



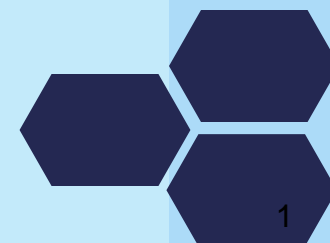
Bài giảng

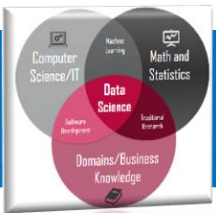
NHẬP MÔN

KHOA HỌC DỮ LIỆU

(Data Science)

ThS. Phạm Đình Tài
0985.73.39.39
pdtai@ntt.edu.vn

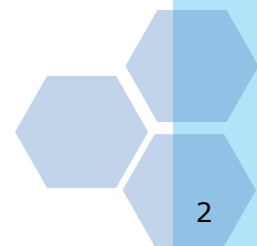


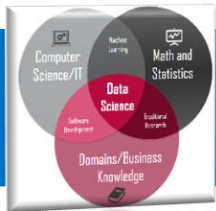


CHƯƠNG

3

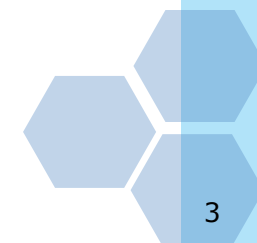
Thư viện NumPy

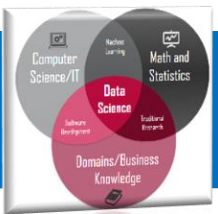




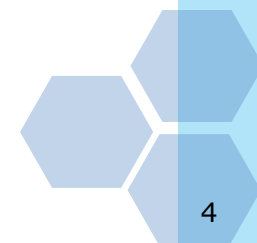
Nội dung

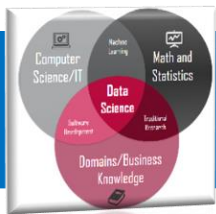
- 1. Một số gói python cho KHDL*
- 2. Giới thiệu về NumPy*
- 3. Khởi tạo mảng và chỉ số*
- 4. Các phép toán trên mảng*
- 5. Một số thao tác thông dụng*





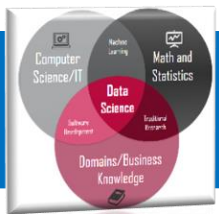
Một số gói python cho Data Science





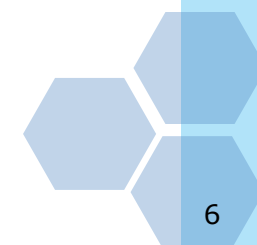
Một số gói python cho KHD

- Ngôn ngữ python có hệ thống các gói rất phong phú, hỗ trợ nhiều lĩnh vực khác nhau, từ xây dựng ứng dụng, xử lý web, xử lý text, xử lý ảnh,...
- ✓ Sử dụng pip để tải các gói mới về từ internet
- Một số gói dành cho lập trình thông thường:
 - ✓ **os**: xử lý file và tương tác với hệ điều hành
 - ✓ **networkx** và **igraph**: làm việc với dữ liệu đồ thị, có thể làm việc với dữ liệu rất lớn (đồ thị hàng triệu đỉnh)
 - ✓ **regular expressions**: tìm kiếm mẫu trong dữ liệu text
 - ✓ **BeautifulSoup**: trích xuất dữ liệu từ file HTML hoặc từ website



Một số gói python cho KHDL

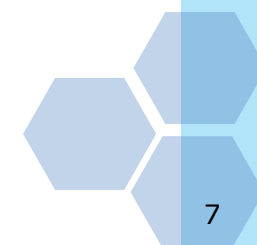
- **NumPy (Numerical Python):** là gói chuyên về xử lý dữ liệu số (nhiều chiều); gói cũng chứa các hàm đại số tuyến tính cơ bản, biến đổi fourier, sinh số ngẫu nhiên nâng cao,...
- **SciPy (Scientific Python):** dựa trên Numpy, cung cấp các công cụ mạnh cho khoa học và kỹ nghệ, chẳng hạn như biến đổi fourier rời rạc, đại số tuyến tính, tối ưu hóa và ma trận thưa
- **Matplotlib:** chuyên sử dụng để vẽ biểu đồ, hỗ trợ rất nhiều loại biểu đồ khác nhau





Một số gói python cho KHDL

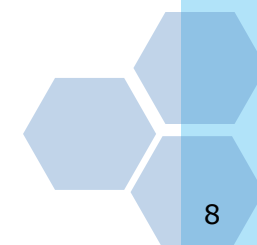
- **Pandas:** chuyên sử dụng cho quản lý và tương tác với dữ liệu có cấu trúc, được sử dụng rộng rãi trong việc thu thập và tiền xử lý dữ liệu.
- **Scikit Learn:** chuyên về học máy, dựa trên **NumPy**, **SciPy** và **matplotlib**; thư viện này có sẵn nhiều công cụ hiệu quả cho học máy và thiết lập mô hình thống kê chẳng hạn như các thuật toán phân lớp, hồi quy, phân cụm và giảm chiều dữ liệu
- **Statsmodels:** cho phép người sử dụng khám phá dữ liệu, ước lượng mô hình thống kê và kiểm định





Một số gói python cho KHDL

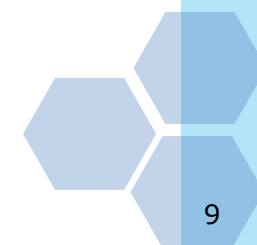
- **Seaborn:** dự trên matplotlib, cung cấp các công cụ diễn thị (visualization) dữ liệu thống kê đẹp và hiệu quả, mục tiêu của gói là sử dụng việc diễn thị như là trọng tâm của khám phá và hiểu dữ liệu.
- **Bokeh:** để tạo các ô tương tác, biểu đồ tổng quan trên nền web, rất hiệu quả khi tương tác với dữ liệu lớn và trực tuyến.
- **Blaze:** gói dựa trên Numpy và Pandas hướng đến dữ liệu phân tán hoặc truyền phát, là công cụ mạnh mẽ tạo diễn thị về dữ liệu cực lớn

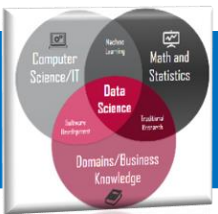




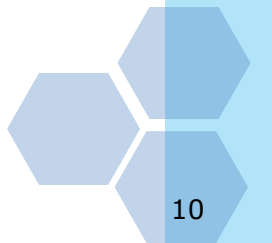
Một số gói python cho KHDL

- **Scrapy:** chuyên về thu thập thông tin trên web, rất phù hợp với việc lấy các dữ liệu theo mẫu
- **SymPy:** tính toán chuyên ngành dùng cho số học, đại số, toán rời rạc và vật lý lượng tử
- **Theano:** gói chuyên dùng tính toán hiệu quả các mảng nhiều chiều, sử dụng rộng rãi trong học máy
- **TensorFlow:** gói chuyên dùng cho học máy của Google, đặc biệt là các mạng thần kinh nhân tạo
- **Keras:** thư viện cấp cao chuyên về học máy, sử dụng **Theano**, **TensorFlow** hoặc CNTK làm phụ trợ





Giới thiệu về NumPy



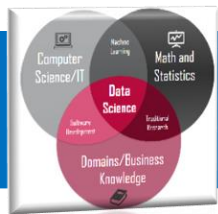


Giới thiệu về NumPy

- NumPy là thư viện bổ sung của python, do không có sẵn, ta phải cài đặt: **pip install numpy**

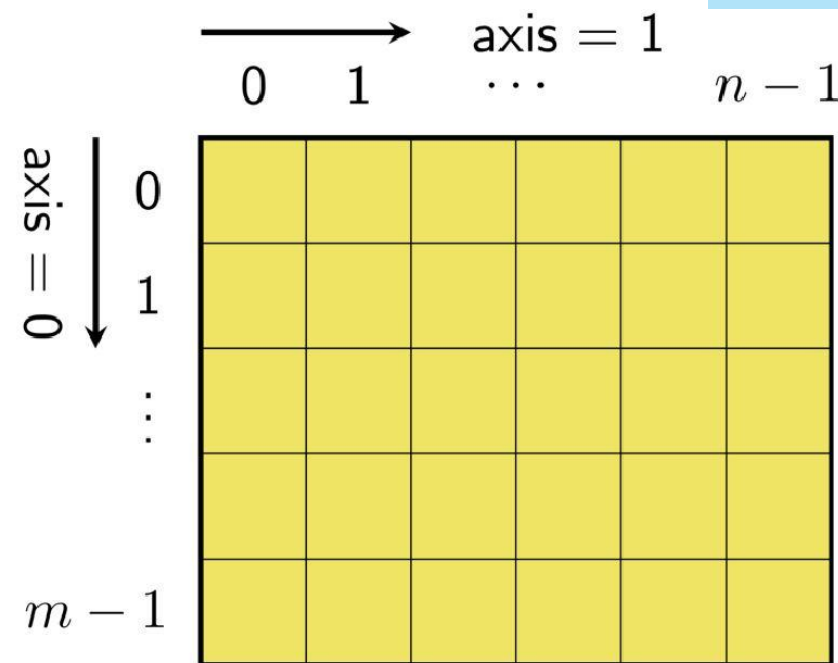
```
Command Prompt
C:\Users\ADMIN>pip install numpy
Collecting numpy
  Downloading numpy-1.22.1-cp39-cp39-win_amd64.whl (14.7 MB)
    | 14.7 MB 3.3 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.22.1
WARNING: You are using pip version 21.1.1; however, version 22.0.2 is available.
You should consider upgrading via the 'c:\users\admin\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.
C:\Users\ADMIN>
```

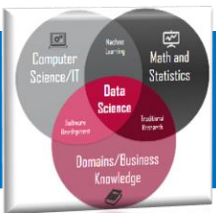
- Một số hệ thống python đã có sẵn numpy thì có thể bỏ qua bước này
- Cách đơn giản nhất để kiểm tra xem hệ thống đã cài numpy hay không là thử import gói xem có bị báo lỗi hay không: **import numpy as np**



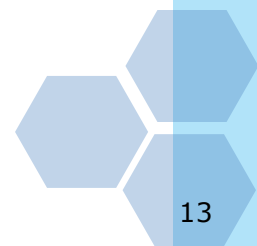
Đặc điểm của NumPy

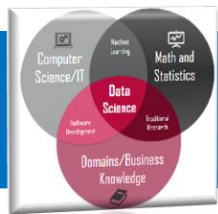
- Đối tượng chính của NumPy là các mảng đa chiều đồng nhất (homogeneous multidimension array)
- ✓ Kiểu dữ liệu phần tử con trong mảng phải giống nhau
- ✓ Mảng có thể một chiều hoặc nhiều chiều
- ✓ Các chiều (**axis**) được đánh thứ tự từ 0 trở đi
- ✓ Số chiều gọi là hạng (**rank**)
- ✓ Có đến 24 kiểu số khác nhau
- ✓ Kiểu **ndarray** là lớp chính xử lý dữ liệu mảng nhiều chiều
- ✓ Rất nhiều hàm và phương thức xử lý ma trận





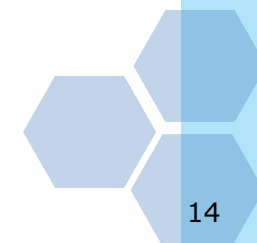
Khởi tạo mạng và chỉ số





Tạo mảng và truy cập

- Một mảng **numpy** là một lưới các giá trị, và tất cả các giá trị có cùng kiểu giá trị, và được lập chỉ mục bởi một số nguyên không âm, số chiều được gọi là rank của mảng **numpy**, và **shape** là một **tuple** các số nguyên đưa ra kích thước của mảng theo mỗi chiều.
- Chúng ta có thể khởi tạo numpy arrays từ **nested Python lists**, và dùng dấu ngoặc vuông để truy cập từng phần tử

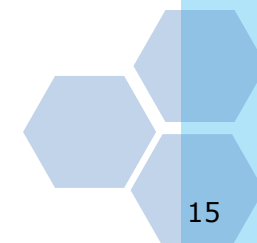




Tạo mảng và truy cập

```
import numpy as np
```

```
a = np.array([1, 2, 3]) # tạo mảng 1 chiều
print(type(a)) # in "<class 'numpy.ndarray'>"
print(a.shape) # in "(3,)"
print(a[0], a[1], a[2]) # in "1 2 3"
a[0] = 5
print(a) # in "[5, 2, 3]"
b = np.array([[1, 2, 3], [4, 5, 6]]) # tạo mảng 2 chiều
print(b.shape) # in "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0]) # in "1 2 4"
print(np.diag([1, 3, 4])) # in ra cái gì?
```

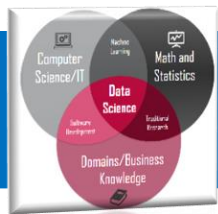




Nhiều cách khởi tạo phong phú

```
import numpy as np
```

```
x = np.range(3.0) # mảng [0. 1. 2.]
a = np.zeros((2, 2)) # mảng 2x2 toàn số 0
b = np.ones((1, 2)) # mảng 1x2 toàn số 1
c = np.full((3, 2, 2), 9) # mảng 3x2x2 toàn số 9
d = np.eye(2) # ma trận đơn vị 2x2
e = np.random.random(3, 2) # mảng 3x2 ngẫu nhiên [0,1)
# mảng 2x3 điền các số từ 1 đến 6, kiểu số nguyên 32 bit
x = np.array([[1, 2, 3], [4, 5, 6]], np.int32)
print(x.ndim, x.size)
print(x.shape) # in "(2, 3)"
print(x.dtype) # in "dtype('int32')"
```

Truy cập theo chỉ số (slicing)

- **Numpy** cung cấp một số cách để truy xuất phần tử trong mảng
- **Slicing**: Tương tự như **list** trong **python**, **numpy arrays** cũng có thể được cắt.

```
import numpy as np
```

```
# mảng 3x4
```

```
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
```

```
# mảng 2x2 trích xuất từ a, dòng 0+1, cột 1+2
```

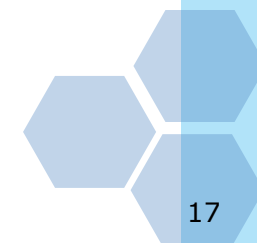
```
b = a[:2, 1:3]
```

```
# chú ý: mảng của numpy tham chiếu chứ không copy dữ liệu
```

```
print(a[0, 1]) # in "2"
```

```
b[0, 0] = 77 # b[0, 0] cũng là a[0, 1]
```

```
print(a[0, 1]) # in "77"
```



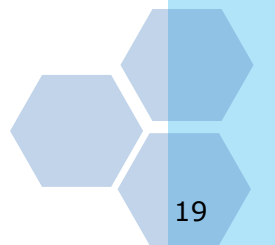


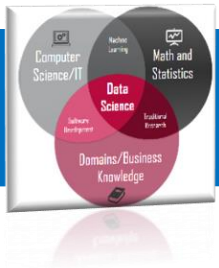
Cẩn thận với slicing

```
row_r1 = a[1, :]    # mảng 1 chiều độ dài 4
row_r2 = a[1:2, :] # mảng 2 chiều 1x4
print(row_r1, row_r1.shape) # in ra "[5 6 7 8] (4,)"
print(row_r2, row_r2.shape) # in ra "[[5 6 7 8]] (1, 4)"
col_r1 = a[:, 1]    # mảng 1 chiều độ dài 3
col_r2 = a[:, 1:2]  # mảng 2 chiều 3x1
print(col_r1, col_r1.shape) # in ra "[ 2 6 10] (3,)"
print(col_r2, col_r2.shape) # in ra "[[ 2]
                             # [ 6]
                             # [10]] (3, 1)"
```



Các phép toán trên mảng

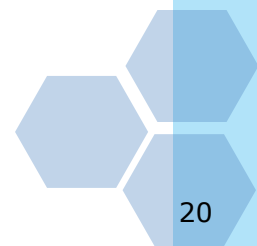


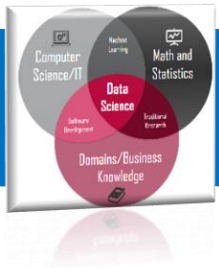


NumPy có nhiều phép toán về mảng

```
import numpy as np
```

```
x = np.array([[1, 2], [3, 4]], dtype=np.float64)
y = np.array([[5, 6], [7, 8]], dtype=np.float64)
print(x + y) # print(np.add(x, y)), xử lý khác list
print(x - y) # print(np.subtract(x, y))
print(x * y) # print(np.multiply(x, y))
print(x / y) # print(np.divide(x, y))
print(np.sqrt(x)) # khai căn tất cả các phần tử
print(2**x) # tính 2 mũ các phần tử trong x
# chú ý: phép nhân/chia thực hiện theo cặp phần tử của
x và y
```





Nhân ma trận (dot) và nghịch đảo

```
import numpy as np
```

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

```
y = np.array([[5, 6], [7, 8]])
```

```
v = np.array([9, 10])
```

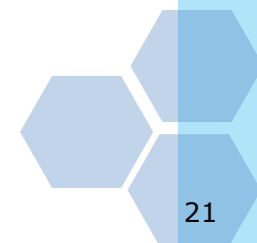
```
w = np.array([11, 12])
```

```
print(v.dot(w)) # tương tự print(np.dot(v, w))
```

```
print(x.dot(v)) # tương tự print(np.dot(x, v))
```

```
print(x.dot(y)) # tương tự print(np.dot(x, y))
```

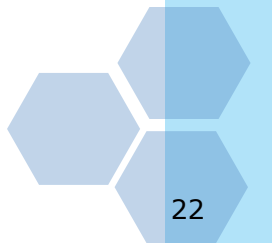
```
print(np.linalg.inv(x)) # tính và in nghịch đảo của x
```





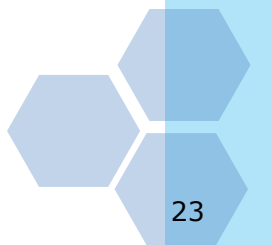
Ma trận chuyển vị

```
import numpy as np
x = np.array([[1, 2], [3, 4]])
print(x) # in ra "[[1 2]
# [3 4]]"
print(x.T) # in ra "[[1 3]
# [2 4]]"
# chú ý: mảng 1 chiều không có chuyển vị
y = np.array([1, 2, 3])
print(y) # in ra "[1 2 3]"
print(y.T) # in ra "[1 2 3]"
z = np.array([[1, 2, 3]])
print(z.T) # đoán xem in ra cái gì?
```





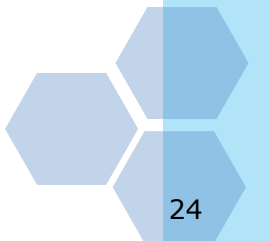
Một số thao tác thông dụng

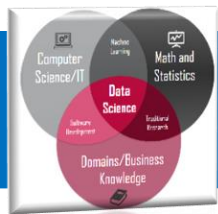




Đọc dữ liệu từ file

```
from io import StringIO
import numpy as np
c = StringIO("0 1\n2 3")
x = np.loadtxt(c) # array([[ 0.,  1.],
# [ 2.,  3.]])
d = StringIO("M 21 72\nF 35 58")
y = np.loadtxt(d, dtype={'names': ('gender', 'age',
'weight'), 'formats': ('S1', 'i4', 'f4')})
print(y) # [('M', 21, 72.0), ('F', 35, 58.0)]
```





Cơ chế broadcasting

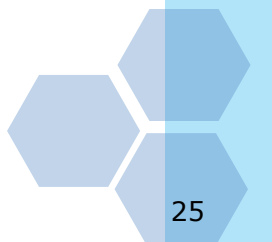
```
import numpy as np
```

```
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)      # in ra "[[ 2  2  4]
               # [ 5  5  7]
               # [ 8  8 10]
               # [11 11 13]]"
```





Tính tổng theo các trục

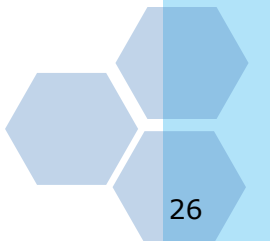
```
import numpy as np
```

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

```
print(np.sum(x)) # tính tổng toàn bộ x, in "10"
```

```
print(np.sum(x, axis=0)) # tính tổng mỗi cột, in "[4 6]"
```

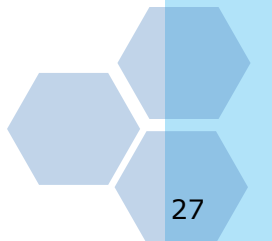
```
print(np.sum(x, axis=1)) # tính tổng mỗi hàng, in "[3 7]"
```





Trích xuất dữ liệu theo dãy

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
# Prints "[1 4 5]"
print(a[[0, 1, 2], [0, 1, 0]])
# Prints "[1 4 5]"
print(np.array([a[0, 0], a[1, 1], a[2, 0]]))
# Prints "[2 2]"
print(a[[0, 0], [1, 1]])
# Prints "[2 2]"
print(np.array([a[0, 1], a[0, 1]]))
```

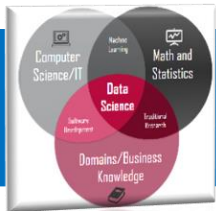




Lọc phần tử theo chỉ số

```
import numpy as np
```

```
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])  
b = np.array([0, 2, 0, 1]) # b là mảng các chỉ số  
print(a[np.arange(4), b]) # in ra "[1 6 7 11]"  
# cộng tất cả các phần tử được lọc thêm 10  
a[np.arange(4), b] += 10  
print(a) # in ra "array([[11, 2, 3],  
# [ 4, 5, 16],  
# [17, 8, 9],  
# [10, 21, 12]])"
```



Lọc dữ liệu theo điều kiện

```
import numpy as np
```

```
a = np.array([[1, 2], [3, 4], [5, 6]])
```

```
bool_idx = (a > 2)
```

```
print(bool_idx) # in ra "[False False]
```

```
# [ True True]
```

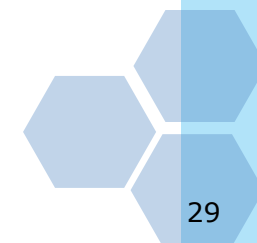
```
# [ True True]]"
```

```
# lọc dữ liệu trong a, trả về một dãy
```

```
print(a[bool_idx]) # Prints "[3 4 5 6]"
```

```
# có thể viết trực tiếp điều kiện (ngắn gọn hơn)
```

```
print(a[a > 2]) # Prints "[3 4 5 6]"
```





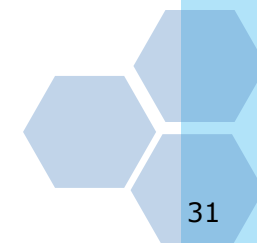
Điều chỉnh cỡ ma trận

```
>>> x = np.array([[1, 3], [4, 4], [4, 2]])
>>> x.shape
(3, 2)
>>> x = np.array([[1, 3], [4, 4], [4, 2]])
>>> x = x.reshape(2, 3) // chỉnh thành 2x3
>>> x
array([[1, 3, 4], [4, 4, 2]])
>>> x = np.array([[1, 3], [4, 4], [4, 2]])
>>> x = x.reshape(2, -1) // tự tính chiều còn lại
>>> x
array([[1, 3, 4], [4, 4, 2]])
```



Elementwise operation

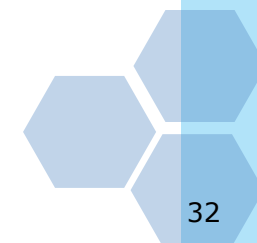
- **Element-wise** là các phép toán cực kỳ phổ biến với tensor trong lập trình mạng neural
- Hiểu đơn giản thì **Element-wise** là phép toán trên các phần tử tương ứng giữa các **tensor**. Hai phần tử được cho là tương ứng nếu hai phần tử chiếm cùng một vị trí trong **tensor**. Vị trí được xác định bởi các **index** được sử dụng để định vị từng phần tử.





Elementwise operation

```
>>> x = np.array([1, 2, 3])
>>> np.log(x) // lấy log cơ số e từng phần tử
array([ 0, 0.69314718, 1.09861229])
>>> np.abs(x) // lấy trị tuyệt đối từng phần tử
array([1, 2, 3])
>>> np.maximum(x, 2) // so sánh từng phần tử với 2 và
lấy max
array([2, 2, 3])
>>> np.minimum(x, 2) // so sánh từng phần tử với 2 và
lấy min
array([1, 2, 2])
>>> x**2 // lũy thừa 2 từng phần tử
array([1, 4, 9])
```





Tính norm cấp 2 của vector

norm cấp 2 của vector là chiều dài của vector đó

$$\# \|\mathbf{x}\|_2 = \|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$$

```
x = np.array([[0, 3], [4, 3], [6, 8]])
```

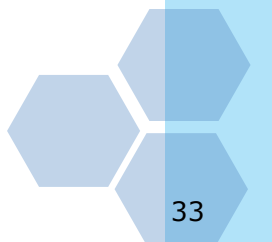
tính norm mỗi dòng, kết quả: array([[3], [5], [10]])

```
np.linalg.norm(x, axis = 1, keepdims = True)
```

```
x = np.array([[0, 6], [4, 0], [3, 8]])
```

tính norm mỗi cột, kết quả: array([[5, 10]])

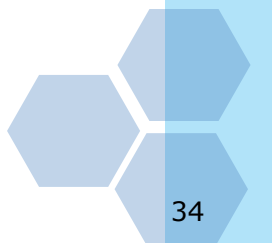
```
np.linalg.norm(x, axis = 0, keepdims = True)
```





Sinh mảng ngẫu nhiên

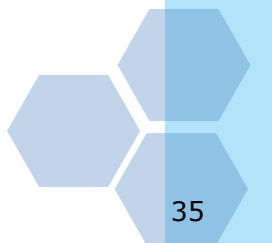
```
np.random.random(3, 2) # mảng 3x2 ngẫu nhiên trong [0,1)
np.random.randn() # một số sinh theo phân phối chuẩn
np.random.randn(3) # mảng 3 số theo phân phối chuẩn
np.random.randn(3, 4) # mảng 3x4 theo phân phối chuẩn
# mảng 2x4 gồm các số nguyên trong [3,15)
np.random.randint(3, 15, (2, 4))
# sinh một dãy là hoán vị ngẫu nhiên của dãy (0, 1, 2,
..., 19)
np.random.permutation(20)
```





Các hàm thống kê

```
import numpy as np
a = np.random.randn(3, 4)
# tính trung bình của cả ma trận a
print(np.mean(a))
# tính trung vị của cột đầu tiên
print(np.median(a[:,0]))
# tính độ lệch chuẩn của từng dòng
print(a.std(axis=0))
# tính phương sai của từng cột
print(a.var(axis=1))
```





Thank You!

ThS. Phạm Đình Tài
0985.73.39.39
pdtai@ntt.edu.vn

