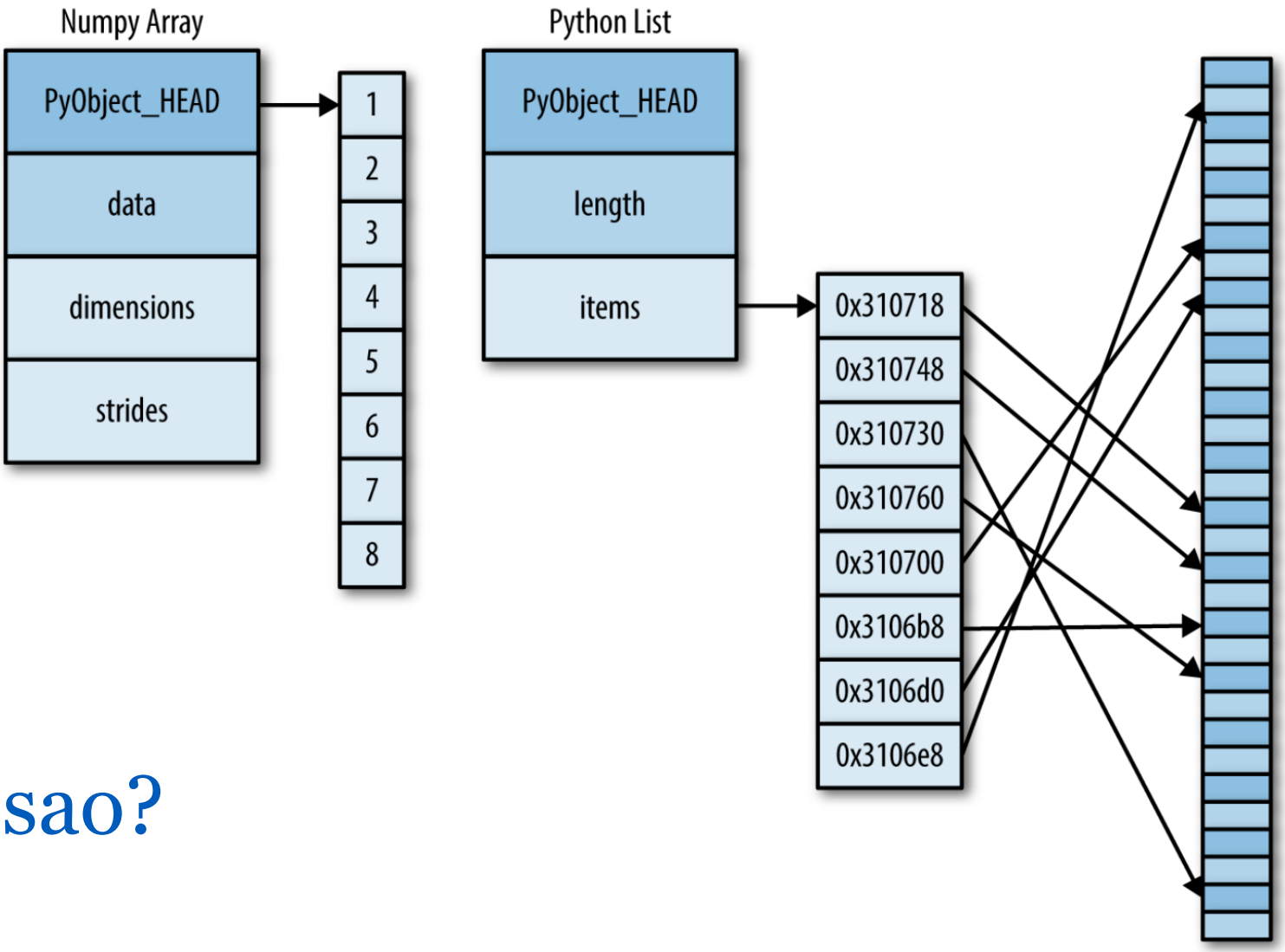
✓ 0.5s Python

```
Wall time: 17 ms
```

Vì sao?

```
struct _longobject {
    long ob_refcnt;
    PyTypeObject *ob_type;
    size_t ob_size;
    long ob_digit[1];
};
```




Numpy Array
PyObject_HEAD
data
dimensions
strides
1
2
3
4
5
6
7
8
Python List
PyObject_HEAD
length
items
0x310718
0x310748
0x310730
0x310760
0x310700
0x3106b8
0x3106d0
0x3106e8

Vì sao?

Mảng có kiểu dữ liệu cố định trong Python

```
import array

L = list(range(10))
A = array.array('i', L)
print(A)

# array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

ndarray

- NumPy cho phép thực hiện các phép toán trên mảng tương tự như các phép toán trên số thực

```
import numpy as np
data = np.random.rand(2,3)
data
```

✓ 0.5s

```
array([[0.11172141, 0.90852808, 0.3276298 ],
       [0.96448868, 0.54006314, 0.50445723]])
```

```
data * 10
```

✓ 0.4s

```
array([[1.11721406, 9.08528077, 3.27629804],
       [9.64488677, 5.40063142, 5.04457226]])
```

```
data + data
```

✓ 0.3s

```
array([[0.22344281, 1.81705615, 0.65525961],
       [1.92897735, 1.08012628, 1.00891445]])
```

ndarray trong NumPy

- Kích thước và kiểu dữ liệu của ndarray

```
data.shape
✓ 0.8s
(2, 3)
```

```
data.ndim
✓ 0.3s
2
```

```
data.dtype
✓ 0.3s
dtype('float64')
```

Khởi tạo mảng

- Tạo từ Python list
  - Sử dụng phương thức array để chuyển mảng trong Python qua mảng trong NumPy

```
np.array([1, 4, 2, 5, 3])
```
✓ 0.4s
```
array([1, 4, 2, 5, 3])
```

```
np.array([3.14, 4, 2, 3])
```
✓ 0.3s
```
array([3.14, 4.  , 2.  , 3.  ])
```

```
np.array([1, 2, 3, 4], dtype='float32')
```
✓ 0.4s
```
array([1., 2., 3., 4.], dtype=float32)
```

```
np.array([range(i,i+3) for i in [2,4,6]])
```
✓ 0.4s
```
array([[2, 3, 4],
       [4, 5, 6],
       [6, 7, 8]])
```

Khởi tạo mảng

- Từ các hàm được xây dựng sẵn trong NumPy

```
np.zeros(10, dtype=int)
```
✓ 0.3s
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
np.ones((2, 4))
```
✓ 0.5s
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

```
np.full((3, 5), 3.14)
```
✓ 0.4s
```
array([[3.14, 3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14, 3.14]])
```

Khởi tạo mảng

Hàm	Mô tả
array	Chuyển dữ liệu kiểu list, tuple, array, ... trong Python qua mảng ndarray trong NumPy
asarray	Chuyển input sang ndarray, nhưng không copy nếu input đã là ndarray
arange	Giống hàm range nhưng trả về một mảng ndarray thay vì list trong Python
ones, ones_like	Tạo mảng tất cả các phần tử đều bằng 1 với shape và dtype cho trước; ones_like tạo ra một mảng khác có các phần tử đều bằng 1 có kích thước giống shape và dtype của 1 mảng cho trước
zeros, zeros_like	Giống ones và ones_like nhưng tất cả phần tử đều bằng 0
empty, empty_like	Tạo một mảng mới bằng cách cấp phát vùng nhớ mới, nhưng giá trị của mảng phụ thuộc vào vùng nhớ
full, full_like	Tạo 1 mảng với shape và dtype cho trước với tất cả giá trị đều gán bằng giá trị chỉ định trước; full_like tạo một mảng khác với kích thước giống share và dtype của mảng cho trước
eye, identity	Tạo 1 ma trận đơn vị NxN (1 trên đường chéo chính và 0 ở những vị trí còn lại)

Các kiểu dữ liệu chuẩn trong NumPy

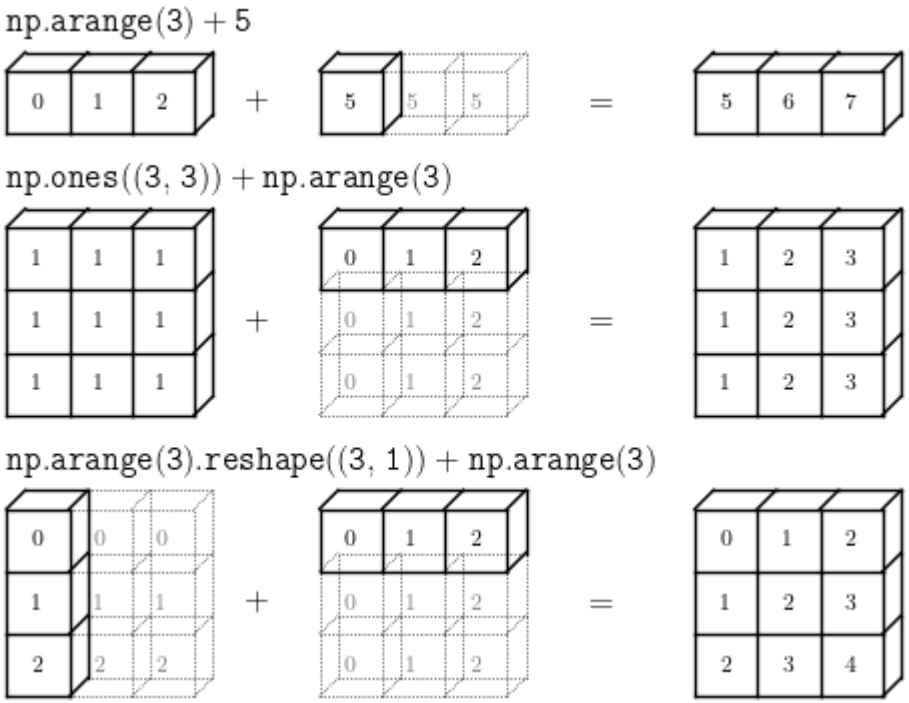
Type	Type code	Description
`int8, uint8`	`i1, u1`	Signed and unsigned 8-bit (1 byte) integer types
`int16, uint16`	`i2, u2`	Signed and unsigned 16-bit integer types
`int32, uint32`	`i4, u4`	Signed and unsigned 32-bit integer types
`int64, uint64`	`i8, u8`	Signed and unsigned 64-bit integer types
`float16`	`f2`	Half-precision floating point
`float32`	`f4 or f`	Standard single-precision floating point; compatible with C float
`float64`	`f8 or d`	Standard double-precision floating point; compatible with C double and Python `float` object
`float128`	`f16 or g`	Extended-precision floating point
`complex64, complex128, complex256`	`c8, c16, c32`	Complex numbers represented by two 32, 64, or 128 floats, respectively
`bool`	`?`	Boolean type storing `True` and `False` values
`object`	`O`	Python object type; a value can be any Python object
`string_`	`S`	Fixed-length ASCII string type (1 byte per character); for example, to create a string dtype with length 10, use `'S10'`
`unicode_`	`U`	Fixed-length Unicode type (number of bytes platform specific); same specification semantics as `string_` (e.g., `'U10'`)

Toán tử số học

- NumPy cung cấp các toán tử để thực hiện các phép tính trên mảng mà không cần sử dụng các lệnh vòng lặp. Người dùng NumPy gọi là vector hóa (vectorization).

```
import numpy as np
arr = np.array([[1., 2., 3.], [4., 5., 6.]])
print(arr)
print(arr*arr)
print(arr-arr)
print(1/arr)
print(arr**0.5)
print(arr*arr)

arr2 = np.array([[0., 4., 1.], [7., 2., 12.]])
print(arr>arr2)
```

Broadcasting



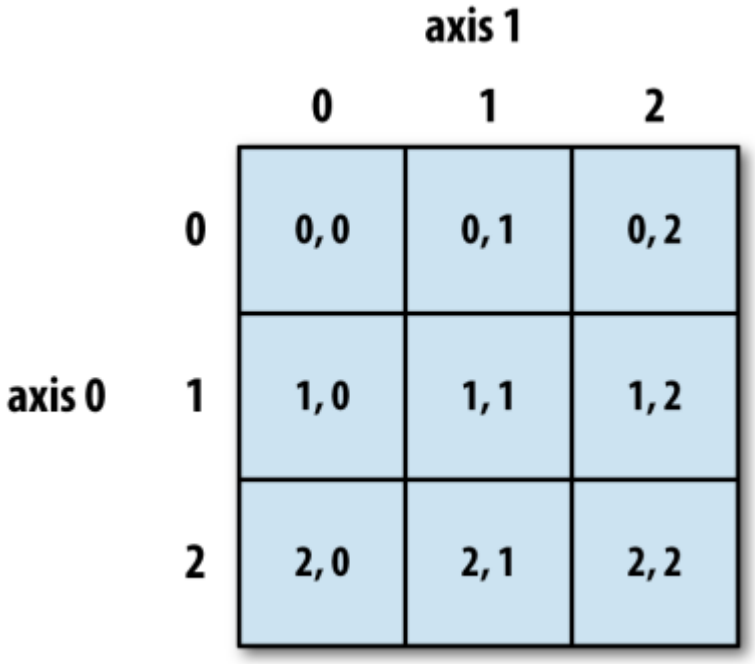
Thao tác trong mảng NumPy

- Thao tác với các phần tử hay các tập con trong mảng của NumPy như thao tác với list của Python

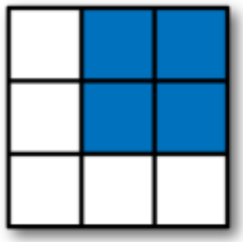
```
import numpy as np
arr = np.arange(10)
print(arr)
print(arr[5])
print(arr[5:-1])
```

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

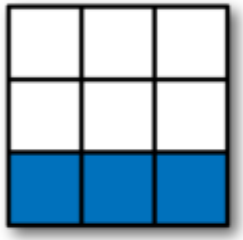
Chỉ mục trong mảng NumPy



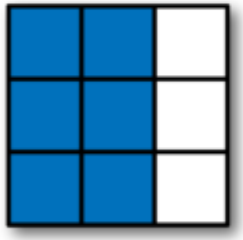
Chỉ mục với slice



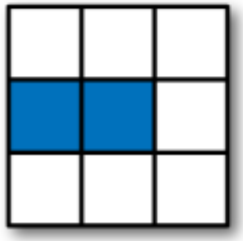
Expression	Shape
arr[:2, 1:]	(2, 2)



Expression	Shape
arr[2]	(3,)
arr[2, :]	(3,)
arr[2:, :]	(1, 3)



Expression	Shape
arr[:, :2]	(3, 2)



Expression	Shape
arr[1, :2]	(2,)
arr[1:2, :2]	(1, 2)

Chỉ mục Boolean

- Có thể dùng mảng Boolean để tạo mảng con cho 1 mảng khác nếu như có cùng chiều dài như nhau

```
import numpy as np
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
data = np.random.randn(7, 4)
print(names)
print(data)
print(names=='Bob')
print(data[names=='Bob'])
```

```
['Bob' 'Joe' 'Will' 'Bob' 'Will' 'Joe' 'Joe']
[[ 0.61351579 -0.72588671  1.52101198  0.70002282]
 [ 0.27705541  0.87402548 -2.05059442  1.41110013]
 [ 0.89114229 -0.39914808  1.23431688 -1.32219814]
 [ 0.51798234  1.63610317 -0.62185666 -0.26285829]
 [-1.04915382  0.19132779 -0.71809904 -0.40998428]
 [ 0.75578892  0.08066942 -1.02714665 -0.08947798]
 [-0.37626032  0.13307081 -0.2355532  -1.01127001]]
[ True False False  True False False False]
[[ 0.61351579 -0.72588671  1.52101198  0.70002282]
 [ 0.51798234  1.63610317 -0.62185666 -0.26285829]]
```

Chỉ mục Boolean

- Hãy cho biết kết quả của các lệnh sau:

a)
```
print(data[names == 'Bob', 2:])
```

b)
```
print(data[names == 'Bob', 3])
```

c)
```
data[data < 0] = 0
print(data)
```

Fancy Indexing

- NumPy cho phép tạo 1 mảng con theo chỉ mục cho trước

```
import numpy as np
names = np.array([   'Tuan', 'Phuong', 'Khoi', 'Hieu',
                     'Duy', 'Thang',  'Duc'])
print(names)
print(names[[0,5,4]])

['Tuan' 'Phuong' 'Khoi' 'Hieu' 'Duy' 'Thang' 'Duc']
['Tuan' 'Thang' 'Duy']
```

Fancy Indexing

- Hãy cho biết kết quả đoạn lệnh sau:

```
arr = np.arange(32).reshape((8, 4))
print(arr[[1, 5, 7, 2], [0, 3, 1, 2]])
print(arr[[1, 5, 7, 2]][:, [0, 3, 1, 2]])
```

- Hai lệnh `print` khác nhau chỗ nào?

Ma trận chuyển vị

```
import numpy as np

arr = np.arange(15).reshape((3, 5))
print(arr)
print(arr.T)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]

[[ 0  5 10]
 [ 1  6 11]
 [ 2  7 12]
 [ 3  8 13]
 [ 4  9 14]]
```

A

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

a
(2, 3)

np.transpose(a)
(3, 2)

Ma trận chuyển vị

- Đối với mảng nhiều chiều, ta cũng có thể chuyển vị mảng như sau:

```
import numpy as np

arr = np.arange(16).reshape((2, 2, 4))
print(arr)
print(arr.transpose((1, 0, 2)))
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]]

 [[ 8  9 10 11]
  [12 13 14 15]]]

[[[ 0  1  2  3]
  [ 8  9 10 11]]

 [[ 4  5  6  7]
  [12 13 14 15]]]
```

Hoán đổi trục ma trận

```
import numpy as np

arr = np.arange(16).reshape((2, 2, 4))
print(arr)
print(arr.swapaxes(1, 2))
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]]
 [[ 8  9 10 11]
  [12 13 14 15]]]

[[[ 0  4]
  [ 1  5]
  [ 2  6]
  [ 3  7]]
 [[ 8 12]
  [ 9 13]
  [10 14]
  [11 15]]]
```

Hàm tiện ích trong NumPy

- Hàm tính toán 1 ngôi

Function	Description
`abs, fabs`	Compute the absolute value element-wise for integer, floating-point, or complex values
`sqrt`	Compute the square root of each element (equivalent to `arr ** 0.5`)
`square`	Compute the square of each element (equivalent to `arr ** 2`)
`exp`	Compute the exponent e^x of each element
`log, log10, log2, log1p`	Natural logarithm (base *e*), log base 10, log base 2, and log(1 + x), respectively
`sign`	Compute the sign of each element: 1 (positive), 0 (zero), or –1 (negative)
`ceil`	Compute the ceiling of each element (i.e., the smallest integer greater than or equal to that number)
`floor`	Compute the floor of each element (i.e., the largest integer less than or equal to each element)
`rint`	Round elements to the nearest integer, preserving the `dtype`
`modf`	Return fractional and integral parts of array as a separate array
`isnan`	Return boolean array indicating whether each value is `NaN` (Not a Number)
`isfinite, isinf`	Return boolean array indicating whether each element is finite (non-`inf`, non-`NaN`) or infinite, respectively
`cos, cosh, sin, sinh, tan, tanh`	Regular and hyperbolic trigonometric functions
`arccos, arccosh, arcsin, arcsinh, arctan, arctanh`	Inverse trigonometric functions
`logical_not`	Compute truth value of `not x` element-wise (equivalent to `~arr`).

Ví dụ:

```
import numpy as np
arr = np.arange(10)
print(arr)
print(np.sqrt(arr))

# [0 1 2 3 4 5 6 7 8 9]
# [0.         1.         1.41421356 1.73205081 2.
   2.23606798 2.44948974 2.64575131 2.82842712 3.
  ]
```

Hàm tiện ích trong NumPy

- Các hàm tính toán 2 ngôi

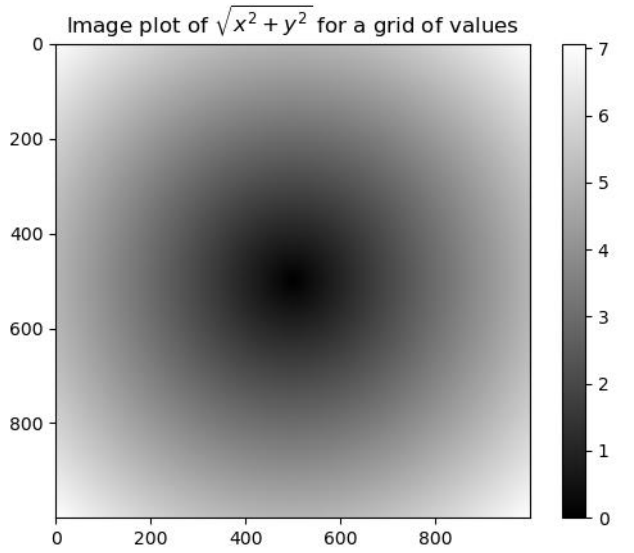
Function	Description
`add`	Add corresponding elements in arrays
`subtract`	Subtract elements in second array from first array
`multiply`	Multiply array elements
`divide, floor_divide`	Divide or floor divide (truncating the remainder)
`power`	Raise elements in first array to powers indicated in second array
`maximum, fmax`	Element-wise maximum; `fmax` ignores `NaN`
`minimum, fmin`	Element-wise minimum; `fmin` ignores `NaN`
`mod`	Element-wise modulus (remainder of division)
`copysign`	Copy sign of values in second argument to values in first argument
`greater, greater_equal, less, less_equal, equal, not_equal`	Perform element-wise comparison, yielding boolean array (equivalent to infix operators `>, >=, <, <=, ==, !=`)
`logical_and, logical_or, logical_xor`	Compute element-wise truth value of logical operation (equivalent to infix operators `& | ^`)

Ví dụ:

```
import numpy as np
arr1 = np.arange(10) / 2
arr2 = np.arange(10) / 3 + 1
print(arr1)
print(arr2)
print(np.greater(arr1, arr2))

# [0.  0.5 1.  1.5 2.  2.5 3.  3.5 4.  4.5]
# [1.         1.33333333 1.66666667 2.         2.3333333
   2.66666667 3.         3.33333333 3.66666667 4.        ]
# [False False False False False False False  True  True  True]
```

Array-Oriented Programming with Arrays

- NumPy cho phép thể hiện nhiều tác vụ xử lý dữ liệu với mảng ngắn gọn mà không cần viết vòng lặp




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

points = np.arange(-5, 5, 0.01)
xs, ys = np.meshgrid(points, points)

z = np.sqrt(xs ** 2 + ys ** 2)
plt.imshow(z, cmap = plt.cm.gray)
plt.colorbar()
plt.title("Image plot of $\sqrt{x^2 + y^2}$ for a grid of values")
plt.show()
```

Điều kiện logic được biểu diễn bởi mảng

- NumPy có các hàm tiện ích khi có các điều kiện logic được biểu diễn bởi mảng
- Cho:

```
xarr = np.array([1.1, 1.2, 1.3, 1.4, 1.5])
yarr = np.array([2.1, 2.2, 2.3, 2.4, 2.5])
cond = np.array([True, False, True, True, False])
```

- Để lựa chọn phần tử, thay vì

```
result = [(x if c else y) for x, y, c in zip(xarr, yarr, cond)]
```

- Ta có thể sử dụng hàm where

```
result = np.where(cond, xarr, yarr)
```

Các hàm toán học, thống kê

- NumPy có rất nhiều hàm liên quan đến thống kê

Method	Description
sum	Sum of all the elements in the array or along an axis; zero-length arrays have sum 0
mean	Arithmetic mean; zero-length arrays have NaN mean
std, var	Standard deviation and variance, respectively, with optional degrees of freedom adjustment (default denominator n)
min, max	Minimum and maximum
argmin, argmax	Indices of minimum and maximum elements, respectively
cumsum	Cumulative sum of elements starting from 0
cumprod	Cumulative product of elements starting from 1

```
arr = np.random.randn(2, 3)
print(arr)
print(arr.mean())
```

```
[[-0.21050017 -0.47941506 -0.29274799]
 [-0.60584632 -1.53666652  0.43055528]]
-0.4491034630493929
```

Mảng Boolean

- Có thể sử dụng sum như 1 cách để đếm các giá trị True trong 1 mảng boolean

```
arr = np.random.randn(100)
print((arr > 0).sum())
```

- Có 2 hàm tiện dụng là any() và all() tương ứng với phép or và add

```
bools = np.array([False, False, True, False])

print(bools.any())      # True
print(bools.all())      # False
```

Sắp xếp

- Sắp xếp mảng 1 chiều

```
arr = np.random.randn(6)
print(arr)
arr.sort()
print(arr)
```

```
[-1.17996359 -0.38383318  0.40114662 -0.36551224 -0.35595998  0.92213374]
[-1.17996359 -0.38383318 -0.36551224 -0.35595998  0.40114662  0.92213374]
```

Sắp xếp

- Sắp xếp mảng nhiều chiều

```
arr = np.random.randn(2,3)
print(arr)
arr.sort(1)
print(arr)
```

```
[[ 1.82029148 -1.28506997 -0.18056308]
 [ 0.77680604  1.32806397 -0.80290937]]

[[-1.28506997 -0.18056308  1.82029148]
 [-0.80290937  0.77680604  1.32806397]]
```

Toán tử liên quan đến tập hợp

- Hàm unique(): tạo tập sắp xếp tăng dần

```
names = np.array(['Bob', 'Alice', 'Peter', 'Alice', 'Bob'])
print(np.unique(names))     # ['Alice' 'Bob' 'Peter']

ints = np.array([3, 3, 3, 2, 2, 1, 1, 4, 4])
print(np.unique(ints))      # [1 2 3 4]
```

Toán tử liên quan đến tập hợp

Method	Description
`unique(x)`	Compute the sorted, unique elements in x
`intersect1d(x, y)`	Compute the sorted, common elements in x and y
`union1d(x, y)`	Compute the sorted union of elements
`in1d(x, y)`	Compute a boolean array indicating whether each element of x is contained in y
`setdiff1d(x, y)`	Set difference, elements in x that are not in y
`setxor1d(x, y)`	Set symmetric differences; elements that are in either of the arrays, but not both

Nhập/ xuất mảng bằng File

- Ta có thể lưu 1 mảng vào File

```
arr = np.arange(10)
np.save('some_array', arr)
```

- Sau đó load mảng từ file

```
arr1 = np.load('some_array.npy')
print(arr1)
```

Nhập/ xuất mảng bằng File

- Ta có thể lưu nhiều mảng vào file .npz

```
arr1 = np.arange(10)
arr2 = np.arange(5)
np.savez('array_archive.npz', a=arr1, b=arr2)
```

```
np.savez_compressed('arrays_compressed.npz', a=arr, b=arr)
```

- Load mảng từ file

```
arrz = np.load('array_archive.npz')
print(arrz['b'])
```

Đại số tuyến tính

- NumPy cung cấp cho chúng ta nhiều hàm để tính toán trong lĩnh vực đại số tuyến tính như, nhân ma trận, phân rã ma trận, tính định thức,…

Đại số tuyến tính

Function	Description
`diag`	Return the diagonal (or off-diagonal) elements of a square matrix as a 1D array, or convert a 1D array into a square matrix with zeros on the off-diagonal
`dot`	Matrix multiplication
`trace`	Compute the sum of the diagonal elements
`det`	Compute the matrix determinant
`eig`	Compute the eigenvalues and eigenvectors of a square matrix
`inv`	Compute the inverse of a square matrix
`pinv`	Compute the Moore-Penrose pseudo-inverse of a matrix
`qr`	Compute the QR decomposition
`svd`	Compute the singular value decomposition (SVD)
`solve`	Solve the linear system Ax = b for x, where A is a square matrix
`lstsq`	Compute the least-squares solution to `Ax = b`

Ví dụ:

```
import numpy as np
from numpy.linalg import inv, qr

X = np.random.randn(2, 2)
mat = X.T.dot(X)

print(mat)
print(inv(mat))
print(mat.dot(inv(mat)))
```

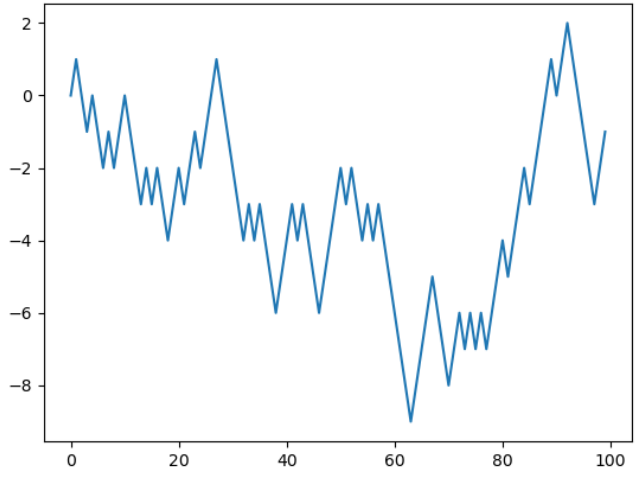
```
[[ 0.26721072 -0.15919219]
 [-0.15919219  2.46121607]]

[[3.89235119 0.25175843]
 [0.25175843 0.42258703]]

[[ 1.00000000e+00 -2.58814336e-18]
 [-1.09420189e-16  1.00000000e+00]]
```

Số ngẫu nhiên

- Các hàm liên quan

Function	Description
seed	Seed the random number generator
permutation	Return a random permutation of a sequence, or return a permuted range
shuffle	Randomly permute a sequence in-place
rand	Draw samples from a uniform distribution
randint	Draw random integers from a given low-to-high range
randn	Draw samples from a normal distribution with mean 0 and standard deviation 1 (MATLAB-like interface)
binomial	Draw samples from a binomial distribution
normal	Draw samples from a normal (Gaussian) distribution
beta	Draw samples from a beta distribution
chisquare	Draw samples from a chi-square distribution
gamma	Draw samples from a gamma distribution
uniform	Draw samples from a uniform [0, 1) distribution

Ví dụ 1: Random Walks



- Sinh ngẫu nhiên bước đi bắt đầu tại 0, bước tới trước 1 hoặc bước lui -1

```
import random
import matplotlib.pyplot as plt

position = 0
walk = [position]
steps = 1000
for i in range(steps):
    step = 1 if random.randint(0, 1) else -1
    position += step
    walk.append(position)

plt.plot(walk[:100])
plt.show()
```

Random Walks với NumPy

```
import numpy as np

nsteps = 1000
draws = np.random.randint(0, 2, size=nsteps)
steps = np.where(draws > 0, 1, -1)
walk = steps.cumsum()

print(walk.min())
print(walk.max())

print((np.abs(walk) >= 10).argmax())
```

Mô phỏng nhiều Random Walks

```
import numpy as np

nwalks = 5000
nsteps = 1000

draws = np.random.randint(0, 2, size=(nwalks, nsteps)) #0 or 1
steps = np.where(draws > 0, 1, -1)

walks = steps.cumsum(1)

hits30 = (np.abs(walks) >= 2).any(1)
print(hits30.sum())       # Number that hit 30 or -30

crossing_times = (np.abs(walks[hits30]) >= 2).argmax(1)
print(crossing_times.mean())
```

Ví dụ 2: Độ lệch chuẩn & Phương sai

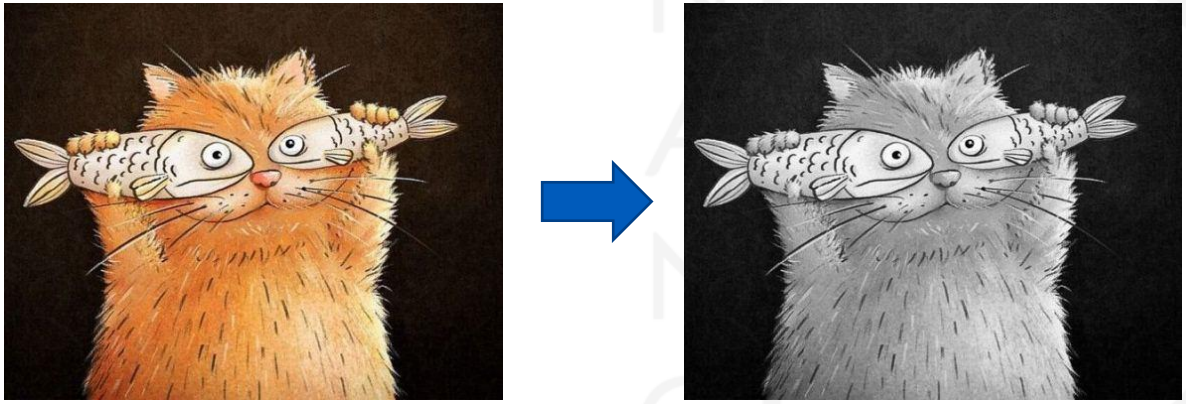
- Ref: Standard Deviation and Variance (mathsisfun.com)

```
np.random.seed(15)
arr = np.random.randint(low=0, high=100, size=10)
print(arr.mean())          # 42.2
print(arr.std())           # 30.261526729495987
```

```
arr = np.random.normal(loc=42, scale=30, size=10)
```

Ví dụ 3: Image process

```
import numpy as np
from PIL import Image

im = np.array(Image.open('cat.jpg').convert('L'))
print(type(im))

gr_im = Image.fromarray(im).save('gr_cat.png')
```



Ref: GitHub - Image processing examples with Numpy, Scipy, and Scikit-image

Tổng kết

- Mảng đa chiều
- Hàm tiện ích trong NumPy
- Array-Oriented Programming with Arrays
- Xuất nhập File
- Đại số tuyến tính
- Số ngẫu nhiên

Tài liệu tham khảo

- Jake VanderPlas, Python Data Science Handbook, O'Reilly Media, Inc., 548 pages, 2016.
- Peter Bruce, Andrew Bruce, Practical Statistics for Data Scientists: 50 Essential Concepts - 1st Edition, O'Reilly Media; 1st edition, 90 pages, 2017
- Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python: A Guide for Data Scientists 1st Edition, 392 pages, 2016
- Lillian Pierson, Data Science For Dummies - 2nd Edition, John Wiley & Sons Inc., 385 pages, 2017.
- David R. Anderson, Dennis J. Sweeney, Thomas A. Williams, Statistics for Business and Economics, South-Western College Pub., 880 pages, 2010.
- Các nguồn internet khác.