

Ôn tập Python, Numpy, Matplotlib và Pandas

TS. Đỗ Như Tài
dntai@sgu.edu.vn

Ngôn ngữ Python

NỘI DUNG

Bài 1.

Mở đầu

Bài 2.

Kiểu dữ liệu và phép toán

Bài 3.

Chuỗi và danh sách

Bài 4.

Tập hợp

Bài 5.

Ngoại lệ và xử lý tập tin

Bài 2.1

Mở đầu

- 1. Giới thiệu ngôn ngữ python**
- 2. Cách thực hiện câu lệnh, chương trình**
- 3. Biến, Kiểu dữ liệu, Khối lệnh**
- 4. Nhập và Xuất dữ liệu**
- 5. Ví dụ minh họa**

Giới thiệu ngôn ngữ Python

- ❑ Python lần đầu được giới thiệu vào tháng 12/1989
- ❑ Tác giả là Guido van Rossum (Hà Lan)
 - Sinh năm 1956
 - Hiện đang làm cho Google
- ❑ Python kế thừa từ ngôn ngữ ABC
- ❑ Python 2 được giới thiệu năm 2000
 - Hỗ trợ unicode
 - Mã python 2 rất phổ biến
- ❑ Python 3 được phát hành năm 2008
- ❑ Hiện đã có phiên bản 3.7





Giới thiệu ngôn ngữ Python

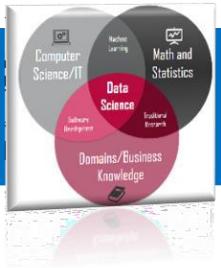


- Được xếp vào loại “ngôn ngữ kịch bản” (scripting programming language)
- *Thích hợp với DevOps (viết code cũng là vận hành)*
- *Khai báo biến tự nhiên, phong phú và động*
- *Nhiều phép tính cấp cao được cung cấp sẵn*
- *Thường được thông dịch thay vì biên dịch*
- Những người cuồng python (pythonista) cho rằng ngôn ngữ này trong sáng và tiện dụng đến mức ta có thể dùng nó cho mọi khâu lập trình (chứ không phải chỉ viết script)



Giới thiệu ngôn ngữ python

- Là ngôn ngữ mã nguồn mở
- Vừa hướng thủ tục, vừa hướng đối tượng
- Hỗ trợ module và hỗ trợ gói (package)
- Xử lý lỗi bằng ngoại lệ (exception)
- Kiểu dữ liệu động ở mức cao
- Có khả năng tương tác với các module viết bằng ngôn ngữ lập trình khác
- Có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface)



ƯU ĐIỂM Python

- Có ngũ pháp đơn giản, dễ đọc
- Viết mã ngắn gọn hơn những chương trình tương đương được viết trong C, C++, C#, Java,...
- Có các bộ thư viện chuẩn và các module ngoài, đáp ứng gần như mọi nhu cầu lập trình
- Có khả năng chạy trên nhiều nền tảng (Windows, Linux, Unix, OS/2, Mac, Amiga, máy ảo .NET, máy ảo Java, Nokia Series 60,...)
- Có cộng đồng lập trình rất lớn, hệ thống thư viện chuẩn, mã nguồn chia sẻ nhiều



NHƯỢC ĐIỂM Python

- Chương trình chạy chậm
- Giao tiếp với các thư viện viết bằng các ngôn ngữ khác tương đối khó khăn
- Yếu trong hỗ trợ tính toán trên di động
- Gõ lỗi đòi hỏi kinh nghiệm
- Kém hỗ trợ các cơ sở dữ liệu



Cách thực hiện câu lệnh, chương trình Python



Cài đặt

www.python.org/downloads/

Python PSF Docs PyPI Jobs Community

python™ Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

Download the latest version for Windows

[Download Python 3.10.2](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases



Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	bugfix	2020-10-05	2025-10	PEP 596



Khởi chạy

❖ Python có 2 chế độ thực thi

- Chế độ thực thi: chỉ ra chương trình cần thực hiện
- Chế độ dòng lệnh: chạy từng lệnh một



Khởi chạy

Chế độ thực thi:

Trình dịch python sẽ nạp, dịch và chạy chương trình đó
“python chuongtrinh1.py” chạy file chuongtrinh1.py

```
Microsoft Windows [Version 10.0.19044.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>python chuongtrinh1.py
Nhap mon Data Science

C:\Users\ADMIN>
```



Khởi chạy

☐ Chế độ dòng lệnh: chạy từng lệnh một

- Chế độ dòng lệnh: “python”
- Lúc này trình thông dịch python sẽ chờ người dùng gõ từng dòng lệnh
- Gõ dòng lệnh nào xong, python chạy liền dòng đó
- Chấm dứt chế độ này bằng cách gõ lệnh: “**quit()**”

```
Microsoft Windows [Version 10.0.19044.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>python
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> a=20*10
>>> b=a+25
>>> b
225
>>> quit()

C:\Users\ADMIN>
```



Soạn thảo mã python

- *Làm thế nào để viết chương trình python (.py)?*
- *Dùng phần mềm soạn thảo văn bản thô (txt) bất kỳ để soạn và lưu file ở dạng .py rồi dịch bằng python*

Có những phần mềm thích hợp cho việc này hơn

- IDLE
- Sublime Text
- Notepad++
- PyCharm
- Spyder
- Rodeo
- ...

```
File Edit Selection Find View Goto Tools Project Preferences Help
C:\DNN\lesson3a.py (DNN) - Sublime Text
FOLDERS
  DNN
    lesson2a.py
    lesson2b.py
    lesson2c.py
    lesson2d.py
    lesson2f.py
    lesson3a.py
1 import matplotlib.image as mpimg
2
3 # First, Load the image
4 filename = "/tmp/MarshOrchid.jpg"
5 image = mpimg.imread(filename)
6
7 # Print out its shape
8 print(image.shape)
9
10 import matplotlib.pyplot as plt
11 plt.imshow(image)
12 plt.show()
Line 6, Column 1
Tab Size: 4
Python
```



Biên dịch mã python

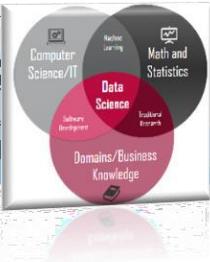
- Trường hợp cần thiết, mã python có thể được biên dịch, kết quả dịch là chương trình dạng bytecode cho máy ảo python (*Tương tự như trường hợp của ngôn ngữ java*)
- Mã lệnh dịch được lưu vào file với đuôi .pyc
- Việc biên dịch có nhiều lợi điểm, chẳng hạn như khi sử dụng câu lệnh import một thư viện nào đó, thì có thể sử dụng luôn mã pyc có sẵn thay vì phải dịch lại từ đầu

Biến Kiểu dữ liệu Khối lệnh



Biến

- Biến = vùng bộ nhớ được đặt tên (để dễ thao tác)
- Biến trong python:
 - Có tên, phân biệt chữ hoa/thường
 - Không cần khai báo trước
 - Không cần chỉ ra kiểu dữ liệu
 - Có thể thay đổi sang kiểu dữ liệu khác
 - Nên gán giá trị ngay khi bắt đầu xuất hiện
- Ví dụ:
 - n = 12 # biến n là kiểu nguyên
 - n = n + 0.1 # biến n chuyển sang kiểu thực



Biến

- Tên biến có thể chứa chữ cái hoặc chữ số hoặc gạch dưới (_), kí tự bắt đầu không được dùng chữ số
 - *Không được trùng với từ khóa (tất nhiên)*
 - *Từ python 3 được dùng chữ cái unicode*
- Tất cả mọi biến trong python đều là các đối tượng, vì thế nó có kiểu và vị trí trong bộ nhớ (id)

```
C:\Dev\Python36\python.exe
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 100
>>> b = 1.
>>> type(a), type(b)
(<class 'int'>, <class 'float'>)
>>> id(a), id(b)
(1961916992, 1753621799176)
>>>
```



Dữ liệu kiểu chuỗi

- Dữ liệu kiểu chuỗi rất quan trọng trong lập trình python và trong các vấn đề của khoa học dữ liệu
- Khai báo dữ liệu kiểu chuỗi có thể nằm bên trong cặp nháy đơn ('), hoặc nháy kép ("") hoặc 3 dấu nháy kép liên tiếp ("""")

```
name = 'matt'  
# chuỗi trong nó có chứa dấu nháy đơn  
with_quote = "I ain't gonna"  
# chuỗi có nội dung nằm trên 2 dòng  
longer = """This string has multiple lines in it"""
```



Escape sequence

- Escape sequence là một phương pháp để viết các kí tự đặc biệt trong python
- *Tương tự như các ngôn ngữ lập trình khác*

Tên	Kí hiệu	Giải thích
Alert	\a	Phát ra một tiếng bíp
Backspace	\b	Đưa con trỏ về lại một khoảng trắng
Newline	\n	Đưa con trỏ xuống dòng tiếp theo
Horizontal tab	\t	In một horizontal tab
Single quote	\'	In ra kí tự '
Double quote	\"	In ra kí tự "
Blackslash	\\\	In ra kí tự \



Chuỗi tràn

- Đặt vấn đề: bạn thao tác với các đường dẫn file, các chuỗi này sẽ có dạng **Ô đĩa : \Thư mục\Thư mục**
- Nếu tên thư mục bắt đầu với các chữ cái **t, n, a, v, b, ...** và kết hợp với kí tự ****. thành escape sequence, điều này có thể gây nhầm lẫn khi viết nội dung các chuỗi
- Python cho phép sử dụng một dạng chuỗi, gọi là
- chuỗi tràn, bằng cách bỏ qua escape sequence
- Cú pháp: **r' nội dung chuỗi'**

```
>>> a = r'\neu moi ngay' # chuoi tran, bo qua Escape Sequence \n
>>> print (a)
\nieu moi ngay
>>>
```

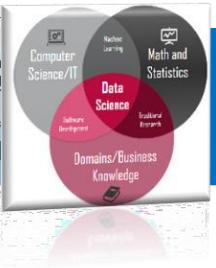


Chú thích (comment)

- Python sử dụng kí tự **#** để chú thích các đoạn code
- Tất cả các nội dung sau kí tự **#** sẽ không được dịch

```
Command Prompt - python
Microsoft Windows [Version 10.0.19044.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>python
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Chuong trinh python") #Day la cau lenh in ra chuoi
Chuong trinh python
>>>
```



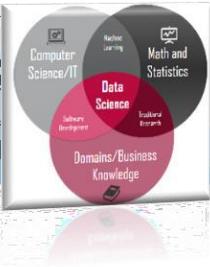
Nhập và Xuất dữ liệu



Xuất dữ liệu

Sử dụng hàm print để in dữ liệu ra màn hình

```
>>> print(42)
42
>>> print("a = ", a)
a = 3.564
>>> print("a = \n", a)
a =
3.564
>>> print("a","b")
a b
>>> print("a","b",sep="")
ab
>>> print(192,168,178,42,sep=". ")
192.168.178.42
>>> print("a","b",sep=":-) ")
a:-)b
```



Nhập dữ liệu

- Sử dụng hàm input để nhập dữ liệu từ bàn phím

```
name = input("What's your name? ")  
print("Nice to meet you " + name + "!")  
age = input("Your age? ")  
print("You are already " + age + " years old, " + name  
+ "!")
```

- Có thể kết hợp chuyển kiểu nếu muốn tường minh

```
age = int(input("Your age? "))  
print("You are already %d years old!", age)
```



Ví dụ minh họa



Giải phương trình bậc 2

Cho phương trình sau: $ax^2+bx+c=0$ ($a \neq 0$), được gọi là phương trình bậc 2 với ẩn là x .

Công thức nghiệm: Ta gọi $\Delta=b^2-4ac$. Khi đó:

- > $\Delta>0$: phương trình tồn tại 2 nghiệm:

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a}, \quad x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

- > $\Delta=0$, phương trình có nghiệm kép $x=-b/2a$
- > $\Delta<0$, phương trình đã cho vô nghiệm.

Trong trường hợp $b=2b'$, để đơn giản ta có thể tính $\Delta'=b'^2-ac$, tương tự như trên:

- > $\Delta'>0$: phương trình có 2 nghiệm phân biệt.

$$x_1 = \frac{-b' + \sqrt{\Delta'}}{a}, \quad x_2 = \frac{-b' - \sqrt{\Delta'}}{a}$$

- > $\Delta'=0$: phương trình có nghiệm kép $x=-b'/a$
- > $\Delta'<0$: phương trình vô nghiệm.



Giải phương trình bậc 2

```
a = float(input("A = "))  
b = float(input("B = "))  
c = float(input("C = "))  
delta = b*b-4*a*c
```

Nhập a,b,c kiểu số thực và tính delta

```
if delta==0:  
    print("Nghiệm kép: x = ", str(-b/2/a))
```

Biện luận các trường hợp của delta

```
if delta<0:  
    print("Phương trình vô nghiệm")
```

Các khối lệnh con được viết thut vào so với khối cha

```
if delta>0:  
    print("x1 = " + str((-b+delta**0.5)/2/a))  
    print("x2 = " + str((-b-delta**0.5)/2/a))
```

Tính căn bậc 2 bằng phép lũy thừa 0.5



Bài 2.2

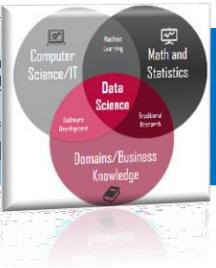
Kiểu dữ liệu & phép toán

- 1. Kiểu dữ liệu và phép toán liên quan*
- 2. Cấu trúc rẽ nhánh*
- 3. Vòng lặp*
- 4. Hàm*
- 5. Bài tập*



Ôn lại

- Biến không cần khai báo trước, không cần chỉ kiểu
- Dữ liệu chuỗi nằm trong cặp nháy đơn ('), nháy kép (""), hoặc ba dấu nháy ("""") – nếu viết nhiều dòng
- Sử dụng chuỗi thoát (escape sequence) để khai báo các ký tự đặc biệt
 - Sử dụng chuỗi “trần”: **r "nội dung"**
- Dùng dấu thăng (#) để viết dòng chú thích
- Dùng hàm **print** để in dữ liệu
- Dùng hàm **input** để nhập dữ liệu
 - Có thể kết hợp với hàm chuyển đổi kiểu



Kiểu dữ liệu & phép toán liên quan



Kiểu số

❖ Python viết số nguyên theo nhiều hệ cơ số

- `A = 1234 # hệ cơ số 10`
- `B = 0xAF1 # hệ cơ số 16`
- `C = 0o772 # hệ cơ số 8`
- `D = 0b1001 # hệ cơ số 2`

❖ Chuyển đổi từ số nguyên thành string ở các hệ cơ số khác nhau

- `K = str(1234) # chuyển thành str ở hệ cơ số 10`
- `L = hex(1234) # chuyển thành str ở hệ cơ số 16`
- `M = oct(1234) # chuyển thành str ở hệ cơ số 8`
- `N = bin(1234) # chuyển thành str ở hệ cơ số 2`



Kiểu số

- ❖ Từ python 3, số nguyên không có giới hạn số chữ số
- ❖ Số thực (float) trong python có thể viết kiểu thông thường hoặc dạng khoa học
 - `x = 12.34`
 - `y = 314.15279e-2` # dạng số nguyên và phần mũ 10
- ❖ Python hỗ trợ kiểu số phức, với chữ j đại diện cho phần ảo
 - `A = 3+4j`
 - `B = 2-2j`
 - `print(A+B)` # sẽ in ra (5+2j)

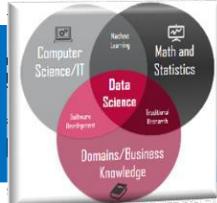


Phép toán

- ❖ Python hỗ trợ nhiều phép toán số, logic, so sánh và phép toán bit
 - Các phép toán số thông thường: `+`, `-`, `*`, `%`, `**`
 - Python có 2 phép chia:
 - **Chia đúng (/)**: `10/3` # `3.3333333333333335`
 - **Chia nguyên (//)**: `10/3` # `3` (nhanh hơn phép /)
 - Các phép logic: `and`, `or`, `not`
 - Python không có phép `xor` logic, trường hợp muốn tính phép xor thì thay bằng phép so sánh khác (`bool(a) != bool(b)`)
 - Các phép so sánh: `<`, `<=`, `>`, `>=`, `!=`, `==`
 - Các phép toán bit: `&`, `|`, `^`, `~`, `<<`, `>>`
 - Phép kiểm tra tập (`in`, `not in`): `1 in [1, 2, 3]`



Cáu trúc rẽ nhánh



Câu trúc rẽ nhánh if-else

if expression:

```
# if-block
```

if expression:

```
# if-block
```

elif 2-expression:

```
# 2-if-block
```

elif 3-expression:

```
# 3-if-block
```

elif n-expression:

```
# n-if-block
```

if expression:

```
# if-block
```

else:

```
#else-block
```

if expression:

```
# if-block
```

elif 2-expression:

```
# 2-if-block
```

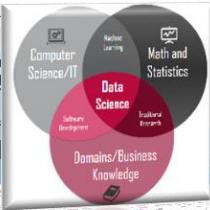
...

elif n-expression:

```
# n-if-block
```

else:

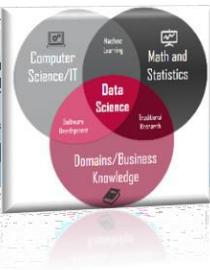
```
#else-block
```



Cấu trúc rẽ nhánh if-else

Chú ý: python nhạy cảm với việc viết khối mã

```
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = int(input("Your age? "))
print("You are already", age, "years old,", 
name, "!")
if age>=18:
    print("Đủ tuổi đi bầu cử")
    if age>100:
        print("Có vẻ sai sai!")
else:
    print("Nhỏ quá")
```



“phép toán” if

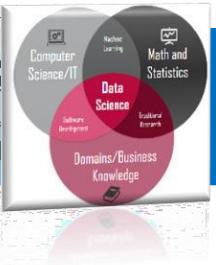
- Python có cách sử dụng **if** khá kì cục (theo cách nhìn của những người đã biết lệnh **if** trong một ngôn ngữ khác)
- Nhưng cách viết này rất hợp lý xét về mặt ngôn ngữ và cách đọc điều kiện logic

Cú pháp: **A if <điều-kiện> else B**

Giải thích: phép toán trả về A nếu điều-kiện là đúng, ngược lại trả về B

Ví dụ:

x = A if A > B else B # x là max của A và B



Vòng lặp



Vòng lặp while

while expression:

```
# while-block
```

while expression:

```
# while-block-1
```

continue

```
# while-block-2
```

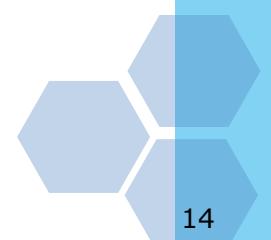
while expression:

```
# while-block
```

else:

```
#else-block
```

- Lặp **while** trong python tương đối giống trong các ngôn ngữ khác
- Trong khối lệnh while (lệnh lặp nói chung) có thể dung **continue** hoặc **break** để về đầu hoặc cuối khối lệnh
- Khối “**else**” sẽ được thực hiện sau khi toàn bộ vòng lặp đã chạy xong
 - Khối này sẽ không chạy nếu vòng lặp bị “**break**”





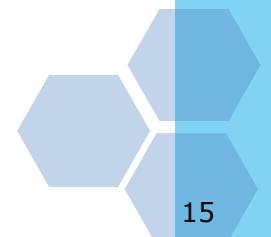
Vòng lặp for

```
for variable_1, variable_2, variable_n in sequence:  
    # for-block
```

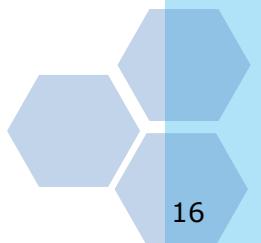
```
for variable_1, variable_2, variable_n in sequence:  
    # for-block  
else:  
    #else-block
```

- Vòng lặp for sử dụng để duyệt danh sách, khối else làm việc tương tự như ở vòng lặp while
- Dùng hàm **range(a, b)** để tạo danh sách gồm các số từ a đến b-1, hoặc tổng quát hơn là **range(a, b, c)** trong đó c là bước nhảy

```
for d in range(10,20): # in các số từ 10 đến 19  
    print(d)  
for d in range(20,10,-1): # in các số từ 20 đến 11  
    print(d)
```



Hàm





Hàm

- Cú pháp khai báo hàm rất đơn giản

```
def <tên-hàm>(<danh-sách-tham-số>) :  
<lệnh 1>
```

...

```
<lệnh n>
```

- Ví dụ: hàm tính tích 2 số

```
def tich(a, b) :  
    return a*b
```

Hàm trả về kết quả bằng lệnh **return**, nếu không trả về thì coi như trả về **None**



Hàm

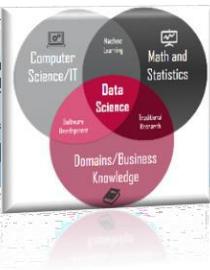
- Hàm có thể chỉ ra giá trị mặc định của tham số
`def tich(a, b = 1):
 return a*b`
- Như vậy với hàm trên ta có thể gọi thực hiện nó:
`print(tich(10, 20)) # 200
print(tich(10)) # 10
print(tich(a=5)) # 5
print(tich(b=6, a=5)) # 30`
- Chú ý: các tham số có giá trị mặc định phải đứng cuối danh sách tham số



Bài 2.3

Chuỗi và danh sách

1. *Kiểu dữ liệu tuần tự (sequential data type)*
2. *String (chuỗi)*
3. *List (danh sách)*
4. *Tuple (hàng)*
5. *Range (miền)*



Ôn lại

- Python hỗ trợ kiểu số rất mạnh và nhiều loại phép tính phong phú
- Sử dụng **if** cho tất cả các nhu cầu rẽ nhánh
- Phép toán **if** cho phép viết lệnh một cách tự nhiên
- Vòng lặp **while** tương tự như các ngôn ngữ khác
 - Ngoại trừ việc có thể có thêm khối **else**
- Vòng lặp **for** cho phép lần lượt thực hiện lặp với các giá trị nhận được từ một danh sách
- Sử dụng từ khóa **def** để định nghĩa một hàm, hàm có thể có các tham số mặc định



Kiểu dữ liệu tuần tự (sequential data type)



Kiểu dữ liệu tuần tự

- **Kiểu dữ liệu tuần tự:** kiểu dữ liệu chứa bên trong nó các dữ liệu con nhỏ hơn và thường được xử lý bằng cách lấy ra từng phần-tử-một (bằng vòng for)
 - Các kiểu dữ liệu chứa bên trong nó các dữ liệu nhỏ hơn thường được gọi là các container (bộ chứa)
 - Khái niệm “tuần tự” nhấn vào việc xử lý từng phần tử một, nhưng không nhất thiết đây là cách xử lý duy nhất
- Có 3 kiểu tuần tự thông dụng là **list**, **tuple** và **range**
- Có nhiều kiểu khác như **string**, **bytes**, **bytearray**,...hoặc các lập trình viên có thể tự tạo kiểu riêng theo nhu cầu .



String (chuỗi)



Kiểu chuỗi

- Một chuỗi được xem như một hàng (tuple) các chuỗi con độ dài 1
 - Trong python không có kiểu kí tự (character)
 - Nội dung của chuỗi không thay đổi được, khi ghép thêm nội dung vào chuỗi thực chất là tạo ra chuỗi mới
- Hàm **len(s)** trả về độ dài (số chữ) của s
Phép toán với chuỗi:
 - Phép nối chuỗi (+): **s = "Good" + " " + "Morning!"**
 - Phép nhân bản (*): **s = "AB" * 3 # số nguyên**
 - Kiểm tra nội dung: **s in '1ABABABCD' # True**



Chỉ mục trong chuỗi

- Các phần tử (các chữ) trong chuỗi được đánh số thứ tự và có thể truy cập vào từng phần tử theo chỉ số.
- Python duy trì 2 cách đánh chỉ mục khác nhau:
 - *Đánh từ trái qua phải: chỉ số đánh từ 0 trở đi cho đến cuối chuỗi*
 - *Đánh từ phải qua trái: chỉ số đánh từ -1 giảm dần về đầu chuỗi*

D	H	T	H	U	Y	L	O	I
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1



Cắt chuỗi

- Dựa trên chỉ mục, phép cắt chuỗi cho phép lấy nội dung bên trong của chuỗi bằng cú pháp như sau
 - <chuỗi>[vị trí A : vị trí B]**
 - <chuỗi>[vị trí A : vị trí B : bước nhảy]**
- Giải thích:
 - Tạo chuỗi con bắt đầu từ <vị-trí-A> đến trước <vị-trí-B>
(Tức là chuỗi con sẽ không gồm vị trí B)
 - Nếu không ghi <vị-trí-A> thì mặc định là lấy từ đầu
 - Nếu không ghi <vị-trí-B> thì mặc định là đến hết chuỗi
 - Nếu không ghi <bước-nhảy> thì mặc định bước là 1
 - Nếu <bước-nhảy> giá trị âm thì sẽ nhận chuỗi ngược lại



Cắt chuỗi

```
s = '0123456789'  
print(s[3:6]) # 345  
print(s[3:]) # 3456789  
print(s[:6]) # 012345  
print(s[-7:-4]) # 345  
print(s[-4:-7]) #  
print(s[-4:-7:-1]) # 654  
print(s[:len(s)]) # 0123456789  
print(s[:len(s)-1]) # 012345678  
print(s[:]) # 0123456789  
print(s[len(s):: -1]) # 9876543210  
print(s[len(s)-1:: -1]) # 9876543210  
print(s[len(s)-2:: -1]) # 876543210
```



Định dạng chuỗi

- Dùng toán tử %: **<chuỗi> % (<các tham số>)**
 - Bên trong <chuỗi> có các kí hiệu đánh dấu nơi đặt lần lượt các tham số
 - Nếu đánh dấu %s: thay thế bằng tham số dạng chuỗi
 - Nếu đánh dấu %d: thay thế bằng tham số dạng nguyên
 - Nếu đánh dấu %f: thay thế bằng tham số dạng thực
- **Ví dụ:**
 - `"Chao %s, gio la %d gio" % ('txnam', 10)`
 - `"Can bac 2 cua 2 = %f" % (2**0.5)`
 - `"Can bac 2 cua 2 = %10.3f" % (2**0.5)`
 - `"Can bac 2 cua 2 = %10f" % (2**0.5)`
 - `"Can bac 2 cua 2 = %.7f" % (2**0.5)`



Định dạng chuỗi

- Python cho phép định dạng chuỗi ở dạng f-string

```
myname = 'DHNTT'
s = f'This is {myname}.' # 'This is DHNTT.'
w = f'{s} {myname}' # 'This is DHNTT. DHNTT '
z = f'{{s}} {s}' # '{s} This is DHNTT.'
```

- Mạnh mẽ nhất là định dạng bằng format

```
# điền lần lượt từng giá trị vào giữa cặp ngoặc nhọn
'a: {}, b: {}, c: {}'.format(1, 2, 3)
# điền nhưng không lần lượt
'a: {1}, b: {2}, c: {0}'.format('one', 'two', 'three')
'two same values: {0}, {0}'.format(1, 2)
# điền và chỉ định từng giá trị
'1: {one}, 2: {two}'.format(one=111, two=222)
```



Định dạng chuỗi

Định dạng bằng format cho phép căn lề phong phú

```
# căn giữa: 'aaaa'
'{:^10}'.format('aaaa')
# căn lề trái: 'aaaa      '
'{:<10}'.format('aaaa')
# căn lề phải '      aaaa'
'{:>10}'.format('aaaa')
# căn lề phải, thay khoảng trắng bằng -: '-----aaaa'
'{:->10}'.format('aaaa')
# căn lề trái, thay khoảng trắng *: 'aaa*****'
'{:*<10}'.format('aaaa')
# căn giữa, thay khoảng trắng bằng +: '+++aaaa+++' 
'{:+^10}'.format('aaaa')
```



Các phương thức của chuỗi

- Các phương thức chỉnh dạng
 - `capitalize()`: viết hoa chữ cái đầu, còn lại viết thường
 - `upper()`: chuyển hết thành chữ hoa
 - `lower()`: chuyển hết thành chữ thường
 - `swapcase()`: chữ thường thành hoa và ngược lại
 - `title()`: chữ đầu của mỗi từ viết hoa, còn lại viết thường
- Các phương thức căn lề
 - `center(width [,fillchar])`: căn lề giữa với độ dài width
 - `rjust(width [,fillchar])`: căn lề phải
 - `ljust(width [,fillchar])`: căn lề trái



Các phương thức của chuỗi

- Các phương thức cắt phần dư
 - **strip([chars])**: loại bỏ những ký tự đầu hoặc cuối chuỗi thuộc vào danh sách [chars], hoặc ký tự trống
 - **rstrip([chars])**: làm việc như strip nhưng cho bên phải
 - **lstrip([chars])**: làm việc như strip nhưng cho bên trái
- Tách chuỗi
 - **split(sep, maxsplit)**: tách chuỗi thành một danh sách, sử dụng dấu ngăn cách sep, thực hiện tối đa maxsplit lần
 - *Tách các số nhập vào từ một dòng: `input("Test: ").split(',')`*
 - **rsplit(sep, maxsplit)**: thực hiện như split nhưng theo hướng ngược từ phía cuối chuỗi



Các phương thức của chuỗi

■ Các phương thức khác

- **join(list)**: ghép các phần tử trong list bởi phần gạch nối là nội dung của chuỗi, ví dụ: '-.join(['1', '2', '3'])
- **replace (old, new [,count])**: thế nội dung old bằng nội dung new, tối đa count lần
- **count(sub, [start, [end]])**: đếm số lần xuất hiện của sub
- **startswith(prefix)**: kiểm tra đầu có là prefix không
- **endswith(prefix)**: kiểm tra cuối có là prefix không
- **find(sub[, start[, end]])**: tìm vị trí của sub (-1: không có)
- **rfind(sub[, start[, end]])**: như find nhưng tìm từ cuối
- **islower()**, **isupper()**, **istitle()**, **isdigit()**, **isspace()**



List (danh sách)



Giới thiệu và khai báo

- List = dãy các đối tượng (một loại **array** đa năng)
- Khai báo trực tiếp bằng cách liệt kê các phần tử con đặt trong cặp ngoặc vuông ([]), ngăn cách bởi phẩy

```
[1, 2, 3, 4, 5] # list 5 số nguyên
['a', 'b', 'c', 'd'] # list 4 chuỗi
[[1, 2], [3, 4]] # list 2 list con
[1, 'one', [2, 'two']] # list hỗn hợp
[] # list rỗng
```

- Kiểu chuỗi (**str**) trong python có thể xem như một list đặc biệt, bên trong gồm toàn các **str** độ dài 1



Khởi tạo list

- Tạo list bằng constructor

```
11 = list([1, 2, 3, 4]) # list 4 số nguyên
12 = list('abc') # list 3 chuỗi con
13 = list() # list rỗng
```

- Tạo list bằng list comprehension

```
# list 1000 số nguyên từ 0 đến 999
x = [n for n in range(1000)]
# list gồm 10 list con là các cặp [x, x2]
# với x chạy từ 0 đến 9
Y = [[x, x*x] for x in range(10)]
```



Phép toán, chỉ mục và cắt

- Giữa list và str có sự tương đồng nhất định
 - List cũng hỗ trợ 3 phép toán: ghép nối (+), nhân bản (*) và kiểm tra nội dung (in)
 - List sử dụng hệ thống chỉ mục và các phép cắt phần con tương tự như str
- Điểm khác biệt là nội dung của list có thể thay đổi

```
ll = list([1, 2, 3, 4])
ll[-1] = list('abc')
print(ll)
# [1, 2, 3, ['a', 'b', 'c']]
```



Các phương thức của list

Một số phương thức thường hay sử dụng

- `count(sub, [start, [end]])`: đếm số lần xuất hiện của sub
- `index(sub[, start[, end]])`: tìm vị trí xuất hiện của sub, trả về ValueError nếu không tìm thấy
- `clear()`: xóa trống list
- `append(x)`: thêm x vào cuối list
- `extend(x)`: thêm các phần tử của x vào cuối list
- `insert (p, x)`: chèn x vào vị trí p trong list
- `pop(p)`: bỏ phần tử thứ p ra khỏi list (trả về giá trị của phần tử đó), nếu không chỉ định p thì lấy phần tử cuối .



Các phương thức của list

Một số phương thức thường hay sử dụng

- `copy()`: tạo bản sao của list (tương tự `list[:]`)
- `remove(x)`: bỏ phần tử đầu tiên trong list có giá trị x, báo lỗi `ValueError` nếu không tìm thấy
- `reverse()`: đảo ngược các phần tử trong list
- `sort(key=None, reverse=False)`: mặc định là sắp xếp các phần tử từ bé đến lớn trong list bằng cách so sánh trực tiếp giá trị

```
x = "Nguyen Van An".split()  
x.sort(key=str.lower)  
print(x)
```



Tuple (hàng)



Tuple là một dạng readonly list

- Tuple = dãy các đối tượng (list), nhưng không thể bị thay đổi giá trị trong quá trình tính toán
- Như vậy str giống tuple nhiều hơn list
- Khai báo trực tiếp bằng cách liệt kê các phần tử con đặt trong cặp ngoặc tròn (), ngăn cách bởi phẩy

```
(1, 2, 3, 4, 5) # tuple 5 số nguyên  
('a', 'b', 'c', 'd') # tuple 4 chuỗi  
(1, 'one', [2, 'two']) # tuple hỗn hợp  
(1,) # tuple 1 phần tử  
() # tuple rỗng
```



Tuple và list nhiều điểm giống nhau

- Tuple có thể tạo bằng constructor hoặc tuple comprehension
- Tuple hỗ trợ 3 phép toán: `+`, `*`, `in`
- Tuple cho phép sử dụng chỉ mục và cắt
- Các phương thức thường dùng của tuple
 - `count(v)` : đếm số lần xuất hiện của v trong tuple
 - `index(sub[, start[, end]])` : tương tự như str và list
- Tuple khác list ở điểm nào?
 - *Chiếm ít bộ nhớ hơn*
 - *Nhanh hơn*



Range (miền)



Range là một tuple đặc biệt?

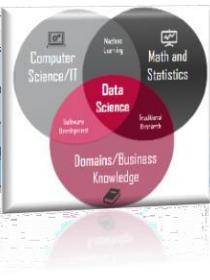
- Chúng ta đã làm quen với range khi dùng vòng for
 - **range(stop)** : tạo miền từ 0 đến stop-1
 - **range(start, stop[, step])** : tạo miền từ start đến stop-1, với bước nhảy là step
 - Nếu không chỉ định thì step = 1
 - Nếu step là số âm sẽ tạo miền đếm giảm dần ($start > stop$)
- Vậy range khác gì một tuple đặc biệt
 - Range chỉ chứa số nguyên
 - Range nhanh hơn rất nhiều
 - Range chiếm ít bộ nhớ hơn
 - Range vẫn hỗ trợ chỉ mục và cắt (nhưng khá đặc biệt)



Bài 2.4

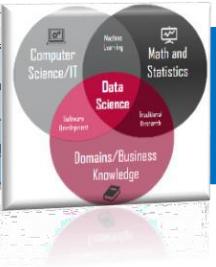
Tập hợp

1. ***Set (tập hợp) và Frozenset (tập hợp tĩnh)***
2. ***Dictionary (từ điển)***
3. ***Module và Package***



Ôn lại

- **Kiểu dữ liệu tuần tự**: là kiểu dữ liệu cho phép xử lý dữ liệu bằng cách xử lý từng-phần-tử-con-một
- **Danh sách (list)**: dãy các phần tử, khai báo bên trong cặp ngoặc vuông, nội dung có thể thay đổi
- **Hàng (tuple)**: dãy các phần tử, khai báo bên trong cặp ngoặc tròn, nội dung cố định (không thay đổi)
- **Range (miền)**: có thể xem như một dạng tuple đặc biệt gồm các số nguyên, chuyên dùng cho lặp for
- **Chuỗi (str)**: một dạng tuple đặc biệt gồm nhiều chuỗi có độ dài 1 ký tự

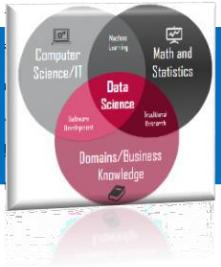


Ôn lại

- Các kiểu dữ liệu này có chung đặc điểm:
 - *Bản chất là các đối tượng, được viết một cách tự nhiên*
 - *Rất nhiều phương thức hỗ trợ việc xử lý*
 - *Sử dụng chung 2 hệ thống chỉ mục (âm và dương)*
 - *Sử dụng chung kĩ thuật cắt lát (bằng chỉ mục)*
 - *Sử dụng chung 3 phép toán: +, *, in*
- Chuỗi có rất nhiều kĩ thuật định dạng nội dung
- List và Tuple có thể được tạo bằng comprehension
- Nhiều hàm dựng sẵn (built-in) xử lý các kiểu dữ liệu này: `len`, `max`, `min`, `all`, `any`, `filter`, `sorted`, `sum`, `zip`, ...



Set (tập hợp) và Frozenset (tập hợp tĩnh)



Giới thiệu và khởi tạo

- **Set** = tập hợp các đối tượng (không trùng nhau)
- Khai báo trực tiếp bằng cách liệt kê các phần tử con đặt trong cặp ngoặc nhọn {}, ngăn cách bởi phẩy.

ví dụ: `basket = {'apple', 'orange', 'apple', 'pear'}`
`print(basket)`

`#{'orange', 'pear', 'apple'} : xóa trùng nhau`

ví dụ: `a = {5,2,3,1,4}`
`print("a=", a) #a = {1, 2, 3, 4, 5}`

- Tạo set bằng constructor

```
s1 = set([1, 2, 3, 4]) # {1, 2, 3, 4}
s2 = set((1, 1, 1)) # {1}
s3 = s1 - s2 # {2, 3, 4}
s4 = set(range(1,100)) # {1, 2, 3, ..., 98, 99}
```



Khởi tạo Set

- **Tạo set** bằng set comprehension

```
# a = {'r', 'd'}
```

```
a = {x for x in 'abracadabra' if x not in 'abc'}
```

- **Set** không thể chứa những đối tượng mutable (có thể bị thay đổi), mặc dù chính set lại có thể thay đổi

```
a = set(([1,2],[2,3])) # lỗi list
```

```
a = set(((1,2),(2,3))) # {(1, 2), (2, 3)}
```

```
a.add("abc") # {(1, 2), "abc", (2, 3)}
```

- **Frozenset** giống set, nhưng không thể bị thay đổi

```
b = frozenset(((1,2),(2,3))) # {(1,2), (2,3)}
```

```
b.add("abc") # lỗi
```



Set

- **Set** Hỗn hợp

Ví dụ:

```
my_set = {8.0, "Sinh viên", (1, 2, 3)}  
print("ketqua_Set=", my_set)  
#Output: ketqua_Set= {'Sinh viên', 8.0, (1, 2, 3)}
```

Ví dụ:

```
# Khởi tạo my_set  
my_set = {1,3}  
my_set.add(2) #thêm  
my_set.update([2,3,4]) # Cập nhật  
my_set.update([4,5], {1,6,8}) # Cập nhật set  
print(my_set)  
  
# Output: {1, 2, 3, 4, 5, 6, 8}
```



Các phép toán trên set

```
a = set('abracadabra') # {'d', 'r', 'c', 'b', 'a'}
```

```
b = set('alacazam') # {'z', 'c', 'm', 'l', 'a'}
```

Phép Hiệu: thuộc a nhưng không thuộc b

```
print(a - b) # {'r', 'd', 'b'}
```

Phép Hợp: thuộc a hoặc b

```
# {'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
```

```
print(a | b)
```

Phép Giao: thuộc cả a và b

```
print(a & b) # {'a', 'c'}
```

Phép Xor: thuộc hoặc a, hoặc b nhưng không phải cả 2

```
# {'r', 'd', 'b', 'm', 'z', 'l'}
```

```
print(a ^ b)
```



Các phương thức của set

Một số phương thức thường hay sử dụng

- **add(e)** : thêm e vào tập hợp
- **clear()** : xóa mọi phần tử trong tập hợp
- **copy()** : tạo một bản sao của tập hợp
- **difference(x)** : tương đương với phép trừ đi x
- **difference_update(x)** : loại bỏ những phần tử trong x khỏi tập
- **discard(e)** : bỏ e khỏi tập
- **remove(e)** : bỏ e khỏi tập, báo lỗi nếu không tìm thấy e
- **union(x)** : tương đương với phép hợp với x
- **intersection(x)** : tương đương với phép giao với x



Các phương thức của set

Một số phương thức thường hay sử dụng

- **`isdisjoint(x)`** : trả về True nếu tập không có phần chung nào với **x**
- **`issubset(x)`** : trả về True nếu tập là con của **x**, tương đương với phép so sánh **<=x**
- **`issuperset(x)`** : trả về True nếu x là tập con của tập, tương đương với phép so sánh **>=x**
- **`pop()`** : lấy một phần tử ra khỏi tập (không biết trước)
- **`symmetric_difference(x)`** : tương đương với phép **$^{\wedge}x$**



Các phương thức của set

Các hàm thường dùng trên set bao gồm **all()**, **any()**, **enumerate()**, **len()**, **max()**, **min()**, **sorted()**, **sum()**. Chức năng của những hàm này khá giống list, tuple.

```
num = {1,3,5,6,8,20,7,8,29,33}
print("Số lớn nhất là :", max(num))
#Số lớn nhất là : 33
```



Vòng lặp for của set()

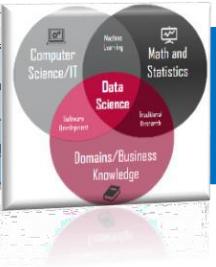
- Không thể truy cập các phần tử trong một Set bằng cách sử dụng chỉ mục, vì Set không lưu trữ các phần tử theo thứ tự nhập ban đầu.
- Sử dụng vòng lặp for để truy cập các phần tử của một Set.

Ví dụ:

```
setFruits = {'lemon', 'orange', 'apple', 'pear'}  
for x in setFruits:  
    print(x)
```

Ví dụ:

```
for letter in set("chivuong"):  
    print(letter)
```



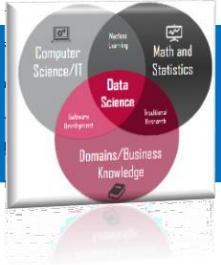
Dictionary (từ điển)



Dictionary

- Từ điển là một danh sách các từ (key) và định nghĩa của nó (value)
 - Yêu cầu các key không được trùng nhau, như vậy có thể xem từ điển như một loại set
- Từ điển có thể khai báo theo cú pháp của set

```
dict = {1: 'one', 2: 'two', 3: 'three'}
print(dict[1]) #'one'
dict[4]='four'
print(dict)
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```



Dictionary

Python Dictionary

```
py_dict = { 1: 'Apple', 2: 'OnePlus' }
```

The diagram illustrates a Python dictionary named `py_dict`. It contains two items: Item 1 with key `1` and value `'Apple'`, and Item 2 with key `2` and value `'OnePlus'`. The code is annotated with arrows pointing from the words "key" and "value" to the corresponding parts in the dictionary definition. Brackets below the code group the key-value pairs into "Item 1" and "Item 2".



Dictionary

- Chú ý: chỉ những loại dữ liệu immutable (không thể thay đổi) mới có thể dùng làm key của từ điển

```
dict = { (1,2,3) : "abc" , 3.1415 : "abc" }
```

```
dict = { [1,2,3] : "abc" } # lỗi
```

- Một số phép toán / phương thức thường dùng**

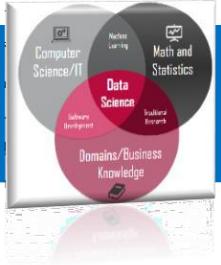
- **len(d)** : trả về độ dài của từ điển (số cặp key-value)
- **del d[k]**: xóa key k (và value tương ứng)
- **k in d**: trả về True nếu có key k trong từ điển
- **k not in d**: trả về True nếu không có key k trong từ điển
- **pop(k)** : trả về value tương ứng với k và xóa cặp này đi
- **popitem()** : trả về (và xóa) một cặp (key, value) tùy ý



Dictionary

Một số phép toán / phương thức thường dùng

- **get (k)** : lấy về value tương ứng với key k
 - Khác phép [] ở chỗ get trả về None nếu k không phải là key
- **update (w)** : ghép các nội dung từ từ điển w vào từ điển hiện tại (nếu key trùng thì lấy value từ w)
- **items ()** : trả về list các cặp (key, value)
- **keys ()** : trả về các key của từ điển
- **values ()** : trả về các value của từ điển
- **pop (k)** : trả về value tương ứng với k và xóa cặp này đi
- **popitem ()** : trả về (và xóa) một cặp (key, value) tùy ý



Dictionary (từ điển)

Dùng zip để ghép 2 list thành từ điển

```
l1 = ["a", "b", "c"]
```

```
l2 = [1,2,3]
```

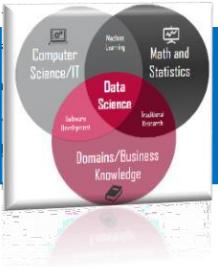
```
c = zip(l1, l2)
```

```
for i in c:  
    print(i)
```

```
('a', 1)
```

```
('b', 2)
```

```
('c', 3)
```



Dictionary (từ điển)

❖ Tạo mới từ điển

- Ví dụ: Tạo 1 từ điển

tạo một dict với key là các số nguyên

```
Dict = {1: 'Nguyen', 2: 'Van', 3: 'An'}
```

```
print(Dict)
```

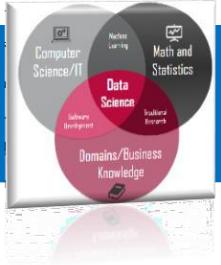
{1: 'Nguyen', 2: 'Van', 3: 'an'}

- # tạo một dict với key hỗn hợp

```
Dict = {'Name': 'dict', 1: [1, 2, 3, 4]}
```

```
print(Dict)
```

{'Name': 'dict', 1: [1, 2, 3, 4]}



Dictionary (từ điển)

❖ Thêm các giá trị vào từ điển

▪ Ví dụ:

tạo một từ điển

Dict = {}

thêm các giá trị

Dict[0] = 'Xin chao'

Dict[1] = 'lop hoc Khoa học du lieu'

print(Dict)

{0: ' Xin chao ', 1: ' Khoa học du lieu'}



Dictionary (từ điển)

❖ Truy cập giá trị của từ điển

▪ Ví dụ:

tạo một dict mới

```
Dict = {1: 'Lớp', 2: 'Khoa học', 3: 'Dữ liệu'}
```

truy cập theo cách thông thường

```
print(Dict[1])
```

Lớp

truy cập bằng phương thức get()

```
print(Dict.get(3))
```

Dữ liệu



Dictionary (từ điển)

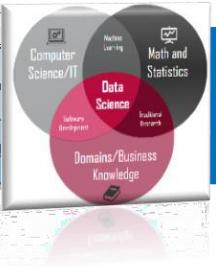
❖ Chuyển từ điển thành chuỗi: str()

```
Dict = { 'Ten' : 'An' , 'Tuoi' :20 } ;  
print("Chuỗi tương đương là : ", % str (Dict) )  
#Chuỗi tương đương là : { 'Ten' : 'An' , 'Tuoi' : 20 }
```

❖ Cập nhật 1 giá trị của từ điển: update()

Hàm *update()* sẽ thêm vào một giá trị mới, hoặc sẽ cập nhật một giá trị nếu có các khóa trùng nhau.

```
Dict = { 'Cúc' :18 , 'Lan' :19 , 'Hồng' :20 , 'Mai' :27 }  
Dict.update({ 'Lan' :21 }) #cập nhật tuổi cho Lan  
Dict.update({ 'Đào' :25 }) #thêm một phần tử mới là Đào  
print(Dict)  
#{ 'Cúc' :18 , 'Lan' :21 , 'Hồng' :20 , 'Mai' :27 , 'Đào' :25 }
```



Module và Package



Module

- Một file mã nguồn trong python được xem là một **module**
 - Có phần mở rộng **.py**
 - Mọi hàm, biến, kiểu trong file là các thành phần của module
- **Sử dụng module:**
 - Khai báo import module đó: **import <tên-module>**
 - Có thể khai báo import cùng lúc nhiều module cách nhau bởi *dấu phẩy*
 - Nếu muốn sử dụng các hàm, biến trong module thì cần viết tường minh tên module đó
 - Hoặc có thể import riêng một hàm hoặc nhiều hàm, cú pháp: **from <tên-module> import fuc1, fuc2, ..., fucN**



Package

- **Package** = Thư mục các module (lưu trữ vật lý)
 - `import numpy`
 - `A = array([1, 2, 3]) # lỗi`
 - `A = numpy.array([1, 2, 3]) # ok`
 - `import numpy as np`
 - `B = np.array([1, 2, 3]) # ok`
 - `from numpy import array`
 - `C = array([1, 2, 3]) # ok`
- **Module** và **Package** giúp quản lý tốt hơn mã nguồn
- **Gom, nhóm** các hàm, biến, lớp xử lý cùng một chủ đề, giúp phân cấp và sử dụng dễ dàng hơn



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 KHDL

- 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 (đô 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

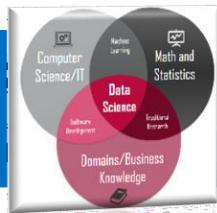


Một số gói python cho KHDL

- **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.

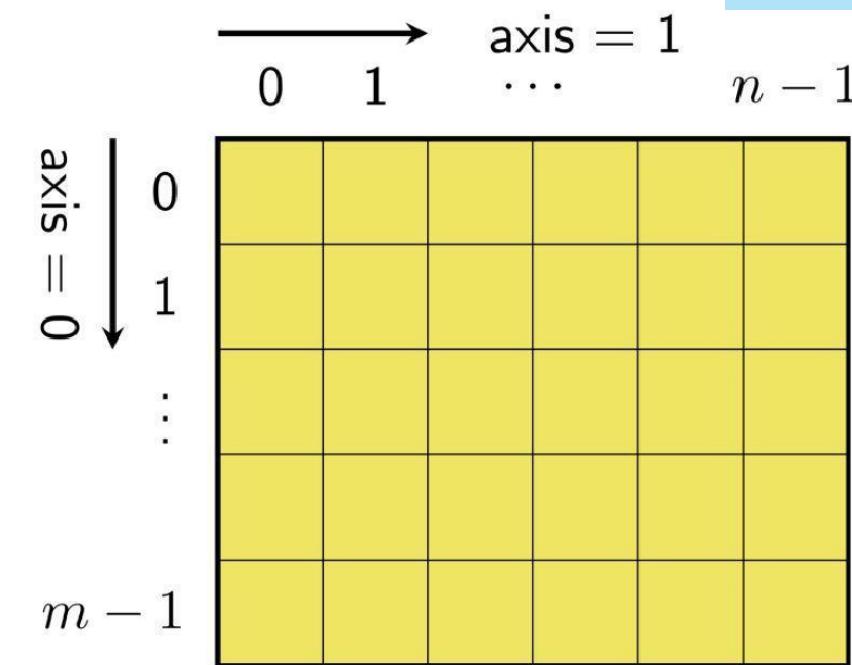
C:\Users\ADMIN>
```

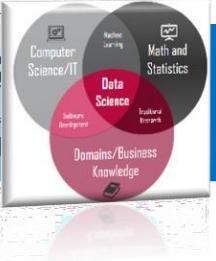
- Một số hệ thống python đã có sẵn numpy thì có thể bỏ qua bước này
- Cách đơn giản nhất để kiểm tra xem hệ thống đã cài numpy hay không là thử import gói xem có bị báo lỗi hay không: **import numpy as np**



Đặc điểm của NumPy

- Đối tượng chính của NumPy là các mảng đa chiều đồng nhất (homogeneous multidimention 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 mảng 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

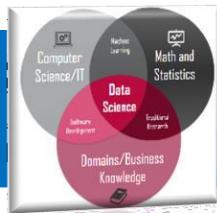
a = np.array([1, 2, 3]) # tạo mảng 1 chiều
print(type(a)) # in "<class 'numpy.ndarray'>"
print(a.shape) # in "(3,)"
print(a[0], a[1], a[2]) # in "1 2 3"
a[0] = 5
print(a) # in "[5, 2, 3]"
b = np.array([[1, 2, 3], [4, 5, 6]]) # tạo mảng 2 chiều
print(b.shape) # in "(2, 3)"
print(b[0, 0], b[0, 1], b[1,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

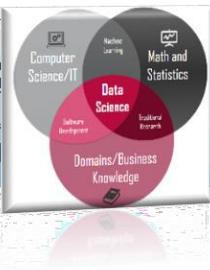


Truy cập theo chỉ số (slicing)

- 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.

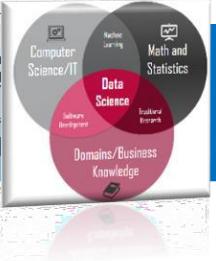
```
import numpy as np

# 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 dây

```
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 coi 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

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

```
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))
```



Thư viện matplotlib



Nội dung

1. Giới thiệu và cài đặt *matplotlib*
2. Vẽ biểu đồ đơn giản
3. Một số loại biểu đồ thông dụng trong *matplotlib*
 1. Biểu đồ dãBiểu đồ dạng cột (bar plot)
 2. Biểu đồ bánh (pie chart)
 3. Biểu đồ bánh (pie chart)
4. Một số chức năng hữu ích



Giới thiệu và cài đặt matplotlib



Giới thiệu matplotlib

- “**matplotlib**” là thư viện chuyên vẽ biểu đồ, mở rộng từ [numpy](#)
- Có mục tiêu đơn giản hóa tối đa công việc vẽ biểu đồ để “chỉ cần vài dòng lệnh”
- Hỗ trợ rất nhiều loại biểu đồ, đặc biệt là các loại được sử dụng trong nghiên cứu hoặc kinh tế như biểu đồ dòng, đường, tần suất ([histograms](#)), phổ, tương quan, [errorcharts](#), [scatterplots](#),...
- Cấu trúc của [matplotlib](#) gồm nhiều phần, phục vụ cho các mục đích sử dụng khác nhau.



Giới thiệu matplotlib

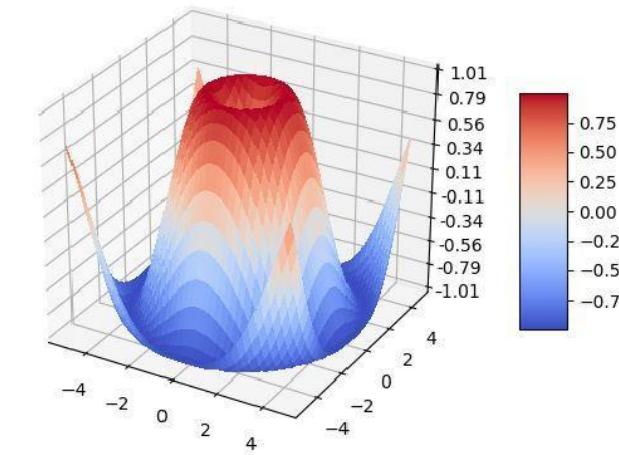
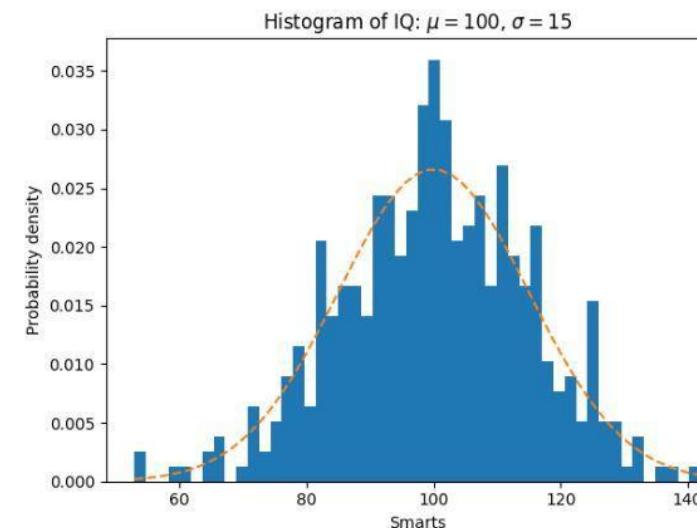
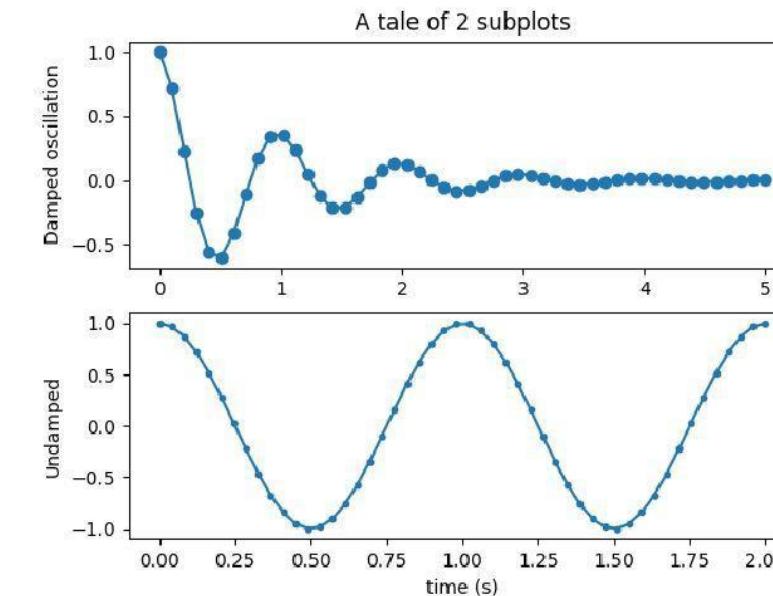
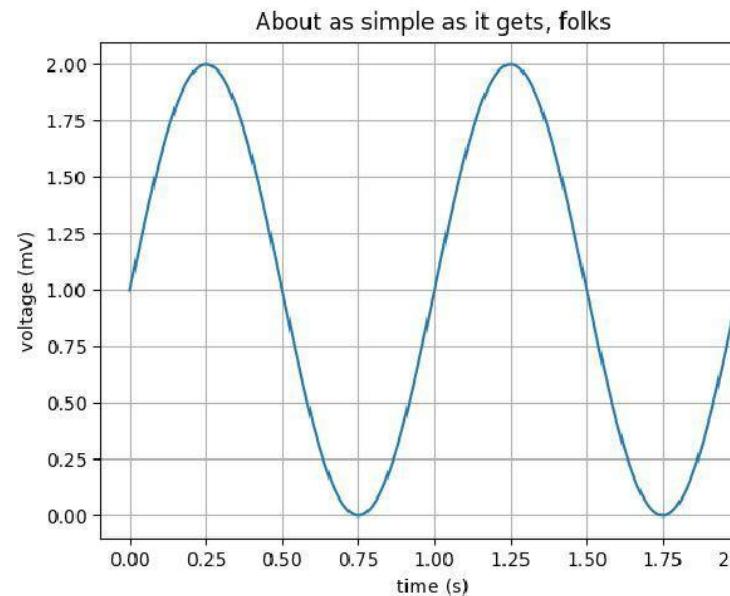
- Ngoài các [API](#) liên quan đến vẽ biểu đồ, matplotlib còn bao gồm một số interface: [Object-Oriented API](#), [The Scripting Interface \(pyplot\)](#), [The MATLAB Interface \(pylab\)](#)
 - ✓ Các interface này giúp chúng ta thuận tiện trong việc thiết lập chỉ số trước khi thực hiện vẽ biểu đồ Interface pylab hiện đã không còn được phát triển
 - ✓ Hầu hết các ví dụ trong slide này đều sử dụng pyplot
 - ✓ Sử dụng [Object-Oriented API](#) hoặc trực tiếp các API của matplotlib sẽ cho phép can thiệp sâu hơn vào việc vẽ biểu đồ (hầu hết project sẽ không có nhu cầu này).

Cài đặt: “pip install matplotlib”

```
Command Prompt
C:\Users\ADMIN>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.5.1-cp39-cp39-win_amd64.whl (7.2 MB)
    |████████| 7.2 MB 3.3 MB/s
Collecting packaging>=20.0
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
    |████████| 40 kB 2.7 MB/s
Collecting pillow>=6.2.0
  Downloading Pillow-9.0.0-cp39-cp39-win_amd64.whl (3.2 MB)
    |████████| 3.2 MB 3.2 MB/s
Collecting fonttools>=4.22.0
  Downloading fonttools-4.29.0-py3-none-any.whl (895 kB)
    |████████| 895 kB 6.4 MB/s
Collecting pyparsing>=2.2.1
  Downloading pyparsing-3.0.7-py3-none-any.whl (98 kB)
    |████████| 98 kB 2.4 MB/s
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    |████████| 247 kB 6.4 MB/s
Requirement already satisfied: numpy>=1.17 in c:\users\admin\appdata\local\programs\python\python39\lib\site-packages (from matplotlib) (1.22.1)
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.2-cp39-cp39-win_amd64.whl (52 kB)
    |████████| 52 kB 314 kB/s
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pyparsing, python-dateutil, pillow, packaging, kiwisolver, fonttools, cycler, matplotlib
Successfully installed cycler-0.11.0 fonttools-4.29.0 kiwisolver-1.3.2 matplotlib-3.5.1 packaging-21.3 pillow-9.0.0 pyparsing-3.0.7 python-dateutil-2.8.2 six-1.16.0
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>
```

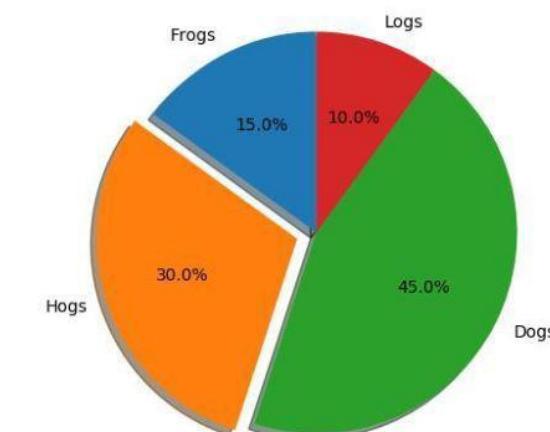
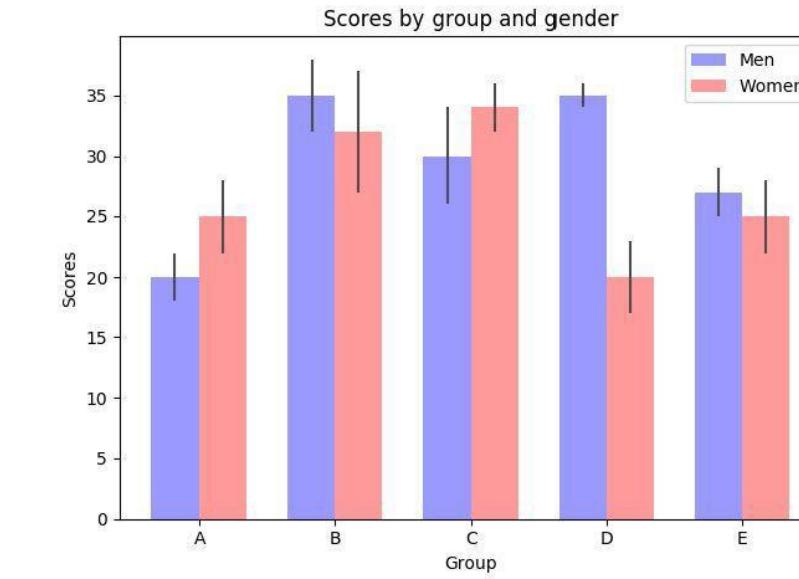
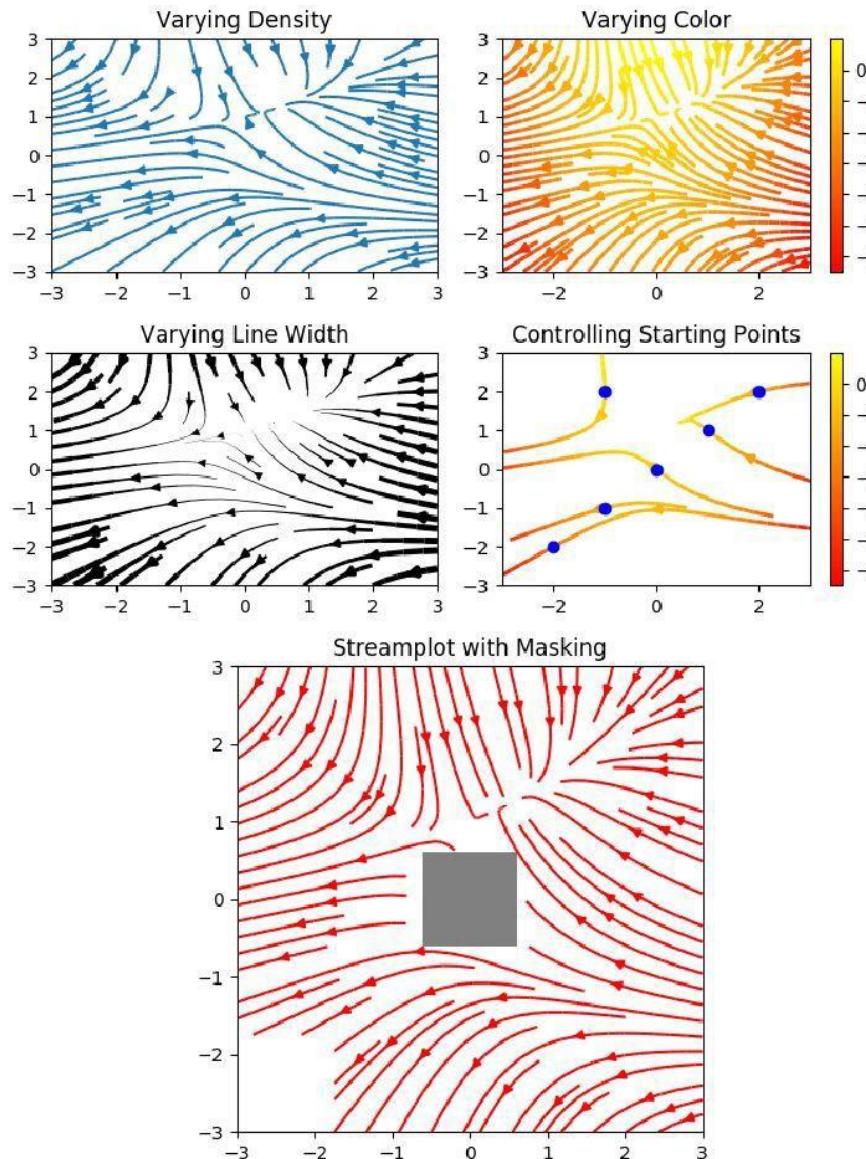


Một số biểu đồ vẽ bằng matplotlib



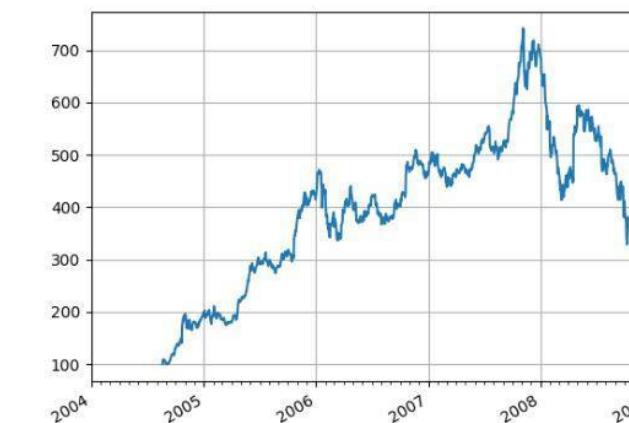
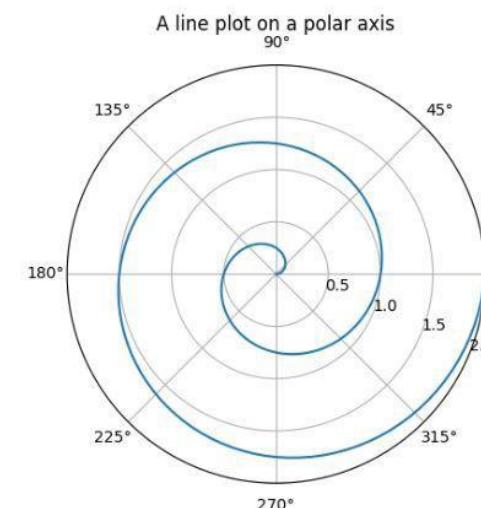
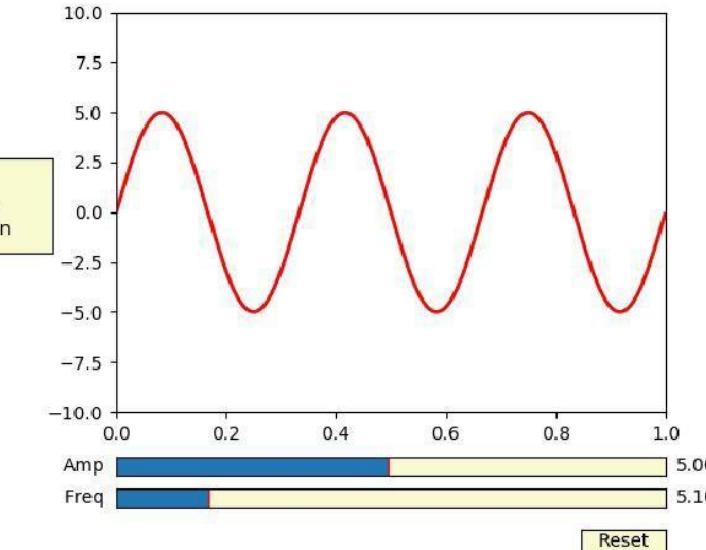
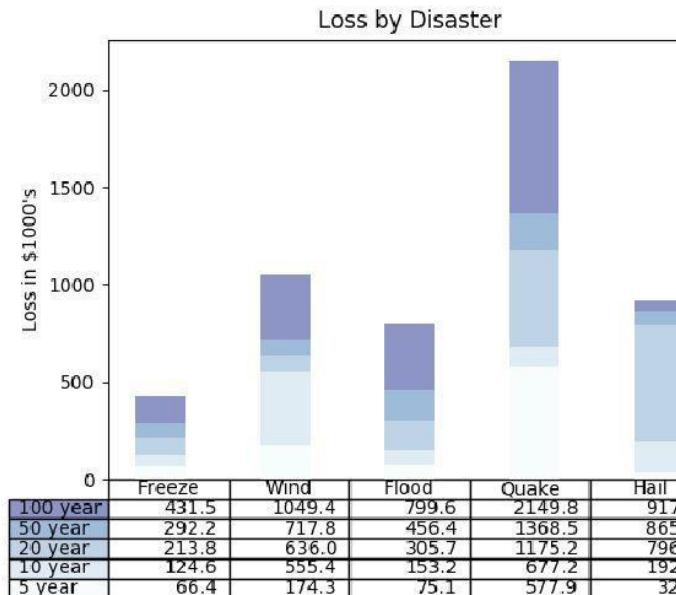


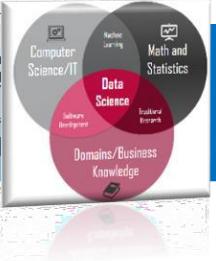
Một số biểu đồ vẽ bằng matplotlib





Một số biểu đồ vẽ bằng matplotlib





Vẽ biểu đồ đơn giản



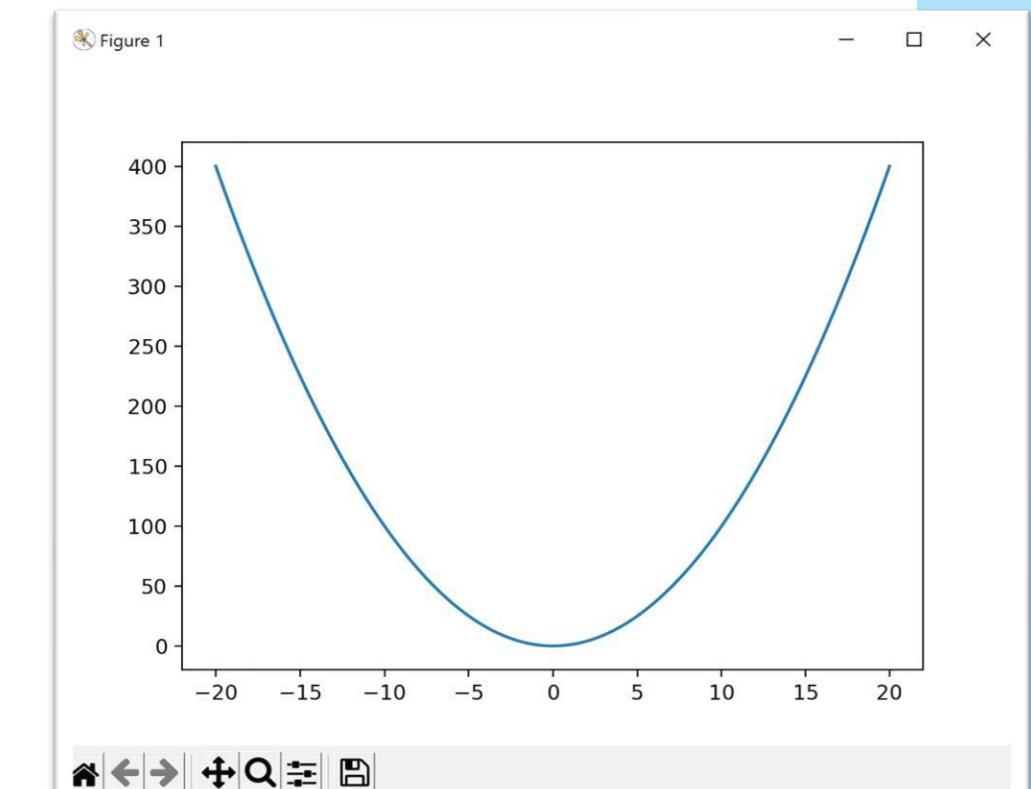
Ví dụ: vẽ biểu đồ $y = x^2$

```

import numpy as np          # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot

# chia đoạn từ -20 đến 20 thành 1000 đoạn
x = np.linspace(-20, 20, 1000)
# tính y
y = x * x
# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()

```





Ví dụ: Vẽ biểu đồ $y = x^2$

```
import numpy as np          # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot

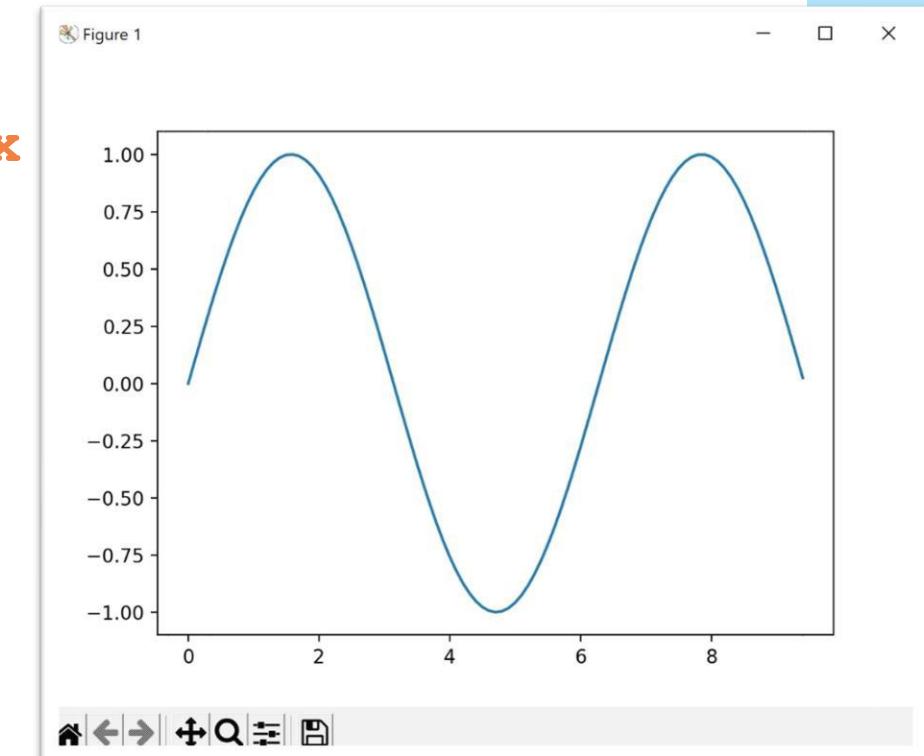
# chia đoạn từ -20 đến 20 thành 1000 đoạn
x = np.linspace(-20, 20, 1000)
# tính y
y = x * x
# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()
```



Ví dụ: Vẽ biểu đồ hình sin

```
import numpy as np          # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
```

```
# chia đoạn từ 0 đến  $3\pi$  thành
các đoạn con 0.1
x = np.arange(0, 3 * np.pi, 0.1)
# tính sin tương ứng với từng phần tử của x
y = np.sin(x)
# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()
```



Ví dụ: Biểu đồ cả SIN và COS

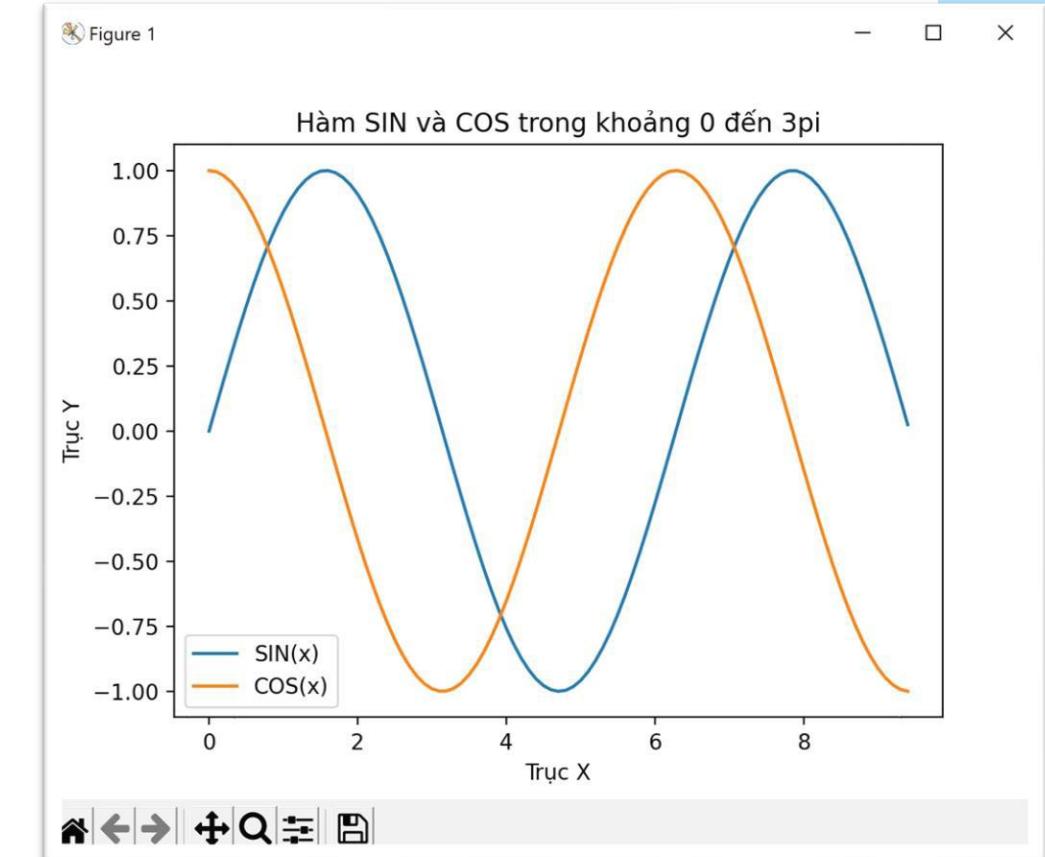


```

import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('Trục X')
plt.ylabel('Trục Y')
plt.title('Hàm SIN và COS trong khoảng 0 đến 3pi')
plt.legend(['SIN(x)', 'COS(x)'])
plt.show()

```





Các bước vẽ biểu đồ với matplotlib

- Điều kiện cần: đã có sẵn dữ liệu
- Có thể có 4 bước cơ bản:

1. Chọn loại biểu đồ phù hợp

- Tùy thuộc rất nhiều vào loại dữ liệu
- Tùy thuộc vào mục đích sử dụng của người dùng

2. Thiết lập các thông số cho biểu đồ

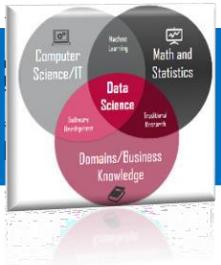
- Thông số của các trục, ý nghĩa, tỉ lệ chia,...
- Các điểm nhấn trên bản đồ
- Góc nhìn, mẫu tô, màu và các chi tiết khác
- Các thông tin bổ sung

3. Vẽ biểu đồ

4. Lưu ra file



Một số loại biểu đồ thông dụng trong matplotlib



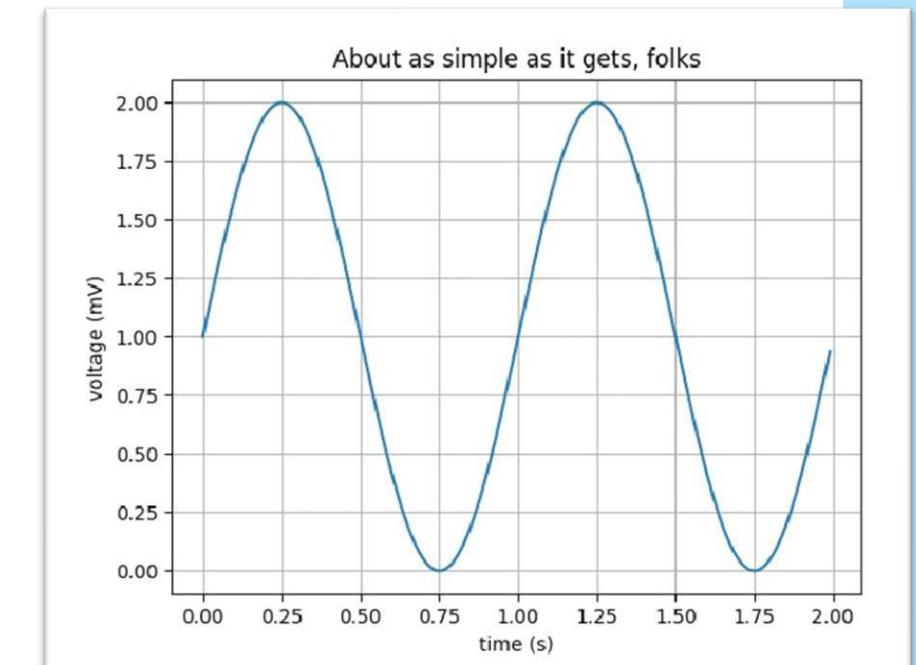
1.

Biểu đồ dạng đường (line plot)



Line plot

- Biểu đồ thể hiện tương quan giữa X và Y
- **Cú pháp:**
 - ✓ `plot([x], y, [fmt], data=None, **kwargs)`
 - ✓ `plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)`
- “**fmt**” là quy cách vẽ đường
- “**data**” là nhãn của dữ liệu
- ****kwargs:** tham số vẽ đường
- Vẽ nhiều lần trên một biểu đồ
- Kết quả trả về là một list các đối tượng Line2D





Line plot: fmt

- “**fmt**” gồm 3 phần **fmt** = ' [color] [marker] [line] '
- [color] – viết tắt tên màu:
 - ✓ 'b' – **blue**
 - ✓ 'g' – **green**
 - ✓ 'r' – **red**
 - ✓ 'c' – **cyan**
 - ✓ 'm' – **magenta**
 - ✓ 'y' – **yellow**
 - ✓ 'b' – **black**
 - ✓ 'w' – **white**
 - ✓ #rrggbb – chỉ ra mã màu theo hệ **RGB**

Line plot: fmt

- [marker] – cách đánh dấu dữ liệu:

- ✓ 'o' – hình tròn
- ✓ 'v' – tam giác xuông ('^', '<', '>')
- ✓ '*' – ngôi sao
- ✓ '.' – chấm
- ✓ 'p' – ngũ giác
- ✓ ...

- [line] – cách vẽ đường:

- ✓ '-' – nét liền
- ✓ '--' – nét đứt
- ✓ '-.' – gạch chấm
- ✓ ':' – đường chấm

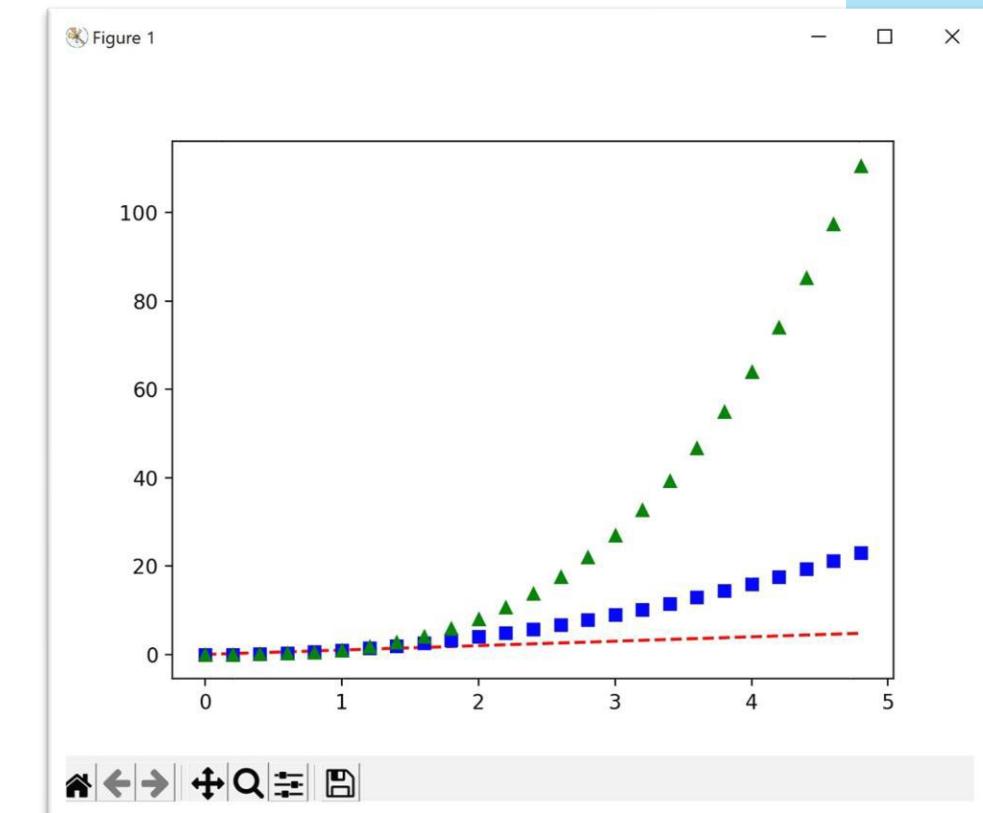


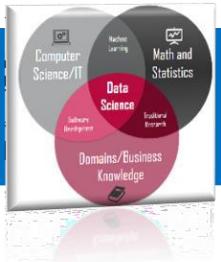
Ví dụ

```

import numpy as np
import matplotlib.pyplot as plt
# chia đoạn 0-5 thành các bước 0.2
t = np.arange(0., 5., 0.2)
# Vẽ 3 đường:
#- màu đỏ nét đứt: y = x
#- màu xanh dương, đánh dấu ô vuông:
y = x^2
#- màu xanh lá, đánh dấu tam giác:
y = x^3
plt.plot(t, t, 'r--', t, t**2, 'bs',
         t, t**3, 'g^')
plt.show()

```





2.

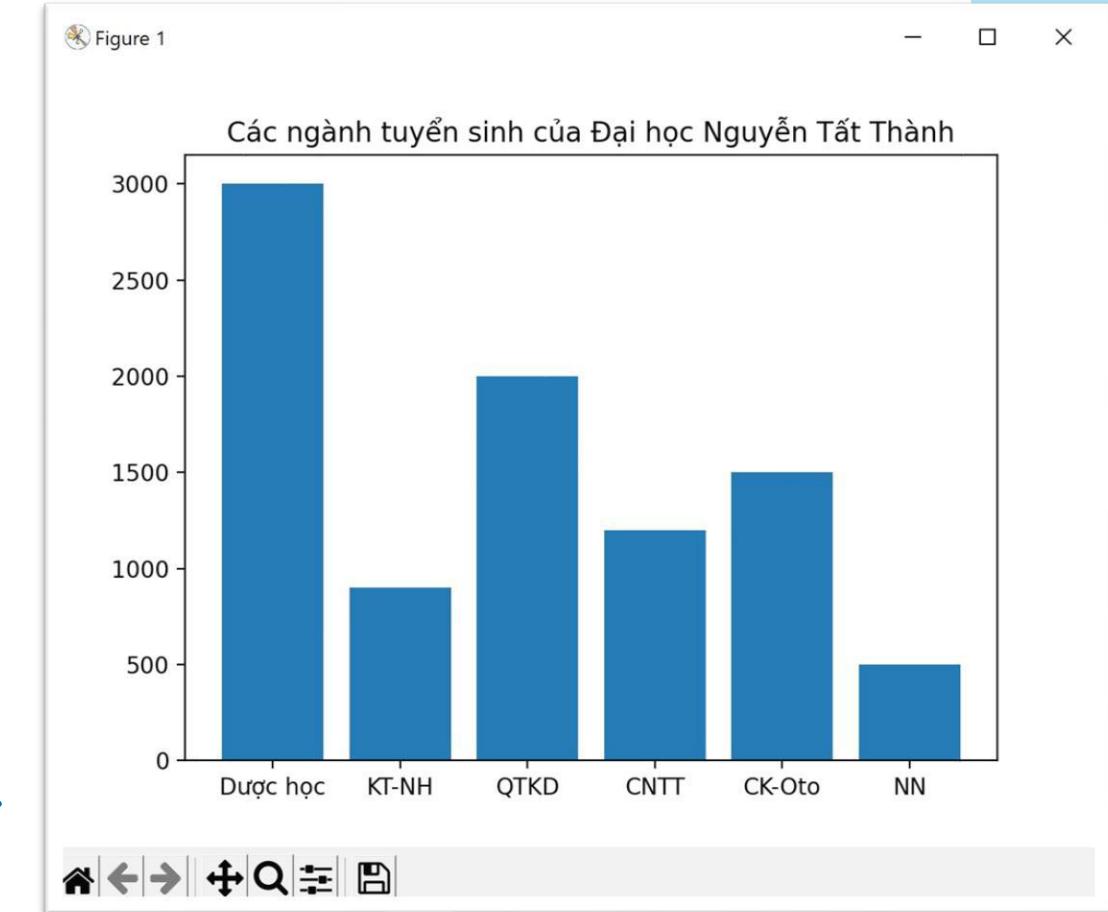
Biểu đồ dạng cột (bar plot)

Biểu đồ dạng cột chiều đứng

```

import matplotlib.pyplot as plt
D = { 'Dược học': 3000,
'KT-NH': 900,
'QTKD': 2000,
'CNTT': 1200,
'CK-Oto': 1500,
'NN': 500 }
plt.bar(range(len(D)), D.values(),
align='center')
plt.xticks(range(len(D)),
D.keys())
plt.title('Các ngành tuyển sinh
của Đại học Nguyễn Tất Thành')
plt.show()

```

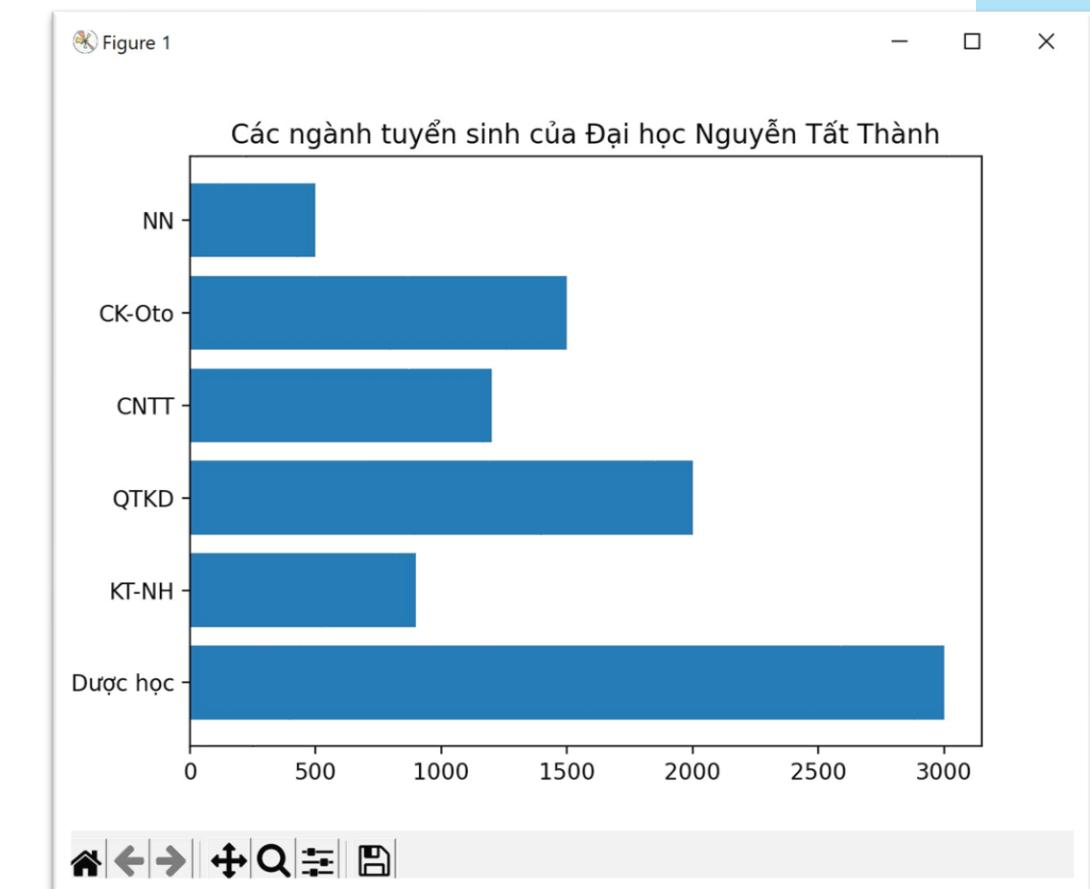


Biểu đồ dạng cột chiều ngang

```

import matplotlib.pyplot as plt
D = { 'Dược học': 3000,
'KT-NH': 900,
'QTKD': 2000,
'CNTT': 1200,
'CK-Oto': 1500,
'NN': 500 }
plt.bar(range(len(D)),
D.values(), align='center')
plt.xticks(range(len(D)),
D.keys())
plt.title('Các ngành tuyển sinh
của Đại học Nguyễn Tất Thành')
plt.show()

```

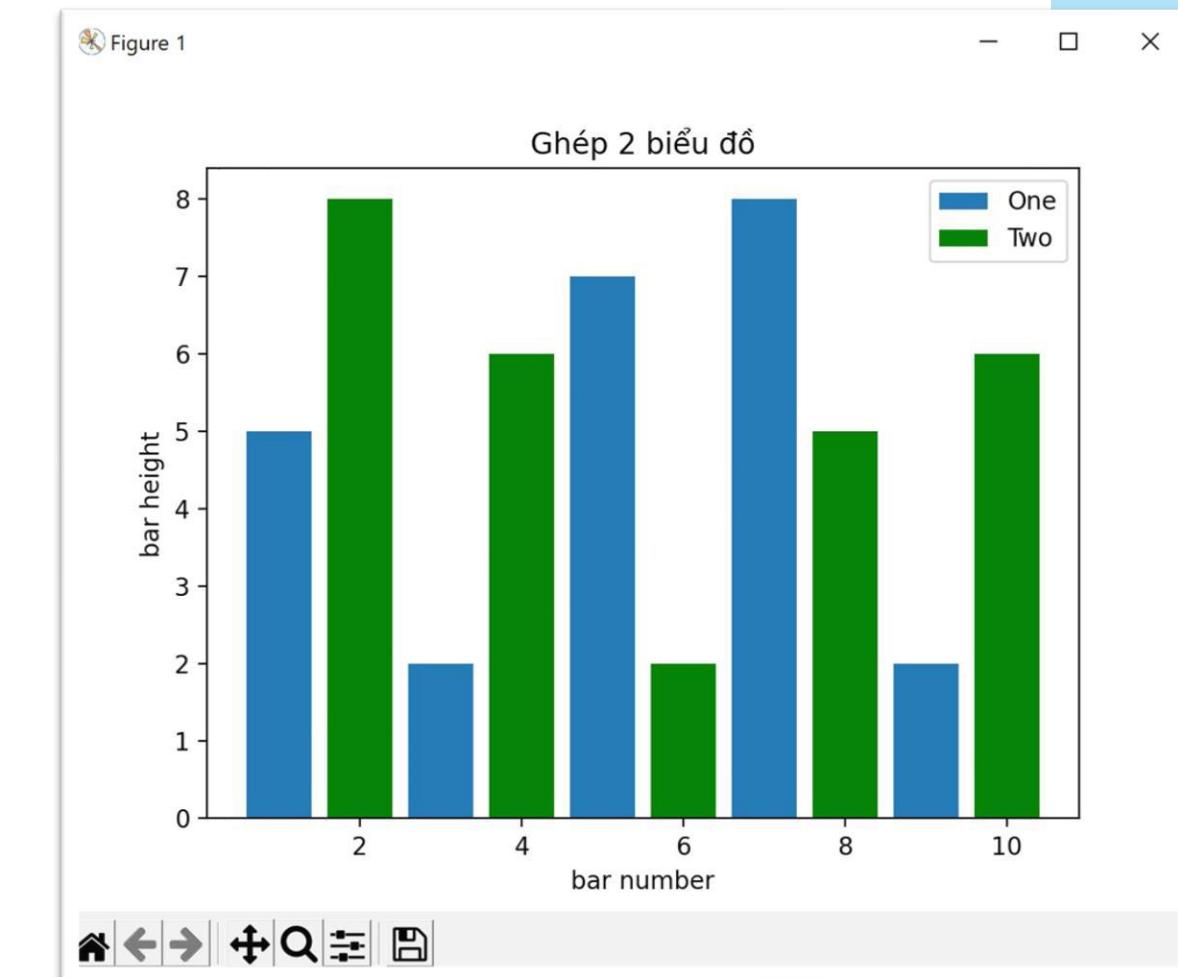




Có thể ghép 2 biểu đồ

Import matplotlib.pyplot as plt

```
plt.bar([1,3,5,7,9],[5,2,7,8,2],  
label="One")  
plt.bar([2,4,6,8,10],[8,6,2,5,6]  
, label="Two", color='g')  
plt.legend()  
plt.xlabel('bar number')  
plt.ylabel('bar height')  
plt.title('Ghép 2 biểu đồ')  
plt.show()
```





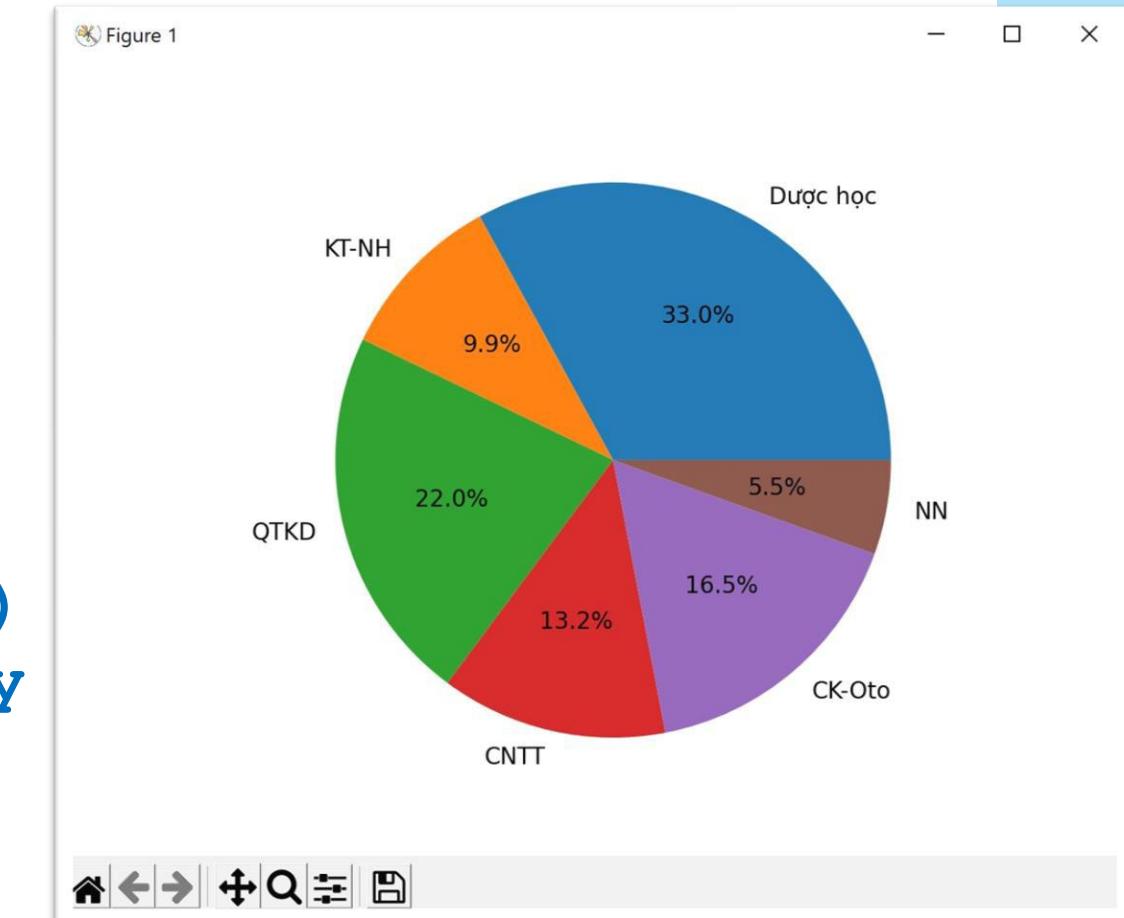
3.

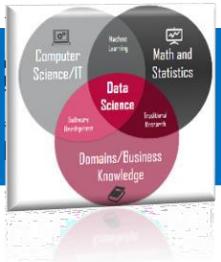
Biểu đồ bánh (pie chart)

Làm rõ tương quan chi tiết / tổng thể



```
import matplotlib.pyplot as plt
D = { 'Dược học': 3000,
      'KT-NH': 900,
      'QTKD': 2000,
      'CNTT': 1200,
      'CK-Oto': 1500,
      'NN': 500 }
plt.pie(D.values(),
        labels=D.keys(), autopct='%.1f%%')
plt.axis('equal') # trục x = trục y
plt.show()
```

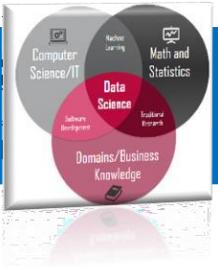




4.

Một số chức năng hữu ích

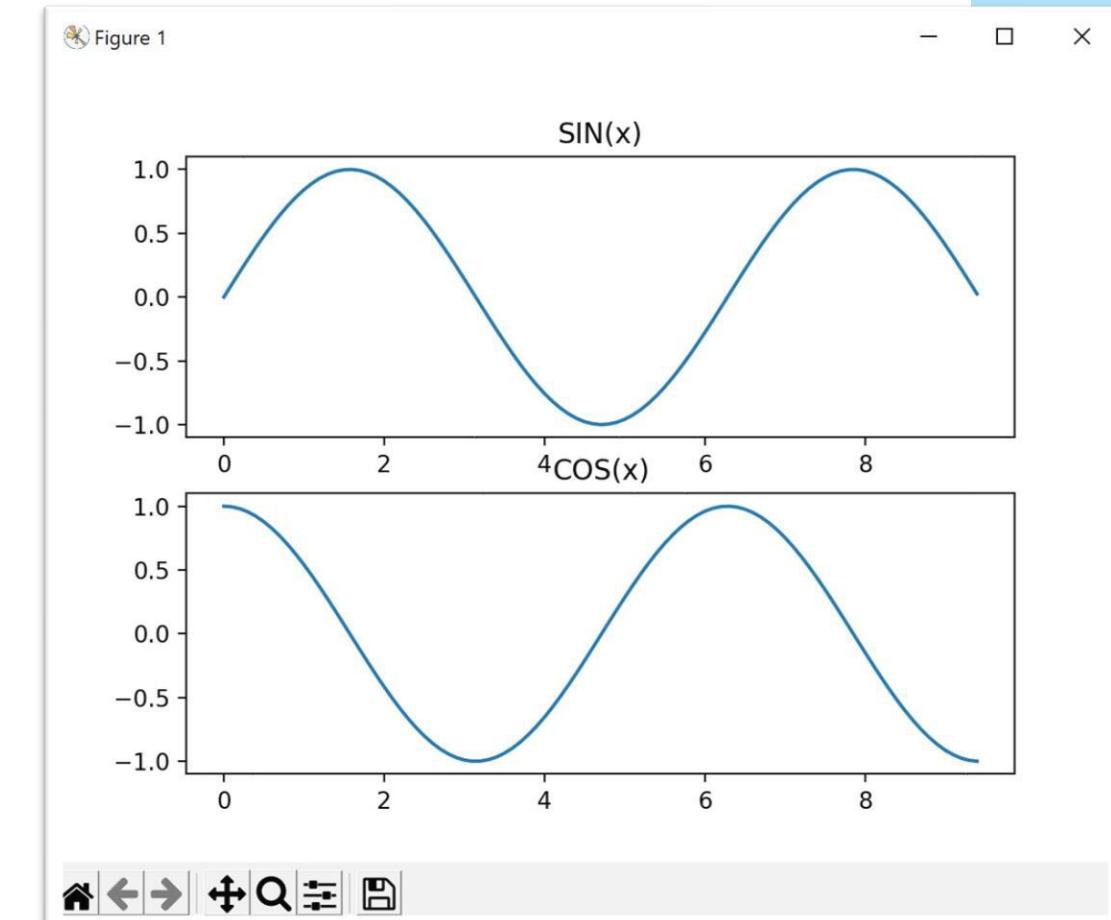
Chia thành các biểu đồ con



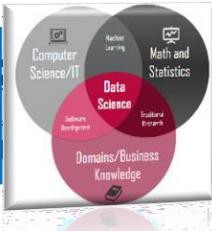
```

import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
plt.subplot(2, 1, 1) # biểu đồ 1
plt.plot(x, y_sin)
plt.title('SIN(x)')
plt.subplot(2, 1, 2) # biểu đồ 2
plt.plot(x, y_cos)
plt.title('COS(x)')
plt.show()

```

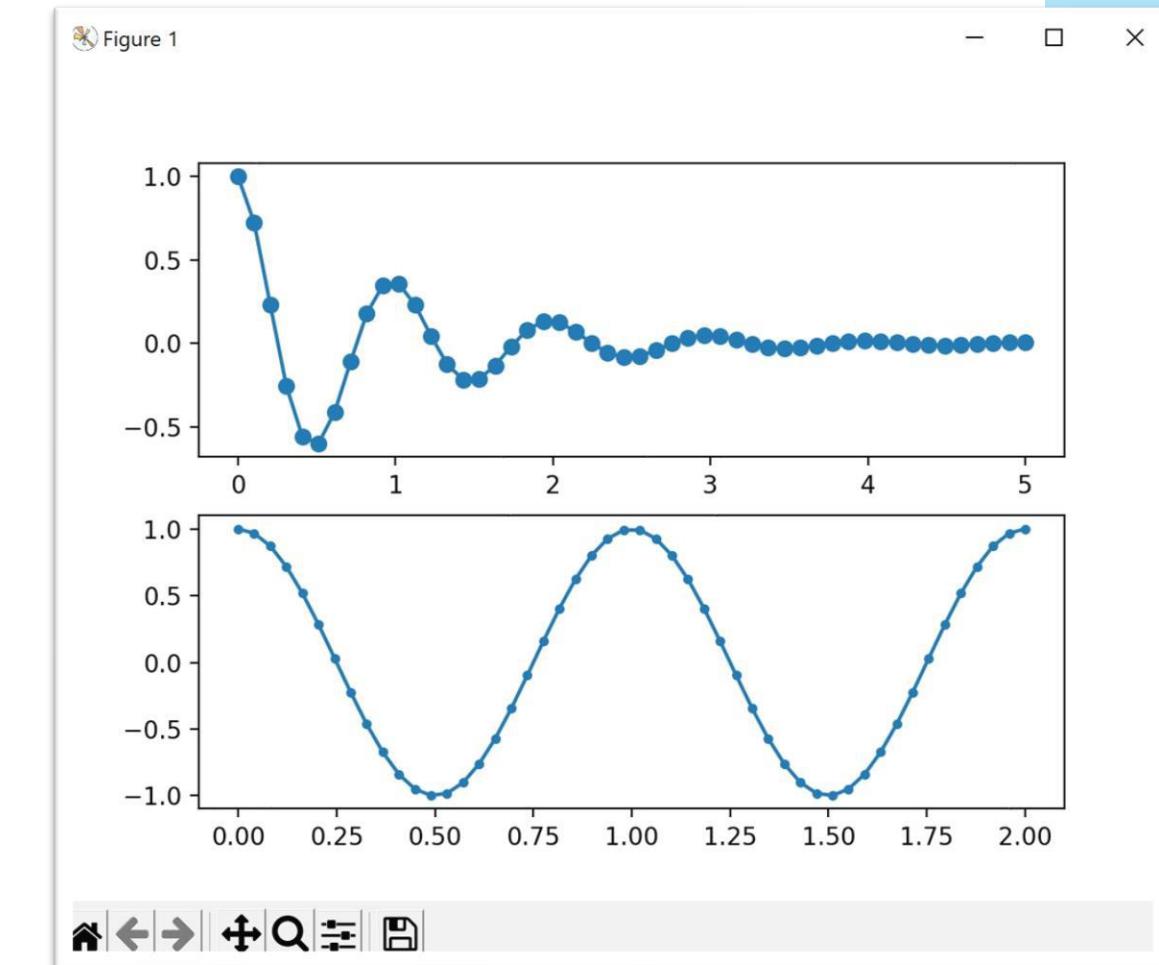


Chia thành các biểu đồ con



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

x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)
y1 = np.cos(2 * np.pi * x1) *
np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)
plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'o-')
plt.subplot(2, 1, 2)
plt.plot(x2, y2, '.-')
plt.show()
```

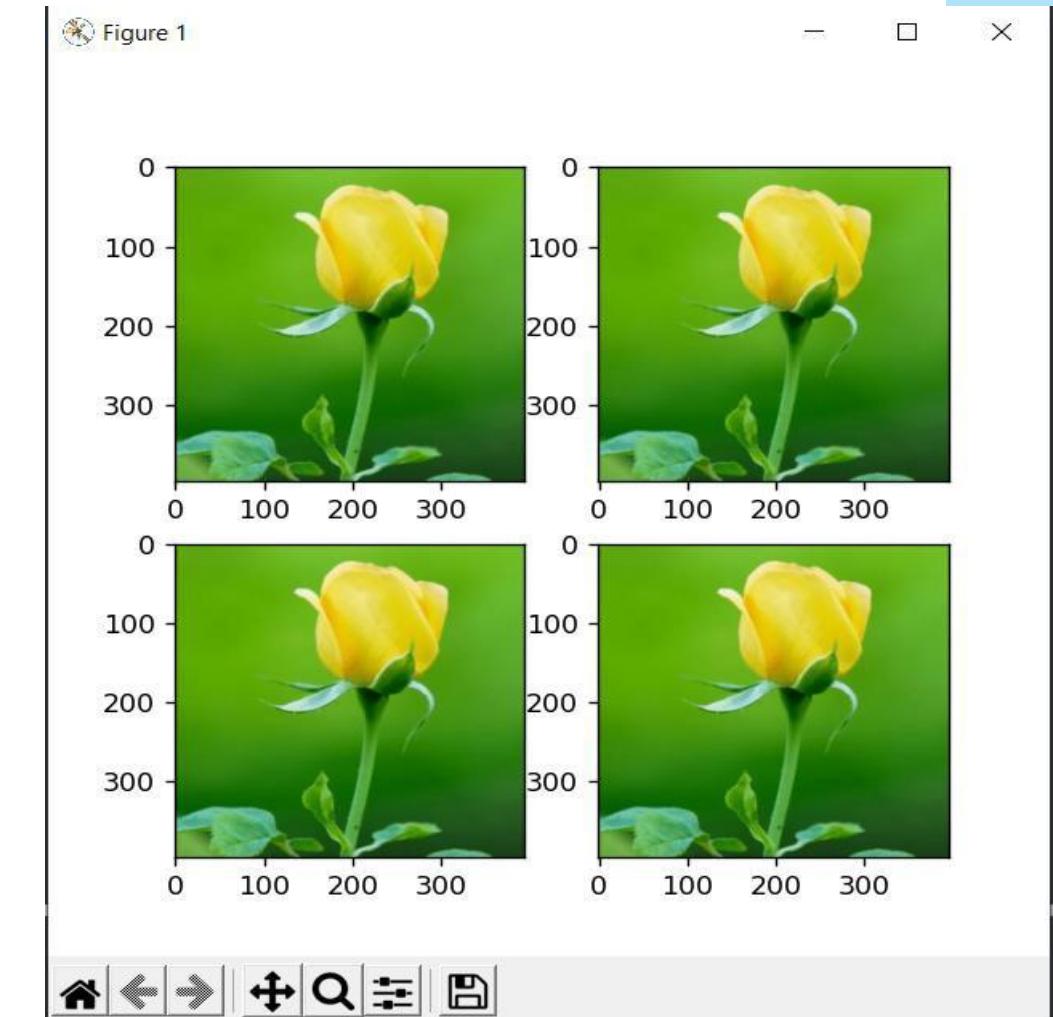




Nạp ảnh (png_jpg)

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

```
image = mpimg.imread("hoa hồng.jpg")
fig, axs = plt.subplots(2, 2,
figsize=(5, 5))
axs[0, 0].imshow(image)
axs[1, 0].imshow(image)
axs[0, 1].imshow(image)
axs[1, 1].imshow(image)
plt.show()
```

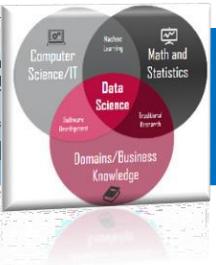




Lưu biểu đồ ra file

```
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
  
image = mpimg.imread("hoa.png")  
fig, axs = plt.subplots(2, 2, figsize=(5, 5))  
axs[0, 0].imshow(image)  
axs[1, 0].imshow(image)  
axs[0, 1].imshow(image)  
axs[1, 1].imshow(image)  
  
plt.savefig('1.jpg')  
plt.savefig('1.pdf')
```





Thư viện Panda



Nội dung

1. *Giới thiệu và cài đặt pandas*
2. *Cấu trúc dữ liệu trong pandas*
3. *Làm việc với series*
4. *Làm việc với dataframe*
5. *Bài tập*
6. *Hướng dẫn bài tập buổi trước*
7. *Làm việc với panel*
8. *Chọn và nhóm phần tử*
9. *Sử dụng pandas trong bài toán thực tế*



Phần 1

Giới thiệu và cài đặt pandas



Cài đặt: “pip install pandas”

- “pandas” là thư viện mở rộng từ numpy, chuyên để xử lý dữ liệu cấu trúc dạng bảng
- Tên “pandas” là dạng số nhiều của “panel data”

```
c:\ Command Prompt
(c) Microsoft Corporation. All rights reserved.

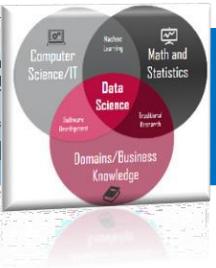
C:\Users\PHATTAI>pip install pandas
Collecting pandas
  Downloading pandas-1.4.0-cp310-cp310-win_amd64.whl (10.6 MB)
    |██████████| 10.6 MB 2.2 MB/s
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\phattai\appdata\local\programs\python\python310\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.21.0 in c:\users\phattai\appdata\local\programs\python\python310\lib\site-packages (from pandas) (1.22.1)
Collecting pytz>=2020.1
  Downloading pytz-2021.3-py2.py3-none-any.whl (503 kB)
    |██████████| 503 kB 2.2 MB/s
Requirement already satisfied: six>=1.5 in c:\users\phattai\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Installing collected packages: pytz, pandas
Successfully installed pandas-1.4.0 pytz-2021.3
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Users\PHATTAI\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' command.

C:\Users\PHATTAI>
```



Đặc điểm nổi bật của pandas

- Đọc dữ liệu từ nhiều định dạng
- Liên kết dữ liệu và tích hợp xử lý dữ liệu bị thiếu
- Xoay và chuyển đổi chiều của dữ liệu dễ dàng
- Tách, đánh chỉ mục và chia nhỏ các tập dữ liệu lớn dựa trên nhãn
- Có thể nhóm dữ liệu cho các mục đích hợp nhất và chuyển đổi
- Lọc dữ liệu và thực hiện query trên dữ liệu
- Xử lý dữ liệu chuỗi thời gian và lấy mẫu



Phần 2

Cấu trúc dữ liệu trong pandas



Cấu trúc dữ liệu trong pandas

- Dữ liệu của pandas có 3 cấu trúc chính:
 - ✓ Series (loạt): cấu trúc 1 chiều, mảng dữ liệu đồng nhất
 - ✓ Dataframe (khung): cấu trúc 2 chiều, dữ liệu trên các cột là đồng nhất (có phần giống như table trong SQL, nhưng với các dòng được đặt tên)
 - ✓ Panel (bảng): cấu trúc 3 chiều, có thể xem như một tập các dataframe với thông tin bổ sung
- Dữ liệu series gần giống kiểu array trong numpy, nhưng có 2 điểm khác biệt quan trọng:
 - ✓ Chấp nhận dữ liệu thiếu (NaN – không xác định)
 - ✓ Hệ thống chỉ mục phong phú (giống dictionary?)



Cấu trúc dataframe

- Dữ liệu 2 chiều
- Các cột có tên
- Dữ liệu trên cột là đồng nhất (series?)
- Các dòng có thể có tên
- Có thể có ô thiếu dữ liệu

	country	population	area	capital
BR	Brazil	200	8515767	Brasilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria



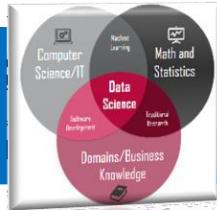
Cấu trúc panel

- Dữ liệu 3 chiều
- Một tập các dataframe
- Các dataframe có cấu trúc tương đồng
- Có thể có các thông tin bổ sung cho từng dataframe

Major	Minor	Open	Close
3/31/2015	IBM	23.602	132.903
	APPL	421.412	212.665
	CVX	568.055	409.201
	BHP	487.414	515.413
4/30/2015	IBM	150.868	457.895
	APPL	204.729	957.179
	CVX	90.679	888.687
	BHP	831.527	714.202
5/31/2015	IBM	788.582	922.422
	APPL	329.716	304.964
	CVX	36.578	981.508
	BHP	313.848	882.293

Phần 3

Làm việc với series



Tạo dữ liệu series (1)

```
import pandas as pd  
import numpy as np
```

```
S = pd.Series(np.random.randint(100, size = 4))  
print(S)  
print(S.index)  
print(S.values)
```

```
C:\ Command Prompt  
Microsoft Windows [Version 10.0.19042.1466]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\PHATTAI>"C:\Users\PHATTAI\Desktop\Python\Chương 5\taodulieuseries1.py"  
0    98  
1    21  
2    74  
3    20  
dtype: int32  
RangeIndex(start=0, stop=4, step=1)  
[98 21 74 20]
```



Tạo dữ liệu series (2)

```
import pandas as pd
import numpy as np
```

```
chi_so = ["Ke toan", "KT", "CNTT", "Co khi"]
gia_tri = [310, 360, 580, 340]
S = pd.Series(gia_tri, index=chi_so)
print(S)
print(S.index)
print(S.values)
```

```
C:\Users\PHATTAI>"C:\Users\PHATTAI\Desktop\Python\Chương 5\taodulieuseries2.py"
Ke toan    310
KT         360
CNTT       580
Co khi     340
dtype: int64
Index(['Ke toan', 'KT', 'CNTT', 'Co khi'], dtype='object')
[310 360 580 340]

C:\Users\PHATTAI>
```



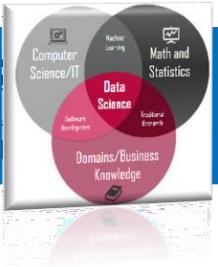
Tạo dữ liệu series (3)

```
import pandas as pd
import numpy as np
```

```
chi_so = ["KT", "KT", "CNTT", "Co khi"]      # trùng nhau
gia_tri = [310, 360, 580, 340]
S = pd.Series(gia_tri, index=chi_so)
print(S)
print(S.index)
print(S.values)
```

```
C:\Users\PHATTAI>"C:\Users\PHATTAI\Desktop\Python\Chương 5\taodulieuseries3.py"
KT      310
KT      360
CNTT    580
Co khi  340
dtype: int64
Index(['KT', 'KT', 'CNTT', 'Co khi'], dtype='object')
[310 360 580 340]
```

Truy vấn dữ liệu thông qua chỉ số



```
import pandas as pd  
import numpy as np
```

```
chi_so = ["KT", "KT", "CNTT", "Co khi"] # trùng nhau  
gia_tri = [310, 360, 580, 340]  
S = pd.Series(gia_tri, index=chi_so)  
print(S['Co khi'])  
print(S['KT'])  
print(S.CNTT)
```

```
C:\Users\PHATTAI>"C:\Users\PHATTAI\Desktop\Python\Chương 5\truyvandulieuthongquachiso.py"  
340  
KT    310  
KT    360  
dtype: int64  
580
```



Phép toán trên series

```

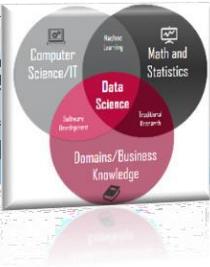
import pandas as pd
import numpy as np

chi_so = ["Ke toan", "KT", "CNTT", "Co
khi"]
gia_tri = [310, 360, 580, 340]
# chỉ số giống nhau thì tính gộp, nếu không
# thì NaN
S = pd.Series(gia_tri, index=chi_so)
P = pd.Series([100, 100], ['CNTT',
'PM'])
Y = S + P
print(Y)

```

CNTT	680.0
Co khi	NaN
KT	NaN
Ke toan	NaN
PM	NaN

dtype: float64



Phép toán trên series

- Nguyên tắc chung của việc thực hiện phép toán trên series như sau:
 - ✓ Nếu là phép toán giữa 2 series, thì các giá trị cùng chỉ số sẽ thực hiện phép toán với nhau, trường hợp không có giá trị ở cả 2 series thì trả về NaN
 - ✓ Nếu là phép toán giữa series và 1 số, thì thực hiện phép toán trên số đó với tất cả các giá trị trong series



Một số phương thức hữu ích

- **S.axes**: trả về danh sách các chỉ mục của S
- **S.dtype**: trả về kiểu dữ liệu các phần tử của S
- **S.empty**: trả về True nếu S rỗng
- **S.ndim**: trả về số chiều của S (1)
- **S.size**: trả về số phần tử của S
- **S.values**: trả về list các phần tử của S
- **S.head(n)**: trả về n phần tử đầu tiên của S
- **S.tail(n)**: trả về n phần tử cuối cùng của S



apply() một hàm khác trên series

```

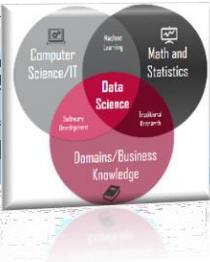
import pandas as pd
import numpy as np

def Tang(x):
    return x if x > 500 else x + 1000

chi_so = ["Ke toan", "KT", "CNTT", "Co khi"]
gia_tri = [310, 360, 580, 340]
S = pd.Series(gia_tri, chi_so)
# áp dụng Tang trên S (không thay đổi S)
print(S.apply(Tang))

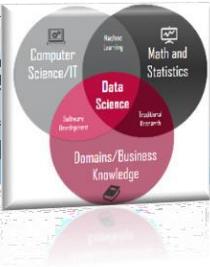
```

Ke toan	1310
KT	1360
CNTT	580
Co khi	1340
dtype: int64	



Phần 4

Làm việc với dataframe



Khởi tạo dataframe

- Cú pháp chung:
`pandas.DataFrame(data, index, columns, dtype, copy)`
- Trong đó:
 - ✓ 'data' sẽ nhận giá trị từ nhiều kiểu khác nhau như list, dictionary, ndarray, series,... và cả các DataFrame khác
 - ✓ 'index' là nhãn chỉ mục hàng của dataframe
 - ✓ 'columns' là nhãn chỉ mục cột của dataframe
 - ✓ 'dtype' là kiểu dữ liệu cho mỗi cột
 - ✓ 'copy' nhận giá trị True/False để chỉ rõ dữ liệu có được copy sang vùng nhớ mới không, mặc định là False



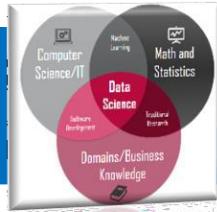
Tạo dataframe từ list

```

crimes_rates = {
    "Year": [1960, 1961, 1962, 1963, 1964],
    "Population": [179323175, 182992000, 185771000, 188483000,
    191141000],
    "Total": [3384200, 3488000, 3752200, 4109500, 4564600],
    "Violent": [288460, 289390, 301510, 316970, 364220]
}
crimes_dataframe = pd.DataFrame(crimes_rates)
print(crimes_dataframe)

```

	Population	Total	Violent	Year
0	179323175	3384200	288460	1960
1	182992000	3488000	289390	1961
2	185771000	3752200	301510	1962
3	188483000	4109500	316970	1963
4	191141000	4564600	364220	1964



Tạo dataframe từ list các dictionary

```

data = [
    { 'MIT' : 5000, 'Stanford' : 4500, "DHTL":15000},
    { 'MIT' : 1, 'Stanford' : 2, "DHTL":200}
]
df = pd.DataFrame(data, index=[ 'NumOfStudents' , "ranking"])
print(df)
print(df.DHTL.dtype)

```

	DHTL	MIT	Stanford
NumOfStudents	15000	5000	4500
ranking		1	2
dtype('int64')			



Tạo dataframe từ dictionary series

```
data = {  
    "one": pd.Series([1,23,45], index = [1,2,3]),  
    "two": pd.Series([1000,2400,1132,3434], index =  
[1,2,3,4])  
}  
  
df = pd.DataFrame(data)  
print(df)
```

	one	two
1	1.0	1000
2	23.0	2400
3	45.0	1132
4	NaN	3434



Đọc dữ liệu từ file .csv

- Nội dung của file brics.csv:
 - ✓ Số liệu về các quốc gia thuộc khối BRICS
 - ✓ Sử dụng dấu phẩy để ngăn giữa các dữ liệu
 - ✓ Mỗi dữ liệu trên 1 dòng
 - ✓ Dòng đầu tiên là tên các cột

,country,population,area,capital
BR,Brazil,200,8515767,Brasilia
RU,Russia,144,17098242,Moscow
IN,India,1252,3287590,New Delhi
CH,China,1357,9596961,Beijing
SA,South Africa,55,1221037,Pretoria



Đọc dữ liệu từ file .csv

```
import pandas as pd
```

```
d = pd.read_csv("brics.csv")
```

```
print(d)
```

	country	population	area	capital	
0	BR	Brazil	200	8515767	Brasilia
1	RU	Russia	144	17098242	Moscow
2	IN	India	1252	3287590	New Delhi
3	CH	China	1357	9596961	Beijing
4	SA	South Africa	55	1221037	Pretoria



Đọc dữ liệu từ file .csv

```
import pandas as pd
```

```
# đọc dữ liệu và quy định cột 0 dùng làm chỉ số dòng
d = pd.read_csv("brics.csv", index_col = 0)
print(d)
```

	country	population	area	capital
BR	Brazil	200	8515767	Brasilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria



Truy cập theo từng cột

- Sử dụng tên cột làm chỉ số hoặc dùng luôn tên cột
- Việc truy cập này trả về tham chiếu đến dữ liệu, vì vậy có thể sử dụng phép gán để cập nhật dữ liệu theo cột

```
print(brics["country"])
```

BR Brazil

RU Russia

IN India

CH China

SA South Africa

Name: country, dtype: object

```
print(brics.country)
```

BR Brazil

RU Russia

IN India

CH China

SA South Africa

Name: country, dtype: object



Thêm một cột (1)

- Bằng cách sử dụng một cột mới chưa có

```
brics["on_earth"] = [True, True, True, True, True]  
print(brics)
```

	country	population	area	capital	on_earth
BR	Brazil	200	8515767	Brasilia	True
RU	Russia	144	17098242	Moscow	True
IN	India	1252	3287590	New Delhi	True
CH	China	1357	9596961	Beijing	True
SA	South Africa	55	1221037	Pretoria	True



Thêm một cột (2)

- Bằng cách sử dụng một cột mới chưa có và thiết lập công thức phù hợp

```
brics["density"] = brics["population"] / brics["area"] * 1000000
print(brics)
```

	country	population	area	capital	on_earth	density
BR	Brazil	200	8515767	Brasilia	True	23.485847
RU	Russia	144	17098242	Moscow	True	8.421918
IN	India	1252	3287590	New Delhi	True	380.826076
CH	China	1357	9596961	Beijing	True	141.398928
SA	South Africa	55	1221037	Pretoria	True	45.043680



Truy cập vào từng ô trên dataframe

- Bằng cách kết hợp chỉ mục dòng và cột

```
print(brics.loc["CH","capital"])
```

Beijing

```
print(brics["capital"].loc["CH"])
```

Beijing

```
print(brics.loc["CH"]["capital"])
```

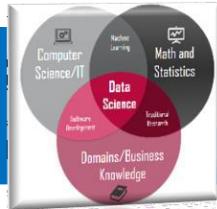
Beijing



Xóa dòng hoặc cột bằng drop

```
# tạo ra dataframe mới bằng cách xóa 2 cột
print(d.drop(["area", "population"], axis=1))
# trường hợp muốn xóa trên d, thêm tham số inplace=True
d.drop(["area", "population"], axis=1, inplace=True)
print(d)
```

	country	capital
BR	Brazil	Brasilia
RU	Russia	Moscow
IN	India	New Delhi
CH	China	Beijing
SA	South Africa	Pretoria



Tính tổng và tổng tích lũy

tính tổng của cột population, trả về tổng

```
print(d.population.sum())
```

tính tổng của cột population, trả về các tổng trong quá trình cộng

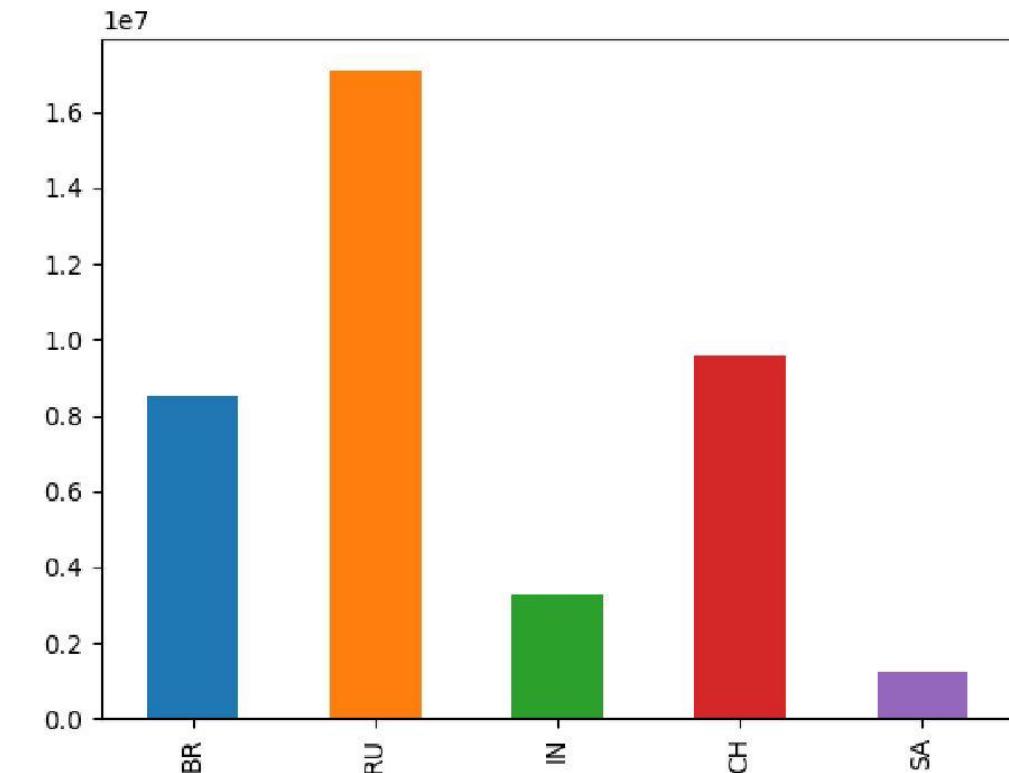
```
print(d.population.cumsum())
```

```
3008  
BR    200  
RU    344  
IN    1596  
CH    2953  
SA    3008
```

Name: population, dtype: int64

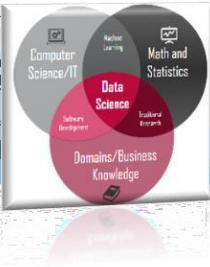
Kết hợp giữa pandas và matplotlib

```
import pandas as pd  
import matplotlib.pyplot as plt  
d = pd.read_csv("brics.csv", index_col = 0)  
d.area.plot(kind='bar')  
plt.show()
```



Phần 7

Làm việc với panel



Cấu trúc panel

- Panel được sử dụng nhiều trong kinh tế lượng
- Dữ liệu có 3 trục:
- Items (trục 0): mỗi item là một dataframe bên trong
- Major axis (trục 1 – trục chính): các dòng
- Minor axis (trục 2 – trục phụ): các cột
- Không được phát triển tiếp (thay bởi MultiIndex)

		Open	Close
Major	Minor		
3/31/2015	IBM	23.602	132.903
	APPL	421.412	212.665
	CVX	568.055	409.201
	BHP	487.414	515.413
4/30/2015	IBM	150.868	457.895
	APPL	204.729	957.179
	CVX	90.679	888.687
	BHP	831.527	714.202
5/31/2015	IBM	788.582	922.422
	APPL	329.716	304.964
	CVX	36.578	981.508
	BHP	313.848	882.293



Tạo panel

- Cú pháp:
`pandas.Panel(data, items, major_axis, minor_axis, dtype, copy)`
- Trong đó:
 - 'data' có thể nhận các kiểu dữ liệu sau: ndarray, series, map, lists, dict, hằng số và cả dataframe khác
 - 'items' là axis = 0
 - 'major_axis' là axis = 1
 - 'minor_axis' là axis = 2
 - 'dtype' là kiểu dữ liệu mỗi cột
 - 'copy' nhận giá trị True/False để khởi tạo dữ liệu có chia sẻ memory hay không



Tạo panel

```
import pandas as pd  
import numpy as np  
  
data = np.random.rand(2,3,4)  
p = pd.Panel(data)  
print(p)
```

```
<class 'pandas.core.panel.Panel'>  
Dimensions: 2 (items) x 3 (major_axis) x 4  
(minor_axis)  
Items axis: 0 to 1  
Major_axis axis: 0 to 2  
Minor_axis axis: 0 to 3
```



Tạo panel

p.to_frame()			0	1
	major	minor		
	0	0	0.335571	0.010409
		1	0.267106	0.843688
		2	0.840885	0.211749
		3	0.049653	0.722182
	1	0	0.755207	0.282777
		1	0.674844	0.543207
		2	0.634314	0.433802
		3	0.290120	0.613040
	2	0	0.322059	0.263548
		1	0.341035	0.702612
		2	0.634411	0.917126
		3	0.281678	0.809592



Phần 8

Chọn và nhóm phần tử



Chọn với iloc, loc và ix

- Pandas có 3 phương pháp chọn phần tử
 - ✓ 1. Dùng iloc: chọn theo chỉ số hàng và cột
 - ❖ Cú pháp: `data.iloc[<row selection>, <column selection>]`
 - ❖ Tham số có thể là số nguyên, list các số nguyên, slice object với các số nguyên (ví dụ 2:7), mảng boolean,...
 - ✓ 2. Dùng loc: chọn theo nhãn hàng hoặc nhãn cột
 - ❖ Cú pháp: `data.loc[<row selection>, <column selection>]`
 - ❖ Tham số là nhãn (chứ không phải chỉ số)
 - ✓ 3. Dùng ix: lai giữa 2 cách trên, nếu truyền tham số là số nguyên thì nó làm việc như iloc, truyền kiểu giá trị khác thì nó làm việc như loc



Nhóm phần tử

```
df2 = pd.DataFrame({'X' : ['B', 'B', 'A', 'A'], 'Y' : [1, 2, 3, 4]})  
df2.groupby(['X']).sum()
```

	Y
X	
A	7
B	3

```
df2.groupby(['X'], sort=False).sum()
```

	Y
X	
B	3
A	7



Nhóm phần tử

```
df3 = pd.DataFrame({'X' : ['A', 'B', 'A', 'B'], 'Y' : [1, 4, 3, 2]})  
df3.groupby(['X']).get_group('A')
```

	X	Y
0	A	1
2	A	3

```
df3.groupby(['X']).get_group('B')
```

	X	Y
1	B	4
3	B	2

Phần 9

Sử dụng pandas trong bài toán thực tế





Dữ liệu kết quả xổ số

- Dữ liệu kết quả xổ số (độc đắc) từ ngày 1-1-2000 đến ngày 21-5-2018 (hôm qua)
- Lưu ở định dạng csv, 2 cột:
 - ✓ Cột 1: ngày ra số
 - ✓ Cột 2: số độc đắc
 - ❖ Dạng số (nếu không đủ 5 chữ số thì có nghĩa là đã bị xóa các chữ số 0 ở đầu)
 - ❖ Có thể không có dữ liệu (mỗi năm có 4 ngày không quay xổ số)
- Bài toán (vui + khoa học): phân tích các chiến lược chơi số để mà người dân hay theo

Đọc và tiền xử lý dữ liệu

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

```
# đọc dữ liệu từ file csv, chuyển dữ liệu cột 1 sang date  
df = pd.read_csv("kqxs.csv", index_col = 0, parse_dates=True)
```

```
# xóa bỏ các dòng không có dữ liệu  
df.dropna(inplace=True)  
# thêm cột mới là 2 số cuối của giải độc đắc  
df['Cuoi'] = df.So % 100
```



Khảo sát dữ liệu

```
# trích xuất cột mới thành dữ liệu series để dễ xử lý  
s = pd.Series(df.Cuoi, dtype='int64')
```

```
# xem phân bổ dữ liệu: biểu đồ histogram, 100 nhóm  
s.plot('hist', bins=100)  
plt.show()
```

```
# một dạng phân bổ dữ liệu khác: biểu đồ bar, đếm tần suất  
s.value_counts().sort_index().plot('bar')  
plt.show()
```



Viết hàm tính số tiền thu về

thử bộ số myNums, kết quả về là result, số tiền chơi là money

```
def one_day(myNums, result, money):
```

```
    pay = len(myNums) * money
```

```
    get = money * 70 if result in myNums else 0
```

```
    return get-pay
```

chơi nhiều ngày bộ số myNums, kết quả về là results

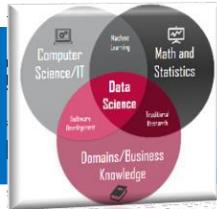
```
def many_day(myNums, results, money):
```

```
    total = 0
```

```
    for x in results:
```

```
        total += one_day(myNums, x, money)
```

```
    return total
```



Chiến lược: nuôi một số

money = 1000

thử chiến lược chơi: nuôi một con

```
print("Chơi con 76 toàn năm 2000:", many_day([76],  
s[0:367], money))  
print("Chơi con 76 toàn bộ các năm:", many_day([76], s,  
money))
```

thử chiến lược chơi: nuôi nhiều con

```
print("Nuôi nhiều số toàn năm 2000:", many_day([76, 92,  
3, 10, 51, 45], s[0:367], money))  
print("Nuôi nhiều số toàn bộ các năm:", many_day([76,  
92, 3, 10, 51, 45], s, money))
```



Chiến lược: thống kê

thống kê con ra nhiều nhất rồi chơi

```
x = s[0:362].value_counts().idxmax()
y = s.value_counts().idxmax()
print("Chơi theo số ra nhiều nhất năm 2000:", x,
many_day([x], s, money))
print("Chơi theo số ra nhiều nhất các năm:", y,
many_day([y], s, money))
```



Chiến lược: ngẫu nhiên

chơi ngẫu nhiên, mỗi ngày một con

```
total = 0
for d in s:
    total -= money
m = np.random.randint(100)
if (m == d): total += 70 * money
print("Chơi ngẫu nhiên:", total)
```



Thư viện scikit-learn



Nội dung

1. *Mối quan hệ giữa Khoa học Dữ liệu và Học máy*
2. *Một số loại bài toán học máy*
3. *Thư viện học máy scikit-learn*



Phần 1

Mối quan hệ giữa Khoa học Dữ liệu và Học máy



Khoa học dữ liệu là gì?

- Hầu hết các ngành khoa học từ xưa đến nay đều giải quyết vấn đề dựa trên **lập luận và tri thức**
 - ✓ Ngành toán: dựa trên các mệnh đề, công thức, lập luận... để chứng minh bài toán
 - ✓ Ngành vật lý: dựa trên các quan sát, thực nghiệm, tính toán,... kiểm chứng các giả thiết
 - ✓ Ngành hóa học:...
 - ✓ ...
 - ✓ Ta gọi các ngành khoa học này là “knowledge-driven” (dẫn dắt bởi tri thức)
- Có ngành có chút ngoại lệ, ví dụ: **ngành xác suất**

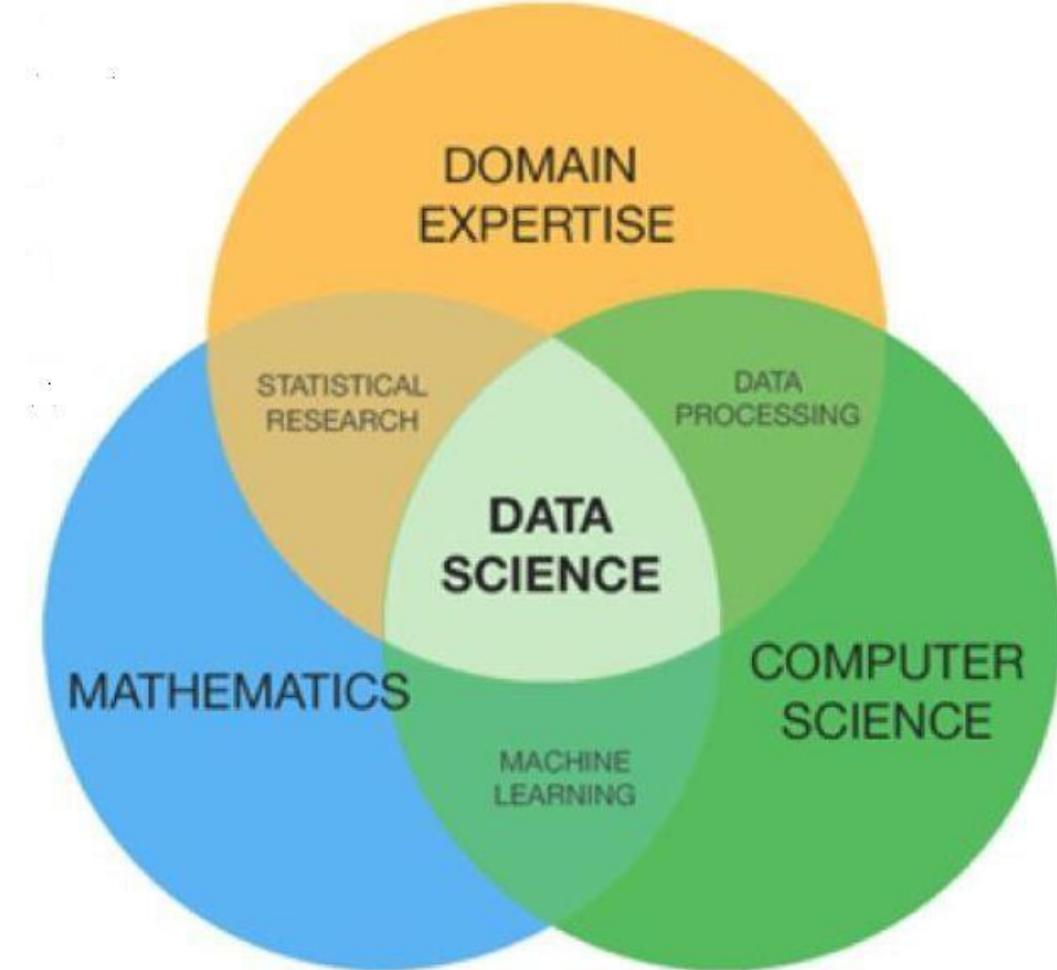


Khoa học dữ liệu là gì?

- Với quan điểm như vậy, tất cả những quan sát mà không được chứng minh chặt chẽ thường được cho là “không khoa học”
 - ✓ Chẳng hạn: chuồn chuồn bay thấp thì mưa
- Khoa học dữ liệu ≠ Khoa học thông thường ở quan điểm: tìm tri thức từ dữ liệu (dẫn dắt bởi dữ liệu – “data-driven”)
 - ✓ Chúng ta rút ra tri thức bằng việc tìm tòi từ dữ liệu chứ không nhất thiết phải chứng minh nó
 - ✓ Tất nhiên tri thức tìm ra phải có tính ổn định (luôn có cùng kết quả nếu sử dụng cùng một phương pháp)

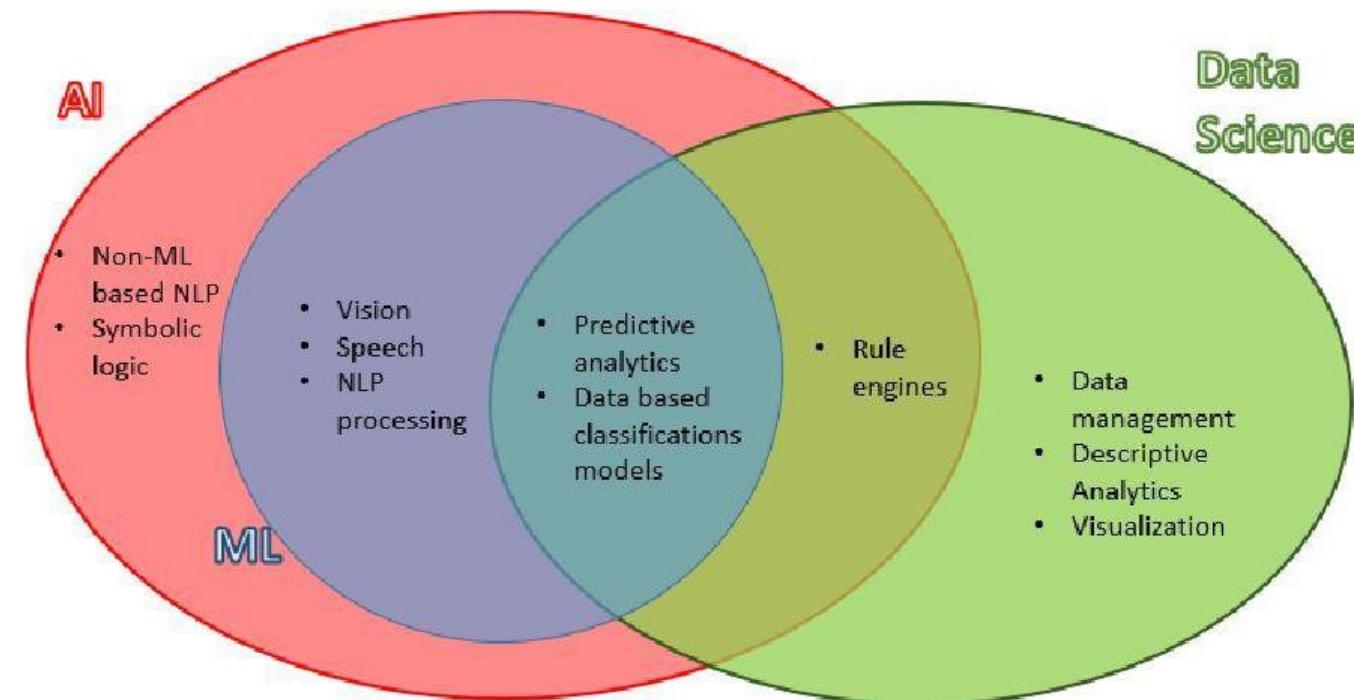
Khoa học Dữ liệu và Học máy

- Không có sơ đồ nào minh họa đầy đủ mối quan hệ giữa hai khái niệm này
- Nhiều người (chẳng hạn như Nate Silver) cho rằng ngành khoa học dữ liệu chỉ là một dạng thống kê

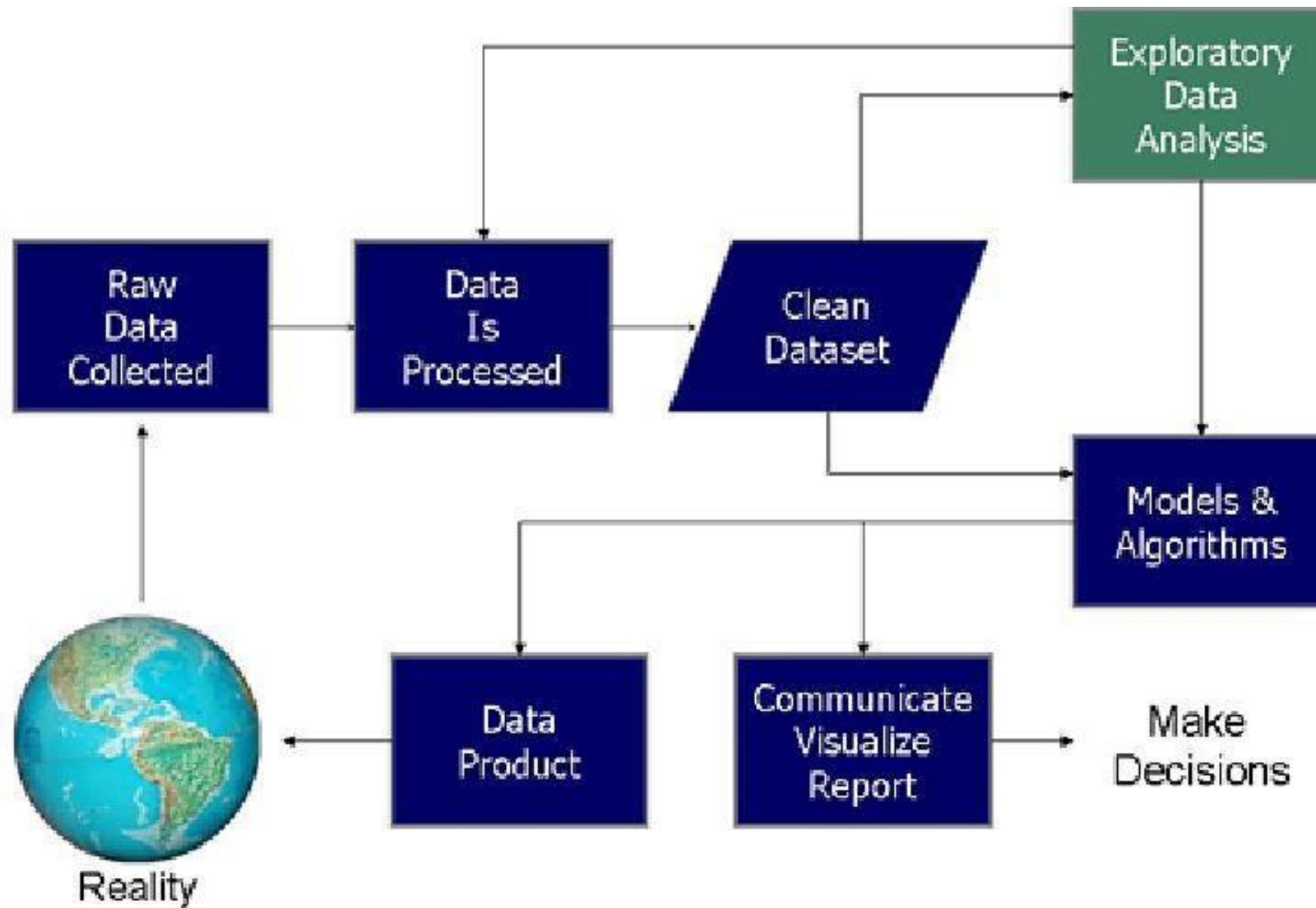


Khoa học Dữ liệu và Học máy

- Học máy là phương pháp quan trọng để xử lý dữ liệu trong ngành data science, bên cạnh những phương pháp truyền thống khác



Quá trình xử lý của khoa học dữ liệu





Ví dụ: hệ thống phát hiện thư rác

1. Thu thập mẫu thư (gồm cả thư rác và thư thường)
2. Xác định đề bài (phân lớp hay đánh giá)
3. Xử lý dữ liệu
4. Chọn mô hình học máy phù hợp với bài toán phân loại thư rác
5. Huấn luyện mô hình
6. Hiệu chỉnh, tinh chỉnh mô hình
7. Áp dụng thực tế (chạy trên email server thực)
8. Tiếp tục cập nhật theo phản hồi của người dùng



Phần 2

Một số loại bài toán học máy



Một số bài toán thực tế

- Hệ thống phân loại email
- Nhận dạng chữ viết từ ảnh
- Ước lượng giá cả của sản phẩm
- Dự báo thời tiết
- Đánh giá trạng thái của người qua ảnh/video
- Trả lời tự động (chat bot)
- Gợi ý sản phẩm phù hợp với nhu cầu khách hàng
- Tự động chơi trò chơi
- Mô phỏng giọng nói của một người nào đó



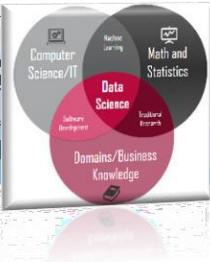
Các lớp bài toán cơ bản

- Học có giám sát (**supervised learning**): học cách tiên đoán đầu ra theo mẫu cho trước
 - ✓ Tập mẫu cho trước, cho cả đầu bài và kết quả
 - ❖ Cho email, chỉ rõ trước đây là spam, đây không phải spam
 - ✓ Mô hình được huấn luyện trên tập mẫu
 - ✓ Thử nghiệm bằng cách cho đầu bài, mô hình tiên đoán kết quả, mô hình đoán càng chính xác càng tốt
 - ❖ Cho một email mới, máy tính đoán xem có phải spam không?
 - ✓ Có 2 loại cơ bản:
 - ❖ Hồi quy (regression): đầu ra là số hoặc vector
 - ❖ Phân lớp (classification): đầu ra thường là xác suất dự báo



Các lớp bài toán cơ bản

- Học không giám sát (*unsupervised learning*): tự khai phá các đặc trưng nội tại hợp lý của đầu vào
 - ✓ Chỉ cho mẫu vào, không cho biết đầu ra
 - ❖ Cho tập băng ghi âm lời nói của một người
 - ✓ Hệ thống tự học trên các mẫu mà không có định hướng
 - ❖ Tạo ra một đoạn phát âm theo ngữ điệu của người đã cho
 - ✓ Một vài chiến lược cơ bản:
 - ❖ Biến đổi dữ liệu đầu vào có số chiều cao thành dữ liệu có số chiều thấp hơn
 - ❖ Dữ liệu có số chiều cao nhưng các đặc trưng thành phần có tính “kinh tế” (*economical*) hơn
 - ❖ Gom cụm dữ liệu đầu vào



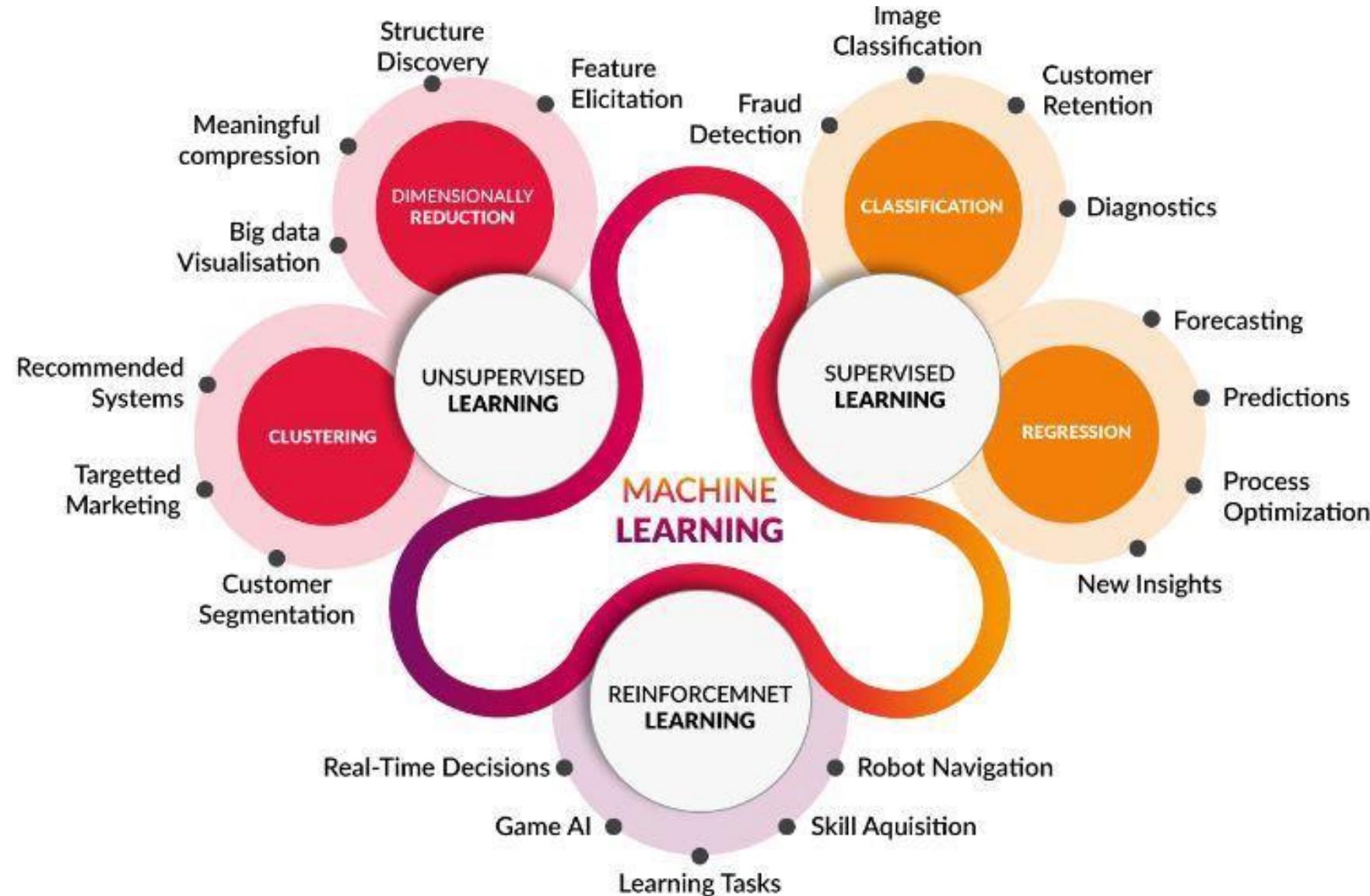
Các lớp bài toán cơ bản

Học tăng cường (reinforcement learning): hiệu chỉnh các siêu tham số (hyperparameter) để cực đại hóa lợi ích trong tương lai

- Cho bối cảnh và các quy tắc
 - ❖ Bàn cờ Vây và các quy tắc của trò chơi cờ Vây
- Ứng với mỗi hành động (hoặc chuỗi hành động), có một phần thưởng tương ứng
 - ❖ Đặt một quân sẽ bị mất điểm, không được hoặc được điểm
- Hệ thống tự điều chỉnh chuỗi hành động sao cho được phần thưởng lớn nhất
 - ❖ Hệ thống học cách chơi để thắng người chơi giỏi nhất



Các lớp bài toán cơ bản





Phần 3

Thư viện học máy scikit-learn



Thư viện học máy scikit-learn

- Scikit-learn xuất phát là một dự án trong một cuộc thi lập trình của Google vào năm 2007, người khởi xướng dự án là David Cournapeau
- Sau đó nhiều viện nghiên cứu và các nhóm ra nhập, đến năm 2010 mới có bản đầu tiên (v0.1 beta)
- Scikit-learn cung cấp gần như tất cả các loại thuật toán học máy cơ bản (khoảng vài chục) và vài trăm biến thể của chúng, cùng với đó là các kĩ thuật xử lý dữ liệu đã được chuẩn hóa
- Cài đặt: `pip install scikit-learn scipy`

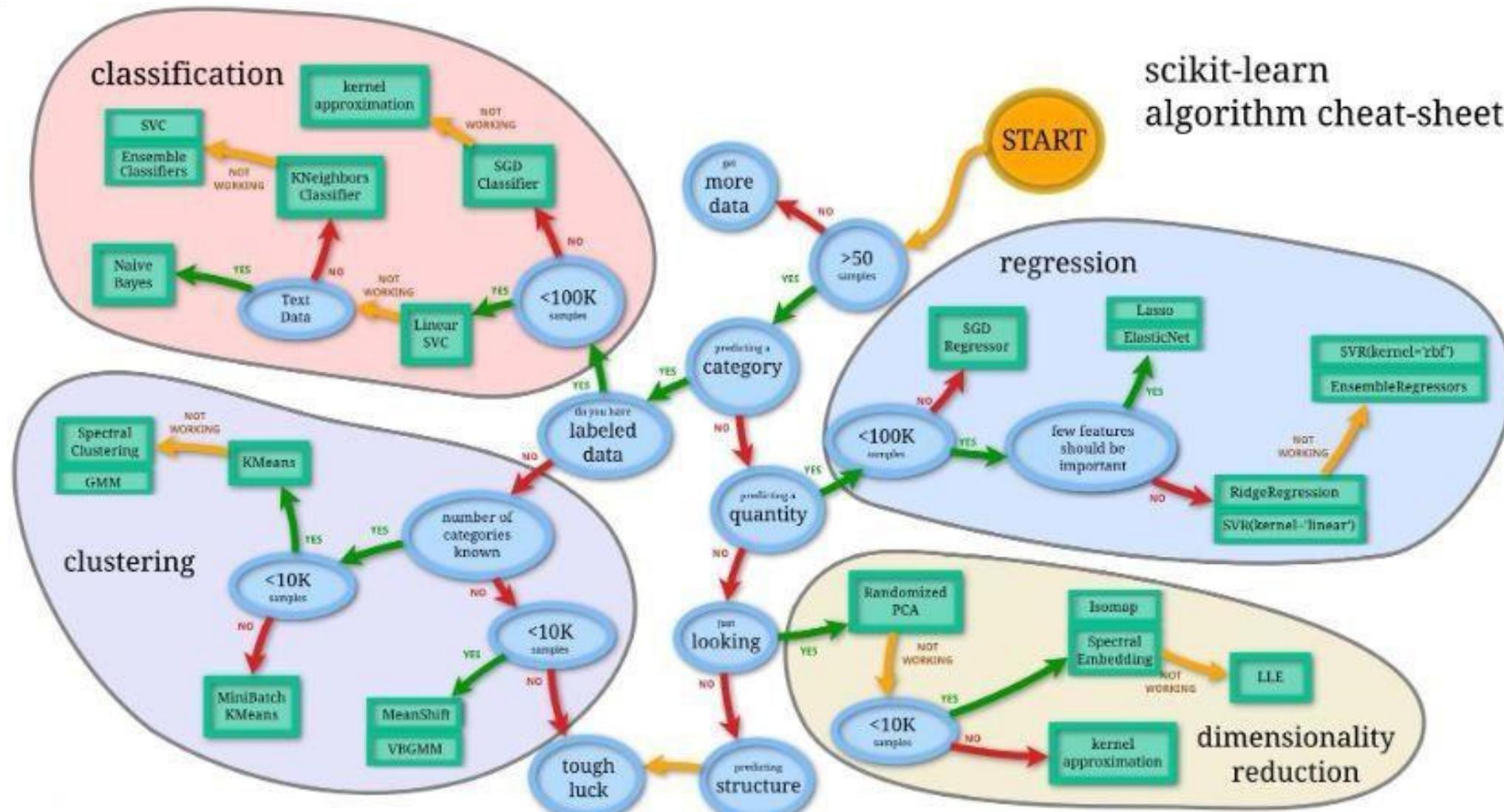


Chọn thuật toán học máy phù hợp

- Scikit-learn xuất phát là một dự án trong một cuộc thi lập trình của Google vào năm 2007, người khởi xướng dự án là David Cournapeau
- Sau đó nhiều viện nghiên cứu và các nhóm ra nhập, đến năm 2010 mới có bản đầu tiên (v0.1 beta)
- Scikit-learn cung cấp gần như tất cả các loại thuật toán học máy cơ bản (khoảng vài chục) và vài trăm biến thể của chúng, cùng với đó là các kĩ thuật xử lý dữ liệu đã được chuẩn hóa
- Cài đặt: `pip install scikit-learn scipy`



Chọn thuật toán học máy phù hợp





Ví dụ: dự báo cân nặng của người

- Tập mẫu quan sát có n người
 - ✓ Gồm tên, chiều cao, cân nặng
 - ✓ Và nhiều loại chỉ số khác nữa
- Xây dựng một mô hình dự báo về cân nặng người, dựa trên các chỉ số còn lại
 - ✓ Trong trường hợp bài toán của ta, chúng ta cố gắng dự báo cân nặng từ chiều cao
 - ✓ Thực tế thì cân nặng phụ thuộc vào nhiều thông số khác nữa, như giới tính, vòng eo,...

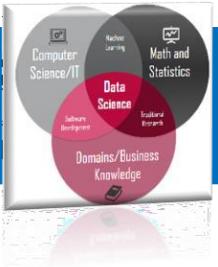
	A	B	C
1	Ten	Cao	Nang
2	A	147	49
3	B	150	50
4	C	153	51
5	D	155	51
6	E	168	60
7	F	170	62
8	G	173	68
9	H	175	65
10	I	178	66
11	J	180	71
12	K	183	68
13	L	165	59
14	M	163	58
15	N	160	56
16	O	158	54
17	P	169	62
18	Q	172	63
19	S	170	62
20	T	176	62
21	U	180	69

Dự báo sử dụng hồi quy tuyến tính

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import linear_model, metrics

# đọc dữ liệu từ file csv
df = pd.read_csv("nguois.csv", index_col = 0)
print(df)
# vẽ biểu đồ minh họa dataset
plt.plot(df.Cao, df.Nang, 'ro')
plt.xlabel('Chiều cao (cm)')
plt.ylabel('Cân nặng (kg)')
plt.show()
```

Dự báo sử dụng hồi quy tuyến tính



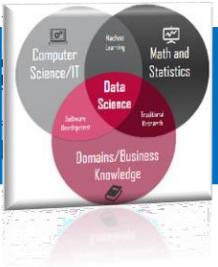
```

# sử dụng hồi quy tuyến tính
X = df.loc[:, ['Cao']].values           # X là dữ liệu đầu vào
y = df.Nang.values                      # y là dữ liệu đầu ra
model = linear_model.LinearRegression() # loại mô hình
model.fit(X, y)                         # tập huấn trên dữ liệu

# in một số thông tin về mô hình
mse = metrics.mean_squared_error(model.predict(X), y)
print("Tổng bình phương sai số trên tập mẫu:", mse)
print("Hệ số hồi quy:", model.coef_)
print("Sai số:", model.intercept_)
print(f"Công thức: [Nặng] = {model.coef_} x [Cao] + {model.intercept_}")

```

Dự báo sử dụng hồi quy tuyến tính



vẽ lại sơ đồ

```
plt.scatter(X, y, c='b')
plt.plot(X, model.predict(X))
plt.show()
```

dự báo một số tình huống

```
while True:
```

```
    x = float(input("Nhập chiều cao (nhập 0 để dừng): "))
```

```
    if x <= 0: break
```

```
    print("Người cao", x, "cm, dự báo cân nặng",
model.predict([[x]]))
```



Mở rộng: thêm cột giới tính

- Vẫn dữ liệu cũ, bổ sung thêm cột giới tính (Nam/Nu)
- Sử dụng phương pháp cũ, để xem giới tính ảnh hưởng như thế nào đến cân nặng

	A	B	C	D
1	Ten	Gioitinh	Cao	Nang
2	A	Nu	147	49
3	B	Nu	150	50
4	C	Nu	153	51
5	D	Nam	155	51
6	E	Nu	168	60
7	F	Nam	170	62
8	G	Nu	173	68
9	H	Nam	175	65
10	I	Nam	178	66
11	J	Nam	180	71
12	K	Nam	183	68
13	L	Nam	165	59
14	M	Nu	163	58
15	N	Nu	160	56
16	O	Nu	158	54
17	P	Nam	169	62
18	Q	Nam	172	63
19	S	Nu	170	62
20	T	Nam	176	62
21	U	Nam	180	69

Dự báo sử dụng hồi quy tuyến tính

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import linear_model, metrics

# đọc dữ liệu từ file csv
df = pd.read_csv("nguois.csv", index_col = 0)
print(df)
# thêm cột mới, giới tính Nam = 1, giới tính Nữ = 0
df['GT'] = df.Gioitinh.apply(lambda x: 1 if x=='Nam' else 0)
print(df)
```

Dự báo sử dụng hồi quy tuyến tính

```

# sử dụng hồi quy tuyến tính
X = df.loc[:, ['Cao', 'GT']].values # X là dữ liệu đầu vào
y = df.Nang.values # y là dữ liệu đầu ra
model = linear_model.LinearRegression() # loại mô hình
model.fit(X, y) # tập huấn trên dữ liệu

# in một số thông tin về mô hình
mse = metrics.mean_squared_error(model.predict(X), y)
print("Tổng bình phương sai số trên tập mẫu:", mse)
print("Hệ số hồi quy:", model.coef_)
print("Sai số:", model.intercept_)
print(f"Công thức: [Nặng] = {model.coef_} x [Cao, Giới tính] + {model.intercept_}")

```

Dự báo sử dụng hồi quy tuyến tính



dự báo một số tình huống

while True:

```
x = float(input("Nhập chiều cao (nhập 0 để dừng): "))
if x <= 0: break
print("Nam giới cao", x, "cm, dự báo cân nặng",
model.predict([[x, 1]]))
print("Nữ giới cao", x, "cm, dự báo cân nặng",
model.predict([[x, 0]]))
```