

Bài 02: Lập trình Python cơ bản (t)



AI Academy Vietnam

1. Các toán tử trong Python

Toán tử số học | Toán tử gán | Toán tử so sánh | Toán tử logic | Toán tử membership

2. Cấu trúc điều khiển

Dạng 1, dạng 2, dạng 3, dạng 4

3. Cấu trúc vòng lặp

Vòng lặp While | Vòng lặp for | Break, continue

4. Cấu trúc dữ liệu cơ sở

Tuple | set | Dictionary

5. Ngoại lệ và xử lý ngoại lệ

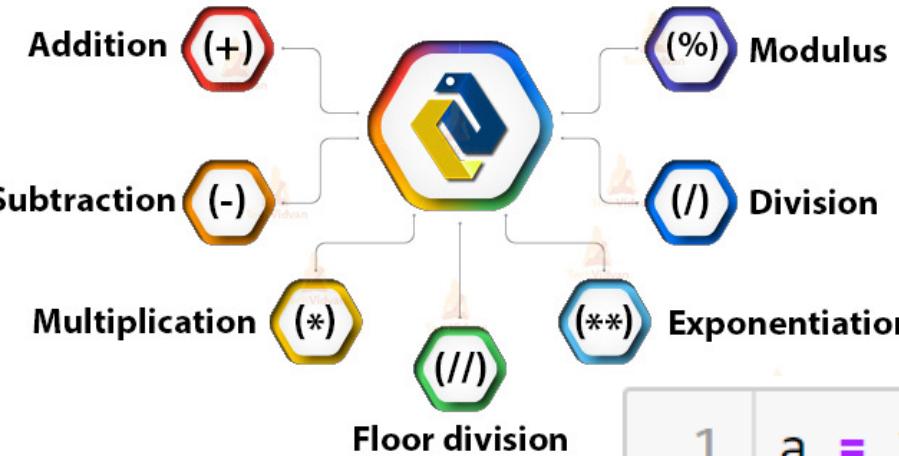


1. Các toán tử trong Python



Python
Programming³

Python Arithmetic Operators



```
1 a = 10
2 b = 8
3 #-----
4 tong = a + b          # Tổng của hai số (+)
5 hieu = a - b          # Hiệu của hai số (-)
6 tich = a*b            # Tích của hai số (*)
7 thuong = a/b           # Thương của hai số (/)
8 thuong_nguyen = a//b  # Phép chia Lấy phần nguyên (//)
9 thuong_du = a % b     # Phép chia Lấy phần dư (%)
10 mu = a**b            # Tính giá trị a Lũy thừa b (**)
```

Các toán tử gán

Toán tử	Mô tả	Ví dụ
=	Toán tử này dùng để gán giá trị của một đối tượng cho một giá trị	$c = a$ (lúc này c sẽ có giá trị = a)
+=	Toán tử này cộng rồi gán giá trị cho đối tượng	$c += a$ (tương đương với $c = c + a$)
-=	Toán tử này trừ rồi gán giá trị cho đối tượng	$c -= a$ (tương đương với $c = c - a$)
*=	Toán tử này trừ rồi gán giá trị cho đối tượng	$c *= a$ (tương đương với $c = c * a$)
/=	Toán tử này chia rồi gán giá trị cho đối tượng	$c /= a$ (tương đương với $c = c / a$)
%	Toán tử này chia hết rồi gán giá trị cho đối tượng	$c \%= a$ (tương đương với $c = c \% a$)
**=	Toán tử này lũy thừa rồi gán giá trị cho đối tượng	$c **= a$ (tương đương với $c = c ** a$)
//=	Toán tử này chia làm tròn rồi GÁN giá trị cho đối tượng	$c //= a$ (tương đương với $c = c // a$)

Các toán tử so sánh

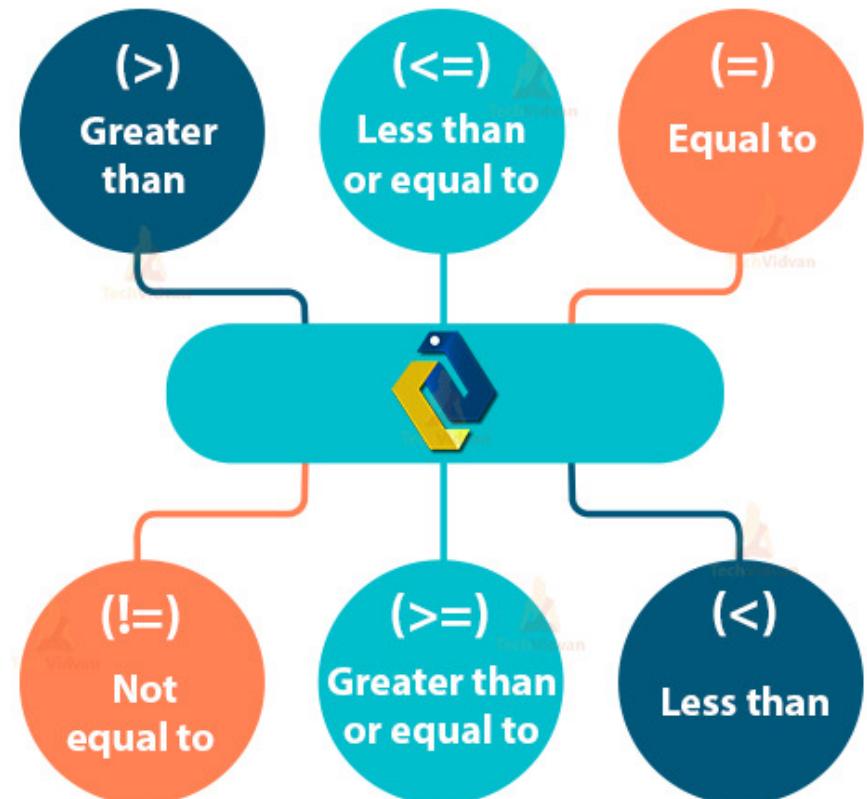
Toán tử	Mô tả	Ví Dụ (a =8, b=10)
<code>==</code>	So sánh giá trị của các đối số xem có bằng nhau hay không. Nếu bằng nhau thì kết quả trả về sẽ là True và ngược lại sẽ là False.	<code>a == b // False</code>
<code>!=</code>	So sánh giá trị của các đối số xem có khác nhau hay không. Nếu khác nhau thì kết quả trả về sẽ là True và ngược lại sẽ là False.	<code>a != b //True</code>
<code><</code>	Dấu < đại diện cho phép toán nhỏ hơn, nếu đối số 1 nhỏ hơn đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	<code>a < b //True</code>
<code>></code>	Dấu > đại diện cho phép toán lớn hơn, nếu đối số 1 lớn hơn đối số 2 thì kết quả trả về là True và ngược lại sẽ là False.	<code>a > b //False</code>
<code><=</code>	Dấu > đại diện cho phép toán nhỏ hơn hoặc bằng, nếu đối số 1 nhỏ hơn hoặc bằng đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	<code>a <= b //True</code>
<code>>=</code>	Dấu > đại diện cho phép toán lớn hơn hoặc bằng, nếu đối số 1 lớn hơn hoặc bằng đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	<code>a>= b //False</code>

Các toán tử so sánh (t)

```
1 #Kết quả của các phép so sánh có kiểu dữ liệu Boolean
2 a = 8
3 b = 10
4 kt = a>b
5 print(type(kt))
6 -----
7 print('1) SS lớn hơn (a>b): ', a>b)
8 print('2) SS nhỏ hơn (a<b): ', a<b)
9 print('3) SS bằng (a==b): ', a==b)
10 print('4) SS lớn hơn hoặc bằng (a>=b): ', a>=b)
11 print('5) SS nhỏ hơn hoặc bằng (a<=b): ', a<=b)
12 print('6) SS khác (a!=b): ', a!=b)
```

```
<class 'bool'>
1) SS lớn hơn (a>b): False
2) SS nhỏ hơn (a<b): True
3) SS bằng (a==b): False
4) SS lớn hơn hoặc bằng (a>=b): False
5) SS nhỏ hơn hoặc bằng (a<=b): True
6) SS khác (a!=b): True
```

PYTHON RELATIONAL OPERATORS



Học viên nhập code và đọc kết quả của các phép so sánh ở trên!

Các toán tử logic, xác thực

Các toán tử logic

Toán tử	Mô tả	Ví dụ
and	Nếu 2 vé của toán tử này đều là True thì kết quả sẽ là True và ngược lại nếu 1 trong 2 vé là False thì kết quả trả về sẽ là False.	$x < 5 \text{ and } x < 10$
or	Nếu 1 trong 2 vé là True thì kết quả trả về sẽ là True và ngược lại nếu cả 2 vé là False thì kết quả trả về sẽ là False.	$x < 5 \text{ or } x < 4$
not	Đây là dạng phủ định, nếu biểu thức là True thì nó sẽ trả về là False và ngược lại.	$\text{not}(x < 5 \text{ and } x < 10)$

Các toán tử xác thực

Toán tử	Mô tả	Ví dụ: a=4, b=5
is	Toán tử này sẽ trả về True nếu $a == b$ và ngược lại	$a \text{ is } b // \text{False}$
is not	Toán tử này sẽ trả về True nếu $a != b$ và ngược lại	$a \text{ is not } b // \text{True}$

Toán tử membership

Toán tử	Mô tả	Ví dụ
in	Nếu 1 đối số thuộc một tập đối số nó sẽ trả về True và ngược lại.	a in b
not in	Nếu 1 đối số không thuộc một tập đối số nó sẽ trả về True và ngược lại.	a not in b

```
1 #D/ Các toán tử membership
2 a = [4, 6, 9 , 0]
3 kt1 = 4 in a      #Kiểm tra 1 phần tử có trong danh sách không?
4 kt2 = 0 not in a #Kiểm tra 1 phần tử không có trong danh sách không?
5
6 print('Kết quả 1: ', kt1)
7 print('Kết quả 2: ', kt2)
```

Kết quả 1: True
Kết quả 2: False

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+X, -X, ~X	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
		Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, Identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

2. Cấu trúc điều khiển (if)



Python
Programming 11

Câu lệnh điều kiện

- Câu lệnh điều kiện là một trong những câu lệnh cơ bản của bất cứ ngôn ngữ lập trình nào.

Ngôn ngữ tự nhiên	Ngôn ngữ lập trình
<p>Nếu Bạn đủ 18 tuổi thì <i>“Bạn được kết hôn”</i>, Ngược lại thì <i>“Bạn chưa được kết hôn”</i></p>	<p>if (age > = 18): <i>print('Bạn được kết hôn!')</i> else: <i>print('Bạn chưa được kết hôn!')</i></p>

```
1 #Câu Lệnh điều kiện
2 so_tien = input ('Nhập vào số tiền bạn có: ')
3 so_tien = int(so_tien)
4 if (so_tien >= 1000000000):
5     print('Bạn đã là một tỷ phú!')
6 else:
7     print('Bạn còn phải kiếm nhiều tiền hơn!')
```

Các dạng câu lệnh điều kiện

- Dạng 1:

if (điều kiện1):

Nhóm lệnh 1

(Nếu điều kiện1 đúng thì thực hiện nhóm lệnh 1)

In [2]:

```
num = 3
if num > 0:
    print(num, "là số dương.")
print("Thông điệp này luôn được in.")
```

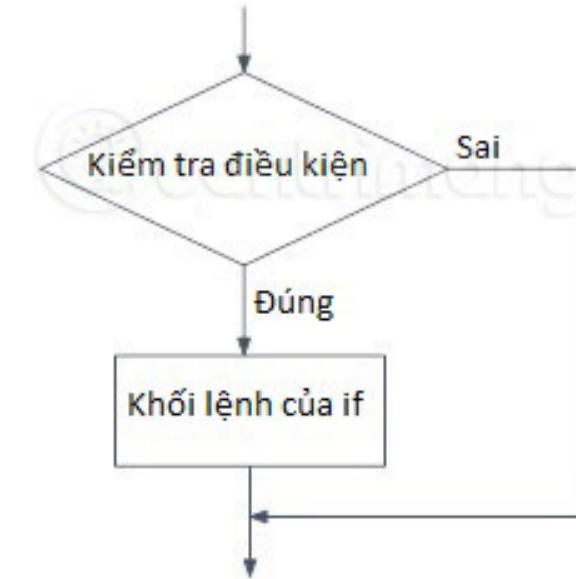
3 là số dương.

Thông điệp này luôn được in.

In [3]:

```
num = -1
if num > 0:
    print(num, "là số dương.")
print("Thông điệp này luôn được in.")
```

Thông điệp này luôn được in.



Ví dụ: Kiểm tra số chẵn

Nhập vào 1 số nguyên N, kiểm tra nếu N là số chẵn hiển thị thông báo.

- **Yêu cầu:**
Nhập vào một số: 12
- **Kiểm tra và hiển thị thông báo:**
Số 12 là số chẵn!

12

Odd

13

Even

Các dạng câu lệnh điều kiện (t)

- Dạng 2:

if (điều kiện1):

Nhóm lệnh 1

else:

Nhóm lệnh 2

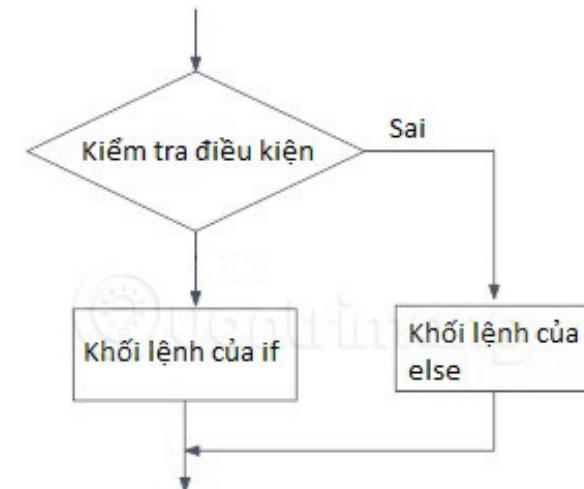
(Nếu điều kiện1 đúng thì thực hiện nhóm lệnh 1, nếu sai thực hiện nhóm lệnh 2)

```
In [5]: num = 3
if num >= 0:
    print("So duong")
else:
    print("So am")
```

So duong

```
In [6]: num = -1
if num >= 0:
    print("So duong")
else:
    print("So am")
```

So am



Ví dụ: Kiểm tra số chẵn – lẻ

Nhập vào 1 số nguyên N, kiểm tra nếu N là số chẵn hiển thị thông báo “Đây là số chẵn!”, ngược lại thông báo “Đây là số lẻ!”

12

13

Odd

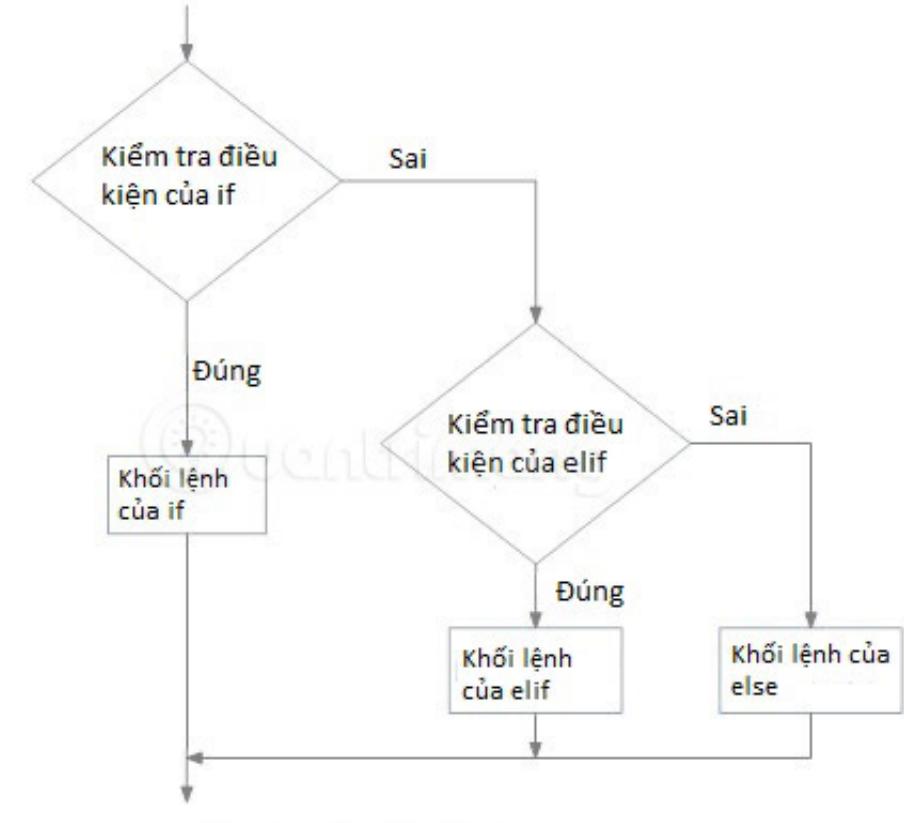
Even

Nhập vào một số:12
Đây là số chẵn!

Nhập vào một số:13
Đây là số lẻ!

Các dạng câu lệnh điều kiện (t)

- Dạng 3 (if lồng nhau):
if (điều kiện1):
 Nhóm lệnh 1
elif (điều kiện 2):
 Nhóm lệnh 2
 ...
elif (điều kiện n):
 Nhóm lệnh n
else:
 Nhóm lệnh x

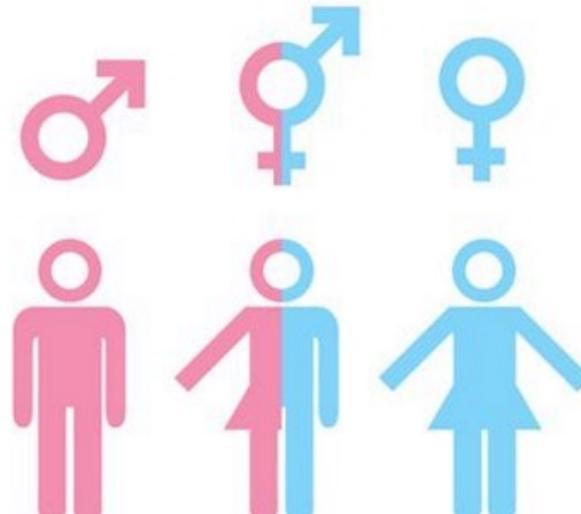


(Kiểm tra điều kiện1 đúng thì thực hiện nhóm lệnh 1, nếu sai kiểm tra điều kiện 2 đúng thực hiện nhóm lệnh 2 ...nếu tất cả các điều kiện sai thực hiện nhóm lệnh x)

Ví dụ: Chào theo giới tính

Nhập vào giới tính (0=Nam | 1=Nữ)

- Nếu nhập vào 0 → Chào anh đẹp trai!
- Nếu nhập vào 1 → Chào chị xinh gái!
- Nếu nhập khác 0 hoặc 1 → Cảnh báo: Giới tính không xác định!



Nhập giới tính (0:Nam - 1:Nữ):0
Chào anh đẹp trai!

Nhập giới tính (0:Nam - 1:Nữ):1
Chào chị xinh gái!

Nhập giới tính (0:Nam - 1:Nữ):2
Cảnh báo: Giới tính không xác định!

Các dạng câu lệnh điều kiện (t)

- Dạng 4 (if lồng nhau – Nested if):

if (điều kiện1):

if (điều kiện 1.1):

Nhóm lệnh 1.1

elif (điều kiện 1.2):

Nhóm lệnh 1.2

else:

Nhóm lệnh 1.x

else:

Nhóm lệnh 2

```
num = float(input("Nhập một số: "))
if num >= 0:
    if num == 0:
        print("Số Không")
    else:
        print("Số dương")
else:
    print("Số âm")
```

Nhập một số: 0
Số Không

Thực hành

3. Cấu trúc vòng lặp trong Python



Python
Programming²¹

Câu lệnh vòng lặp

- Cũng như câu lệnh điều kiện, Câu lệnh vòng lặp là một trong những câu lệnh cơ bản của bất cứ ngôn ngữ lập trình nào.
- Để giải quyết bài toán, chúng ta cần thực hiện một công việc nào đó lặp đi lặp lại rất nhiều lần. Số lần lặp đó có thể biết trước hoặc không biết trước.

Ngôn ngữ tự nhiên	Ngôn ngữ lập trình
<p>Tính tổng các số từ 1 đến 10:</p> <p>$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10$</p>	<pre>tong = 0 for i in range(1,11): tong = tong + i print('Tổng từ 1 đến 10 là:', tong)</pre>

Có 2 kiểu vòng lặp trong python:

- * Vòng lặp while
- * Vòng lặp for

Vòng lặp While

- Vòng lặp while sử dụng khi **không biết trước số lần lặp**.
- Cú pháp:

While <điều kiện>:

Nhóm lệnh 1

```
1 n = int(input('Em sinh tháng mấy?'))
2 i=1
3 while(i<=n):
4     print(i, ' I Love You!')
5     i=i+1
6
7 #Câu Lệnh Lặp ngoài vòng Lặp while
8 print('-----AIACADEMY-----')
```

```
Em sinh tháng mấy?3
1 ) I Love You!
2 ) I Love You!
3 ) I Love You!
-----AIACADEMY-----
```



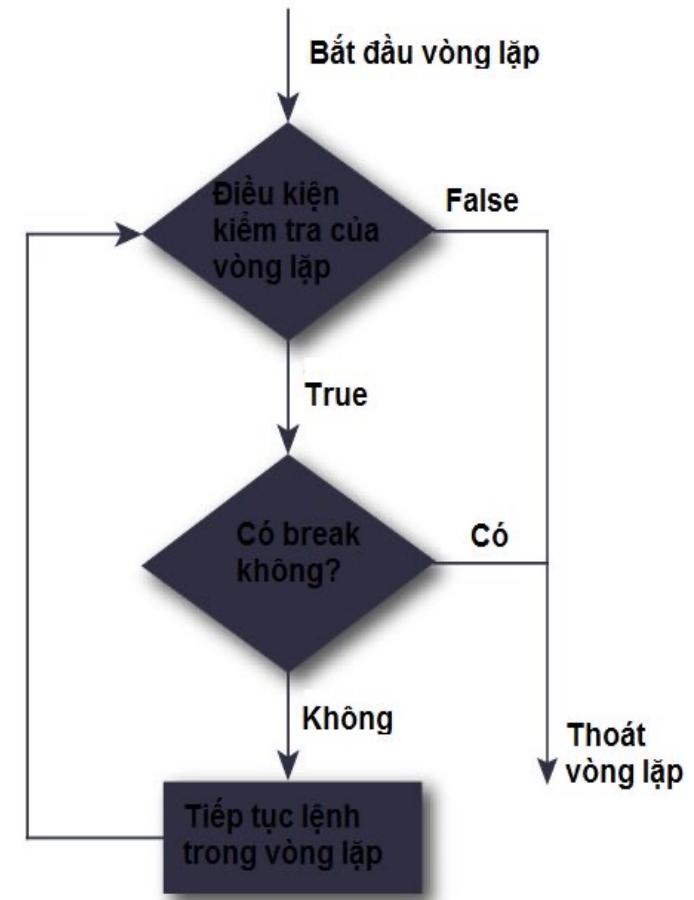


Lệnh break

- Lệnh **break** kết thúc vòng lặp chứa nó và truyền điều khiển đến lệnh tiếp theo sau khối lệnh của vòng lặp đó.

```
1 n = int(input('Em sinh tháng mấy? '))
2 i=1
3 while(i<=n):
4     print(i, ' I Love You!')
5
6     #Chỉ hiển thị tối đa 3 Lần
7     if (i==3):
8         break #Thoát ra khỏi vòng Lặp while
9
10    i=i+1
11
12 #Câu Lệnh Lặp ngoài vòng Lặp while
13 print('-----AIACADEMY-----')
```

```
Em sinh tháng mấy? 8
1 ) I Love You!
2 ) I Love You!
3 ) I Love You!
-----AIACADEMY-----
```

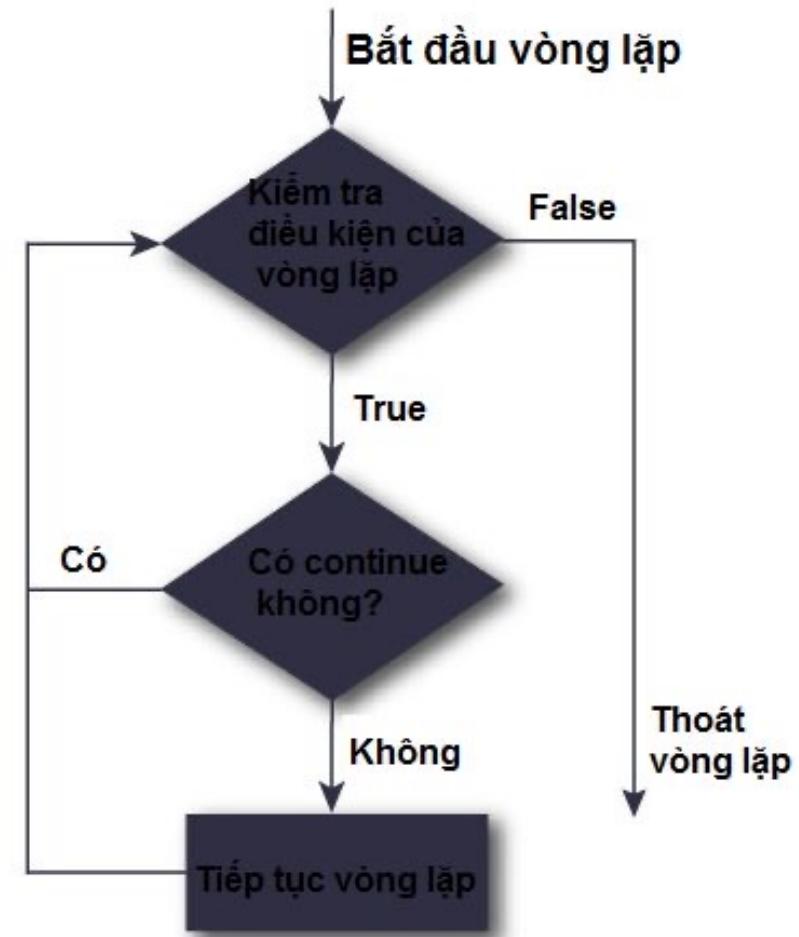


Lệnh continue

- Lệnh **continue** bỏ qua phần còn lại của khối lệnh bên trong vòng lặp, áp dụng cho vòng lặp tiếp theo. Nghĩa là vòng lặp không chấm dứt nó sẽ tiếp tục với số lần lặp kế tiếp

```
1 n = 20
2 i = 1
3 while (i<=n):
4     i = i+1
5     if (i%3!=0):
6         continue
7         #Bỏ qua các câu Lệnh phía sau nếu ko chia hết cho 3
8         print(i)
9
10 #Câu Lệnh Lặp ngoài vòng Lặp while
11 print('-----AIACADEMY-----')
```

```
3
6
9
12
15
18
21
-----AIACADEMY-----
```



Lệnh whileelse

- Cấu trúc:

while <điều kiện>:

<Nhóm lệnh A>

else:

<Nhóm lệnh B>

 ↗ 1 số bài toán cần
(nếu có break thì stop)

- Thực hiện nhóm lệnh A cho đến khi điều kiện còn đúng.
- Nếu điều kiện sai thực hiện nhóm lệnh B.
(ít nhất thực hiện nhóm lệnh B 1 lần)

```
1 counter = 4
2 while counter < 3:
3     print("Inside loop")
4     counter = counter + 1
5 else:
6     print("Inside else")
7 #-----
8 print('----Bên ngoài vòng lặp while----')
```

Inside else

----Bên ngoài vòng lặp while----

Lệnh while True

- Cấu trúc:

while True:

<Nhóm lệnh thực hiện khi điều kiện đúng>

if <điều kiện dừng>:

break

Với cấu trúc này các **câu lệnh sẽ thực hiện lặp đi lặp lại**, cho đến khi biểu thức <điều kiện dừng> thỏa mãn. Lúc đó câu lệnh **if** sẽ giúp cho lệnh **break** được thực thi và dừng vòng lặp.

```
1 #chỉ cho phép nhập tháng sinh 1 - 12
2 while True:
3     n = int(input('Em sinh tháng mấy? '))
4     if (1<= n <= 12):
5         #'Tháng sinh nhập vào hợp lệ!'
6         break;
7     print('Tháng không đúng, vui lòng nhập lại')
8 #Câu lệnh ngoài vòng Lặp while
9 print('Chào em cô gái tháng ', n)
```

```
Em sinh tháng mấy? 15
Tháng không đúng, vui lòng nhập lại
Em sinh tháng mấy? 10
Chào em cô gái tháng 10
```

Vòng lặp for

- Vòng lặp for sử dụng khi **biết trước số lần lặp**.
- Cú pháp:

for <biến chạy> in <dãy>:

Nhóm lệnh 1

<Biến chạy> sẽ lần lượt nhận các giá trị của các thành phần có trong <dãy>. Dãy có thể là một danh sách (list), chuỗi ký tự (str), dãy số....

```
1 #Tính 10! = 1*2*3*4*5*6*7*8*9*10
2 #Tổng 10 = 1+2+3+4+5+6+7+8+9+10
3 n = 10
4 tich = 1
5 tong = 0
6 for i in range (1, n+1):
7     #Mỗi Lần Lặp biến i tăng Lên 1
8     tich = tich*i
9     tong = tong+i
10
11 print ('10! = ', tich)
12 print('10+ = ', tong)
```

10! = 3628800
10+ = 55

Vòng lặp for (2)

- **Vòng lặp for với chuỗi ký tự:** Biến chạy sẽ lần lượt nhận các giá trị là các ký tự trong chuỗi ký tự.

```
1 #Vòng lặp for với chuỗi ký tự:  
2 st = 'HUMG IN MY MIND'  
3 for i in st:  
4     print('ký tự: ', i)
```

1

ký tự: H
ký tự: U
ký tự: M
ký tự: G
ký tự:
ký tự: I
ký tự: N
ký tự:
ký tự: M
ký tự: Y
ký tự:
ký tự: M
ký tự: I
ký tự: N
ký tự: D

```
1 #Đếm số ký tự M trong chuỗi  
2 st = 'HUMG IN MY MIND'  
3 dem = 0  
4 for i in st:  
5     if (i=='M'): dem=dem+1  
6 print('Số ký tự M có trong chuỗi là: ', dem)
```

2

Số ký tự M có trong chuỗi là: 3

Vòng lặp for (3)

- **Vòng lặp for với danh sách:** Biến chạy sẽ lần lượt nhận các giá trị là các phần tử trong danh sách.

```
1 #Vòng Lặp for với danh sách
2 hoc_sinh = ['Lê Thùy Dung', 'Trần Đức Hùng',
3             'Nguyễn Lan Anh', 'Mai Phương Thúy',
4             'Trần Thanh Thủy', 'Kiều Thành Công']
5
6 print('Danh sách học sinh bao gồm:')
7 tt = 1
8 for i in hoc_sinh:
9     print( tt, ') ', i)
10    tt = tt+1
```

Danh sách học sinh bao gồm:

```
1 ) Lê Thùy Dung
2 ) Trần Đức Hùng
3 ) Nguyễn Lan Anh
4 ) Mai Phương Thúy
5 ) Trần Thanh Thủy
6 ) Kiều Thành Công
```

Vòng lặp for (4)

- **Vòng lặp for với lệnh range():** Lệnh range() trong Python kết hợp với vòng lặp for sẽ trở nên rất hữu hiệu trong việc kiểm soát giá trị bắt đầu, kết thúc và bước nhảy của biến chạy.
- Cú pháp:

```
for <biến chạy> in range(<bắt đầu>, <kết thúc>, <bước nhảy>):  
    nhóm lệnh thực hiện
```

- **<bắt đầu>** là giá trị khởi gán ban đầu cho biến chạy (mặc định = 0)
- **<kết thúc>** là giá trị kết thúc cho biến chạy, nhưng không bao gồm chính nó (< kết thúc>)
- **<bước nhảy>** là giá trị mà biến nhảy tăng thêm sau mỗi lần lặp (mặc định = 1)

Vòng lặp for (4)

- Vòng lặp for với lệnh range():

1 #Lệnh for với range()
2 for i in range(5):
3 #Giá trị khởi đầu mặc định = 0
4 #Bước nhảy mặc định = 1
5 print('i = ',i)

i = 0
i = 1
i = 2
i = 3
i = 4

1

1 #Lệnh for với range(m,n)
2 for i in range(5,10):
3 #Giá trị khởi đầu m = 5
4 #Giá trị kết thúc n = 10
5 #Bước nhảy mặc định = 1
6 print('i = ',i)

i = 5
i = 6
i = 7
i = 8
i = 9

2

1 #Lệnh for với range(m,n,d)
2 for i in range(2,11,2):
3 #Giá trị khởi đầu m = 2
4 #Giá trị kết thúc n = 11
5 #Bước nhảy d = 2
6 print('i = ',i)

i = 2
i = 4
i = 6
i = 8
i = 10

3

Vòng lặp for lồng nhau

- Trong một số bài toán chúng ta cần kết hợp và sử dụng nhiều câu lệnh lặp đặt lồng nhau để giải quyết.

```
1 #Hiển thị bảng cửu chương từ 2 -> 9
2 for i in range(2,10):
3     print('Bảng cửu chương ', i)
4     for j in range(1,11):
5         print (i , ' x ', j, ' = ', i*j)
6     print('-----')
```

Bảng cửu chương 2

2	x	1	=	2
2	x	2	=	4
2	x	3	=	6
2	x	4	=	8
2	x	5	=	10
2	x	6	=	12
2	x	7	=	14
2	x	8	=	16
2	x	9	=	18
2	x	10	=	20

Break, continue cho cả while và for

```
for var in sequence:  
    # codes inside for loop  
    if condition:  
        break  
        # codes inside for loop  
  
    # codes outside for loop
```

```
for var in sequence:  
    # codes inside for loop  
    if condition:  
        continue  
        # codes inside for loop  
  
    # codes outside for loop
```

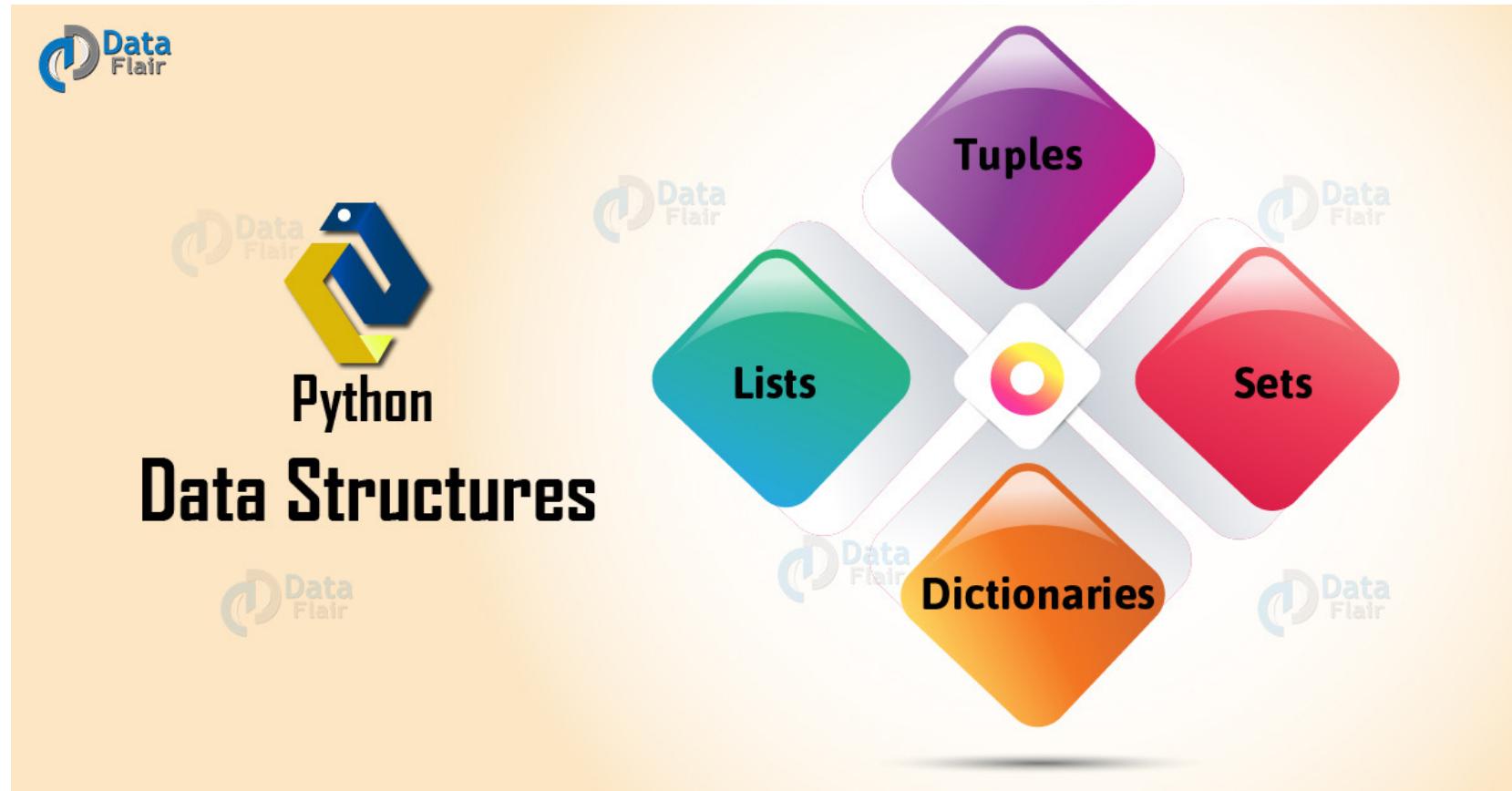
```
while test expression:  
    # codes inside while loop  
    if condition:  
        break  
        # codes inside while loop  
  
    # codes outside while loop
```

```
while test expression:  
    # codes inside while loop  
    if condition:  
        continue  
        # codes inside while loop  
  
    # codes outside while loop
```

Thực hành

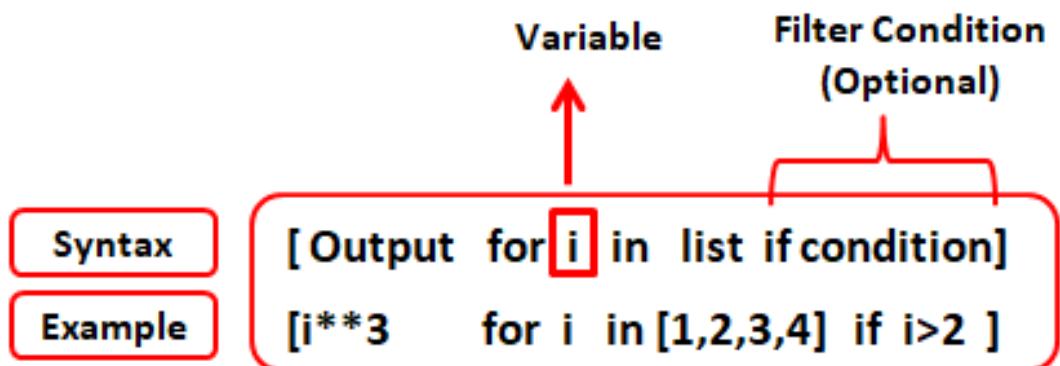


4. Các cấu trúc dữ liệu cơ sở



List comprehension

- List comprehension là một cách dễ dàng, ngắn gọn và nhanh chóng để xây dựng một danh sách theo một điều kiện nào đó.
- List comprehension sử dụng một biểu thức đi kèm với lệnh for hoặc lệnh if được đặt trong cặp dấu ngoặc vuông []



```
1 #Tạo danh sách gồm 10 phần tử có giá trị là 3**x (x = [0,9])
2 list_cub3 = [3**x for x in range(10)]
3 list_cub3
```

[1, 3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049]

```
1 #Tạo danh sách gồm các phần tử chia hết cho 4 nhỏ hơn 50
2 list_4 = [x for x in range(50) if x%4==0]
3 list_4
```

[0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48]

Thực hành

4.2 Tuple

Tuple

- Tuple giống như các list với ngoại lệ là dữ liệu một khi được nhập vào bộ dữ liệu không thể thay đổi bất kể điều gì.

LIST	TUPLE
Được sử dụng cho các loại dữ liệu đồng nhất	Thường được sử dụng cho các loại dữ liệu không đồng nhất
Có thể thay đổi trong môi trường	Bất biến trong môi trường giúp lắp lại nhanh hơn
Không có yếu tố bất biến	Các yếu tố bất biến có thể được sử dụng làm key cho từ điển
Không đảm bảo rằng dữ liệu được bảo vệ chống ghi	Việc thực hiện một bộ dữ liệu không thay đổi đảm bảo rằng nó được bảo vệ chống ghi

Tuple

- Khai báo tuple:

<Tên_danh_sách> = (<Phần tử 1>, <Phần tử 2>, ..., <Phần tử n>)

```
1 #Khai báo biến Tuple
2 Tuple1 = (1,2,5,6)
3 Tuple2 = ('a', "b", 'c', "d")
4 Tuple3 = () #empty tuple
5 Tuple4 = 5,3,1
6 Tuple5 = ("London", "Tokyo", "Korea", 1986,1640, 1948)
7 print(Tuple1)
8 print(Tuple2)
9 print(Tuple3)
10 print(Tuple4)
11 print(Tuple5)
```

```
(1, 2, 5, 6)
('a', 'b', 'c', 'd')
()
(5, 3, 1)
('London', 'Tokyo', 'Korea', 1986, 1640, 1948)
```

```
1 #Lưu ý: nếu chỉ có một phần tử
2 Tup1 = (5)
3 Tup2 = (5,) #phải có thêm dấu ,
4 print(type(Tup1))
5 print(type(Tup2))
```

```
<class 'int'>
<class 'tuple'>
```

Tuple

- Tuple là kiểu dữ liệu không thay đổi được nên nó chỉ có phương thức **count()** và **index()**

```
1 #Tuple ko cho phép thay đổi dữ liệu:  
2 num = (1,2,3)  
3 num[1] = 20 #Thay đổi phần tử tại index=1
```

```
TypeError Traceback (most recent call last)  
<ipython-input-53-e200330d0003> in <module>  
      1 #Tuple ko cho phép thay đổi dữ liệu:  
      2 num = (1,2,3)  
----> 3 num[1] = 20 #Thay đổi phần tử tại index=1
```

TypeError: 'tuple' object does not support item assignment

```
1 #Tuple ko cho phép xóa dữ liệu:  
2 tup = (1,2,3,4,5)  
3 del tup[2] #Xóa phần tử tại index 2
```

```
TypeError Traceback (most recent call last)  
<ipython-input-54-2a4ac9218560> in <module>  
      1 #Tuple ko cho phép xóa dữ liệu:  
      2 tup = (1,2,3,4,5)  
----> 3 del tup[2] #Xóa phần tử tại index 2
```

TypeError: 'tuple' object doesn't support item deletion

```
1 #count(n): đếm số phần tử trong danh sách có giá trị n  
2 (1,2,2,2,1,4,2).count(2)
```

4

```
1 #index(n): chỉ số phần tử đầu tiên trong tuple có giá trị n  
2 (1, 3, 5, 7, 9, 7).index(7)
```

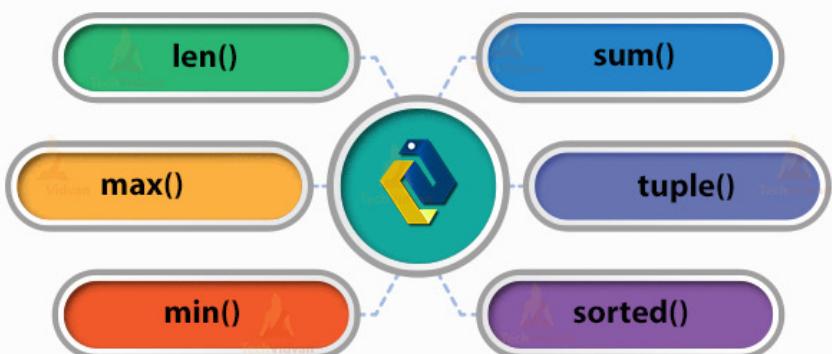
3

Tuple

```
1 #Truy cập tới các phần tử trong Tuple:  
2 #Tương tự như với List  
3 print(Tuple5)  
4 print(Tuple5[0])  
5 print(Tuple5[-1])  
6 print(Tuple5[2:4])  
7 print(Tuple5[3:])
```

```
('London', 'Tokyo', 'Korea', 1986, 1640, 1948)  
London  
1948  
('Korea', 1986)  
(1986, 1640, 1948)
```

Python Tuple Functions



```
1 #Khai báo biến kiểu tuple:  
2 tup = (4,8,9,0,5)  
3 type(tup)
```

tuple

```
1 #1.Len: Số phần tử của tuple  
2 print('1. len:',len(tup))  
3 #2.max: phần tử Lớn nhất của tuple  
4 print('2. max:',max(tup))  
5 #3.min: phần tử nhỏ nhất của tuple  
6 print('3. min:',min(tup))  
7 #4.sum: tổng các phần tử của tuple  
8 print('4. sum:',sum(tup))  
9 #5.sorted: sắp xếp các phần tử của tuple  
10 print('5. sort:',sorted(tup))
```

1. len: 5
2. max: 9
3. min: 0
4. sum: 26
5. sort: [0, 4, 5, 8, 9]

Thực hành

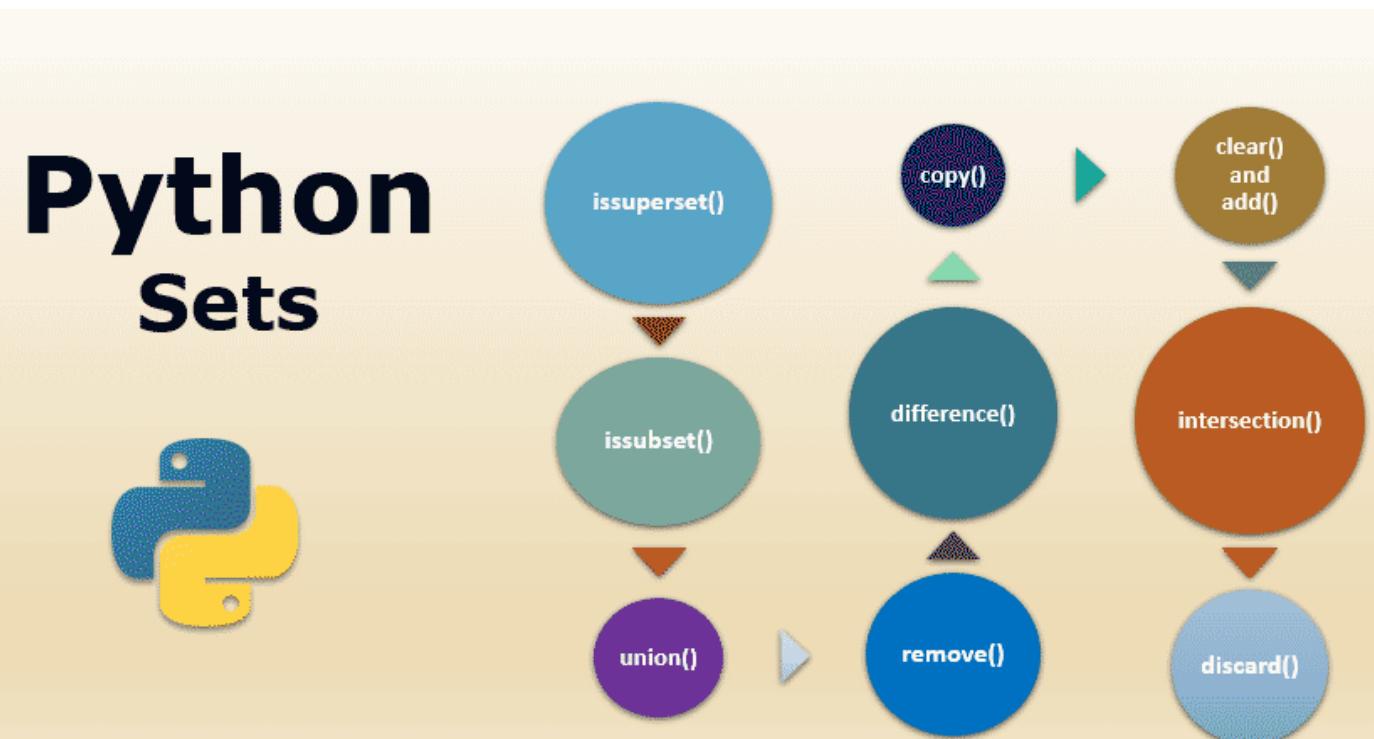
4.3 Set

Set

- Set là bao gồm các phần tử theo thứ tự, không trùng nhau được bọc bởi hai dấu ngoặc nhọn {} hoặc khởi tạo bằng set().
- Các phần tử trong set không được đánh chỉ số

```
1 #Khai báo biến kiểu set
2 a = {1, 2, 7, 5, 5, 1, 3}
3 b = {99, 100, 200, 'yes', 'no'}
4 empty_set = set() #Khai báo set rỗng
5 print(a)
6 print(b)
7 print(empty_set)
8 print(type(empty_set))
```

```
{1, 2, 3, 5, 7}
{99, 100, 200, 'yes', 'no'}
set()
<class 'set'>
```



Phương thức của set

add()	Thêm một phần tử vào set.
clear()	Xóa tất cả phần tử của set.
copy()	Trả về bản sao chép của set.
difference()	Trả về set mới chứa những phần tử khác nhau của 2 hay nhiều set.
difference_update()	Xóa tất cả các phần tử của set khác từ set này.
discard()	Xóa phần tử nếu nó có mặt trong set.
intersection()	Trả về set mới chứa phần tử chung của 2 set.
intersection_update()	Cập nhật set với phần tử chung của chính nó và set khác.
isdisjoint()	Trả về True nếu 2 set không có phần tử chung.
issubset()	Trả về True nếu set khác chứa set này.
issuperset()	Trả về True nếu set này chứa set khác.
pop()	Xóa và trả về phần tử ngẫu nhiên, báo lỗi KeyError nếu set rỗng.
remove()	Xóa phần tử từ set. Nếu phần tử đó không có trong set sẽ báo lỗi KeyError.
symmetric_difference()	Trả về set mới chứa những phần tử không phải là phần tử chung của 2 set.
symmetric_difference_update()	Cập nhật set với những phần tử khác nhau của chính nó và set khác.
union()	Trả về set mới là hợp của 2 set.
update()	Cập nhật set với hợp của chính nó và set khác.

Phương thức của set

- Ví dụ

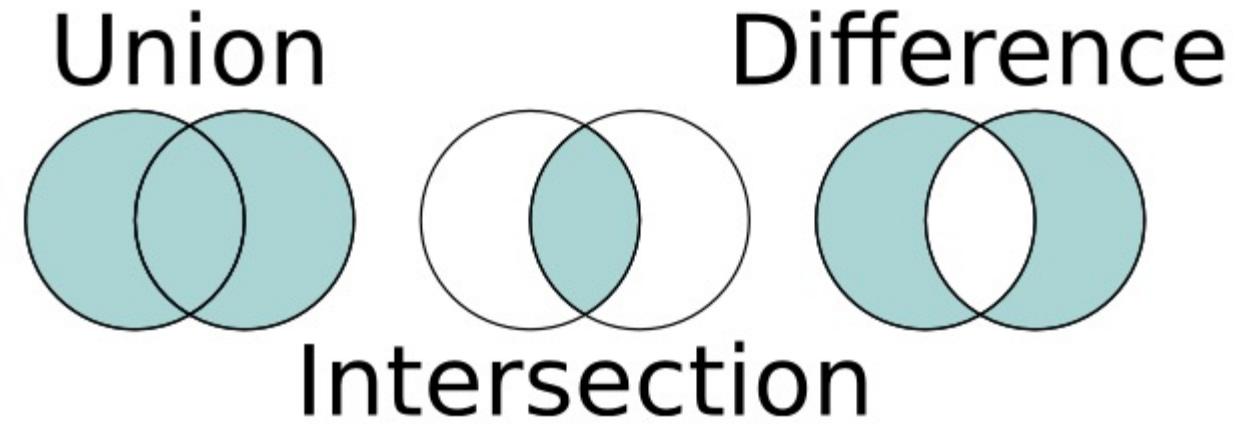
```
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# Intersection of sets
# use & operator
print(A & B)

# use intersection function on A
print(A.intersection(B))

# Union of sets
# use | operator
print(A | B)

# use union function
print(A.union(B))
```



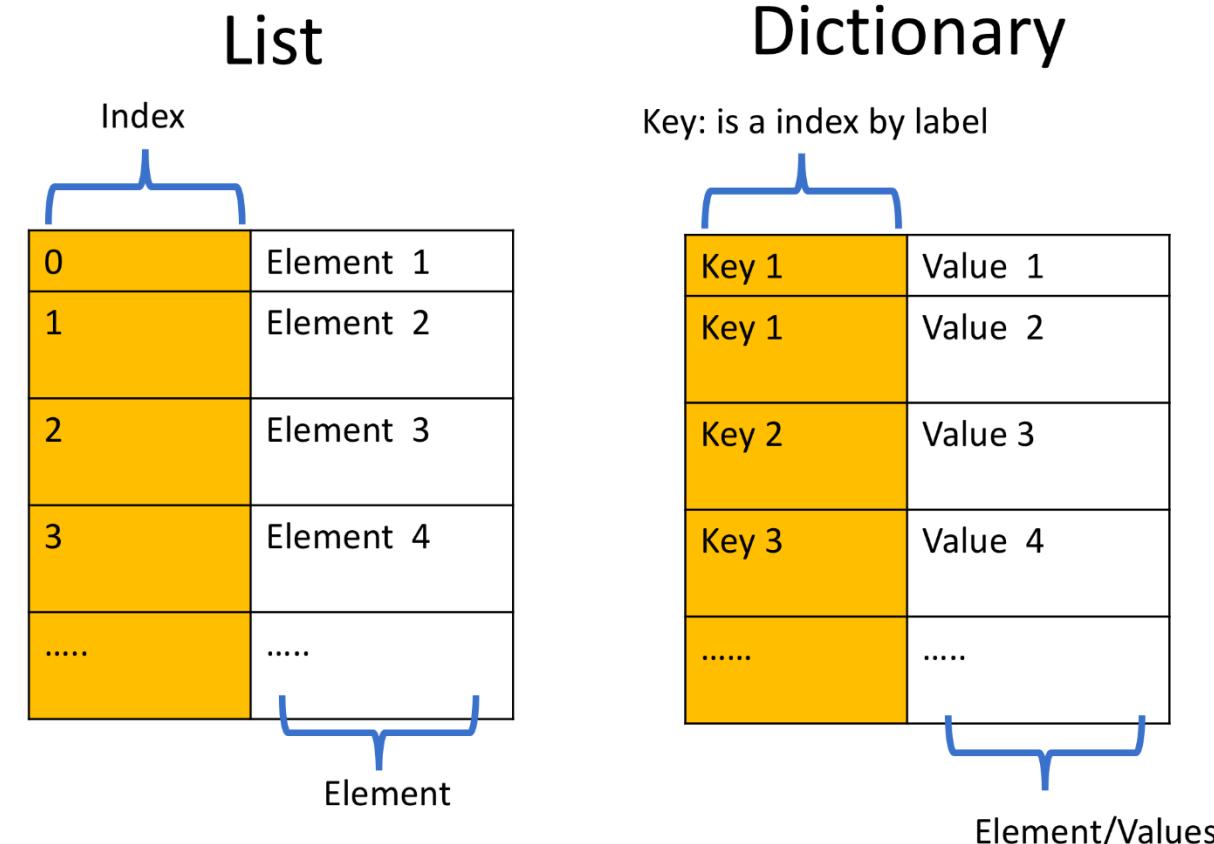
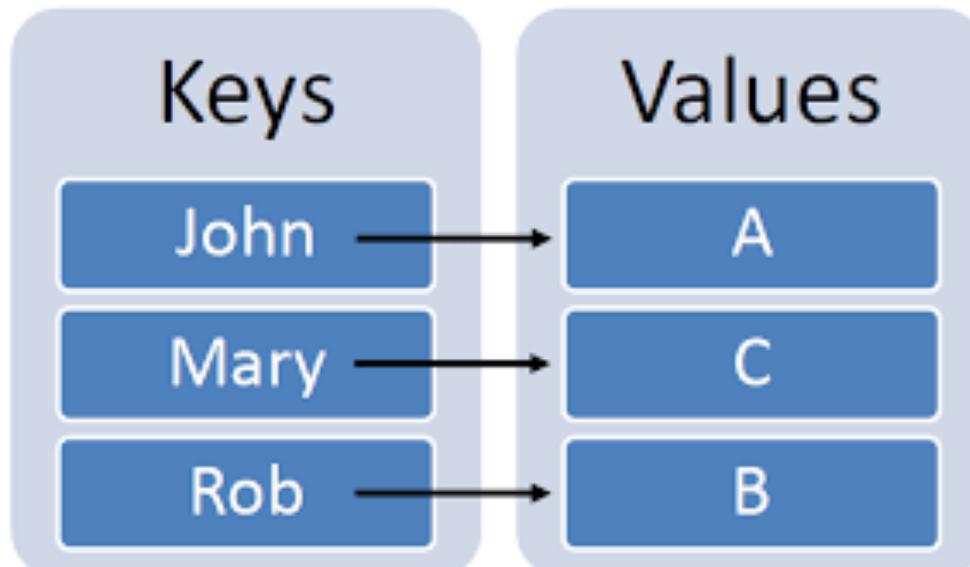
```
{4, 5}
{4, 5}
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
```

Thực hành

4.4 Dictionary

Dictionary

- Dictionary là tập hợp các cặp khóa - giá trị không có thứ tự.
- Nó thường được sử dụng khi chúng ta có một số lượng lớn dữ liệu.
- Các dictionary được tối ưu hóa để trích xuất dữ liệu với điều kiện bạn phải biết được khóa để lấy giá trị.



Dictionary

- Ví dụ:

```
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}

# using dict()
my_dict = dict( {'name': 'John', 1: [2, 4, 3]})

# from sequence having each item as a pair
my_dict = dict([(1, 'apple'), (2, 'ball')])
```

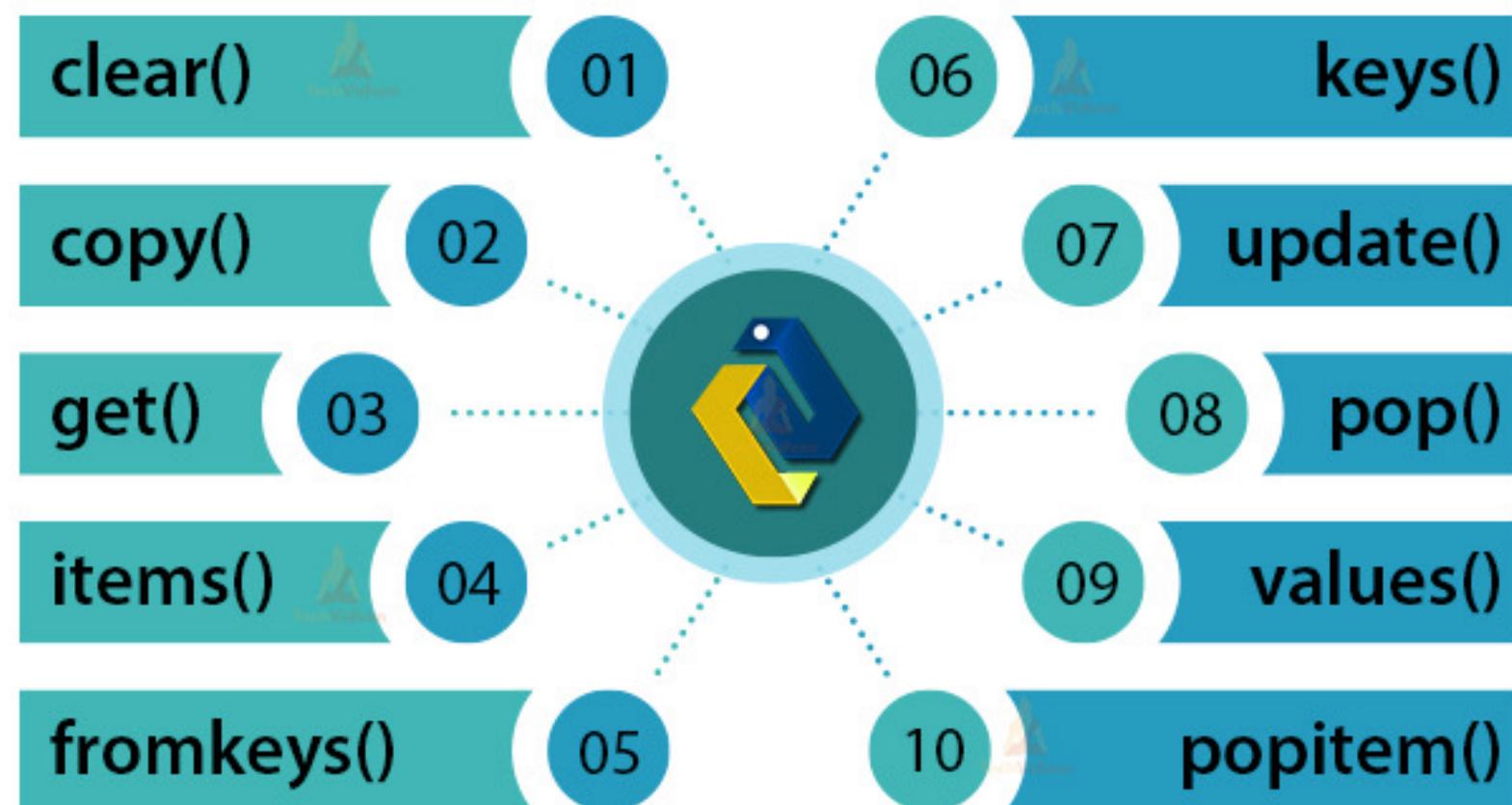
Truy cập phần tử của dictionary:

- Các kiểu dữ liệu lưu trữ khác sử dụng index để truy cập vào các giá trị thì dictionary sử dụng các key.
- Key có thể được sử dụng trong cặp dấu ngoặc vuông hoặc sử dụng get().

```
my_dict = { 'name': 'Jack', 'age': 26}  
          # Output: Jack  
          print(my_dict['name'])  
          # Output: 26  
          print(my_dict.get('age'))
```

- Dictionary có thể thay đổi, nên có thể thêm phần tử mới hoặc thay đổi giá trị của các phần tử hiện có bằng cách sử dụng toán tử gán.
- Nếu key đã có, giá trị sẽ được cập nhật, nếu là một cặp key: value mới thì sẽ được thêm thành phần tử mới.

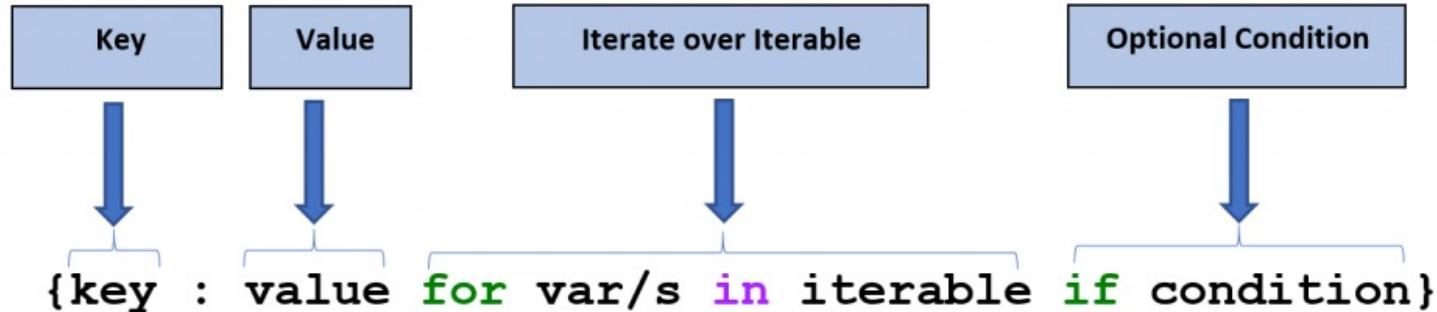
Python Dictionary Methods



Phương thức của dictionary

clear()	Xóa tất cả phần tử của dictionary.
copy()	Trả về một bản sao shallow copy của dictionary.
fromkeys(seq[,v])	Trả về dictionary mới với key từ seq và value bằng v (default là None).
get(key[,d])	Trả về giá trị của key, nếu key không tồn tại, trả về d. (default là None).
items()	Trả lại kiểu xem mới của các phần tử trong dictionary (key, value).
keys()	Trả về kiểu xem mới của các key trong dictionary.
pop(key[,d])	Xóa phần tử bằng key và trả về giá trị hoặc d nếu key không tìm thấy. Nếu d không được cấp, key không tồn tại thì sẽ tạo lỗi KeyError.
popitem()	Xóa và trả về phần tử bất kỳ ở dạng (key, value). Tạo lỗi KeyError nếu dictionary rỗng.
setdefault(key,[,d])	Nếu key tồn tại trả về value của nó, nếu không thêm key với value là d và trả về d (default là None).
update([other])	Cập nhật dictionary với cặp key/value từ other, ghi đè lên các key đã có.
values()	Trả về kiểu view mới của value trong dictionary.

Dictionary comprehension



```
1 #Tự động tạo một biến dictionary theo quy định:  
2 dict_lapphuong = {x:x**3 for x in range(6)}  
3 print('output:', dict_lapphuong)
```

output: {0: 0, 1: 1, 2: 8, 3: 27, 4: 64, 5: 125}

```
1 dict_lapphuong_chan = {x:x**3 for x in range(10) if x%2==0}  
2 print('output:',dict_lapphuong_chan)
```

output: {0: 0, 2: 8, 4: 64, 6: 216, 8: 512}

Thực hành

5. Ngoại lệ (Exception)

- Ngoại lệ là một sự kiện, xảy ra trong quá trình thực thi chương trình làm gián đoạn luồng hướng dẫn bình thường của chương trình.
- Khi một tập lệnh trong python tạo ra một ngoại lệ, nó phải xử lý ngoại lệ đó ngay lập tức nếu không nó sẽ kết thúc và thoát.
- Một số ngoại lệ phổ biến trong Python:
 - ZeroDivisionError:
 - NameError:
 - IndentationError:
 - IOError:
 - EOFError:
 - ...

```
-----  
ZeroDivisionError                                Traceback (most recent call last)  
<ipython-input-4-7fecd9efd1af> in <module>  
      1 a = 10  
      2 b = 0  
----> 3 c = a/b  
      4 print('a/b = ',c)  
  
ZeroDivisionError: division by zero
```

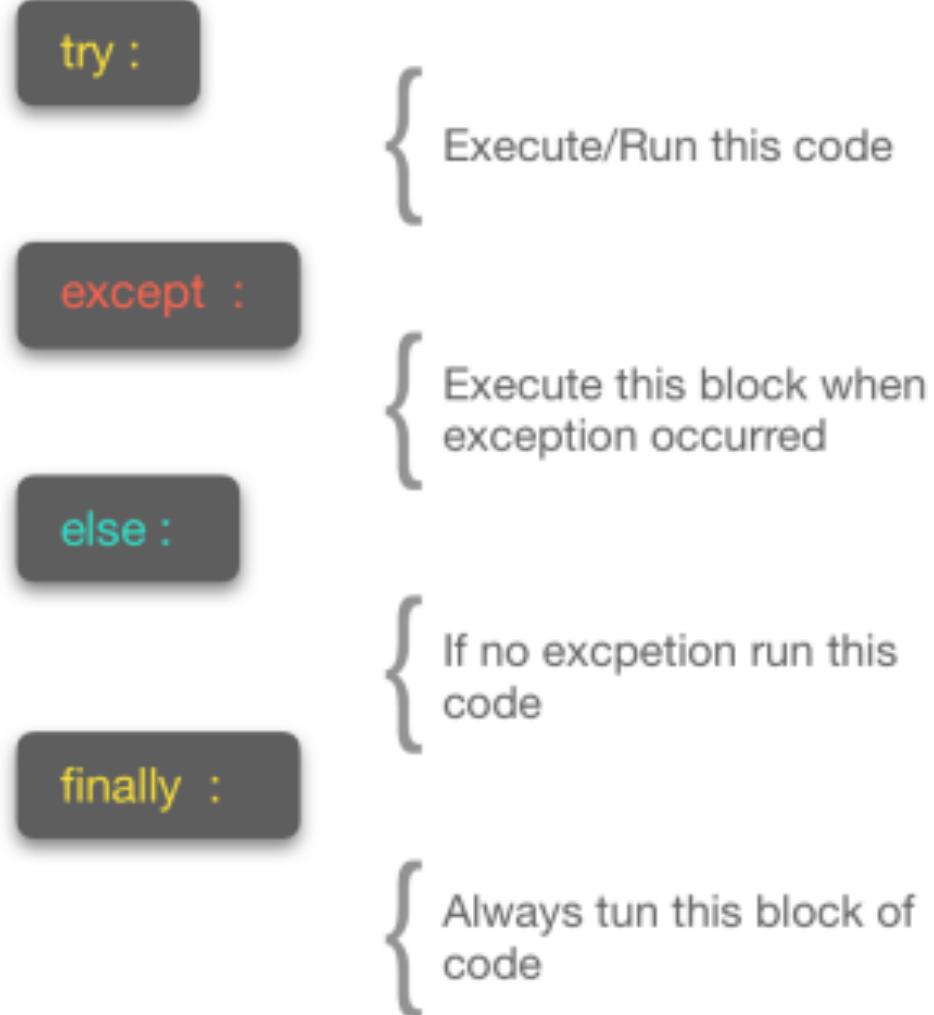
```
-----  
NameError                                         Traceback (most recent call last)  
<ipython-input-2-780a7d5891ae> in <module>  
----> 1 print (A)  
  
NameError: name 'A' is not defined
```

```
-----  
File "<ipython-input-3-6f06c5334b49>", line 3  
      print(--)  
      ^  
  
IndentationError: unexpected indent
```

- Xử lý ngoại lệ:

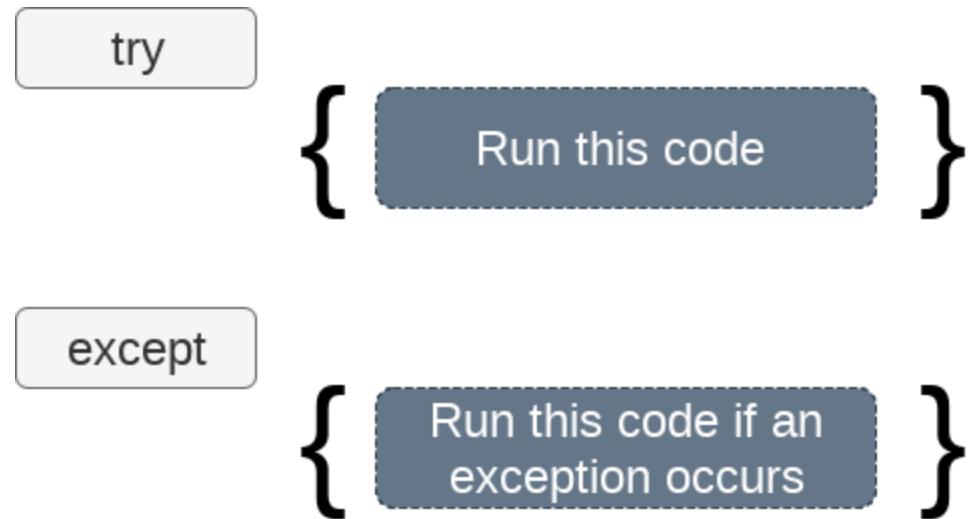
Python try-except

Error & Exceptions Handling



Xử lý ngoại lệ

- Xử lý ngoại lệ:



```
1 #try....except
2 try:
3     a = int(input("Enter a:"))
4     b = int(input("Enter b:"))
5     c = a/b
6 except:
7     print("Can't divide with zero")
```

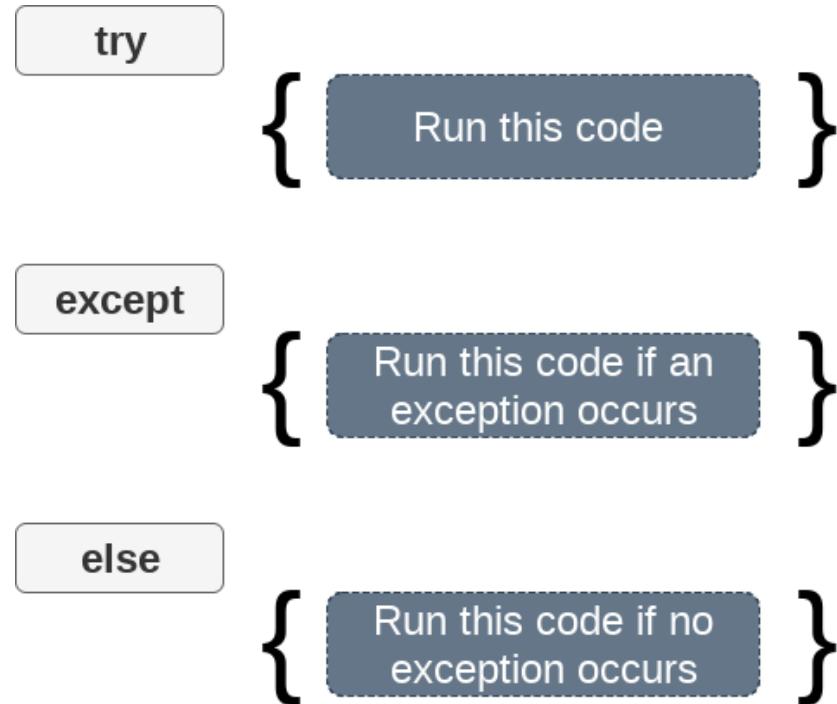
```
Enter a:10
Enter b:0
Can't divide with zero
```

```
1 #Hiển thị ngoại Lệ:
2 try:
3     fh = open("testfile", "r")
4     fh.write("This is my test file for exception handling!!")
5 except Exception as e:
6     print(e)
```

```
[Errno 2] No such file or directory: 'testfile'
```

Xử lý ngoại lệ

- Xử lý ngoại lệ:



```
1 #Try....except...else:  
2 try:  
3     fh = open("testfile.txt", "r")  
4     fh.write("This is my test file for exception handling!!")  
5 except Exception as e:  
6     print("Error: can't find file or read data")  
7     print(e)  
8 else:  
9     print("Written content in the file successfully")
```

Error: can't find file or read data
not writable

- Một số lưu ý:
 - Một câu lệnh try duy nhất có thể có nhiều câu lệnh except.
 - Mã trong khối else thực thi nếu mã trong khối try không phát sinh ngoại lệ.
 - Khối lệnh else là không lệnh không cần sự bảo vệ của khối try

Xử lý ngoại lệ

- Xử lý ngoại lệ:

try

```
{ Run this code }
```

- Khối finally là nơi để đặt khối lệnh cần phải được thực thi, cho dù khối try có đưa ra ngoại lệ hay không.

except

```
{ Run this code if an exception occurs }
```

else

```
{ Run this code if no exception occurs }
```

finally

```
{ Always run this code }
```

```
1 ##Try....except...else...finally
2 try:
3     fh = open("testfile.txt", "r")
4     fh.write("This is my test file for exception handling!!")
5 except IOError as e:
6     print("Error: can't find file or read data")
7     print(e)
8 else:
9     print("Written content in the file successfully")
10 finally:
11     fh.close()
12     print("End process....")
```

Error: can't find file or read data
not writable
End process....



Q & A
Thank you!

Bài 03: Lập trình Python cơ bản (tiếp)



AI Academy Vietnam

Nội dung bài 3

1. Hàm trong Python

- a. Giới thiệu Hàm | Xây dựng hàm trong Python
- b. Return | Tham số của hàm | Phạm vi của biến
- c. Hàm đệ quy | Hàm ẩn danh (Lambda)

2. Lớp và đối tượng trong Python

- a. Tạo lớp – đối tượng | Truy cập phương thức, thuộc tính của đối tượng
- b. Thừa kế

3. Module và Package

- a. Giới thiệu module và cách xây dựng module
- b. Import các package sử dụng

4. Đọc và ghi file với Python

5. Multithreads, Multiprocessing, Functools

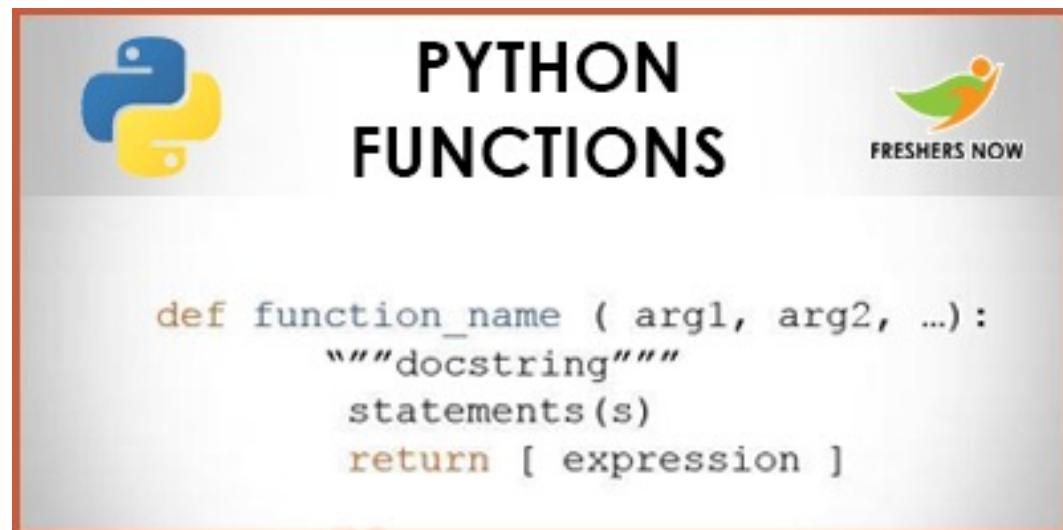


1. Hàm trong Python



Cơ bản về hàm trong Python

- Hàm trong Python là một nhóm các câu lệnh trong chương trình được tổ chức chung với nhau để thực hiện một chức năng hay một nhiệm vụ cụ thể nào đó.
- Sử dụng hàm giúp phân rã chương trình từ một chương trình lớn, phức tạp thành các phần cụ thể nhỏ hơn giúp dễ quản lý, tổ chức, nâng cao khả năng tái sử dụng và chia sẻ công việc.



1 #Ví dụ: Xây dựng 1 Hàm thực hiện chào hỏi
2 def hello_aiv(name):
3 print('Hi ',name,', How are you?')
4 print('Have a nice day!')

1 #Sử dụng hàm đã xây dựng
2 hello_aiv('AIV')

Hi AIV , How are you?
Have a nice day!

1. Cú pháp:

```
def tên_hàm(cá_tham_số):  
    "function_docstring"  
    Các câu lệnh xử lý bên trong hàm  
    return [kết quả trả về]
```

- Từ khóa **def** được sử dụng để bắt đầu phần định nghĩa hàm.
- sau đó là **tên_hàm**, tên hàm được đặt theo quy tắc như tên biến.
- Các tham số được truyền vào bên trong các dấu ngoặc đơn.
- Ở cuối là dấu hai chấm “:”.
- Sau đó là lệnh để được thực thi.
- Kết quả trả về cho hàm được thực hiện thông qua lệnh return

Lưu ý: Hàm không bắt buộc phải có tham số truyền vào hay kết quả trả về

Cơ bản về hàm trong Python

- Trước khi bắt tay vào xây dựng 1 hàm trong Python, cần phải tự **trả lời các câu hỏi**:
 - Hàm này sử dụng nhằm mục đích gì?
 - Hàm này nhận đầu vào là gì?
 - Hàm này trả kết quả ra là gì?

```
1 #Xây dựng hàm tính n!
2 #1)Hàm này dùng để làm gì? - Để tính n!
3 #2)Hàm này nhận dữ liệu vào là gì? - Một số nguyên dương N
4 #3)Hàm trả kết quả là gì? - Một số nguyên dương Là tích của 1*2*...*N
5 def giai_thua(n):
6     #Nhóm câu lệnh xử lý bên trong hàm
7     tich=1
8     for i in range(1,n+1):
9         tich=tich*i
10    #kết quả trả về cho hàm
11    return tich
```

```
1 n = int(input('Nhập vào một số nguyên N:'))
2 print(n,'!=',giai_thua(n))
```

Nhập vào một số nguyên N:10
10 != 3628800

Cơ bản về hàm trong Python

Gọi hàm:

- Hàm sau khi được xây dựng, có thể thực hiện lời gọi hàm ở nơi nào cần dùng đến.

```
1 n = int(input('Nhập vào một số nguyên N:'))
2 print(n, '!=' ,giai_thua(n))
```

```
Nhập vào một số nguyên N:10
10 != 3628800
```

```
1 #Gọi hàm giai_thua đã xây dựng
2 #Tính 12!
3 print('12! = ', giai_thua(12))
```

```
12! = 479001600
```

- Các lệnh mà chúng ta đã được học và sử dụng trước đây như: print(), input(), type(), int(), float(), str()... Đây thực chất là các **hàm được Python định nghĩa sẵn**.

Lệnh return:

- Lệnh **return <kết quả trả về>** được sử dụng để trả kết quả xử lý thông qua tên hàm.
- Lệnh return có thể có hoặc không.
- Trong trường hợp **không cung cấp <kết quả trả về>**, thì hàm **return** này sẽ trả về **None**. Nói cách khác, lệnh return được sử dụng để thoát khỏi định nghĩa hàm.

```
1 #Hàm không có Lệnh return
2 def hello_aiv(name):
3     print('Hello ',name,', How are you?')
4     print('Have a nice day.....!')
```

```
1 #Hàm trả về 1 giá trị
2 def giai_thua(n):
3     #nhóm các câu Lệnh xử Lý bên trong hàm
4     tich=1
5     for i in range(1,n+1):
6         tich=tich*i
7     #kết quả trả về cho hàm
8     return tich
```

Cơ bản về hàm trong Python

Lệnh `return()` có thể trả về 1 hay nhiều kết quả, nếu có nhiều hơn một kết quả thì ngăn cách nhau bởi **dấu phẩy**.

```
1 #Hàm trả về nhiều giá trị
2 #Hàm tính tổng, hiệu, tích, thương:
3 def all_ab(a,b):
4     add = a+b
5     sub = a-b
6     multi = a*b
7     div = a/b
8     #hàm trả về kết quả là 4 giá trị
9     return add, sub, multi, div
```

```
1 a=10
2 b=6
3 #Lấy kết quả trả về khi thực hiện hàm
4 tong,hieu,tich,thuong = all_ab(a,b)
5
6 #Lưu ý: Thứ tự trả về theo đúng thứ tự đã viết trong
7 #câu Lệnh return
8 print('Tổng ',a,'+',b,'=',tong)
9 print('Hiệu ',a,'-',b,'=',hieu)
10 print('Tích ',a,'*',b,'=',tich)
11 print('Thương ',a,'/',b,'=',thuong)
```

Tổng 10 + 6 = 16
Hiệu 10 - 6 = 4
Tích 10 * 6 = 60
Thương 10 / 6 = 1.6666666666666667

Cơ bản về hàm trong Python

Tham số truyền vào hàm:

- Tham số bắt buộc
- Tham số có mặc định (Default parameter)
- Tham số có độ dài biến (Variable-Length Parameter)

➤ Tham số bắt buộc

```
1 #Xây dựng hàm tính n!
2 #Hàm giai_thua có 1 tham số bắt buộc n
3 def giai_thua(n):
4     tich=1
5     for i in range(1,n+1):
6         tich=tich*i
7     #kết quả trả về cho hàm
8     return tich
```

```
1 #Gọi hàm giai_thua đã xây dựng
2 print('12! = ', giai_thua(12))
```

```
12! = 479001600
```

```
1 #Gọi hàm giai_thua đã xây dựng
2 #Khi không truyền vào tham số
3 print('12! = ', giai_thua())
```

```
-----  
TypeError  
<ipython-input-14-01226097b9b9> in <module>  
      1 #Gọi hàm giai_thua đã xây dựng  
      2 #Khi không truyền vào tham số  
----> 3 print('12! = ', giai_thua())  
  
Traceback (most recent call last)  
TypeError: giai_thua() missing 1 required positional argument: 'n'
```

➤ Tham số mặc định cho hàm:

- Để hạn chế trường hợp báo lỗi khi gọi hàm không cung cấp tham số thì trong Python cũng cung cấp cho chúng ta **thiết lập giá trị mặc định của tham số** khi khai báo hàm. Bằng cách sử dụng dấu = với cú pháp như sau:

```
def ten_ham(param = defaultValue):  
    # code
```

- Trong đó: **defaultValue** là giá trị mặc định của tham số đó mà bạn muốn gán.

Cơ bản về hàm trong Python

➤ Ví dụ:

- Hàm **sum_ab(a,b)** gọi khi truyền tham số và và khi không truyền tham số:

```
1 #Hàm tính tổng
2 def sum_ab(a=5, b =7):
3     total = a + b
4     return total
```

```
1 #Gọi hàm sum_ab() truyền vào 2 tham số
2 print(sum_ab(8,13))
```

21

```
1 #Gọi hàm sum_ab() không truyền vào tham số
2 #Sử dụng tham số mặc định
3 print(sum_ab())
```

12

➤ Tham số có độ dài biến (Variable-Length Parameter)

- Trên thực tế, không phải lúc nào chúng ta cũng biết được chính xác số lượng biến truyền vào trong hàm. Chính vì thế trong Python có cung cấp cho chúng ta khai báo một **param** đại diện cho các biến truyền vào hàm bằng cách thêm dấu * vào trước param đó.

- Ví dụ:

```
1 #Xây dựng hàm tính tổng các số đưa vào
2 def get_sum(*num):
3     tmp=0
4     #duyệt các tham số
5     for i in num:
6         tmp = tmp+i
7     return tmp
8
9 #Gọi hàm và truyền các tham số cho hàm
10 result = get_sum(1,2,3,4,5)
11 print('Kết quả:', result)
```

Kết quả: 15

Cơ bản về hàm trong Python

Phạm vi của biến.

- Một biến chỉ có tác dụng trong phạm vi mà nó khai báo (global – local).
- Khi một biến được khai báo ở trong hàm thì nó chỉ có thể được sử dụng ở trong hàm đó.

```
1 x = 300 #Biến toàn cục, có tác dụng trong toàn bộ chương trình
2 y = 800
3 def myfunc():
4     #Biến địa phương, chỉ có tác dụng trong thân hàm
5     x = 200
6     total = x + y
7     print('(Local) x :', x)
8     print('total :', total)
9 #Gọi hàm
10 myfunc()
11
12 print('-----')
13 print('(global) x:', x)
14
```

```
(Local) x : 200
total : 1000
-----
(global) x: 300
```

```
1 #Thiết lập một biến local thành global
2 def myfunc():
3     global k #Thiết lập biến k là biến global
4     k = 300
5     print('Inside func: k = ',k)
6
7 myfunc()
8
9 print('Outside func: k = ', k)
```

```
Inside func: k = 300
Outside func: k = 300
```

Hàm đệ quy

- Ngôn ngữ Python cho phép hàm gọi đến chính nó, người ta gọi là đệ quy hay quay lui

```
def recurse():
    ...
    recurse()      ← recursive call
    ...
recurse()
```

- Trong giải thuật phải có một điều kiện dừng để đệ quy kết thúc.
- Chương trình sử dụng đệ quy thì dễ hiểu nhưng hao tốn tài nguyên CPU, ảnh hưởng đến thời gian chạy chương trình nếu số lần đệ quy của hàm lớn.

```
1 #Hàm tính n! theo phương pháp đệ quy
2 def giai_thua(n):
3     if n==0:          #Điều kiện dừng để kết thúc đệ quy
4         return 1;      #vì 0! = 1 nên n==0 là vị trí kết thúc của đệ quy
5     else:
6         return giai_thua(n-1)*n  #1*2*....*n
```

```
1 n = int(input("Nhập vào số N:"))
2 print(n,'!=',giai_thua(n))
```

Nhập vào số N:10
10 != 3628800

Hàm ẩn danh - lambda

- Một hàm ẩn danh là một hàm được định nghĩa mà không có tên
- Các hàm bình thường được định nghĩa bằng từ khóa def; hàm ẩn danh được định nghĩa bằng từ khóa lambda

Cú pháp:

```
lambda arguments : Expression
```

- Các hàm lambda có thể có bất kỳ đối số nào nhưng chỉ có một biểu thức.

```
1 #Ví dụ hàm ẩn danh: 1 tham số
2 x = lambda a : a + 10
3 #Lambda a: a + 10 là hàm Lambda
4 #Trong đó: a là tham số truyền vào
5 #           a+10 là biểu thức
6
7 print(x(5))
8 print(x(7))
```

```
# Program to filter out only the even items from a list
my_list = [1, 5, 4, 6, 8, 11, 3, 12]
new_list = list(map(lambda x: x*2, my_list))
print(new_list)
```

```
[2, 10, 8, 12, 16, 22, 6, 24]
```



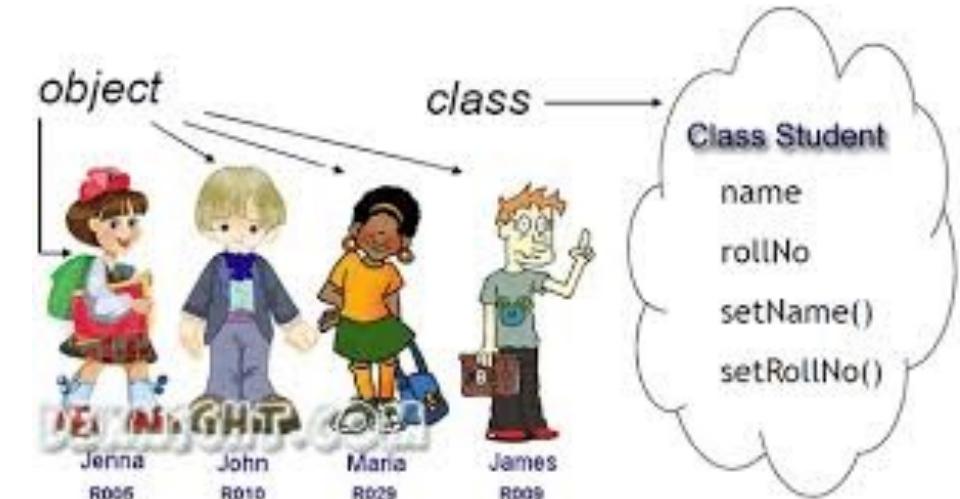
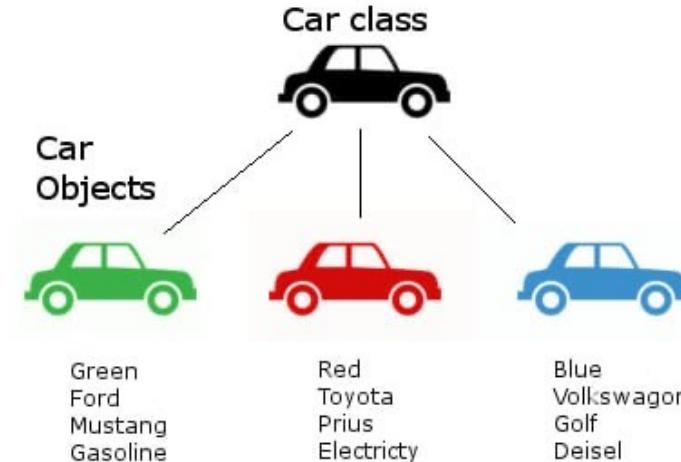
Thực hành



2. Lớp, đối tượng trong Python

1. Giới thiệu lớp

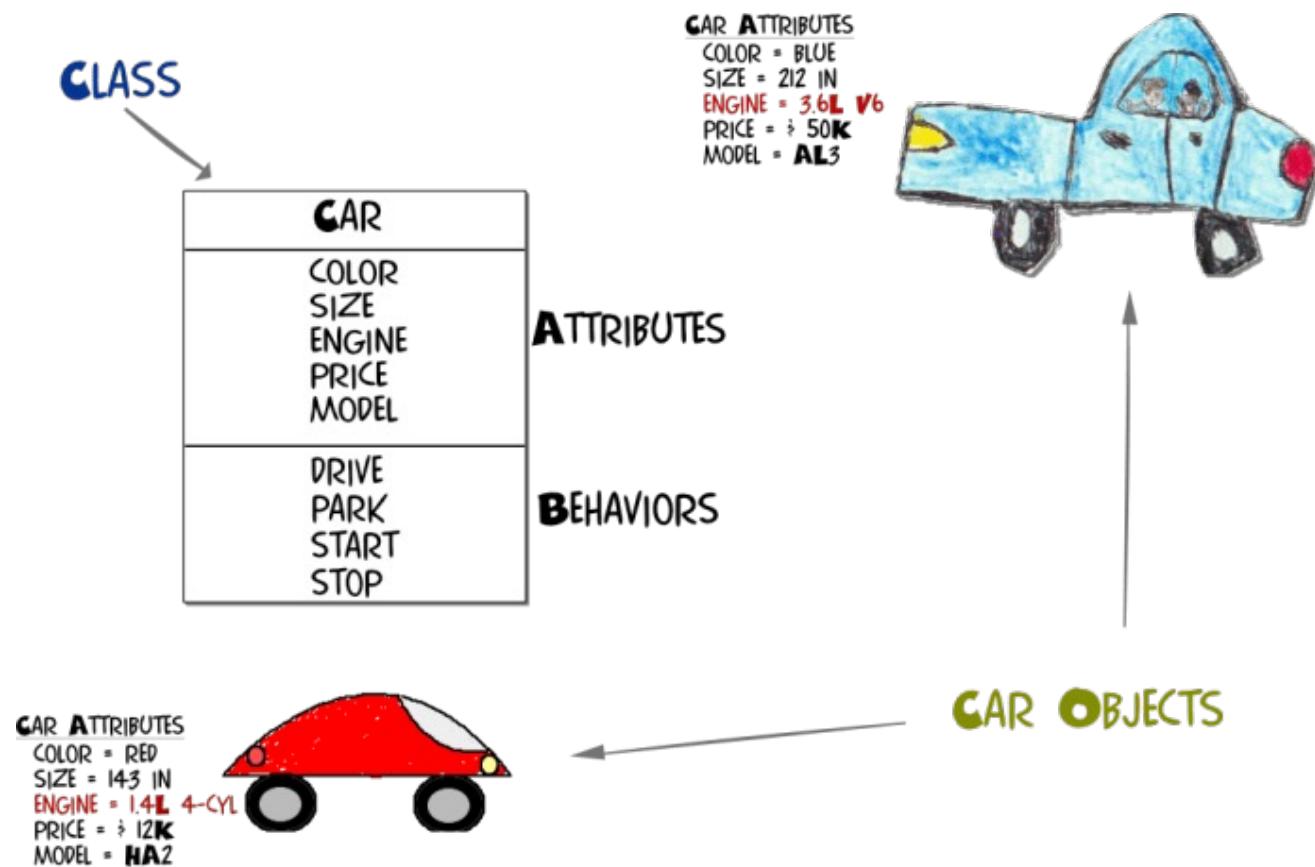
- Python là một ngôn ngữ lập trình hướng đối tượng



- Lớp (class) là một cấu trúc logic dùng để định nghĩa khuôn dạng và tính chất của các Đối tượng (Object).
- Lớp được định nghĩa như là một kiểu dữ liệu mới.

Lớp, đối tượng trong Python

- Mỗi một đối tượng có 2 thành phần chính:



- Các thuộc tính: dùng để chứa các thông tin mô tả các đặc điểm của đối tượng. Sử dụng các biến để lưu giữ những thông tin này. Khi đó các biến được gọi là các thuộc tính.
- Các phương thức: dùng để mô tả các hành vi của đối tượng. Sử dụng các hàm để mô tả các hành vi này. Khi đó các hàm được gọi là các phương thức.

➤ Khai báo Class trong Python

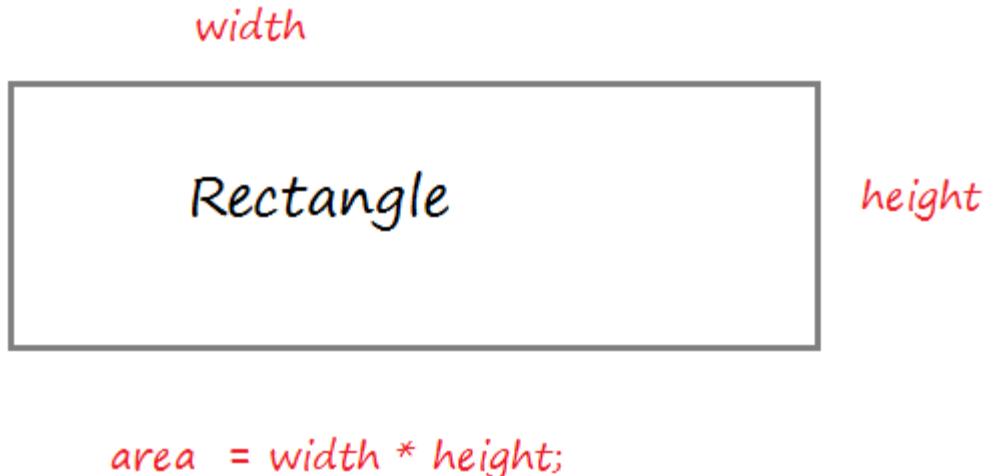
```
class ClassName:  
    'Gồm các thuộc tính, phương thức'  
    # Code ...
```

- Trong đó, **className** là tên của class cần khai báo. Một số lưu ý khi đặt tên cho lớp:
 - Tên lớp nên là một danh từ,
 - Tên lớp được đặt ký tự đầu tiên của mỗi từ là in hoa.
 - Tên lớp nên đơn giản, mang tính mô tả và đầy đủ ý nghĩa
 - Tên lớp không được là một từ khóa nào đó của Python
 - Tên lớp không bắt đầu bằng số, có thể bắt đầu với dấu \$ hoặc ký tự gạch dưới.

Lớp, đối tượng trong Python

➤ Khai báo Class: Rectangle

- có 2 thuộc tính: width, height
- 2 phương thức: getArea(), getPrimeter()



```
1 #Tạo một lớp Rectangle
2
3 class Rectangle:
4     #Lớp Rectangle có 2 thuộc tính: width, height
5
6     #Phương thức khởi tạo đối tượng (Constructor)
7     def __init__(self, width, height):
8         self.width = width
9         self.height = height
10
11    #Phương thức tính diện tích
12    def getArea(self):
13        area = round(self.width * self.height, 1)
14        return area
15
16    #Phương thức tính chu vi
17    def getPerimeter(self):
18        perimeter = round((self.width + self.height)*2, 1)
19        return perimeter
```

➤ Một số khái niệm hướng đối tượng

- **Thuộc tính (Attribute):** Thuộc tính là một thành viên của lớp, Hình chữ nhật có 2 thuộc tính width và height
- **Phương thức (Method):**
 - Phương thức của class tương tự như một hàm thông thường, nhưng nó là một hàm của class. Để sử dụng được cần phải gọi thông qua đối tượng.
 - Tham số đầu tiên của phương thức luôn là self (Một từ khóa ám chỉ chính class đó)
- **Phương thức khởi tạo (Constructor):**
 - Là một phương thức đặc biệt của lớp, luôn có tên là `__init__`. Tham số đầu tiên luôn là self. Chỉ có thể định nghĩa một constructor trong class
 - Constructor được sử dụng để tạo ra một đối tượng.
 - Constructor gán các giá trị từ tham số vào các thuộc tính của đối tượng sẽ được tạo ra

Lớp, đối tượng trong Python

➤ Khởi tạo đối tượng từ lớp

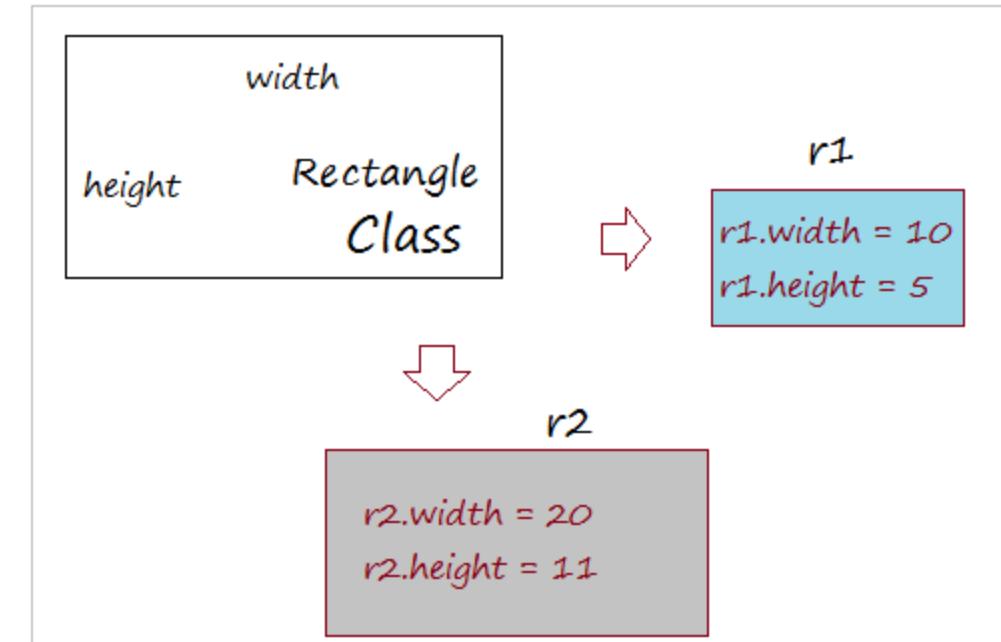
Sau khi đã khai báo được class trong Python rồi, thì để khởi tạo các đối tượng sử dụng cú pháp sau:

```
variableName = className()
```

Trong đó:

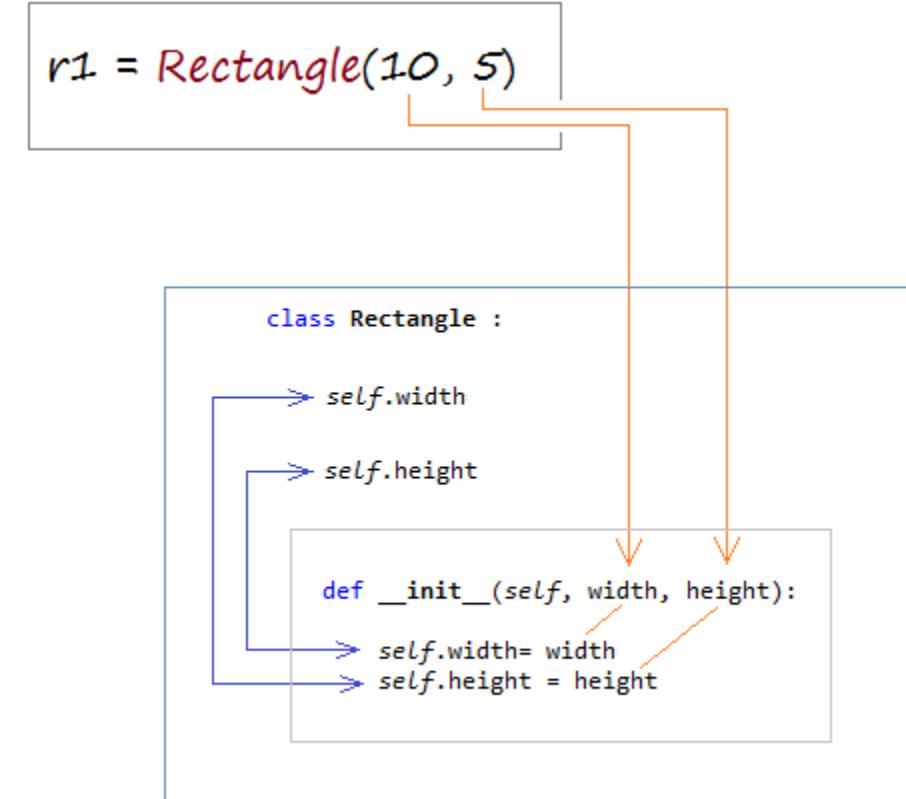
- **variableName** là tên đối tượng.
- **className** là class muốn khởi tạo.

```
1 #Ví dụ: Tạo 2 đối tượng r1, r2 từ class Rectangle  
2 r1 = Rectangle(10,5)  
3  
4 r2 = Rectangle(20,11)
```



➤ Điều gì xảy ra khi khởi tạo một đối tượng

Khi tạo một đối tượng của lớp **Rectangle**, phương thức khởi tạo (constructor) của class đó sẽ được gọi để tạo một đối tượng, và các thuộc tính của đối tượng sẽ được gán giá trị từ tham số



- Sau khi đã khởi tạo được đối tượng sẽ có thể truy cập được các thuộc tính và phương thức trong class đó.
- Bằng cách sử dụng dấu . theo cú pháp sau:

```
# truy cap den thuoc tinh  
object.propertyName
```

```
#truy cap den phuong thuc  
object.methodName()
```

Trong đó:

- object là biến thể hiện lại object.
- propertyName là tên thuộc tính muốn truy xuất.
- methodName là tên phương thức muốn truy xuất.

Lớp, đối tượng trong Python

➤ **Ví dụ:** sẽ truy xuất đến các thuộc tính và phương thức trong class Rectangle

```
1 #Lấy thuộc tính width, height của đối tượng rec1
2 x = r1.width
3 y = r1.height
4 print('----Thuộc tính-----')
5 print('1. Thuộc tính Chiều rộng: ', x)
6 print('2. Thuộc tính Chiều dài: ', y)
7 #Gọi phương thức getArea, getPerimeter của đối tượng rec1
8 dt = r1.getArea()
9 cv = r1.getPerimeter()
10 print('-----Phương thức-----')
11 print('1. Phương thức tính Diện tích: ', dt)
12 print('2. Phương thức tính Chu vi: ', cv)
```

-----Thuộc tính-----

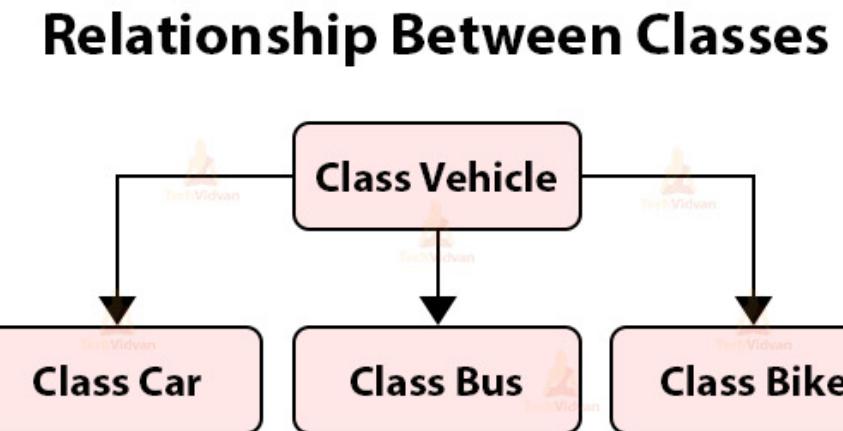
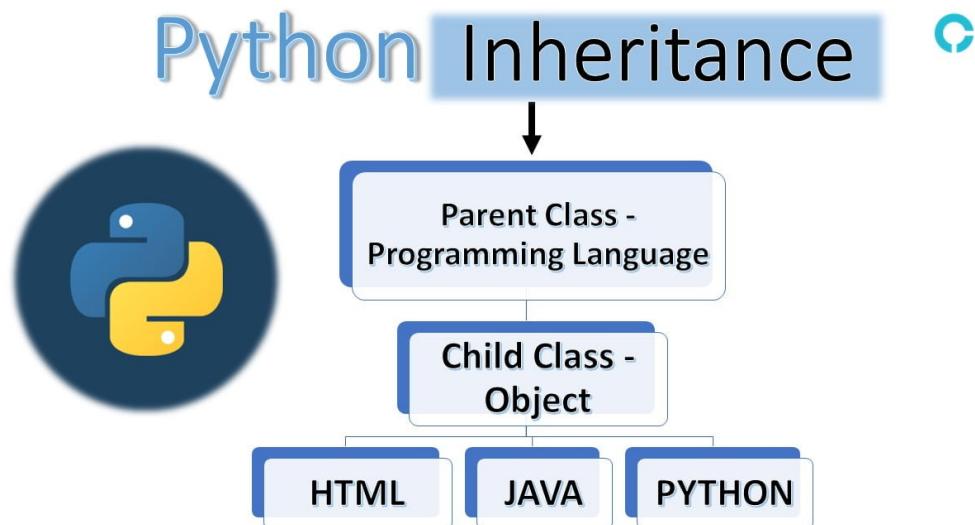
1. Thuộc tính Chiều rộng: 10
2. Thuộc tính Chiều dài: 5

-----Phương thức-----

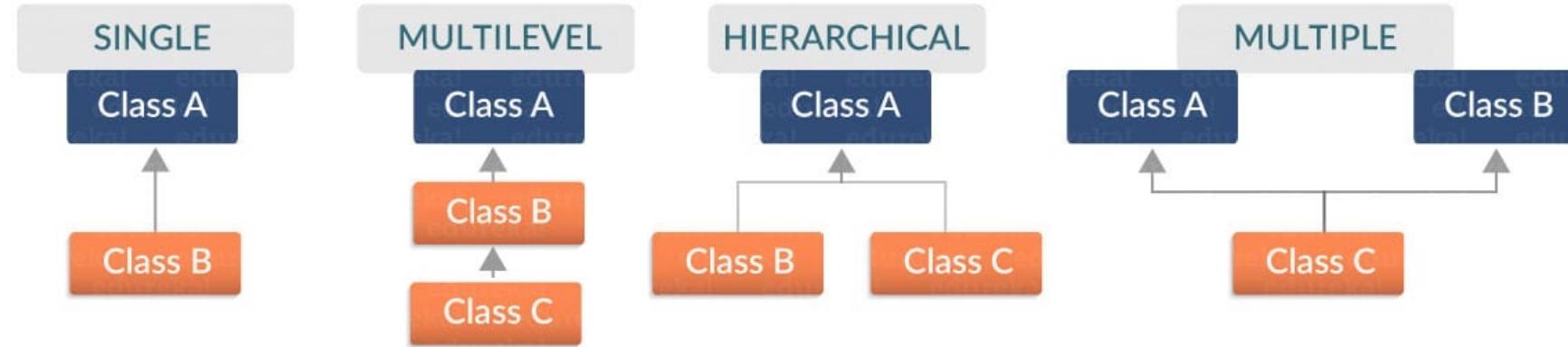
1. Phương thức tính Diện tích: 50
2. Phương thức tính Chu vi: 30

Thừa kế

- Việc thực hiện thao tác thừa kế là đơn giản và hiệu quả, trong đó cho phép tạo ra một lớp mới từ một lớp có sẵn.
- Lớp mới dẫn xuất từ lớp có sẵn có thể tái sử dụng các biến và phương thức của nó mà không cần phải viết lại hoặc sửa lại đoạn mã.
- Khi thực hiện thừa kế, lớp được dẫn xuất từ lớp khác được gọi là phân lớp, lớp dẫn xuất, **Lớp con**; Lớp mà từ đó phân lớp được dẫn xuất thì được gọi là siêu lớp, lớp cơ sở, **Lớp cha**



Types Of Inheritance



- **Đơn thừa kế (Single):** Xảy ra khi một lớp con thừa kế chỉ một lớp cha
- **Thừa kế nhiều mức (Multilevel):** Khi một lớp con dẫn xuất từ một lớp cha và bản thân lớp cha lại là con của một lớp khác.
- **Thừa kế phân cấp (Hierarchical):** Loại thừa kế này xảy ra khi một lớp cha có nhiều hơn một lớp con ở những mức khác nhau.
- **Đa thừa kế (Multiple):** Xảy ra khi một lớp con dẫn xuất từ nhiều hơn một lớp cha



Thùa kế

- **Ví dụ:** Tạo lớp Square thừa kế từ lớp Rectangle (Đơn thừa kế)

```
#Khai báo Lớp Square thừa kế từ Lớp Rectangle
class Square(Rectangle):
    #dùng super() để gọi đến hàm tạo của Lớp cha
    def __init__(self, width = 15, color='red'):
        super().__init__(width, width)
        self.color=color

#Thêm phương thức draw() để vẽ hình vuông theo width và color
def draw(self):
    x = self.width
    c = self.color
    fig, ax = plt.subplots(figsize=(4,4))
    square = plt.Rectangle((0, 0), x, x, color=c)
    ax.add_patch(square)
    automin, automax = plt.xlim()
    plt.xlim(0-10, x+10)
    automin, automax = plt.ylim()
    plt.ylim(0-10, x+10)
    plt.grid(True)
    plt.show()
```

Thùa kế

```
1 #Khai báo đối tượng a thuộc Lớp Square  
2 a = Square(30, 'yellow')
```

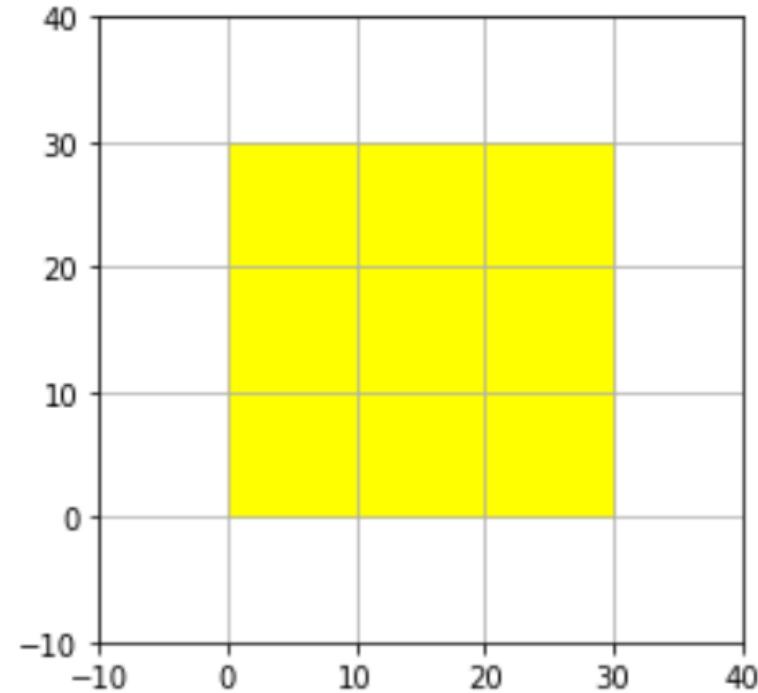
```
1 #Tái sử dụng các thuộc tính của Lớp Rectangle  
2 print(a.width)  
3 print(a.height)  
4  
5 #Bổ sung thêm thuộc tính mới cho Lớp Square  
6 print(a.color)
```

```
30  
30  
yellow
```

```
1 #Tái sử dụng các phương thức của Lớp Rectangle  
2 print('Diện tích hình vuông:', a.getArea())  
3 print('Chu vi hình vuông:', a.getPerimeter())
```

```
Diện tích hình vuông: 900  
Chu vi hình vuông: 120
```

```
1 #Sử dụng phương thức mới draw của Lớp Square  
2 a.draw()
```



Thực hành

3. Module và Package



Module và Package

- Module hiểu ngắn gọn là một file python mà trong đó chứa các khai báo và định nghĩa về hàm số và biến. Các chương trình python sẽ được thiết kế sao cho nội dung được chia nhỏ về các files module để dễ dàng quản lý. Chúng ta có thể dễ dàng import lại các khai báo và định nghĩa từ module này sang module khác.
- Một package sẽ bao gồm nhiều file module



```
1 #gọi package để sử dụng
2 import math
3
4 #Liệt kê tất cả các phương thức, thuộc tính của package math
5 print (dir(math))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'exp_m1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclos_e', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

Module và Package

- Khi chương trình lớn hơn, bạn nên chia nhỏ nó thành các modules khác nhau.
- Module là một file chứa các định nghĩa và câu lệnh Python.
- Module trong Python có phần mở rộng .py.
- Các định nghĩa bên trong một module có thể được sử dụng trong module khác hoặc trình thông dịch Python. Để làm điều này chúng ta sử dụng từ khóa import

```
import math
```

```
import math  
print(math.pi)
```

```
>>> from math import pi  
>>> pi 3.141592653589793
```

Module và Package

- Ngoài sử dụng các module có sẵn trong python ra thì chúng ta cũng có thể viết ra các module của riêng mình và import nó khi muốn sử dụng.
- Ví dụ
 - Tạo file mathplus.py có nội dung sau:

```
def get_sum (a, b):  
    return a + b
```

- Tạo file main.py sử dụng module vừa viết:

```
import mathplus  
  
print(mathplus.get_sum(4,7))  
# Kết quả: 11
```

Module và Package

- Định danh cho modules:
 - Trong Python, có thể gán định danh mới cho modules khi import chúng bằng cách sử dụng keyword as.

```
import modules as newname
# hoặc đổi với from import
from modules import something as newname
```



4. Làm việc với tập tin trong Python

1. File là gì?

- File hay còn gọi là tệp, tập tin. File là tập hợp của các thông tin được đặt tên và lưu trữ trên bộ nhớ máy tính như đĩa cứng, đĩa mềm, CD, DVD,...
- Khi muốn **đọc** hoặc **ghi file**, chúng ta cần phải **mở file** trước. **Khi hoàn thành**, file cần phải **được đóng** lại để các tài nguyên gắn với file được giải phóng.
- Do đó, trong Python, một thao tác với file diễn ra theo thứ tự sau:
 - ✓ Mở tệp tin
 - ✓ Đọc hoặc ghi
 - ✓ Đóng tệp

2. Mở File

Để mở file trong Python chúng ta sử dụng hàm **open** với cú pháp như sau:

```
open(filePath, mode, buffer)
```

Trong đó:

- **filePath** là đường dẫn đến địa chỉ của file.
- **mode** là thông số thiết lập chế độ chúng ta mở file được cấp những quyền gì?
Mặc định mode sẽ bằng **r** (xem các mode ở dưới).
- **buffer** là thông số đệm cho file mặc định thì nó sẽ là 0.

Làm việc với tập tin (File)

Các chế độ mode

Mode	Chú thích
r	Chế độ chỉ được phép đọc.
w	Chế độ ghi file, nếu như file không tồn tại thì nó sẽ tạo mới file và ghi nội dung, còn nếu như file đã tồn tại nó sẽ ghi đè nội dung lên file cũ.
a	Mở file trong chế độ ghi tiếp. Nếu file đã tồn tại rồi thì nó sẽ ghi tiếp nội dung, và nếu như file chưa tồn tại thì nó sẽ tạo một file mới và ghi nội dung vào đó.
r+	Mở file cho cả đọc và ghi
w+	Mở file trong chế độ đọc và ghi đè lên file hiện có nếu file tồn tại, hoặc tạo ra file mới nếu chưa tồn tại
a+	Mở file trong chế độ đọc và ghi tiếp nội dung, còn lại cơ chế giống chế độ a.
x	Tạo file mới để ghi, báo lỗi nếu file đã tồn tại

2. Mở File

➤ Ví dụ:

```
f=open("test.txt") #mở file mode 'r' hoặc 'rt' để đọc
```

```
f=open("test.txt",'w') #mở file mode 'w' để ghi
```

```
import csv  
f=open('data.csv','rt') #mở file mode 'r' hoặc 'rt' để đọc file csv
```

3. Đóng File

- Việc đóng file được xây dựng trong Python bằng hàm `close()` với cú pháp như sau:

```
fileObject.close()
```

- Trong đó, `fileObject` là đối tượng mà chúng ta thu được khi sử dụng hàm `open()`.

4. Đọc file

- Sau khi đã mở được file ra rồi, để đọc được file thì chúng ta sử dụng phương thức **read** với cú pháp:

```
fileObject.read(length);
```

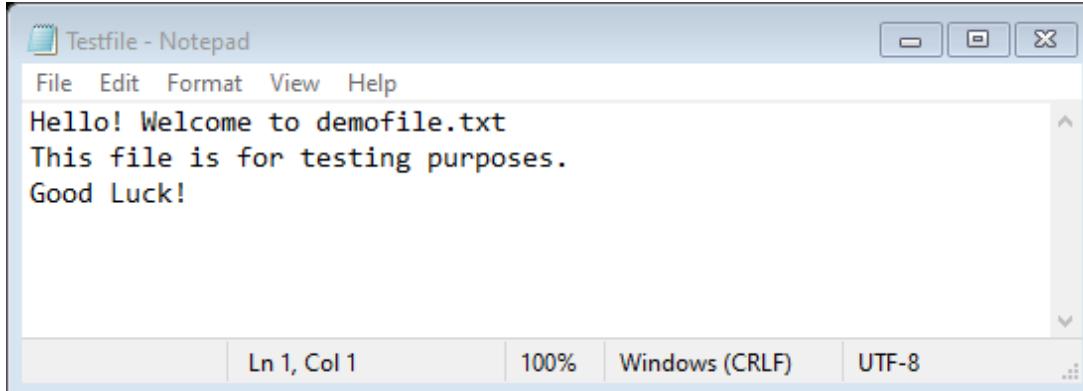
Trong đó:

- fileObject** là đối tượng mà chúng ta thu được khi sử dụng hàm **open()**.
- length** là dung lượng của dữ liệu mà chúng ta muốn đọc, nếu để trống tham số này thì nó sẽ đọc hết file hoặc nếu file lớn quá thì nó sẽ đọc đến khi giới hạn của bộ nhớ cho phép.

Làm việc với tập tin (File)

4. Đọc file

➤ Ví dụ:



```
1 #Mở file để đọc dữ liệu
2 f = open('Testfile.txt',"r")
3
4 #Đọc 10 ký tự đầu tiên của file
5 st1 = f.read(10)
6
7 print(st1, ' -- Số ký tự là: ', len(st1))
```

Hello! Wel -- Số ký tự là: 10

```
1 #Mở file để đọc dữ liệu
2 f = open('Testfile.txt')
3
4 #Đọc nội dung của file vào biến st
5 st = f.read()
6
7 print('Nội dung file:')
8 print(st)
```

Nội dung file:
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!

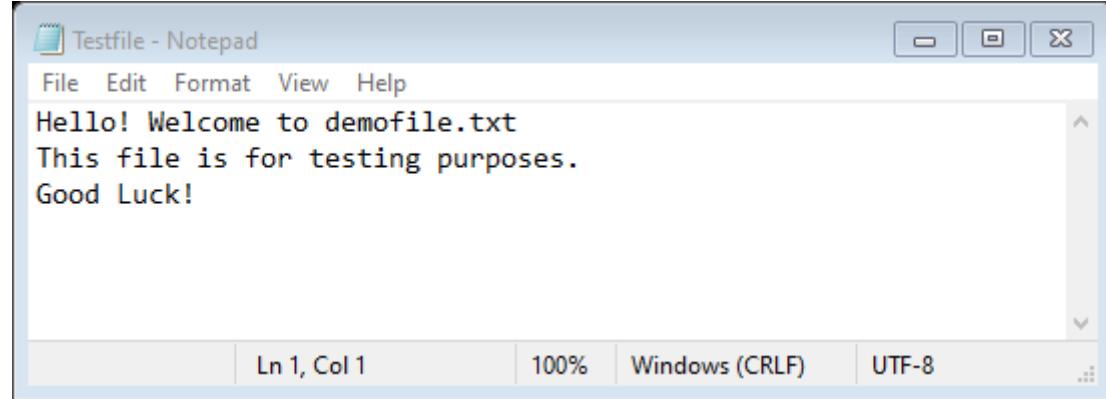
```
1 #Mở file đọc dữ liệu với with:
2 with open('Testfile.txt') as f:
3     lines = f.readlines()
4
5 print(lines)
6 f.close()
```

['Hello! Welcome to demofile.txt\n', 'This

Làm việc với tập tin (File)

4. Đọc file

➤ Ví dụ:



```
1 #Mở file để đọc dữ liệu
2 f = open('Testfile.txt')
3
4 #Đọc từng dòng dữ liệu của file
5 print(f.readline())
6 print(f.readline())
7
8 f.close() #Đóng file dữ liệu
```

Hello! Welcome to demofile.txt

This file is for testing purposes.

```
1 #Mở file để đọc dữ liệu
2 f = open("Testfile.txt", "r")
3
4 #đọc tất cả các dòng của file
5 for x in f:
6     print(x)
7
8 f.close() #Đóng file dữ liệu
```

Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!

5. Ghi file

- Để ghi được file thì bạn phải chắc chắn là đang mở file ở các chế độ cho phép ghi. Và sử dụng phương thức **write** với cú pháp sau:

```
fileObject.write(data)
```

Trong đó:

- fileObject** là đối tượng mà chúng ta thu được khi sử dụng hàm **open()**.
- data** là dữ liệu mà chúng ta muốn ghi vào trong file.

➤ Ví dụ:

```
1 #Mở file với chế độ ghi đè (w):  
2 #nếu như file không tồn tại thì nó sẽ tạo mới file và ghi nội dung,  
3 #còn nếu như file đã tồn tại nó sẽ ghi đè nội dung lên file cũ.  
4 f1 = open('Ghifile.txt', 'w')  
5  
6 #Dữ Liệu muốn ghi vào file  
7 st = 'Welcome to Python for Analysis!'  
8  
9 f1.write(st) #Ghi dữ Liệu vào file  
10 f1.close() #Đóng file
```

5. Ghi file

➤ Ví dụ:

```
1 #Mở file với chế độ ghi tiếp (a):
2 # Nếu file đã tồn tại rồi thì nó sẽ ghi tiếp nội dung,
3 # và nếu như file chưa tồn tại thì nó sẽ tạo một file mới và ghi nội dung vào đó.
4 f1 = open('Ghifile.txt', 'a+')
5
6 #Đữ liệu muốn ghi vào file
7 st = 'This is new line.....'
8
9 f1.write(st) #Ghi tiếp dữ liệu vào file
10 f1.close() #Đóng file
11
12 #open and read the file after the appending:
13 f = open("Ghifile.txt", "r")
14 print(f.read())
```

Welcome to Python for Analysis!This is new line.....

Làm việc với tập tin (File)

➤ Các thuộc tính trong file.

Thuộc tính	Chú thích
file.name	Trả về tên của file đang được mở.
file.mode	Trả về chế độ mode của file đang được mở.
file.closed	Trả về true nếu file đã được đóng, và false nếu file chưa đóng.

➤ Ví dụ:

In ra thông số của file Ghifile.txt ở trên

```
1 #Lấy các thông số của file
2 f2 = open('Ghifile.txt')
3
4 print('1.Tên file:',f2.name)
5 print('2.Chế độ mở file:',f2.mode)
6 print('3.Trạng thái đóng file:',f2.closed)
```

1.Tên file: Ghifile.txt
2.Chế độ mở file: r
3.Trạng thái đóng file: False

Ví dụ với đọc/ghi tệp tin

Bài 1: Ghi dữ liệu vào File “data.txt”

```
# Mở file để ghi
fo = open("data.txt", "w")
# Ghi dữ liệu lên file
fo.write("Tobe or not tobe. \n Nghi lon de thanh cong ! \n");
# Close opened file
fo.close()
print("Ghi file thanh cong !")
```

Bài 2: Đọc và ghi dữ liệu từ một File

```
obj=open("test.txt","w")
obj.write("Chao mung cac ban den voi khoa CNTT")
obj.close()
obj1=open("test.txt","r")
s=obj1.read()
print (s)
obj1.close()
obj2=open("test.txt","r")
s1=obj2.read(20)
print (s1)
obj2.close()
```

Thực hành

5. Multithreading, Multiprocessing, Functools

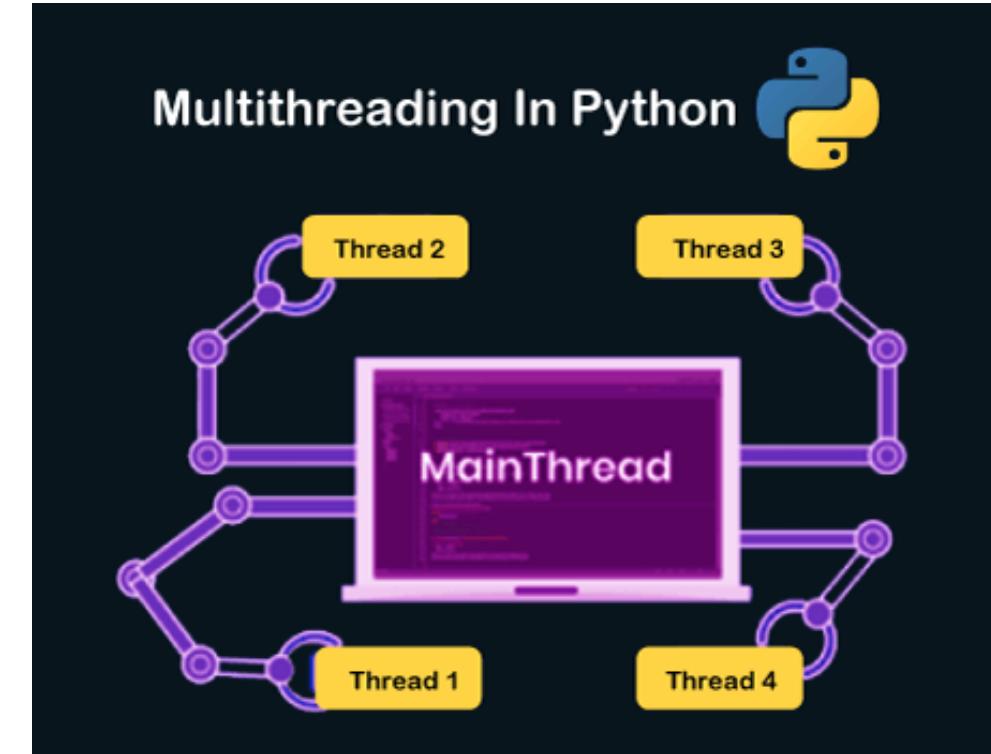


Multithreading

Đa luồng (Multithreading)?

- Một chương trình đa luồng chứa hai hoặc nhiều phần mà có thể chạy đồng thời và mỗi phần có thể xử lý tác vụ khác nhau tại cùng một thời điểm được gọi là Đa luồng. → Giúp sử dụng tốt nhất các tài nguyên sẵn có của máy tính.

- Python cung cấp thread Module và threading Module để bạn có thể bắt đầu một thread mới cũng như một số tác vụ khác trong khi lập trình đa luồng.
- Mỗi một Thread đều có vòng đời chung là bắt đầu, chạy và kết thúc.
- Một Thread có thể bị ngắt (interrupt), hoặc tạm thời bị dừng (sleeping) trong khi các Thread khác đang chạy



<https://docs.python.org/3/library/threading.html#>

Multithreading

So sánh đơn luồng và đa luồng

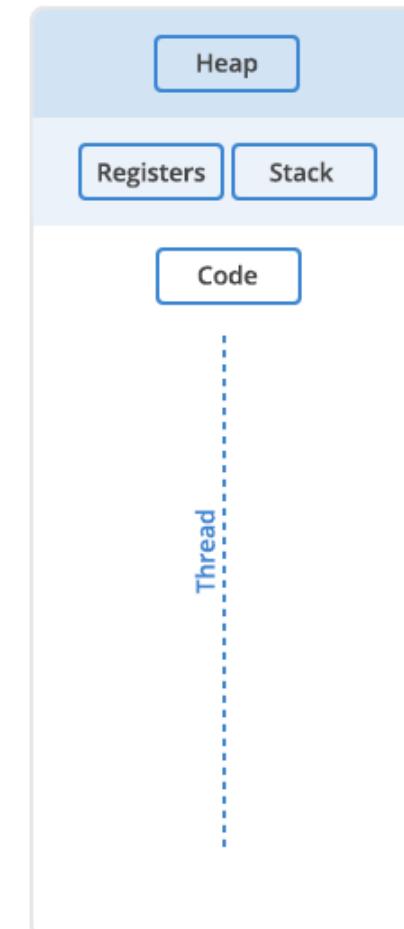
- Xây dựng hàm tính X bình phương và lập phương của các số:

```

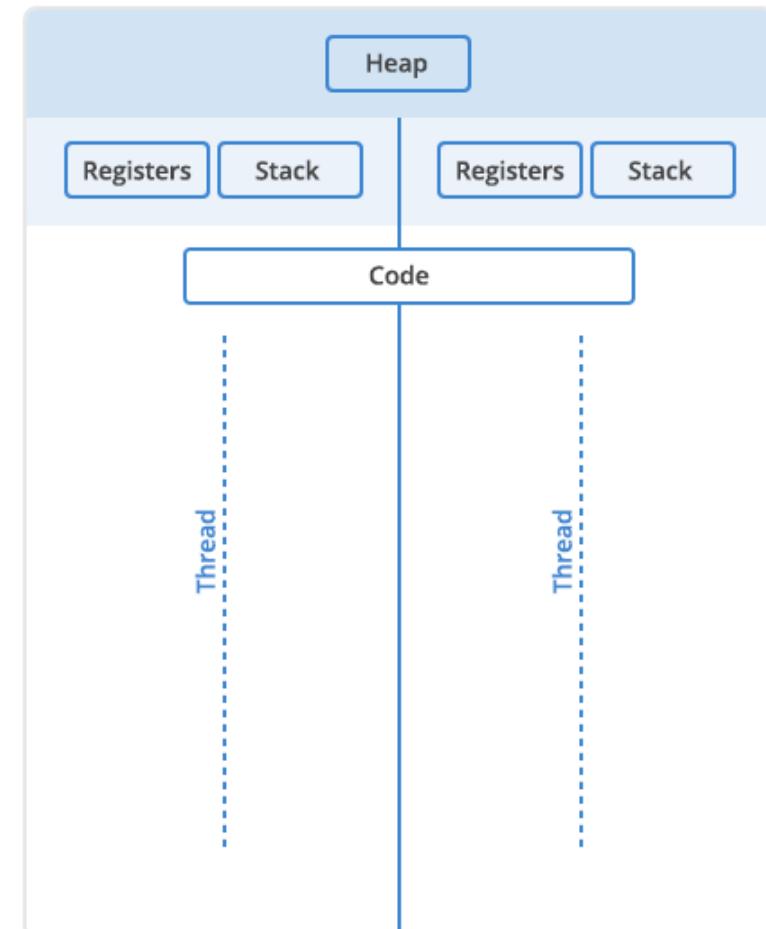
1 #Hàm tính bình phương của x:
2 def cal_square(numbers):
3     print("calculate square number")
4     for n in numbers:
5         time.sleep(0.2)
6         print ('square:', n*n)
7
8 #Hàm tính lập phương của x:
9 def cal_cube(numbers):
10    print("calculate cube number")
11    for n in numbers:
12        time.sleep(0.2)
13        print ('cube:', n*n*n)

```

Single Thread



Multi Threaded



Multithreading

So sánh đơn luồng và đa luồng

```

1 #Thời gian thực hiện với đơn Luồng (tuần tự)
2 import time
3 arr = [15,27,49,68]
4 t = time.time()
5 cal_square(arr) #Thực hiện tính arr bình phương
6 cal_cube(arr)   #Thực hiện tính arr Lập phương
7 print ("done in ", time.time()- t)

```

```

calculate square number
square: 225
square: 729
square: 2401
square: 4624
calculate cube number
cube: 3375
cube: 19683
cube: 117649
cube: 314432
done in  1.682037115097046

```

```

1 #Thời gian thực hiện với 2 luồng
2 from threading import Thread
3 import threading
4 import time
5 arr = [15,27,49,68]
6 try:
7     t = time.time()
8     #Tạo 2 luồng thread
9     t1 = threading.Thread(target=cal_square, args=(arr,))
10    t2 = threading.Thread(target=cal_cube, args=(arr,))
11    t1.start() #Bắt đầu thread 1
12    t2.start() #Bắt đầu thread 2
13    t1.join()  #Chờ tới khi thread 1 hoàn thành
14    t2.join()  #Chờ tới khi thread 2 hoàn thành
15    print ("done in ", time.time()- t)
16 except:
17     print ("error")

```

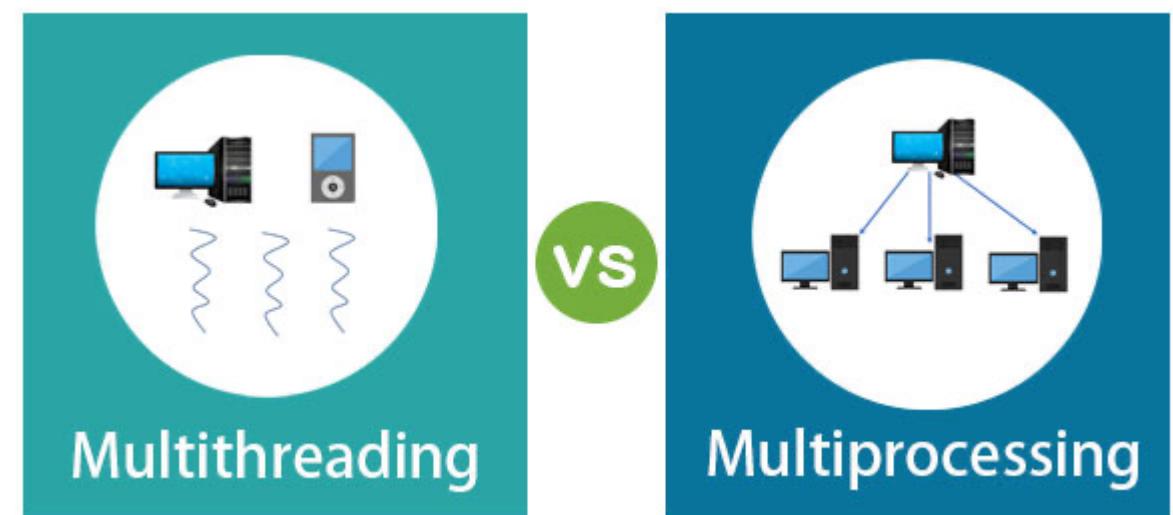
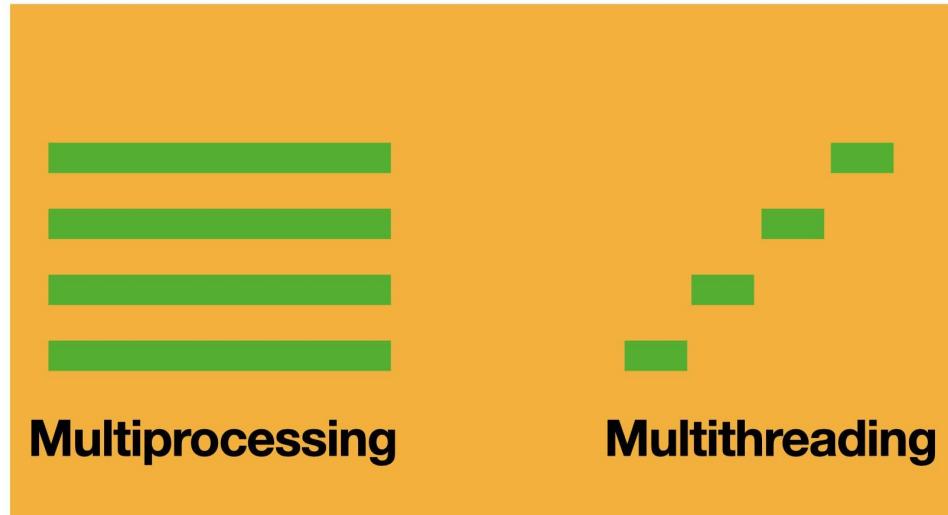
```

calculate square number
calculate cube number
square: 225
cube: 3375
square: 729
cube: 19683
square: 2401
cube: 117649
square: 4624
cube: 314432
done in  0.8385195732116699

```

Multiprocessing

- Xử lý đa tiến trình (Multiprocessing) là khả năng của một hệ thống hỗ trợ nhiều bộ vi xử lý cùng một lúc. Các ứng dụng của hệ thống đa xử lý được chia thành nhiều tiến trình nhỏ và chạy độc lập cùng nhau (xử lý song song) → Cải thiện hiệu suất của hệ thống

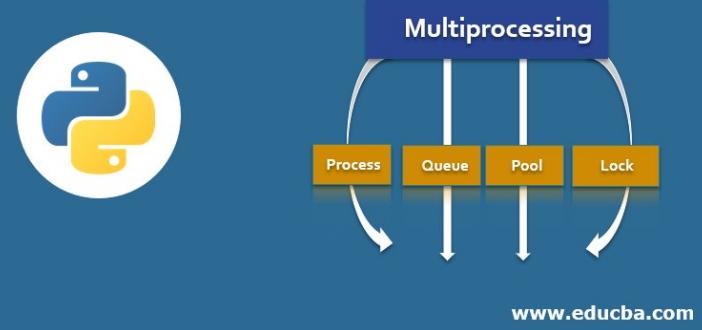


<https://laptrinhx.com/multithreading-vs-multiprocessing-1494474039/>

Multiprocessing

- Thư viện Multiprocessing trong Python là một module hỗ trợ lập trình viên có thể phân chia công việc theo nhiều tiến trình. Bằng cách thông qua những phương thức (API) mà module cung cấp sẵn, chúng ta có thể quản lý được các task một cách dễ dàng.

Python Multiprocessing



```

1 #Sử dụng thư viện multiprocessing kiểm tra số Lượng CPU
2 import multiprocessing
3 print("Số lượng CPU: ", multiprocessing.cpu_count())

```

Số lượng CPU: 4

Multiprocessing

Ví dụ về xử lý đa tiến trình với thư viện Multiprocessing của Python:

```

1 #Hàm tính bình phương của x:
2 def cal_square(numbers):
3     print("calculate square number")
4     for n in numbers:
5         time.sleep(0.2)
6         print ('square:', n*n)
7
8 #Hàm tính lập phương của x:
9 def cal_cube(numbers):
10    print("calculate cube number")
11    for n in numbers:
12        time.sleep(0.2)
13        print ('cube:', n*n*n)

```

```

1 #Tạo 2 tiến trình chạy song song với multiprocessing
2 import multiprocessing
3 import time
4 arr = [15,27,49,68]
5 try:
6     t = time.time()
7     #Tạo 2 tiến trình process
8     p1 = multiprocessing.Process(target=cal_square, args=(arr,))
9     p2 = multiprocessing.Process(target=cal_cube, args=(arr,))
10    p1.start()   #Bắt đầu process 1
11    p2.start()   #Bắt đầu process 2
12    p1.join()    #Chờ tới khi process 1 hoàn thành
13    p2.join()    #Chờ tới khi process 1 hoàn thành
14    print ("done in ", time.time()- t)
15 except:
16     print ("error")

```

```

calculate square number
calculate cube number
square: 225
cube: 3375
square: 729
cube: 19683
square: 2401
cube: 117649
square: 4624
cube: 314432
done in  0.8370239734649658

```

Tham khảo:

<https://phamdinhhanh.github.io/2020/11/30/ParallelComputingPython.html>

<https://docs.python.org/3.1/library/multiprocessing.html>

Functools

Functools là một thư viện của Python, cung cấp các tính năng hữu ích giúp làm việc với các hàm bậc cao dễ dàng hơn.

Molude này bao gồm các chức năng chính:

`@functools.cache()`

`@functools.cached_property()`

`@functools.lru_cache()`

`@functools.lru_cache()`

`@functools.total_ordering`

`@functools.reduce()`

`@functools.singledispatch`

`@functools.wraps()`



Tham khảo:

<https://docs.python.org/3/library/functools.html>



Q & A
Thank you!

Bài 4: LẬP TRÌNH PYTHON CƠ BẢN (Sử dụng thư viện NumPy làm việc với ma trận - 01)

AI Academy Vietnam

1. Một số thư viện quan trọng của Python

2. Giới thiệu thư viện Numpy

3. Cách tạo vector, ma trận (matrix)

- Tạo mảng 1D, 2D, 3D | Tạo mảng với các hàm có sẵn | Tạo mảng từ file dữ liệu

4. Thao tác cơ bản với mảng

- Quan sát thuộc tính | Chuyển đổi kiểu dữ liệu | Truy cập phần tử

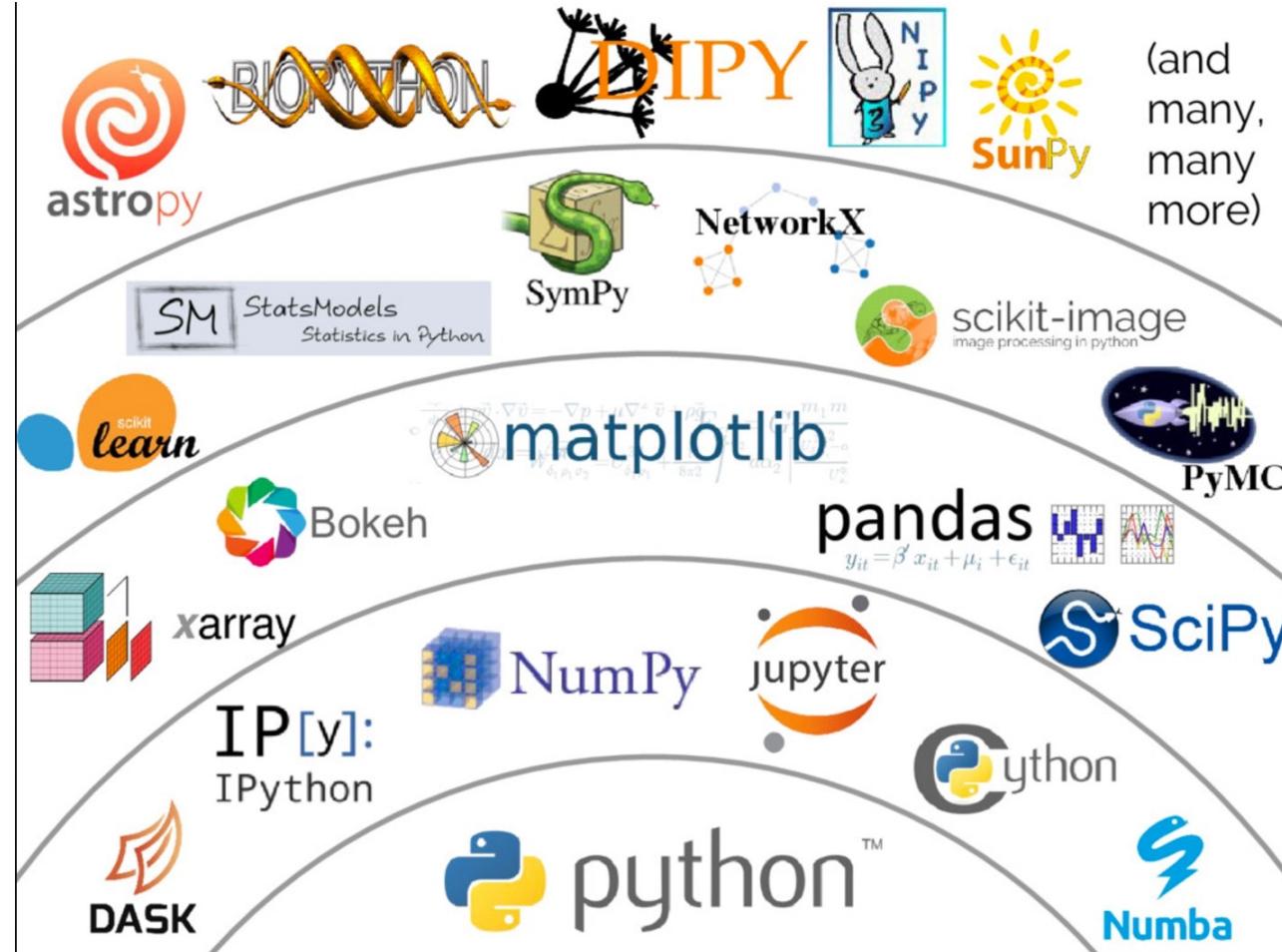
5. Tính toán các đặc trưng thống kê

- Min, Max, Mean, Median, Mode, Range, Std, Corrcoef

1. Một số thư viện quan trọng của Python

Thư viện Python

- Python có hệ thống thư viện rất phong phú, hỗ trợ nhiều lĩnh vực khác nhau.



- Do đó, tùy thuộc vào lĩnh vực nghiên cứu cụ thể, để lựa chọn và sử dụng các thư viện cho phù hợp.

Python libraries for Data Analysis

There are many interesting libraries that have made Python popular with Data Scientists:



Top 10 Python Packages 2020



Data Science in Python

Pandas,

pandas



Scikit-learn, Numpy



Matplotlib



Top 5 Python Libraries for Data Science



Report from Cloud Academy suggests that the top technical skill in demand for data engineers is python. 67 percent of job posts mentioned python.

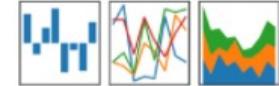
1.) NUMPY



Through NumPy, you can use it as an efficient multi-dimensional container of generic data. It also contains sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code, useful linear algebra, Fourier transform, and random number capabilities

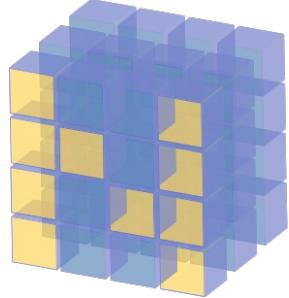
2.) PANDAS

Pandas provides high-performance, easy-to-use data structures and data analysis tools for python. You can store and manage data from tables by performing manipulation over rows and columns.



3.) SCIKIT-LEARN

Scikit-Learn is a powerful library for machine learning in Python. It contains simple and efficient tools for data mining and data analysis. It is built on NumPy, SciPy and matplotlib.



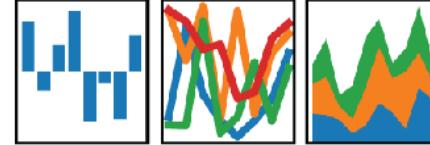
NumPy



OpenCV

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



	BandName	WavelengthMax	WavelengthMin
0	CoastalAerosol	450	430
1	Blue	510	450
2	Green	590	530
3	Red	670	640
4	NearInfrared	880	850
5	ShortWaveInfrared_1	1650	1570
6	ShortWaveInfrared_2	2290	2110
7	Cirrus	1380	1360



Machine Learning with Scikit-Learn



Natural Language Analyses with NLTK

■ Khai báo sử dụng thư viện trong Python

```
In [1]: #Khai báo sử dụng thư viện và kiểm tra phiên bản thư viện đang sử dụng
import numpy as np
print("Thu vien Numpy, Version: ",np.__version__)
```

```
Thu vien Numpy, Version:  1.15.4
```

```
In [2]: #Trong trường hợp thư viện chưa được cài đặt!
import scrapy as sc
print("Thu vien Scrapy, Version: ",sc.__version__)
```

```
-----
ModuleNotFoundError                                     Traceback (most recent call last)
<ipython-input-2-ef1be0ed66f4> in <module>
      1 #Trong trường hợp thư viện chưa được cài đặt!
----> 2 import scrapy as sc
      3 print("Thu vien Scrapy, Version: ",sc.__version__)
```

```
ModuleNotFoundError: No module named 'scrapy'
```

2. Thư viện NumPy

Thư viện Numpy

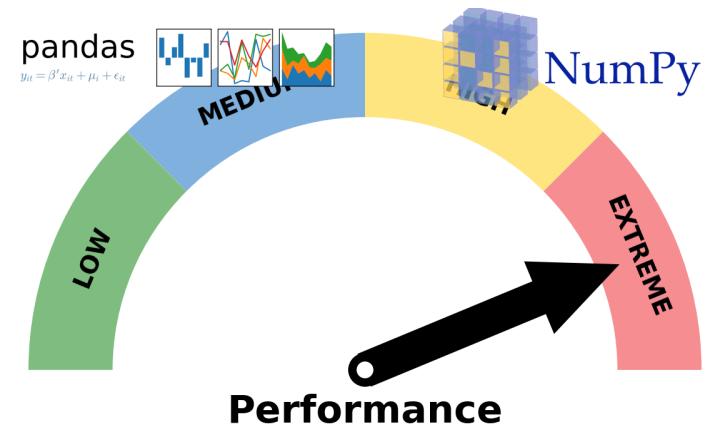
- **Numpy** (Numeric Python): là một thư viện toán học phổ biến và mạnh mẽ của Python.
- Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng “core Python” đơn thuần.
- Ngoài ra, Python cũng hỗ trợ một thư viện khác để mở rộng thêm các tính năng của Numpy là Scipy với ưu thế về các phép hồi quy hay biến đổi Fourier...
- Tham khảo thêm tại: <http://www.numpy.org/>

```
1 big_array = np.random.rand(1000000)
2 %timeit sum(big_array)
3 %timeit np.sum(big_array)
```

10 loops, best of 3: 171 ms per loop
1000 loops, best of 3: 380 µs per loop

```
1 %timeit min(big_array)
2 %timeit np.min(big_array)
```

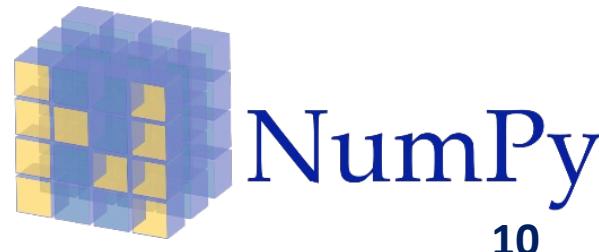
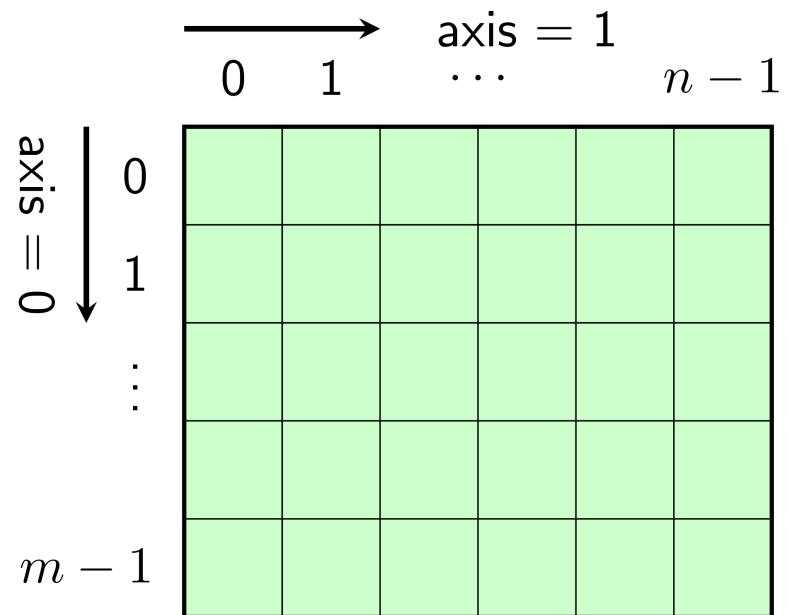
10 loops, best of 3: 103 ms per loop
1000 loops, best of 3: 432 µs per loop



Thư viện NumPy

- Đối tượng chính của NumPy là các mảng đa chiều đồng nhất:

- Kiểu dữ liệu của các phần tử con trong mảng phải **giống nhau**
- Mảng có thể có 1 chiều hoặc nhiều chiều
- Các chiều được đánh số từ 0 trở đi
- Số chiều được gọi là **hạng (rank)**
- Có đến 24 kiểu số khác nhau.
- Kiểu ndarray là lớp chính xử lý dữ liệu mảng nhiều chiều.
- Có rất nhiều hàm và phương thức xử lý mảng



Thư viện Numpy



VINBIGDATA



Academy
Vietnam

1	5	18	23
---	---	----	----

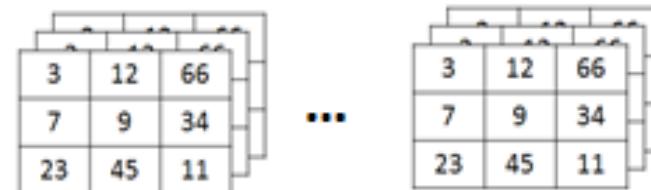
Vector (1D array)
Dimension = 1
(1 index required)

3	12	66
7	9	34
23	45	11

Matrix (2D array)
Dimension = 2
(2 indexes required)

3	12	66
7	9	34
23	45	11

3D array (3rd order Tensor)
Dimension = 3
(3 indexes required)



ND array
Dimension = N
(N indexes required)

3. Khởi tạo mảng

Khởi tạo mảng 1 chiều – 1D (Vector)

```
1 #Khởi tạo mảng 1 chiều với thư viện Numpy
2 import numpy as np
3
4 #Tạo mảng 1 chiều (1D) - row
5 a = np.array([1, 2, 5, 7, 0, 8])
6
7 print(a)
8 print("Loại dữ liệu của biến a:", type(a))
9 print("Kiểu dữ liệu của phần tử trong mảng a:", a.dtype)
10 print("Kích thước của mảng a:", a.shape)
11 print("Số phần tử của mảng a:", a.size)
12 print("Số chiều của mảng a:", a.ndim)
```

[1 2 5 7 0 8]

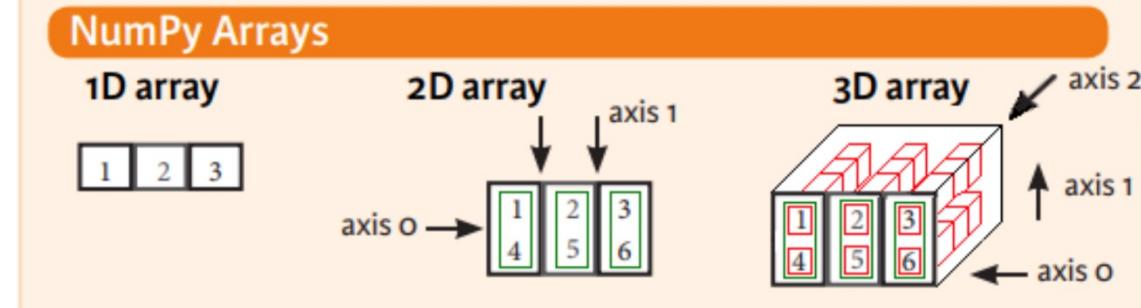
Loại dữ liệu của biến a: <class 'numpy.ndarray'>

Kiểu dữ liệu của phần tử trong mảng a: int32

Kích thước của mảng a: (6,)

Số phần tử của mảng a: 6

Số chiều của mảng a: 1



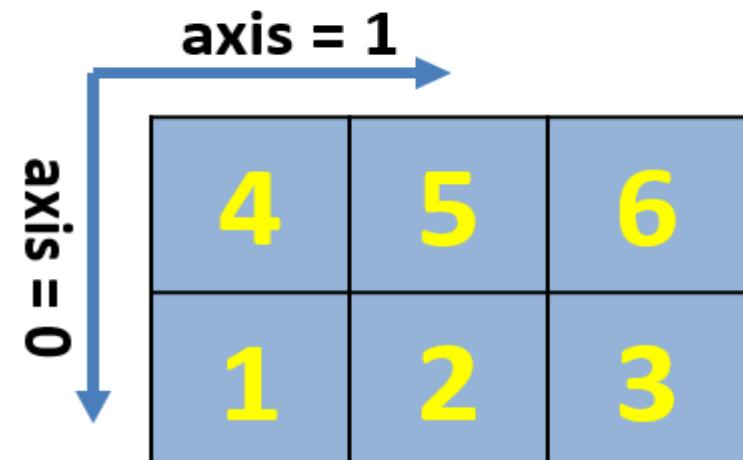
Khởi tạo mảng (2)

• Khởi tạo mảng 2 chiều – 2D (Matrix)

```
1 #Gọi thư viện numpy
2 import numpy as np
3
4 #Tạo mảng 2 chiều (2D - Ma trận)
5 b = np.array([(4, 5, 6.0),(1, 2, 3.5)])
6
7 print(b)
8 print("Loại dữ liệu của biến b:", type(b))
9 print("Kiểu dữ liệu của phần tử trong mảng b:", b.dtype)
10 print("Kích thước của mảng b:", b.shape)
11 print("Số phần tử của mảng b:", b.size)
12 print("Số chiều của mảng b:", b.ndim)
```

```
[[4.  5.  6. ]
 [1.  2.  3.5]]
```

Loại dữ liệu của biến b: <class 'numpy.ndarray'>
Kiểu dữ liệu của phần tử trong mảng b: float64
Kích thước của mảng b: (2, 3)
Số phần tử của mảng b: 6
Số chiều của mảng b: 2





Khởi tạo mảng (3)

- Khởi tạo mảng 3 chiều – 3D

```
1 import numpy as np
2
3 c = np.array([[(2,4,0,6), (4,7,5,6)],
4               [(0,3,2,1), (9,4,5,6)],
5               [(5,8,6,4), (1,4,6,8)]]) #mảng 3 chiều (3D)
6
7 print(c)
8 print("Phần tử đầu tiên của mảng c:",c[0,0,0])
9 print("Kiểu dữ liệu của phần tử trong mảng c:",c.dtype)
10 print("Kích thước của mảng c:",c.shape)
11 print("Số phần tử của mảng c:",c.size)
12 print("Số chiều của mảng c:",c.ndim)
```

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

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

```
[[[5 8 6 4]
  [1 4 6 8]]]
```

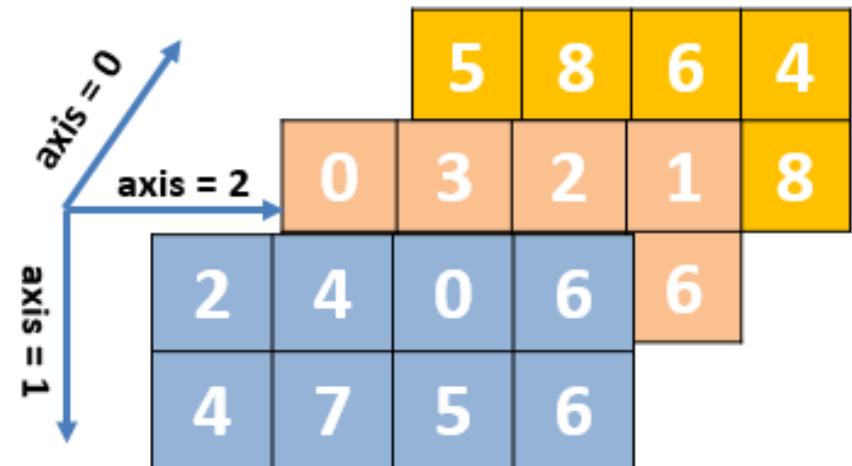
Phần tử đầu tiên của mảng c: 2

Kiểu dữ liệu của phần tử trong mảng c: int32

Kích thước của mảng c: (3, 2, 4)

Số phần tử của mảng c: 24

Số chiều của mảng c: 3



Khởi tạo mảng

Với các hàm có sẵn của NumPy

- Khởi tạo mảng với các hàm sẵn có của Numpy

Initial Placeholders

```
>>> np.zeros ((3, 4))  
>>> np.ones ((2, 3, 4), dtype=np.int16)  
>>> d = np.arange(10, 25, 5)  
  
>>> np.linspace(0, 2, 9)  
  
>>> e = np.full((2, 2), 7)  
>>> f = np.eye(2)  
>>> np.random.random((2, 2))  
>>> np.empty((3, 2))
```

Create an array of zeros
Create an array of ones
Create an array of evenly spaced values (step value)
Create an array of evenly spaced values (number of samples)
Create a constant array
Create a 2X2 identity matrix
Create an array with random values
Create an empty array

Khởi tạo mảng (5)

- Vd 1: Tạo ma trận 0|1 kích thước m x n

```
1 # Phương thức zeros: Tạo ma trận 0 kích thước 5 hàng x 3 cột
2 import numpy as np
3
4 array_zeros = np.zeros((5, 3))
5
6 print(array_zeros)
7 print("Kiểu dữ liệu trong mảng array_zeros:", array_zeros.dtype)
8 print("Kích thước của mảng array_zeros:", array_zeros.shape)
9 print("Số phần tử của mảng array_zeros:", array_zeros.size)
10 print("Số chiều của mảng array_zeros:", array_zeros.ndim)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
Kiểu dữ liệu trong mảng array_zeros: float64
Kích thước của mảng array_zeros: (5, 3)
Số phần tử của mảng array_zeros: 15
Số chiều của mảng array_zeros: 2
```

```
1 # Phương thức ones: Tạo ma trận 1 kích thước 3 hàng x 5 cột
2 import numpy as np
3
4 array_one = np.ones((3, 5), dtype=np.int)
5
6 print(array_one)
7 print("Kiểu dữ liệu trong mảng array_one:", array_one.dtype)
8 print("Kích thước của mảng array_one:", array_one.shape)
9 print("Số phần tử của mảng array_one:", array_one.size)
10 print("Số chiều của mảng array_one:", array_one.ndim)
```

```
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
Kiểu dữ liệu trong mảng array_one: int32
Kích thước của mảng array_one: (3, 5)
Số phần tử của mảng array_one: 15
Số chiều của mảng array_one: 2
```

Khởi tạo mảng (6)

- Vd 2: Tạo ma trận đơn vị cấp n

```
1 #Phương thức eye: Tạo ma trận đơn vị cấp 5
2 import numpy as np
3 array_eye = np.eye(5)
4
5 print(array_eye)
6 print("Kiểu dữ liệu của phần tử trong mảng array_eye:", array_eye.dtype)
7 print("Kích thước của mảng array_eye:", array_eye.shape)
8 print("Số phần tử của mảng array_eye:", array_eye.size)
9 print("Số chiều của mảng array_eye:", array_eye.ndim)
```

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

Kiểu dữ liệu của phần tử trong mảng array_eye: float64

Kích thước của mảng array_eye: (5, 5)

Số phần tử của mảng array_eye: 25

Số chiều của mảng array_eye: 2

Khởi tạo mảng (7)

- Vd 3: Tạo ma trận với các phần tử ngẫu nhiên trong khoảng (0,1)

```
1 #Phương thức random: Tạo một ma trận (7x5) các phần tử ngẫu nhiên [0,1]
2 import numpy as np
3 array_random = np.random.random((7,5))
4
5 print(array_random)
6 print("Kiểu dữ liệu của phần tử trong mảng array_random:", array_random.dtype)
7 print("Kích thước của mảng array_random:", array_random.shape)
8 print("Số phần tử của mảng array_random:", array_random.size)
9 print("Số chiều của mảng array_random:", array_random.ndim)
```

```
[[0.57738653 0.38330643 0.84085595 0.88920867 0.11759141]
 [0.13200344 0.40891213 0.46518628 0.81332657 0.62117097]
 [0.0255157 0.10842881 0.36001561 0.06382023 0.40403947]
 [0.37338483 0.35678386 0.38280971 0.97395415 0.8950108 ]
 [0.86054013 0.93742679 0.13039088 0.599897 0.0071806 ]
 [0.83131566 0.35010989 0.47524521 0.56107776 0.13418245]
 [0.39089618 0.40229334 0.73431802 0.53481456 0.45046071]]
```

Kiểu dữ liệu của phần tử trong mảng array_random: float64

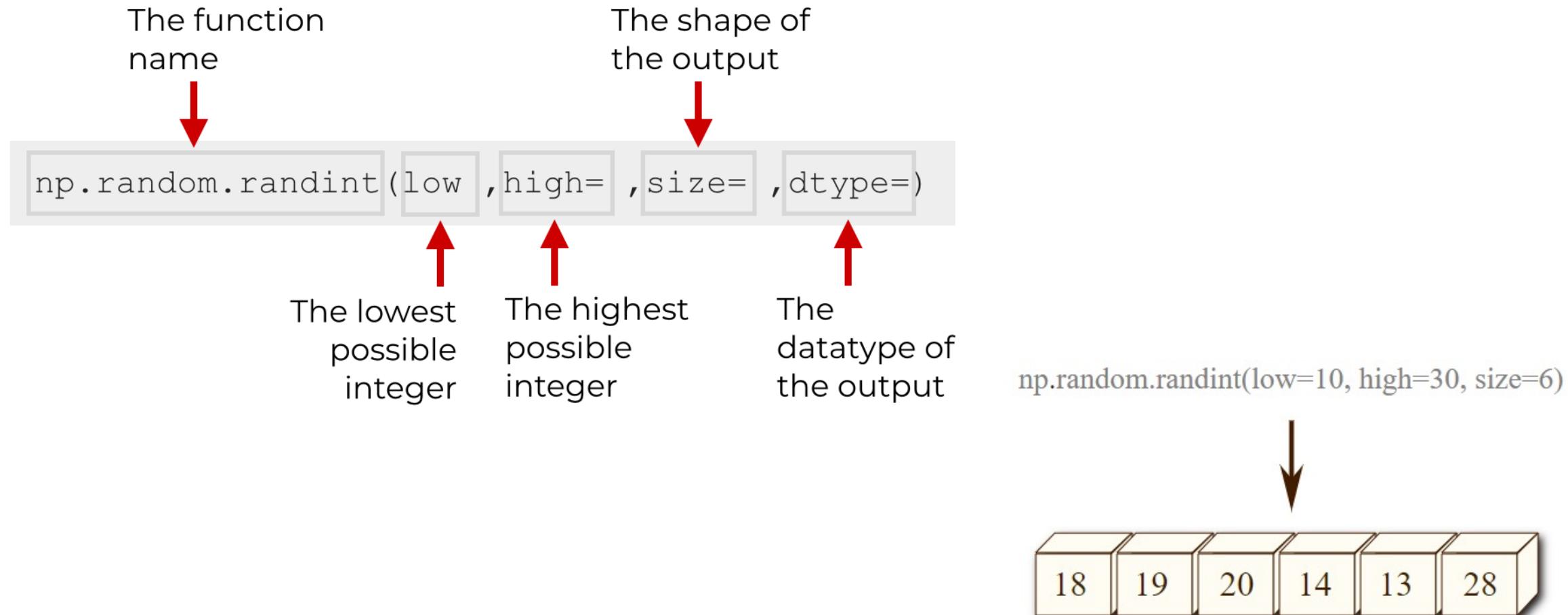
Kích thước của mảng array_random: (7, 5)

Số phần tử của mảng array_random: 35

Số chiều của mảng array_random: 2

Khởi tạo mảng (8)

- Vd 4: Tạo vector, ma trận với các phần tử là số nguyên ngẫu nhiên trong khoảng (low,high)



Khởi tạo mảng (9)

• Vd 5: Tạo vector với các tham số thiết lập

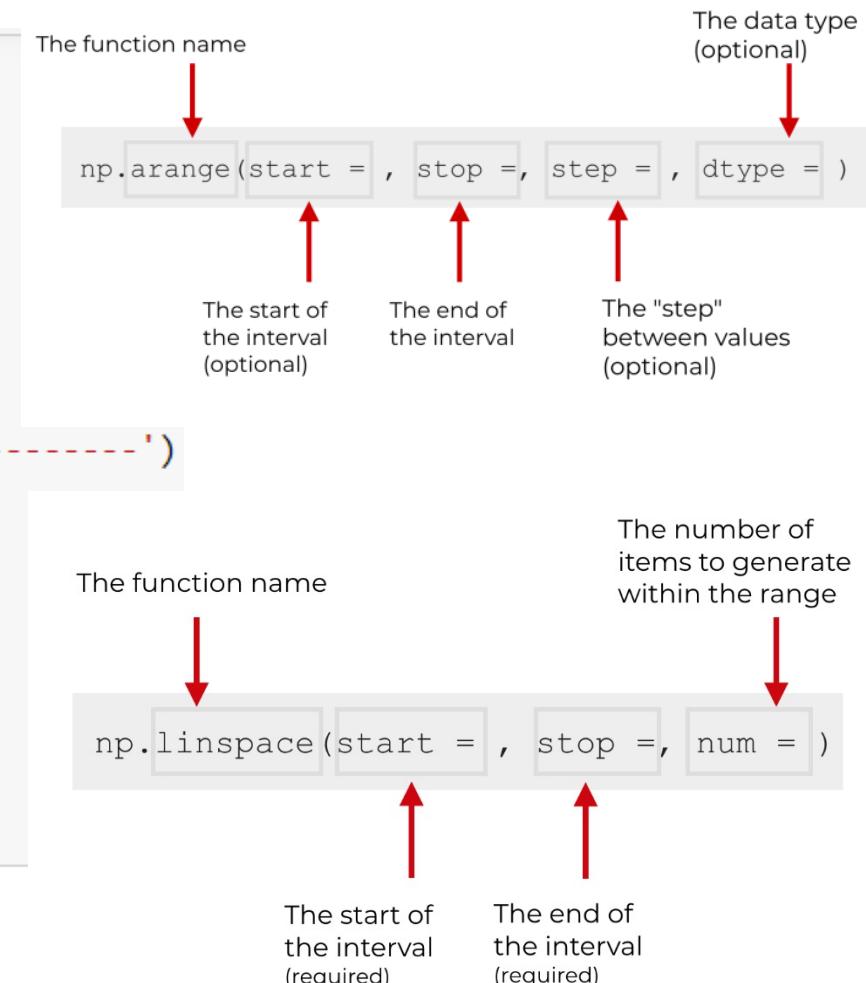
```
1 #Phương thức arange(a, b, steps):
2 #Tạo vector:
3 # Phần tử đầu tiên = a,
4 # kết thúc <b,
5 # mỗi phần tử cách nhau một khoảng = steps
6 d = np.arange(1, 15, 2)
7 print('Vector d:', d)
8 print("Số phần tử của vector d:", d.size)
9
10 print('-----')
11 #Phương thức linspace(a, b, num)
12 #Tạo vector:
13 #Phần tử đầu tiên = a,
14 #Phần tử kết thúc = b,
15 #Số phần tử của ma trận = num
16 f = np.linspace(1,15,11)
17 print('Vector f:', f)
18 print("Số phần tử của vector f:", f.size)
19
```

Vector d: [1 3 5 7 9 11 13]

Số phần tử của vector d: 7

Vector f: [1. 2.4 3.8 5.2 6.6 8. 9.4 10.8 12.2 13.6 15.]

Số phần tử của vector f: 11



Khởi tạo mảng từ file dữ liệu .txt

Khởi tạo mảng (10)

- Bảng điểm của lớp 2A (bao gồm 30 học sinh, tương ứng với 30 cột, của 10 môn học, tương ứng với 10 hàng) lưu trong file Diem_2A.txt

Diem_2A - Notepad

Điểm học sinh i (30 học sinh)										Điểm của môn học j (10 môn học)
1	2	3	4	5	6	7	8	9	10	
2, 4, 3, 7, 5, 6, 5, 6, 8, 9, 3, 6, 1, 9, 8, 7, 3, 9, 5, 1, 1, 6, 5, 1,	3, 5, 3, 10, 9, 1, 9, 8, 3, 1, 6, 0, 7, 10, 8, 5, 2, 7, 7, 1, 1, 6, 1, 6,	1, 10, 4, 9, 6, 9, 0, 2, 3, 1, 8, 6, 8, 4, 2, 9, 2, 9, 5, 0, 4, 1, 7, 3,	6, 3, 0, 8, 3, 7, 7, 2, 6, 8, 7, 3, 4, 1, 5, 9, 1, 0, 2, 10, 4, 6, 8, 6,	4, 3, 6, 7, 4, 5, 2, 6, 9, 4, 3, 9, 9, 4, 5, 7, 2, 10, 9, 4, 0, 5, 3, 1,	2, 3, 8, 10, 4, 5, 9, 5, 4, 7, 10, 1, 8, 4, 3, 9, 6, 3, 6, 7, 4, 7, 3, 5,	9, 9, 1, 10, 9, 9, 5, 9, 6, 3, 9, 5, 1, 10, 7, 10, 2, 8, 8, 1, 8, 4, 5, 4,	8, 8, 4, 8, 0, 4, 4, 8, 6, 7, 1, 3, 1, 6, 8, 8, 4, 6, 8, 4, 0, 1, 8, 2,	6, 7, 8, 9, 10, 9, 2, 2, 6, 1, 10, 9, 6, 3, 9, 5, 9, 8, 1, 1, 8, 8, 8, 6,	7, 8, 7, 8, 6, 10, 10, 6, 8, 10, 8, 9, 8, 8, 5, 10, 8, 7, 8, 7, 9, 9, 8, 7,	

Khởi tạo mảng (11)

- Đọc dữ liệu từ file txt vào biến mảng.

```
1 import numpy as np
2
3 #Đọc dữ liệu từ file Diem_2A.txt
4 path = 'Data_Excercise\Diem_2A.txt'
5 diem_2a = np.loadtxt(path, delimiter=',', dtype=np.int)
6
7 print(diem_2a)
8 print("Kiểu dữ liệu của phần tử trong mảng diem_2a:", diem_2a.dtype)
9 print("Kích thước của mảng diem_2a:", diem_2a.shape)
10 print("Số phần tử của mảng diem_2a:", diem_2a.size)
11 print("Số chiều của mảng diem_2a:", diem_2a.ndim)
```

```
[[ 2  4  3  7  5  6  5  6  8  9  3  6  1  9  8  7  3  3  9  5  1  6  5  1
   4  6  7  1  1  1]
 [ 3  5  3  10  9  1  9  8  3  1  6  0  7  10  8  5  2  7  7  1  1  6  1  6
   3  0  2  2  1  6]
```

```
[ 6  7  8  9  10  9  2  2  6  1  10  9  6  3  9  5  9  8  1  1  8  8  8  6
   6  8  7  3  8  1]
 [ 7  8  7  8  6  10  10  6  8  10  8  9  8  8  5  10  8  7  8  7  9  9  8  7
   7  7  10  8  9  7]]
```

Kiểu dữ liệu của phần tử trong mảng diem_2a: int32
Kích thước của mảng diem_2a: (10, 30)
Số phần tử của mảng diem_2a: 300
Số chiều của mảng diem_2a: 2

4. Các thao tác cơ bản

4.1 Quan sát mảng



VINBIGDATA



AS Academy Vietnam

```
#a.shape: Cho biết kích thước của mảng a:  
print('kích thước của mảng diem_2a:', diem_2a.shape)
```

kích thước của mảng diem_2a: (10, 30)

```
#a.ndim: Cho biết Số chiều của mảng a:  
print('Số chiều của mảng diem_2a:', diem_2a.ndim)
```

Số chiều của mảng diem_2a: 2

```
#a.size: Cho biết số phần tử của mảng a:  
print('Số phần tử của mảng diem_2a: ', diem_2a.size)
```

Số phần tử của mảng diem_2a: 300

```
#a.dtype: Cho biết kiểu dữ liệu của các phần tử trong mảng a  
print('Kiểu dữ liệu của các phần tử trong mảng diem_2a:', diem_2a.dtype)
```

Kiểu dữ liệu của các phần tử trong mảng diem_2a: float64

4.2 Chuyển đổi kiểu dữ liệu

```
1 #a.astype(kiểu mới): Chuyển đổi kiểu dữ liệu của các phần tử
2 a_float = np.linspace(0,15,11)
3 print(a_float)
4 print('Kiểu Dữ liệu: ', a_float.dtype)
5 print('-----')
6 #Chuyển từ kiểu float --> int
7 a_int = a_float.astype(np.int16)
8 print(a_int)
9 print('Dữ liệu sau khi chuyển: ', a_int.dtype)
```

[0. 1.5 3. 4.5 6. 7.5 9. 10.5 12. 13.5 15.]

Kiểu Dữ liệu: float64

[0 1 3 4 6 7 9 10 12 13 15]

Dữ liệu sau khi chuyển: int16

```
1 #Chuyển từ kiểu float --> int
2 a_str = a_int.astype(np.str)
3 print(a_str)
4 print('Dữ liệu sau khi chuyển: ', a_str.dtype)
5 print('-----')
6 #Chuyển từ kiểu float --> boolean
7 a_bol = a_int.astype(np.bool)
8 print(a_bol)
9 print('Dữ liệu sau khi chuyển: ', a_bol.dtype)
```

['0' '1' '3' '4' '6' '7' '9' '10' '12' '13' '15']

Dữ liệu sau khi chuyển: <U6

[False True True True True True True True True True True]

Dữ liệu sau khi chuyển: bool

NumPy dtypes



Basic Type	Available NumPy types	Comments
Boolean	bool	Elements are 1 byte in size
Integer	int8, int16, int32, int64, int128, int	int defaults to the size of int in C for the platform
Unsigned Integer	uint8, uint16, uint32, uint64, uint128, uint	uint defaults to the size of unsigned int in C for the platform
Float	float32, float64, float, longfloat,	Float is always a double precision floating point value (64 bits). longfloat represents large precision floats. Its size is platform dependent.
Complex	complex64, complex128, complex	The real and complex elements of a complex64 are each represented by a single precision (32 bit) value for a total size of 64 bits.
Strings	str, unicode	Unicode is always UTF32 (UCS4)
Object	object	Represent items in array as Python objects.
Records	void	Used for arbitrary data structures in record arrays.

Chuyển đổi kiểu dữ liệu (3)

Data type	Description
bool_	Boolean (True or False) stored as a byte
int_	Default integer type (same as C long ; normally either int64 or int32)
intc	Identical to C int (normally int32 or int64)
intp	Integer used for indexing (same as C ssize_t ; normally either int32 or int64)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float_	Shorthand for float64 .
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_	Shorthand for complex128 .
complex64	Complex number, represented by two 32-bit floats
complex128	Complex number, represented by two 64-bit floats

4.3 Truy cập tới các phần tử

- Truy cập tới phần tử trong một vector (1D)

```
1 #Truy cập tới một phần tử của Vector: a[index]
2 #Note: index phần tử đầu tiên 0
3 #   : index phần tử cuối cùng -1
4 a = np.array([3, 5, 3, 10, 9, 1, 9, 8, 3, 1])
5
6 print('các phần tử của Vector a:\n', a)
7 print('-----')
8 print('phần tử đầu tiên:', a[0])
9 print('phần tử thứ 3:', a[3])
10 print('phần tử cuối cùng:', a[-1])
```

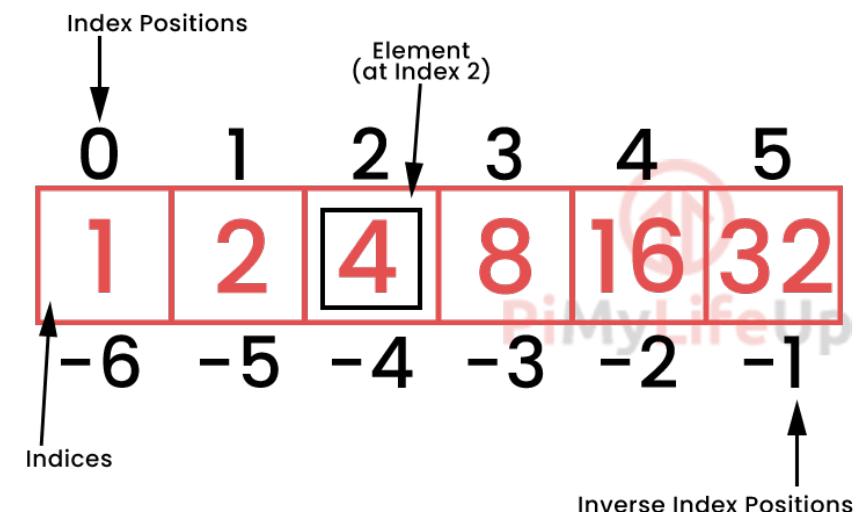
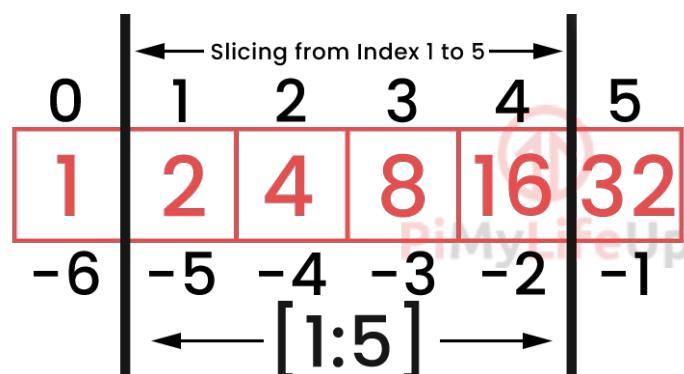
các phần tử của Vector a:

```
[ 3 5 3 10 9 1 9 8 3 1]
```

phần tử đầu tiên: 3

phần tử thứ 3: 10

phần tử cuối cùng: 1



```
1 #Truy cập tới nhiều phần tử của Vector: a[index1:index2]
2 print('các phần tử của Vector a:\n', a)
3 print('-----')
4 print('3 Phần tử đầu tiên:', a[:3])
5 print('Từ phần tử thứ 5 tới hết:', a[5:])
6 print('Từ phần tử 2 đến phần tử <6 của vector:', a[2:6])
```

các phần tử của Vector a:

```
[ 3 5 3 10 9 1 9 8 3 1]
```

3 Phần tử đầu tiên: [3 5 3]

Từ phần tử thứ 5 tới hết: [1 9 8 3 1]

Từ phần tử 2 đến phần tử <6 của vector: [3 10 9 1]

Truy cập tới các phần tử (2)

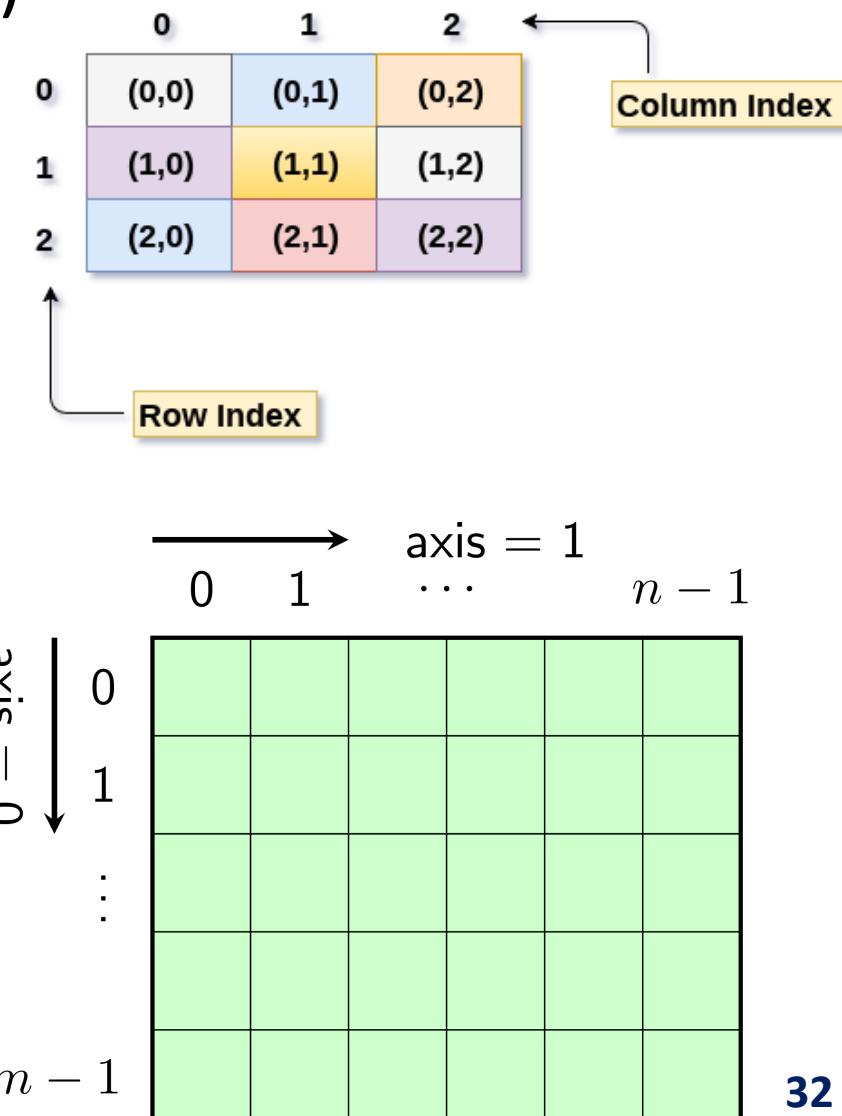
- Truy cập tới các phần tử trong một ma trận (2D)

```
1 #Truy cập tới 1 phần tử của ma trận (2D): a[index_row, index_col]
2 print('Điểm môn học đầu tiên, của học sinh đầu tiên:',diem_2a[0,0])
3 print('Điểm môn học thứ 1, của học sinh thứ 3:',diem_2a[1,3])
4 print('Điểm môn cuối cùng, của học sinh cuối cùng:',diem_2a[-1,-1])
5 print('-----')
6 print('Bảng điểm lớp 2A:\n',diem_2a)
```

Điểm môn học đầu tiên, của học sinh đầu tiên: 2

Điểm môn học thứ 1, của học sinh thứ 3: 10

Điểm môn cuối cùng, của học sinh cuối cùng: 7



Truy cập tới các phần tử (3)

- Truy cập tới các phần tử trong một ma trận (2D)

```
1 #Truy cập tới nhiều phần tử trong ma trận: a[index_row1:index_row2,index_col1:index_col2]
2 #Lấy điểm tất cả các môn (tất cả các hàng) của học sinh 5:
3 diem_hs5 = diem_2a[:,5]
4 print("Điểm các môn của học sinh 5:",diem_hs5)
5
6 #Lấy điểm môn học cuối cùng của tất cả học sinh (tất cả các cột)
7 diem_mon = diem_2a[-1,:]
8 print("Điểm môn học cuối cùng của tất cả học sinh: \n",diem_mon)
9
10 #Lấy điểm 5 môn học đầu tiên của 10 học sinh đầu tiên
11 diem5_hs10 = diem_2a[:5,:10]
12 print("Bảng điểm 5 môn học đầu tiên của 10 học sinh đầu của lớp:\n",diem5_hs10)
```

Điểm các môn của học sinh 5: [6 1 9 7 5 5 9 7 9 10]

Điểm môn học cuối cùng của tất cả học sinh:

```
[ 7 8 7 8 6 10 10 6 8 10 8 9 8 8 5 10 8 7 8 7 9 9 8 7  
7 7 10 8 9 7]
```

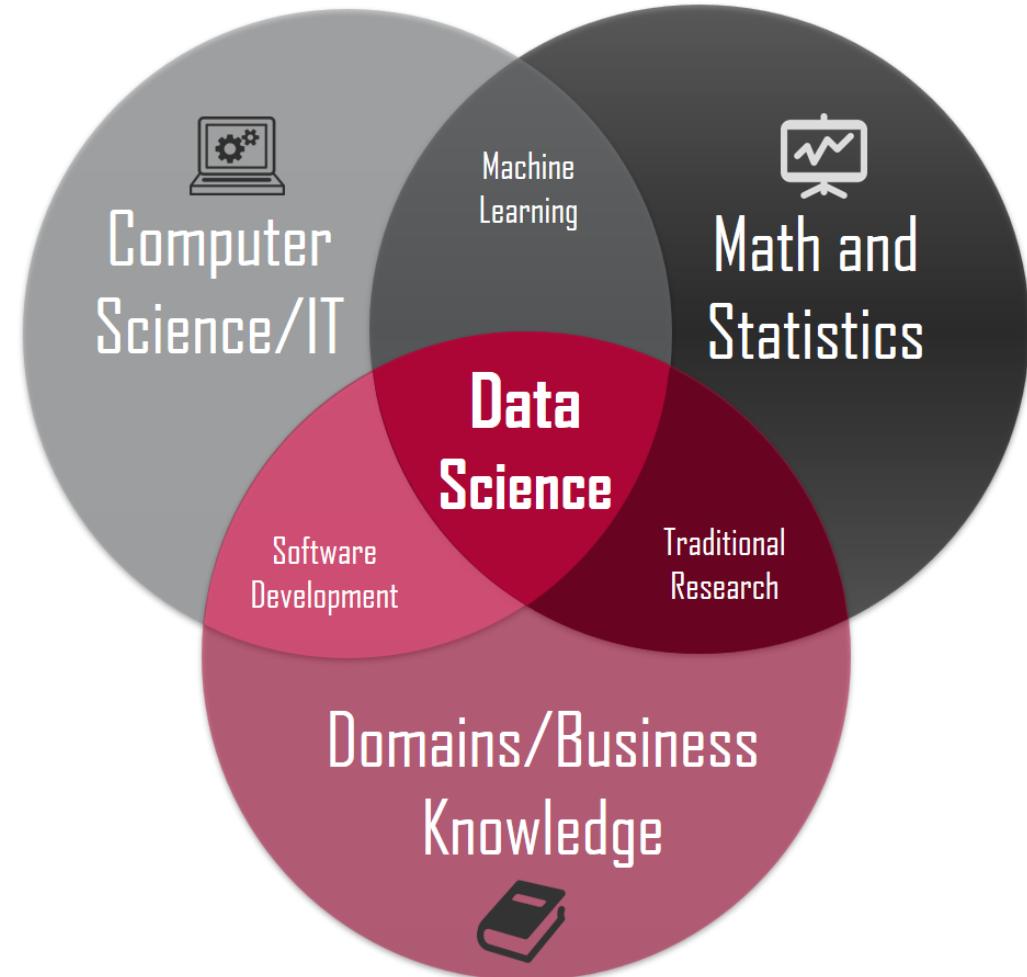
Bảng điểm 5 môn học đầu tiên của 10 học sinh đầu của lớp:

```
[[ 2 4 3 7 5 6 5 6 8 9]  
[ 3 5 3 10 9 1 9 8 3 1]  
[ 1 10 4 9 6 9 0 2 3 1]  
[ 6 3 0 8 3 7 7 2 6 8]  
[ 4 3 6 7 4 5 2 6 9 4]]
```

THỰC HÀNH 1

5. Tính toán các đặc trưng thống kê

5. Các đặc trưng thống kê



Toán học và thống kê có một vai trò rất quan trọng trong khoa học dữ liệu!



5.1 Max - Min

- **a.max():** Lấy giá trị lớn nhất của mảng a
- **b.min():** Lấy giá trị nhỏ nhất của mảng b

```
1 #Max - Min: Xác định giá trị lớn nhất, nhỏ nhất:  
2 #1) Hiển thị điểm cao nhất, thấp nhất của lớp 2A  
3 print('Điểm cao nhất của lớp:',diem_2a.max())  
4 print('Điểm thấp nhất của lớp:',diem_2a.min())
```

Điểm cao nhất của lớp: 10

Điểm thấp nhất của lớp: 0

```
1 #2) Liệt kê điểm cao nhất và thấp nhất theo môn học  
2 for i in range(0,diem_2a.shape[0]):  
3     print('Môn ', i,' : Điểm Max: ', diem_2a[i,:].max(),  
        '-- Điểm Min:', diem_2a[i,:].min())
```

Môn 0 : Điểm Max: 9 -- Điểm Min: 1

Môn 1 : Điểm Max: 10 -- Điểm Min: 0

Môn 2 : Điểm Max: 10 -- Điểm Min: 0

```
1 #3) Liệt kê điểm cao nhất và thấp nhất của mỗi học sinh  
2 for i in range(0,diem_2a.shape[1]):  
3     print('Học sinh ', i,' : Điểm Max: ', diem_2a[:,i].max(),  
        '-- Điểm Min:', diem_2a[:,i].min())
```

Học sinh 0 : Điểm Max: 9 -- Điểm Min: 1

Học sinh 1 : Điểm Max: 10 -- Điểm Min: 3





5.2 Sum

- **a.sum():**Tính tổng tất cả các phần tử của mảng a

```
1 #Sum:Tính tổng các phần tử trong mảng
2 print('Tổng tất các điểm trong của lớp 2A:', diem_2a.sum())
3 print('-----')
4
5 #Tính tổng điểm của từng học sinh:
6 for i in range(0,diem_2a.shape[1]):
7     print('Tổng điểm các môn của học sinh ', i, ' : ', diem_2a[:,i].sum())
```

Tổng tất các điểm trong của lớp 2A: 1731

Tổng điểm các môn của học sinh 0 : 48
Tổng điểm các môn của học sinh 1 : 60
Tổng điểm các môn của học sinh 2 : 47
Tổng điểm các môn của học sinh 3 : 86
Tổng điểm các môn của học sinh 4 : 62
Tổng điểm các môn của học sinh 5 : 68
Tổng điểm các môn của học sinh 6 : 56
Tổng điểm các môn của học sinh 7 : 54
Tổng điểm các môn của học sinh 8 : 59
Tổng điểm các môn của học sinh 9 : 51



5.3 Mean, Median, Mode, Range



VINBIGDATA



AS Academy Vietnam

Statistics – Mean, Median, Mode and Range

EZY MATHS

Mean

$$\text{Mean} = \frac{\text{Total of all values}}{\text{number of values}}$$

3, 3, 4, 5, 5, 8, 9, 15

$$\text{Mean} = \frac{52}{8} = 6.5$$

Collect it all together and share it out evenly

Using the mean to find the total amount

$$\text{Mean} \times \text{Number of values}$$

Ezytown FC have scored an average of 3.8 goals per game in their last 15 matches. How many goals have they scored?

$$3.8 \times 15 = 57 \text{ goals}$$

Median

Median = Middle value
(Numbers written in order)

3, 3, 4, 5, 5, 8, 9, 15

$$\text{Median} = 5$$

Finds the middle value

Use of formula to find location of median

$$\text{Location} = \frac{n + 1}{2}$$

The median of 45 values would be the 23rd number when written in order

$$\frac{45 + 1}{2} = 23$$

Mode

Mode = Most common value/item

3, 3, 4, 5, 5, 8, 9, 15

Mode = 3 and 5

Average usually used for qualitative data

Occurrence of no mode

1, 1, 3, 3, 7, 7

Each value appears twice so there is no mode

Range

Range = Largest - Smallest

3, 3, 4, 5, 5, 8, 9, 15

$$\text{Range} = 15 - 3 = 12$$

Reveals how close/far apart the values are

Interpreting measures of spread

The Smaller the range, the closer and more 'consistent' the values are.

The Larger the range, the more varied and more 'inconsistent' the values are.



Mean

```

1 # a.mean(): Giá trị trung bình của mảng a
2 print('Điểm trung bình của cả lớp 2A:', diem_2a.mean())
3 print('-----')
4 #Tính điểm trung bình của các học sinh trong Lớp:
5 #CÁCH 1:
6 for i in range(0,diem_2a.shape[1]):
7     print('Điểm trung bình của học sinh ', i, ' : ', diem_2a[:,i].mean())

```

Điểm trung bình của cả lớp 2A: 5.77

 Điểm trung bình của học sinh 0 : 4.8
 Điểm trung bình của học sinh 1 : 6.0
 Điểm trung bình của học sinh 2 : 4.7

```

1 #Tính điểm trung bình của các học sinh trong Lớp:
2 #CÁCH 2:
3 mean_2a = diem_2a.mean(axis=0)
4 #axis = 0: theo hàng
5 #axis = 1: theo cột
6 for i in range(0,mean_2a.size):
7     print('Điểm trung bình của học sinh ', i, ' : ', mean_2a[i])

```

Điểm trung bình của học sinh 0 : 4.8
 Điểm trung bình của học sinh 1 : 6.0
 Điểm trung bình của học sinh 2 : 4.7

Mean

Mean = $\frac{\text{Total of all values}}{\text{number of values}}$

3,3,4,5,5,8,9,15

$$\text{Mean} = \frac{52}{8} = 6.5$$

Collect it all together and share it out evenly

Using the mean to find the total amount

$\text{Mean} \times \text{Number of values}$

Ezytown FC have scored an average of 3.8 goals per game in their last 15 matches. How many goals have they scored?

$$3.8 \times 15 = 57 \text{ goals}$$



Median

- **np.median(a):Tìm trung vị của mảng a**

```

1 #median(): Giá trị trung vị trong một tập hợp các phần tử.
2 #Trường hợp số phần tử trong mảng là lẻ
3 a=diem_2a[1,:15]
4
5 print('Mảng a ban đầu: \n', a)
6 print('Số phần tử trong mảng a: ', a.size)
7 print('Mảng a đã sắp xếp: \n', np.sort(a))
8 print('Giá trị trung bình mean: ', np.mean(a))
9 print('Giá trị trung vị median: ', np.median(a))

```

Mảng a ban đầu:

[3 5 3 10 9 1 9 8 3 1 6 0 7 10 8]

Số phần tử trong mảng a: 15

Mảng a đã sắp xếp:

[0 1 1 3 3 3 5 6 7 8 8 9 9 10 10]

Giá trị trung bình mean: 5.533333333333333

Giá trị trung vị median: 6.0

Mảng a ban đầu:

[9 1 1 8 4 7 3 7 1 10]

Số phần tử trong mảng a: 10

Mảng a đã sắp xếp:

[1 1 1 3 4 7 7 8 9 10]

Giá trị trung bình mean: 5.1

Giá trị trung vị median: 5.5

Median

Median = Middle value
(Numbers written in order)

3, 3, 4, 5, 5, 8, 9, 15

Median = 5

Finds the middle value

Use of formula to find location of median

$$\text{Location} = \frac{n + 1}{2}$$

The median of 45 values would be the 23rd number when written in order

$$\frac{45 + 1}{2} = 23$$



Mode

```
1 #C) Mode: là giá trị xuất hiện nhiều nhất trong tập hợp.  
2 #Trong trường hợp không có giá trị nào được Lặp Lại thì không có Mode.  
3 #Liệt kê điểm xuất hiện nhiều nhất theo từng môn học  
4 from scipy import stats as sp #sử dụng thư viện scipy để dùng hàm mode  
5  
6 for i in range(0,diem_2a.shape[0]):  
7     a = sp.mode(diem_2a[i,:])  
8     print('Môn ', i,': Điểm xuất hiện nhiều nhất: ', a[0],  
9           ' số lần: ', a[1])  
10    print(type(a))
```

```
Môn 0 : Điểm xuất hiện nhiều nhất: [1] số lần: [6]  
Môn 1 : Điểm xuất hiện nhiều nhất: [1] số lần: [6]  
Môn 2 : Điểm xuất hiện nhiều nhất: [9] số lần: [8]  
Môn 3 : Điểm xuất hiện nhiều nhất: [6] số lần: [5]  
Môn 4 : Điểm xuất hiện nhiều nhất: [4] số lần: [6]  
Môn 5 : Điểm xuất hiện nhiều nhất: [5] số lần: [5]  
Môn 6 : Điểm xuất hiện nhiều nhất: [9] số lần: [8]  
Môn 7 : Điểm xuất hiện nhiều nhất: [8] số lần: [15]  
Môn 8 : Điểm xuất hiện nhiều nhất: [8] số lần: [7]  
Môn 9 : Điểm xuất hiện nhiều nhất: [8] số lần: [10]  
<class 'scipy.stats.stats.ModeResult'>
```

Mode

Mode = Most common value/item

3, 3, 4, 5, 5, 8, 9, 15

Mode = 3 and 5

Average usually used for qualitative data

Occurrence of no mode

If every value appears equally, there is no mode

1, 1, 3, 3, 7, 7

Each value appears twice so there is no mode



Range

Trong thư viện numpy không có hàm tính range, ta có thể xác định giá trị range bằng cách tính thông qua max - min

```
1 #D) Range: là sự khác biệt, khoảng cách giữa phần tử dưới và phần tử trên,  
2 #giữa giá trị nhỏ nhất (Min) với giá trị lớn nhất (Max) trong tập hợp.  
3 #Xác định độ chênh điểm max - min của từng học sinh  
4  
5 for i in range(0,diem_2a.shape[1]):  
6     print('Độ chênh điểm của học sinh ', i, ' : ',  
7           diem_2a[:,i].max()-diem_2a[:,i].min())
```

Độ chênh điểm của học sinh 0 : 8
Độ chênh điểm của học sinh 1 : 7
Độ chênh điểm của học sinh 2 : 8
Độ chênh điểm của học sinh 3 : 3
Độ chênh điểm của học sinh 4 : 7
Độ chênh điểm của học sinh 5 : 9
Độ chênh điểm của học sinh 6 : 10
Độ chênh điểm của học sinh 7 : 7
Độ chênh điểm của học sinh 8 : 6
Độ chênh điểm của học sinh 9 : 9
Độ chênh điểm của học sinh 10 : 7

Range

Range = Largest - Smallest

3, 3, 4, 5, 5, 8, 9, 15

Range = 15 – 3 = 12

Reveals how close/far apart the values are

Interpreting measures of spread

The Smaller the range, the closer and more 'consistent' the values are.

The Larger the range, the more varied and more 'inconsistent' the values are.

5.4 std

Độ lệch tiêu chuẩn (standard deviation) là đại lượng thường được sử dụng để phản ánh mức độ phân tán của một biến số xung quanh số bình quân.

```

1 #E) Std: Tính độ lệch chuẩn
2 a = np.array([10,1,1,9,12,1,9,12,10])
3 print('Phần tử của mảng a:',a)
4 print('Giá trị trung bình:',a.mean())
5 print('Độ lệch chuẩn:',a.std())
6
7 print('-----')
8 b = np.array([7,7,8,7,8,7,7,7,7])
9 print('Phần tử của mảng b:',b)
10 print('Giá trị trung bình:',b.mean())
11 print('Độ lệch chuẩn:',b.std())

```

Phần tử của mảng a: [10 1 1 9 12 1 9 12 10]

Giá trị trung bình: 7.222222222222222

Độ lệch chuẩn: 4.516089207311461

Phần tử của mảng b: [7 7 8 7 8 7 7 7 7]

Giá trị trung bình: 7.222222222222222

Độ lệch chuẩn: 0.41573970964154905

- Nếu độ lệch chuẩn bằng 0, suy ra các giá trị quan sát cũng chính là giá trị trung bình. Nói cách khác là không có sự biến thiên.
- Nếu độ lệch chuẩn càng lớn, suy ra sự biến thiên xung quanh giá trị trung bình càng lớn.

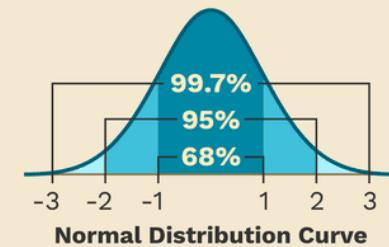
Calculating Standard Deviation

$$S_x = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

n = The number of data points

x_i = Each of the values of the data

\bar{x} = The mean of x_i

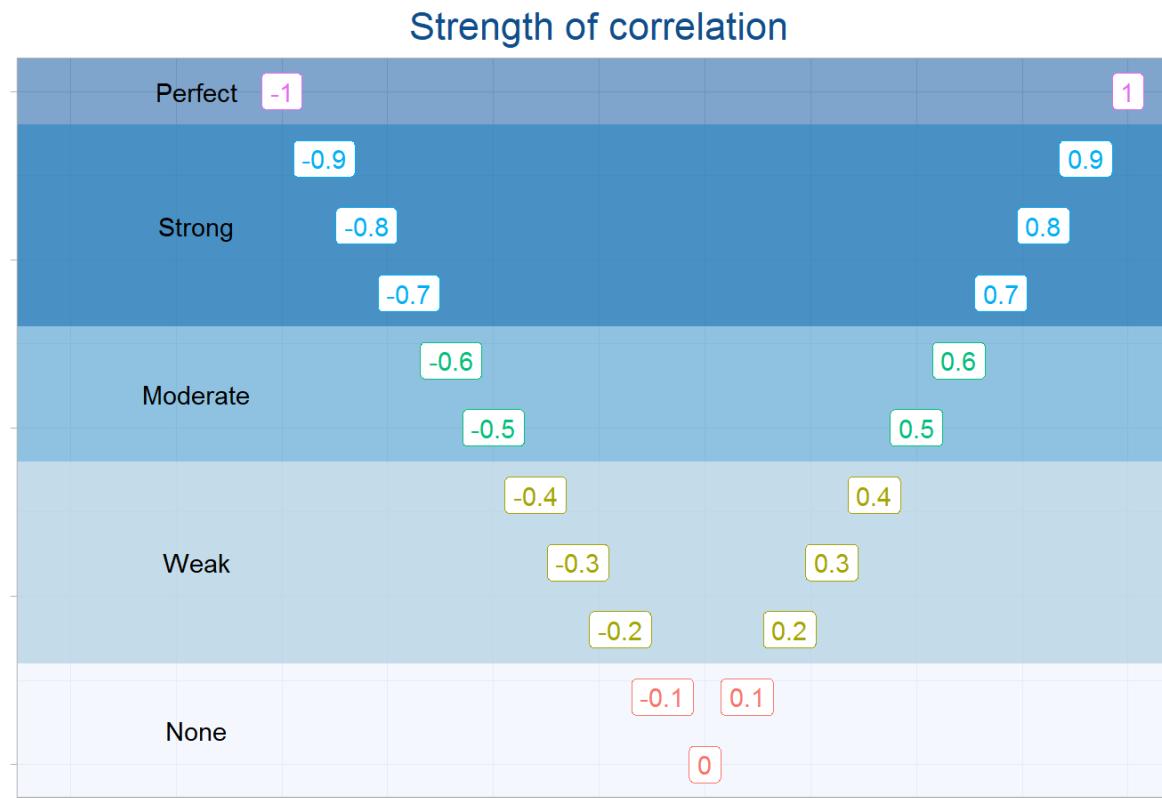


THỰC HÀNH 2

5.5 Hệ số tương quan

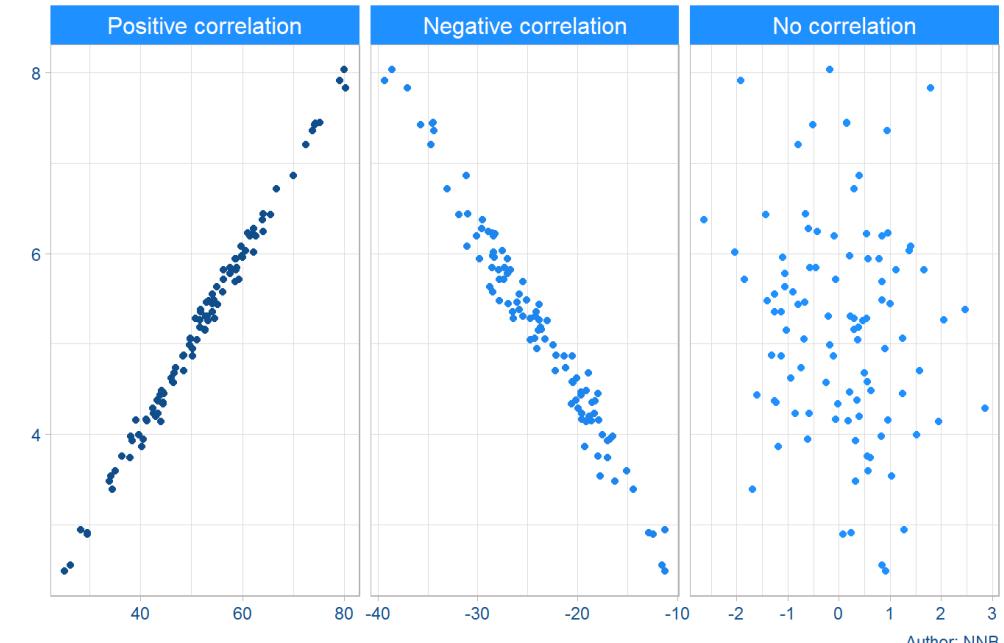
Hệ số tương quan đo lường mức độ quan hệ tuyến tính giữa hai biến.

- Hệ số tương quan không có đơn vị
- Hệ số tương quan nằm trong khoảng [-1, 1]



Correlation Coefficient Formula

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2] [n\sum y^2 - (\sum y)^2]}}$$



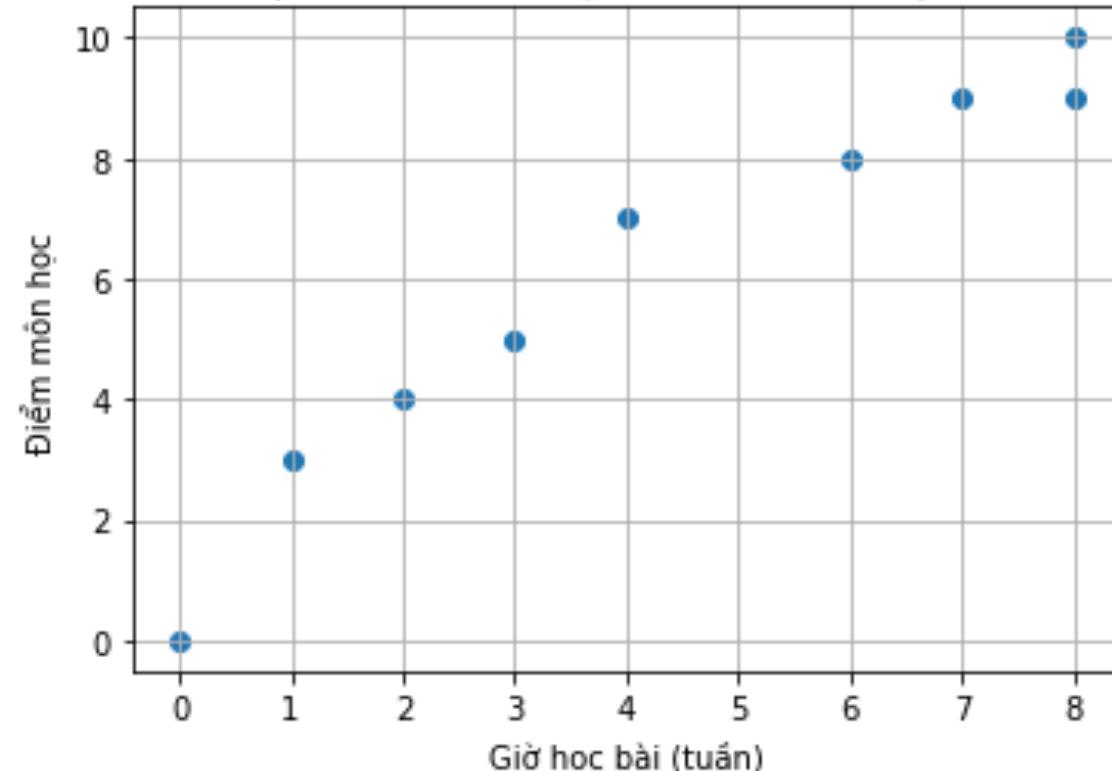
Hệ số tương quan (2)

```
#corrcoef: Hệ số tương quan
#Thời gian dành cho học bài
a_giohoc = np.array([4,7,1,2,8,0,3,8,6])
#Điểm thi nhận được:
b_diem   = np.array([7,9,3,4,9,0,5,10,8])
co = np.corrcoef(a_giohoc,b_diem)
print(type(co))
print('Hệ số tương quan: \n', co)
```

```
<class 'numpy.ndarray'>
Hệ số tương quan:
[[1.          0.96995403]
 [0.96995403 1.         ]]
```

Ví dụ về mối tương quan giữa **thời gian dành cho việc học bài** với **điểm thi nhận được!**

BIỂU ĐỒ THỂ HIỆN MỐI TƯƠNG QUAN GIỮA GIỜ HỌC BÀI VÀ ĐIỂM THI



THỰC HÀNH 3,4,5



Q & A
Thank you!

Bài 5:

LẬP TRÌNH PYTHON CƠ BẢN

(Sử dụng thư viện NumPy làm việc với ma trận - 02)

AI Academy Vietnam

1. Chuyển đổi Vector, Array

- Reshape, Ravel, Concatenate, Split, Hsplit, Vsplit, Flip

2. Tính toán trên mảng

- Arithmetic operators | Abs | Trigonometric | Exponents | logarithms | Round

3. Sắp xếp các phần tử trong mảng (sort)

- Sắp xếp Vector | Sắp xếp Matrix

4. Tìm kiếm các phần tử trong mảng (where)

5. Ma trận vuông

- det(A) | A⁻¹ | diagonal | Matrix below diagonal | trace of matrix | Eigen

6. Phép toán trên 2 ma trận

- np.equal | np.add | np.subtract | np.dot (@)

7. Rank(A), A.T



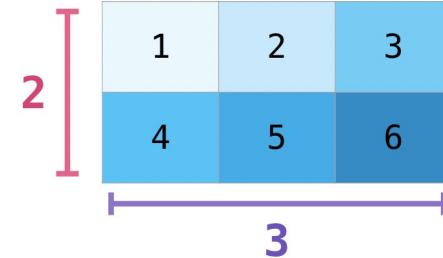
1. Kết hợp, chuyển đổi Vector, Ma trận

1.1 Reshape Arrays (1)

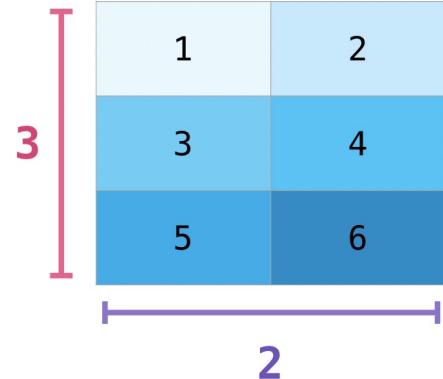
data

1
2
3
4
5
6

data.reshape(2,3)



data.reshape(3,2)



name of the
new array

```
new_array
```

name of the
original NumPy
array

```
old_array
```

a tuple of values
specifying the new
shape

```
.reshape((2, 6))
```

the reshape()
method, called with
"dot" notation

1.1 Reshape Arrays (2)

```
1 # Phương thức a.reshape(m,n)
2 vector_a = np.array([5,7,2,9,10,15,2,9,2,17,28,16],dtype=np.int16)
3 print(vector_a)
4 print('Số phần tử của vector:', vector_a.size)
5 print('-----')
6 #Chuyển đổi vector về matrix (n x m)
7 #Lưu ý: matrix.size =vector.size
8 matrix_a = vector_a.reshape((3,4))
9 print('Reshape về matrix: 3 x 4')
10 print(matrix_a)
11 print('Số phần tử của matrix_a:',matrix_a.size)
12 print('-----')
13 print('Reshape về matrix: 2 x 6')
14 matrix_b = vector_a.reshape((2,6))
15 print(matrix_b)
16 print('Số phần tử của matrix_b:',matrix_b.size)
```

```
[ 5  7  2  9 10 15  2  9  2 17 28 16]
```

```
Số phần tử của vector: 12
```

```
-----
```

```
Reshape về matrix: 3 x 4
```

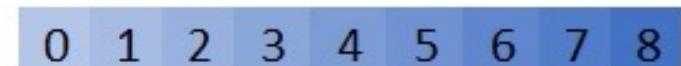
```
[[ 5  7  2  9]
 [10 15  2  9]
 [ 2 17 28 16]]
```

```
Số phần tử của matrix_a: 12
```

```
-----
```

```
Reshape về matrix: 2 x 6
```

```
[[ 5  7  2  9 10 15]
 [ 2  9  2 17 28 16]]
```

A horizontal array of 8 blue squares, each containing a number from 0 to 7 in sequence.

Reshape Vector to Matrix

0	1	2
3	4	5
6	7	8

1.1 Reshape Arrays (3)



VINBIGDATA VINGROUP



Academy
Vietnam

```
a1 = np.arange(1, 13)
```

1	2	3	4	5	6	7	8	9	10	11	12
---	---	---	---	---	---	---	---	---	----	----	----

Sắp xếp thứ tự các phần tử khi reshape array sử dụng thuộc tính
order ='C' | 'F'

→

1	2	3	4
5	6	7	8
9	10	11	12

↓

1	4	7	10
2	5	8	11
3	6	9	12

```
a1.reshape(3, 4) # reshapes or 'fills in' row by row  
a1.reshape(3, 4, order='C') # same results as above
```

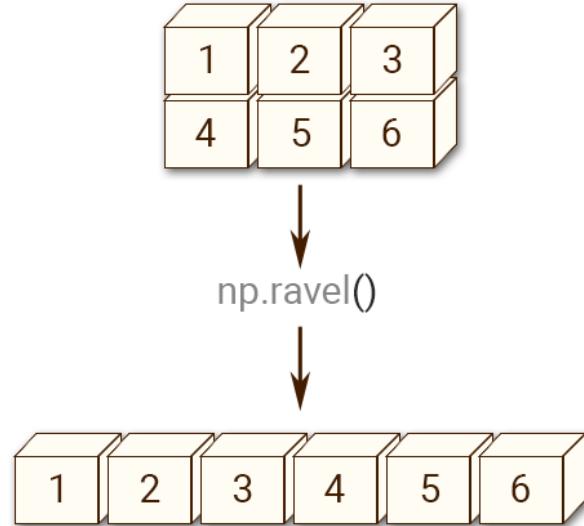
```
a1.reshape(3, 4, order='F') # reshapes column by column
```

1.2 Flatten | ravel Arrays



VINBIGDATA VINGROUP

Academy
Vietnam



```
1 #Chuyển đổi từ Matrix --> Vector
2
3 a1_2d = np.array([(1,2,3,4),(5,6,7,8),(9,10,11,12)])
4 print('Matrix: \n', a1_2d)
5
6 print('-----')
7 print('a) ravel by row (default order='c')')
8 print(a1_2d.ravel())
9
10 print('\n b) ravel by column (order='F')')
11 print(a1_2d.ravel(order='F'))
```

Matrix:

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

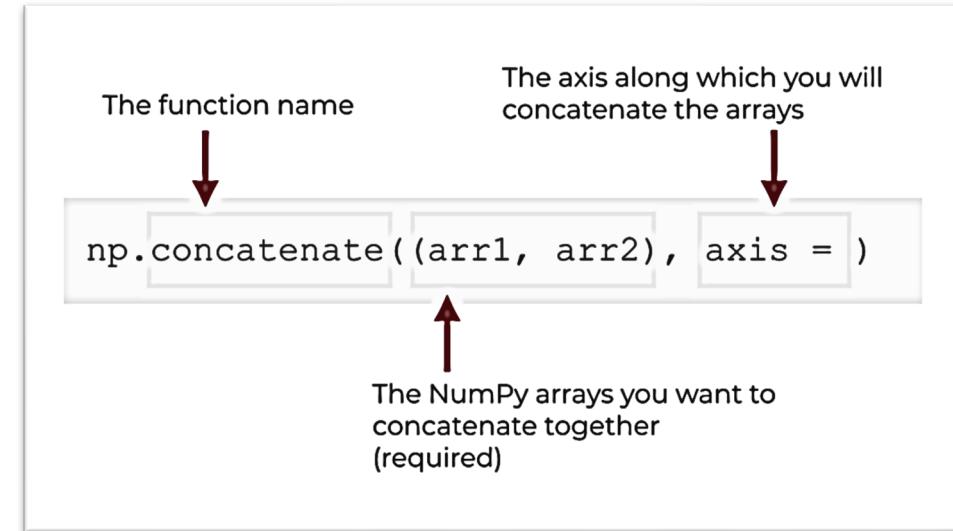
a) ravel by row (default order='C')

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

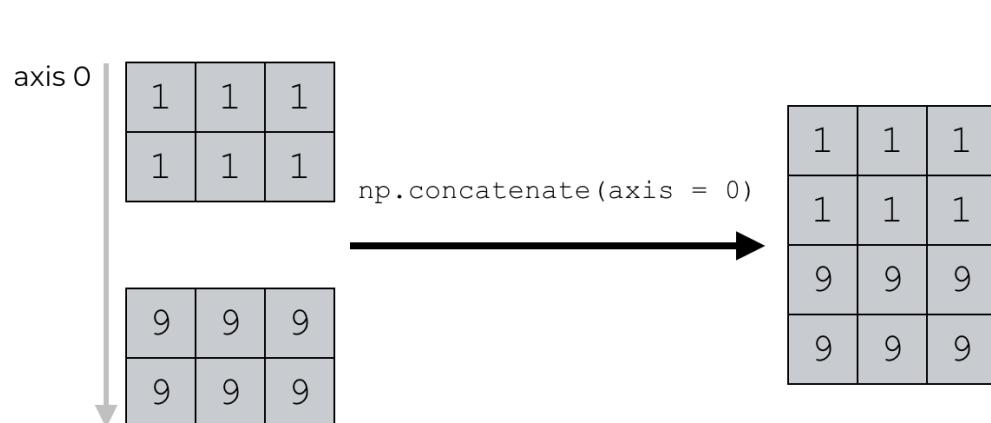
b) ravel by column (order='F')

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

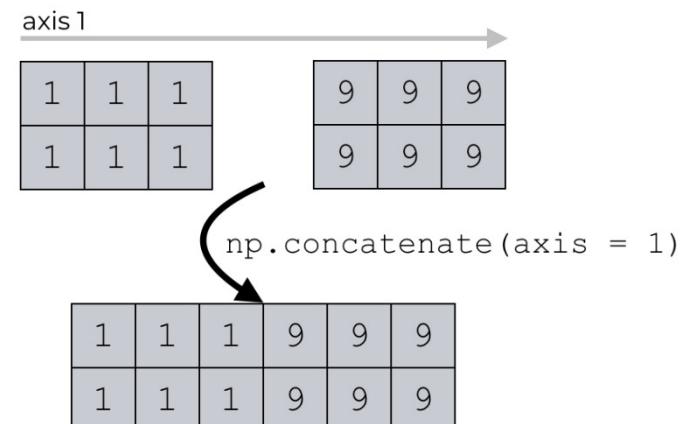
1.3 Concatenate Arrays (1)



Setting `axis=0` concatenates along the row axis



Setting `axis=1` concatenates along the column axis



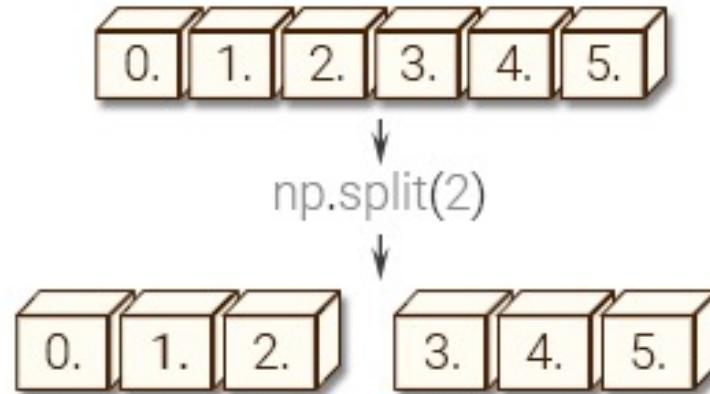
1.4 Split Arrays (1)



VINBIGDATA VINGROUP

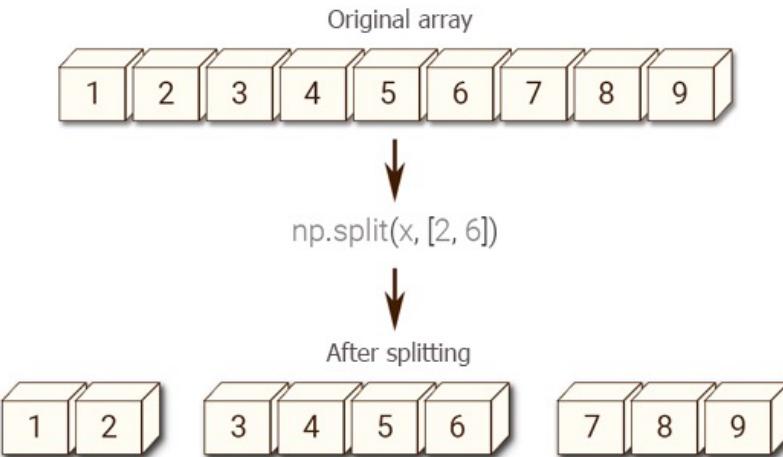
Academy Vietnam

Split: Tách một vector, ma trận thành các vector, ma trận con



```
1 import numpy as np
2 x = np.arange(0,6)
3 print(x)
4
5 #Tách vector x thành 2 vector
6 #có số phần tử bằng nhau
7 x1, x2 = np.split(x, 2)
8 print(x1, x2)
```

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

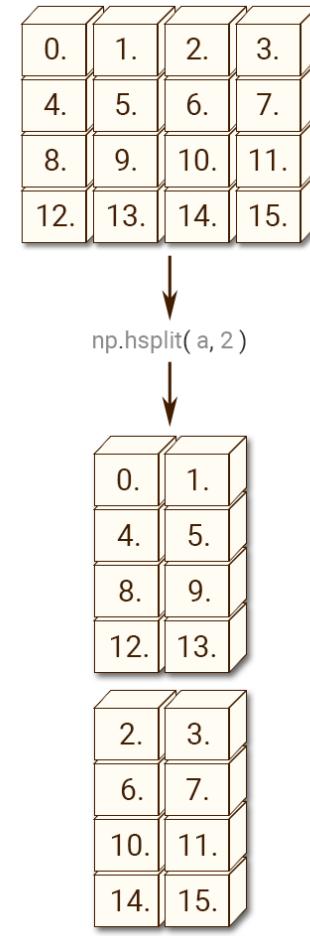
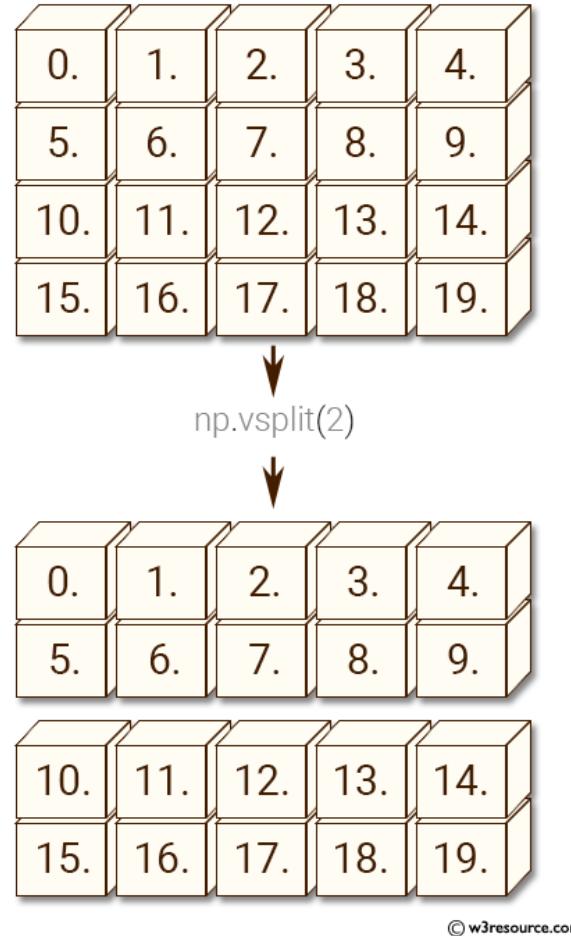


```
1 import numpy as np
2 x = np.arange(1,10)
3 print(x)
4
5 #Tách vector x thành 3 vector
6 #tại các vị trí 2 và 6
7 x1, x2, x3 = np.split(x, [2,6])
8 print(x1, x2, x3)
```

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

1.4 Split Arrays (2)

vsplit, hsplit: Tách một ma trận thành các ma trận con theo hàng, cột



1.5 Flip

Ma trận ban đầu:

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
```

- **np.flip (A,1) ~ np.fliplr(A): Lật ngược ma trận A theo cột.**

```
1 #Lật ma trận theo cột
2 A1 = np.flip(A,1)
3 #Tương đương với
4 A1 = np.fliplr(A)
5 print('Lật ma trận theo cột: \n',A1)
```

Lật ma trận theo cột:

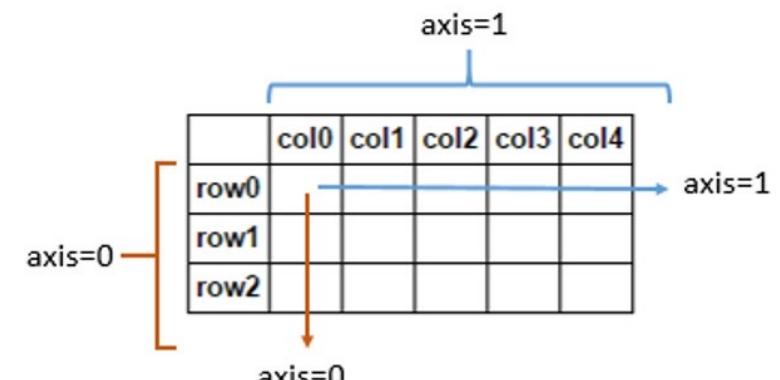
```
[[ 5  4  3  2  1]
 [10  9  8  7  6]
 [15 14 13 12 11]
 [20 19 18 17 16]
 [25 24 23 22 21]]
```

- **np.flip(A,0) ~ np.fipud(A): Lật ngược ma trận A theo hàng.**

```
1 #Lật ma trận theo hàng
2 A2 = np.flip(A,0)
3 #Tương đương với
4 A2 = np.fipud(A)
5 print('Lật ma trận theo hàng: \n',A2)
```

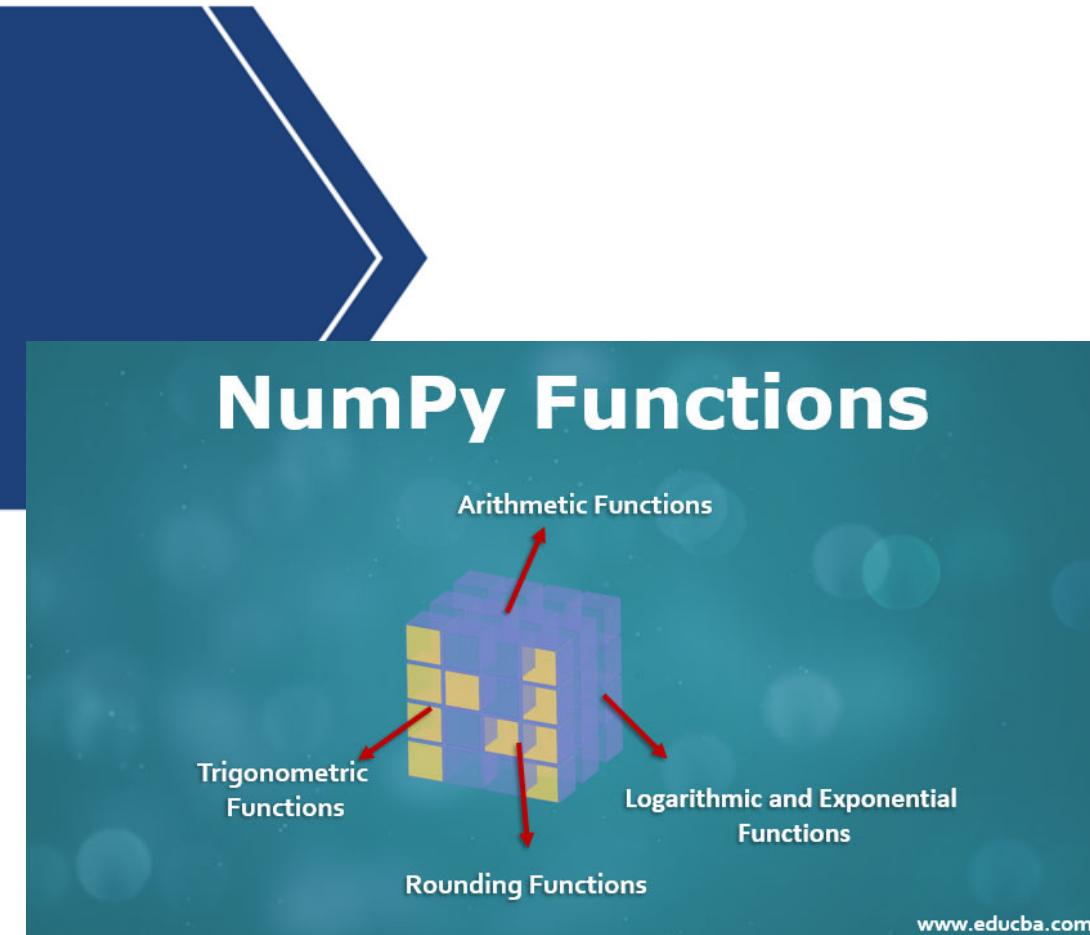
Lật ma trận theo hàng:

```
[[21 22 23 24 25]
 [16 17 18 19 20]
 [11 12 13 14 15]
 [ 6  7  8  9 10]
 [ 1  2  3  4  5]]
```



Thực hành 1

2. Tính toán với NumPy





2.1 Arithmetic operators.

The following table lists the arithmetic operators implemented in NumPy:

Operator	Equivalent ufunc	Description
+	<code>np.add</code>	Addition (e.g., <code>1 + 1 = 2</code>)
-	<code>np.subtract</code>	Subtraction (e.g., <code>3 - 2 = 1</code>)
-	<code>np.negative</code>	Unary negation (e.g., <code>-2</code>)
*	<code>np.multiply</code>	Multiplication (e.g., <code>2 * 3 = 6</code>)
/	<code>np.divide</code>	Division (e.g., <code>3 / 2 = 1.5</code>)
//	<code>np.floor_divide</code>	Floor division (e.g., <code>3 // 2 = 1</code>)
**	<code>np.power</code>	Exponentiation (e.g., <code>2 ** 3 = 8</code>)
%	<code>np.mod</code>	Modulus/remainder (e.g., <code>9 % 4 = 1</code>)



2.1 Arithmetic operators(2).

```
1 import numpy as np
2 x = np.arange(8)
3 print("x      =", x)
4 print('-----')
5 #Các phép toán đã giới thiệu trong buổi 01
6 print("x + 5  =", x + 5)
7 print("x - 5  =", x - 5)
8 print("-x     =", -x)
9 print("x * 2  =", x * 2)
10 print("x / 2   =", x / 2)
11 print("x // 2  =", x // 2)
12 print("x % 2   =", x % 2)
13 print("x ^ 3   =", x**3)
```

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

```
x + 5  = [ 5  6  7  8  9 10 11 12]
x - 5  = [-5 -4 -3 -2 -1  0  1  2]
-x     = [ 0 -1 -2 -3 -4 -5 -6 -7]
x * 2  = [ 0  2  4  6  8 10 12 14]
x / 2   = [0.  0.5 1.  1.5 2.  2.5 3.  3.5]
x // 2  = [0  0  1  1  2  2  3  3]
x % 2   = [0  1  0  1  0  1  0  1]
x ^ 3   = [ 0    1    8   27   64  125  216  343]
```

```
1 print("x      =", x)
2 print('-----')
3 #Sử dụng các phương thức của NumPy
4 print("x + 5  =", np.add(x,5))
5 print("x - 5  =", np.subtract(x, 5))
6 print("-x     =", np.negative(x))
7 print("x * 2  =", np.multiply(x,2))
8 print("x / 2   =", np.divide(x,2))
9 print("x // 2  =", np.floor_divide(x, 2))
10 print("x % 2   =", np.mod(x,2))
11 print("x ^ 3   =", np.power(x,3))
```

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

```
x + 5  = [ 5  6  7  8  9 10 11 12]
x - 5  = [-5 -4 -3 -2 -1  0  1  2]
-x     = [ 0 -1 -2 -3 -4 -5 -6 -7]
x * 2  = [ 0  2  4  6  8 10 12 14]
x / 2   = [0.  0.5 1.  1.5 2.  2.5 3.  3.5]
x // 2  = [0  0  1  1  2  2  3  3]
x % 2   = [0  1  0  1  0  1  0  1]
x ^ 3   = [ 0    1    8   27   64  125  216  343]
```

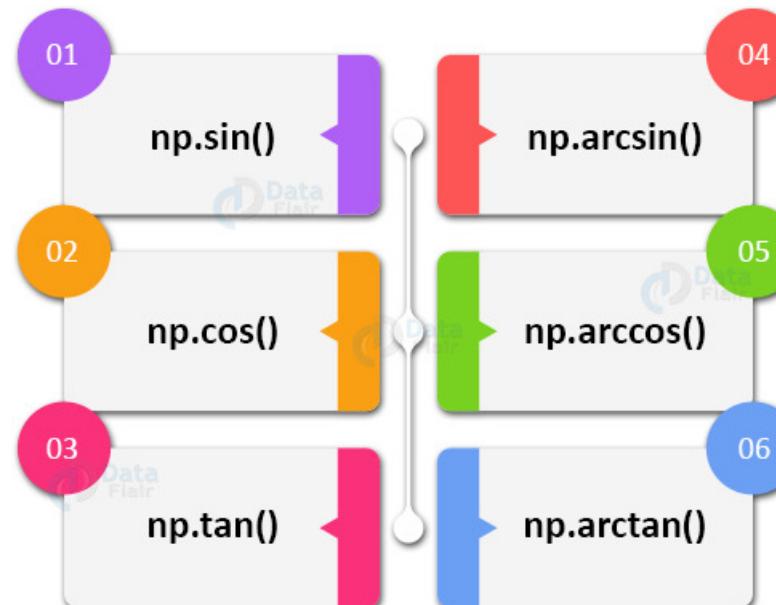
2.2 Abs | Trigonometric functions

- **np.abs() | np.absolute():** để tính giá trị tuyệt đối của các phần tử.

```
1 x = np.array([-2, -1, 0, 1, 2])
2 print(x)
3 print('-----')
4 print(np.abs(x))
5 print(np.absolute(x))
```

```
[ -2 -1  0  1  2]
-----
[ 2  1  0  1  2]
[ 2  1  0  1  2]
```

Trigonometric Functions



```
1 theta = np.linspace(0, np.pi, 3)
2 print("theta      = ", theta)
3 print('-----')
4 print("sin(theta) = ", np.sin(theta))
5 print("cos(theta) = ", np.cos(theta))
6 print("tan(theta) = ", np.tan(theta))
7
```

```
theta      = [0.           1.57079633  3.14159265]
```

```
-----
```

```
sin(theta) = [ 0.000000e+00  1.000000e+00  1.2246468e-16]
cos(theta) = [ 1.000000e+00  6.123234e-17 -1.000000e+00]
tan(theta) = [ 0.000000e+00  1.63312394e+16 -1.22464680e-16]
```

2.3 Exponents and logarithms



Function	Description
exp(arr)	Returns exponential of an input array element wise
expm1(arr)	Returns exponential $\exp(x)-1$ of an input array element wise
exp2(arr)	Returns exponential $2^{**}x$ of all elements in an array
log(arr)	Returns natural log of an input array element wise
log10(arr)	Returns log base 10 of an input array element wise
log2(arr)	Returns log base 2 of an input array element wise
logaddexp(arr)	Returns logarithm of the sum of exponentiations of all inputs
logaddexp2(arr)	Returns logarithm of the sum of exponentiations of the inputs in base 2

2.3 Exponents and logarithms (2)

```

1 x = np.array([1, 2, 3])
2
3 print("x      =", x)
4 print('-----')
5 print("e^x    =", np.exp(x))
6 print("2^x    =", np.exp2(x))
7 print("3^x    =", np.power(3, x))

```

```

x      = [1 2 3]
-----
e^x    = [ 2.71828183  7.3890561  20.08553692]
2^x    = [2. 4. 8.]
3^x    = [ 3  9 27]

```

```

1 x = np.array([1, 2, 4, 100])
2
3 print("x      =", x)
4 print('-----')
5 print("ln(x)   =", np.log(x))
6 print("log2(x)  =", np.log2(x))
7 print("log10(x) =", np.log10(x))

```

```

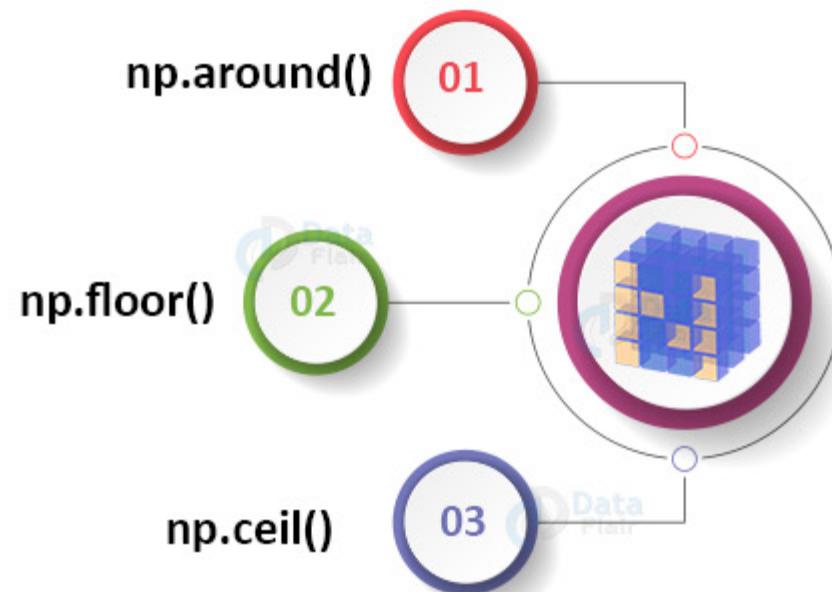
x      = [ 1   2   4 100]
-----
ln(x)   = [0.          0.69314718  1.38629436  4.60517019]
log2(x)  = [0.          1.          2.          6.64385619]
log10(x) = [0.          0.30103    0.60205999  2.          ]

```

2.4 Rounding Functions



Rounding Functions in NumPy



```
1 import numpy as np
2 arr = np.array([20.8999, 67.89899, 54.43409])
3
4 print(arr)
5 print('-----')
6 #1) Làm tròn tới 1 số sau dấu ,
7 print(np.around(arr,1))
8
9 #2)Làm tròn tới 2 số sau dấu ,
10 print(np.around(arr,2))
11
12 #3)Làm tròn xuống số nguyên gần nhất
13 print(np.floor(arr))
14
15 #4)Làm tròn Lên số nguyên gần nhất
16 print(np.ceil(arr))
```

[20.8999 67.89899 54.43409]

[20.9 67.9 54.4]

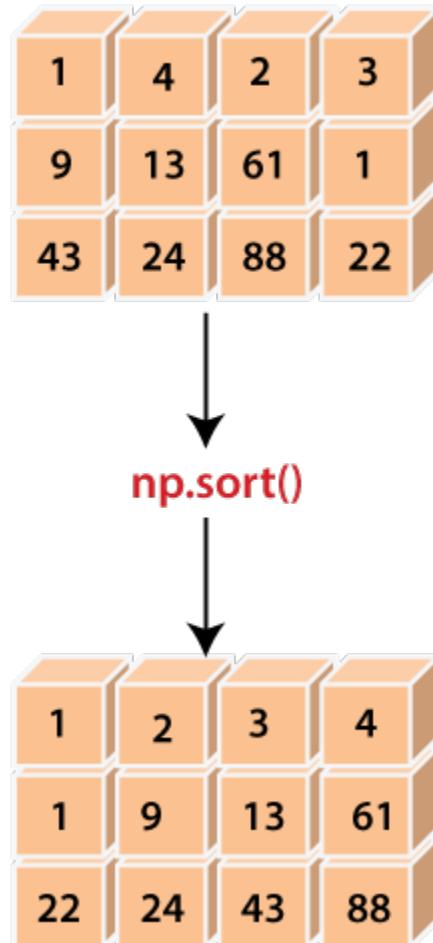
[20.9 67.9 54.43]

[20. 67. 54.]

[21. 68. 55.]

3. Sắp xếp mảng (np.sort)

3. Sắp xếp các phần tử trong mảng (1)



The function name

`np.sort(a= , axis= , kind=)`

The array you want to operate on

The sorting algorithm you want to use to sort the data

`np.sort(a= , axis= , kind=)`

The axis along which you will sort the array

* kind='quicksort' - Default, 'mergesort', 'heapsort', 'stable'

3. Sắp xếp các phần tử trong mảng (2)



VINBIGDATA VINGROUP



Academy
Vietnam

Sắp xếp các phần tử trong một Vector

5	3	1	2	4
---	---	---	---	---



np.sort() SORTS THE
VALUES OF A NUMPY ARRAY

1	2	3	4	5
---	---	---	---	---

```
1 #Sắp xếp các phần tử trong một vector
2 a = np.random.randint(1,33,15)
3
4 print('Vector ban đầu:\n', a)
5 print('-----')
6 #Sắp xếp vector a tăng dần
7 a_sort = np.sort(a)
8
9 #Sắp xếp vector a giảm dần:
10 #1) Lật vector a_sort để sắp xếp giảm dần
11 b_sort = np.flip(a_sort)
12 #2) sử dụng -np.sort(-x)
13 b_sort = -np.sort(-a)
14 print('Vector sắp xếp tăng dần: \n',a_sort)
15
16 print('Vector sắp xếp giảm dần: \n',b_sort)
```

Vector ban đầu:

[1 17 13 15 9 9 23 30 32 10 30 16 4 16 24]

Vector sắp xếp tăng dần:

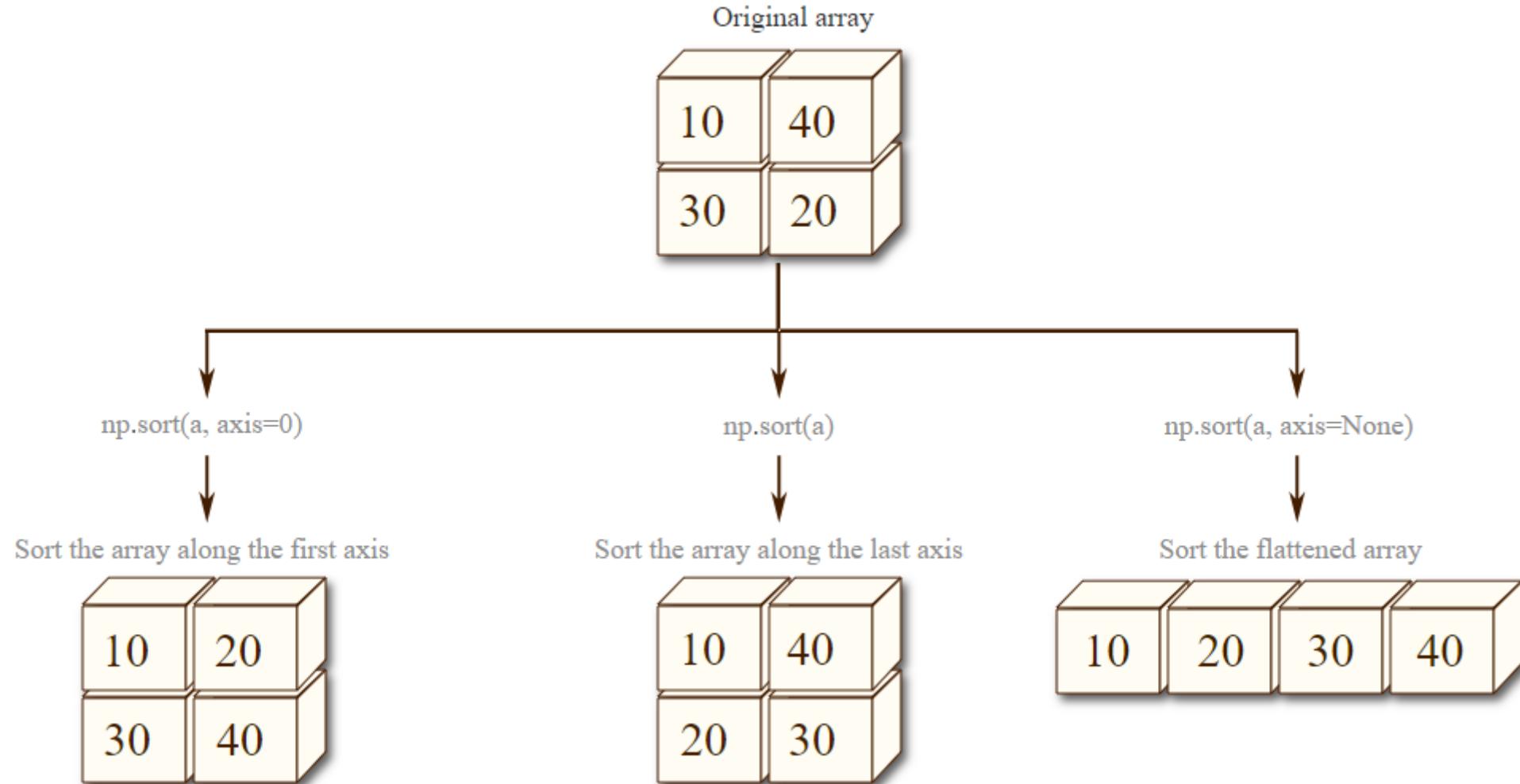
[1 4 9 9 10 13 15 16 16 17 23 24 30 30 32]

Vector sắp xếp giảm dần:

[32 30 30 24 23 17 16 16 15 13 10 9 9 4 1]

3. Sắp xếp các phần tử trong mảng (3)

Sắp xếp các phần tử trong một Ma trận:



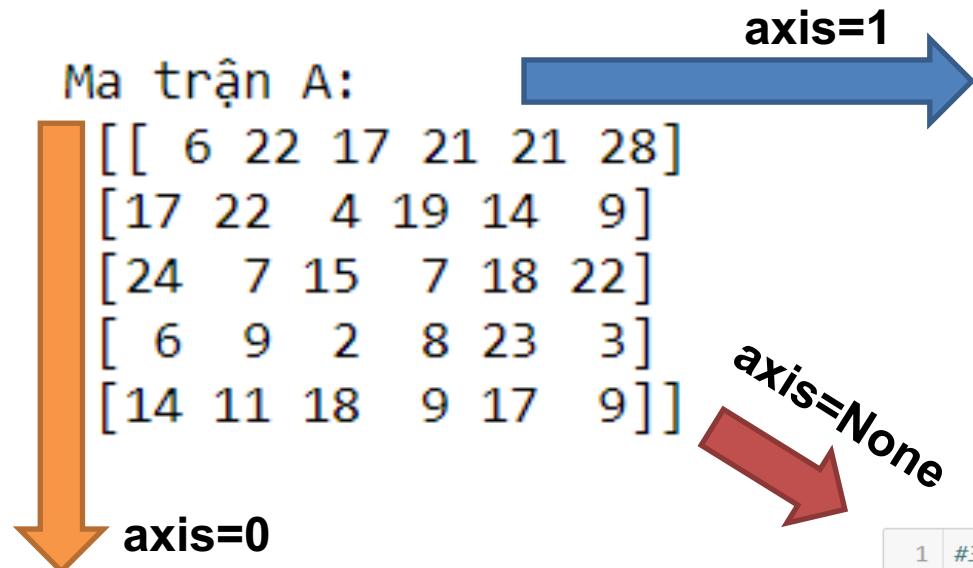
3. Sắp xếp các phần tử trong mảng (4)



VINBIGDATA VINGROUP

Academy
Vietnam

Sắp xếp các phần tử trong một Ma trận:



```
1 #1) Sắp xếp theo cột axis=0
2 a_sort1 = np.sort(A, axis=0)
3 print('Ma trận 1:\n', a_sort1)
```

Ma trận 1:

```
[[ 6  7  2  7 14  3]
 [ 6  9  4  8 17  9]
 [14 11 15  9 18  9]
 [17 22 17 19 21 22]
 [24 22 18 21 23 28]]
```

```
1 #Sắp xếp các phần tử của ma trận A
2 #2) Sắp xếp theo hàng axis=1 | Default
3 a_sort2 = np.sort(A, axis=1)
4 print('Ma trận 2:\n', a_sort2)
```

Ma trận 2:

```
[[ 6 17 21 21 22 28]
 [ 4  9 14 17 19 22]
 [ 7  7 15 18 22 24]
 [ 2  3  6  8  9 23]
 [ 9  9 11 14 17 18]]
```

```
1 #3) Chuyển thành vector và sắp xếp các phần tử tăng dần theo hàng
2 v_sort = np.sort(A, axis=None)
3 print('Vector: \n', v_sort)
4
5 #Sắp xếp tất cả các phần tử theo thứ tự tăng dần theo hàng
6 a_sort3 = np.sort(A, axis=None).reshape(A.shape[0],A.shape[1])
7 print('Ma trận 3:\n', a_sort3)
```

Vector:

```
[ 2  3  4  6  6  7  7  8  9  9  9  9 11 14 14 15 17 17 17 18 18 19 21 21
22 22 22 23 24 28]
```

Ma trận 3:

```
[[ 2  3  4  6  6  7]
 [ 7  8  9  9  9  9]
 [11 14 14 15 17 17]
 [17 18 18 19 21 21]
 [22 22 22 23 24 28]]
```

4. Tìm kiếm (np.where)

4. Tìm kiếm trong mảng: np.where (1)

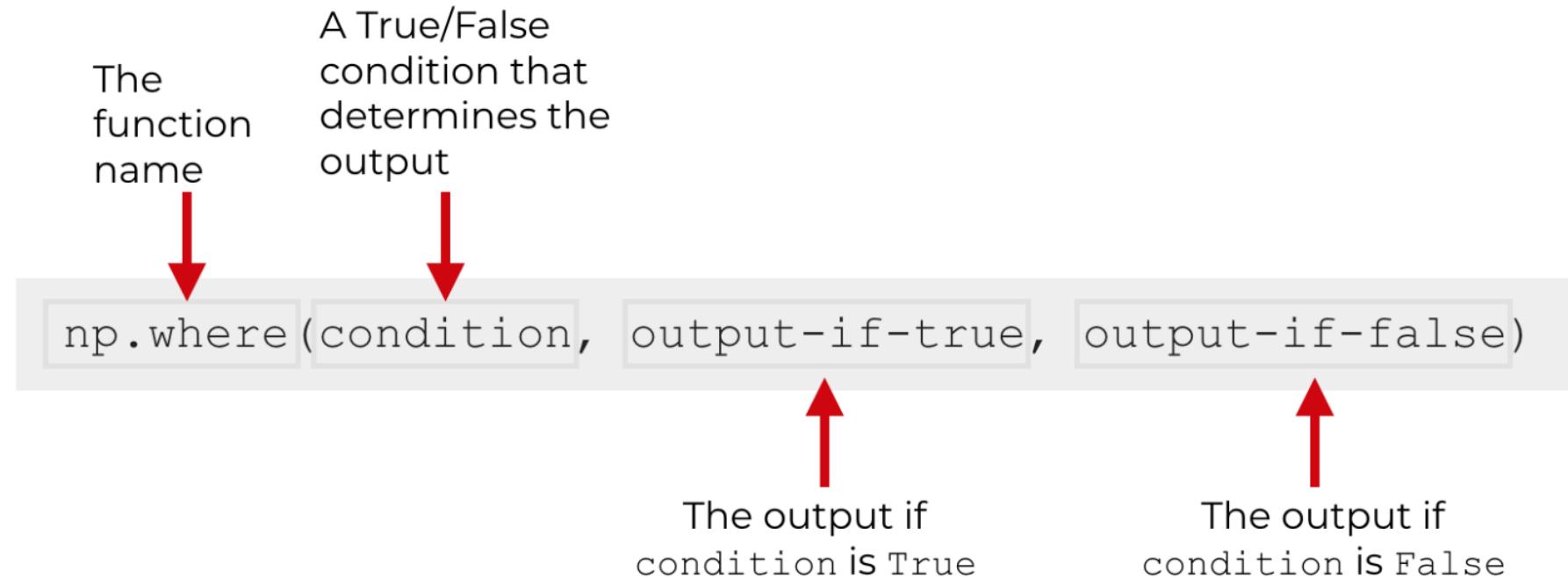


VINBIGDATA VINGROUP



Academy
Vietnam

np.where(): Tìm kiếm một phần tử trong mảng theo điều kiện



4. Tìm kiếm trong mảng: np.where (2)



VINBIGDATA VINGROUP



Academy
Vietnam

Vd1: Tìm kiếm trên vector

```
1 import numpy as np
2 x = np.array([17, 2, 11, 1, 9, 15, 1, 3, 8, 1, 12, 13, 5])
3 #1) Tìm kiếm các phần tử có giá trị ==1
4 t1 = np.where(x==1)
5 print(t1)
6 print('1. Số phần tử thỏa mãn điều kiện = 1: ', t1[0].size)
7 print('-----')
8 #2) Tìm kiếm các phần tử có giá trị >10
9 t2 = np.where(x>10)
10 print(t2)
11 print('2. Số phần tử thỏa mãn điều kiện>10: ', t2[0].size)
12 print('-----')
13 #Tìm kiếm các phần tử có giá trị [5,12)
14 t3 = np.where((x>=5) & (x<12))
15 print(t3)
16 print('3.Số phần tử thỏa mãn điều kiện [5,10): ', t3[0].size)
17
```

(array([3, 6, 9], dtype=int64),)

1. Số phần tử thỏa mãn điều kiện = 1: 3

(array([0, 2, 5, 10, 11], dtype=int64),)

2. Số phần tử thỏa mãn điều kiện>10: 5

(array([2, 4, 8, 12], dtype=int64),)

3.Số phần tử thỏa mãn điều kiện [5,10): 4



4. Tìm kiếm trong mảng: np.where (2)

Vd1: Tìm kiếm trên ma trận:

```
1 import numpy as np
2 #Tìm kiếm trên ma trận
3 arr = np.array([(1, 2, 3, 4, 5, 4, 4),
4                 (7, 3, 4, 8, 9, 6, 7)])
5 #Tìm kiếm phần tử > 4
6 x = np.where(arr >4)
7
8 print('Ma trận A: \n',arr)
9 print('-----')
10 print(x)
11 print('Số phần tử thỏa mãn điều kiện > 4:',x[0].size)
```

Ma trận A:

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

```
-----
(array([0, 1, 1, 1, 1, 1], dtype=int64), array([4, 0, 3, 4, 5, 6], dtype=int64))
Số phần tử thỏa mãn điều kiện > 4: 6
```

Thực hành 2

5. Ma trận vuông

5.1 Định thức det (A)

Định thức của ma trận vuông cấp n là tổng đại số của $n!$ (n giai thừa) số hạng, mỗi số hạng là tích của n phần tử lấy trên các hàng và các cột khác nhau của ma trận A, mỗi tích được nhân với phần tử dấu là +1 hoặc -1 theo phép thế tạo bởi các chỉ số hàng và chỉ số cột của các phần tử trong tích

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^n a_{i,\sigma(i)}$$

Định thức của một ma trận vuông còn được viết như sau

$$\det A = \begin{vmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,n} \end{vmatrix}$$

Áp dụng với các ma trận vuông cấp 1,2,3 ta có:

$$\det [a] = a$$

$$\det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

$$\det \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$$

5.1 Định thức det(A)

np.linalg.det(a): tính định thức của ma trận vuông a

```

1 import numpy as np
2 a = np.array([( 1, 3, 1, 4),
3                 ( 3, 9, 5, 15),
4                 ( 0, 2, 1, 1),
5                 ( 0, 4, 2, 3)])
6 print('Ma trận a:\n',a)
7 det_a = np.linalg.det(a)
8 print('det(a) = ', det_a)

```

Ma trận a:

```

[[ 1  3  1  4]
 [ 3  9  5 15]
 [ 0  2  1  1]
 [ 0  4  2  3]]

```

det(a) = -3.999999999999999

```

1 import numpy as np
2 b = np.array([( 1, 2, 3, 4),
3                 (-2,-1, 4, 1),
4                 ( 3,-4,-5, 6),
5                 ( 1, 2, 3, 4)])
6 print('Ma trận b:\n',b)
7 det_b = np.linalg.det(b)
8 print('det(b) = ', det_b)

```

Ma trận b:

```

[[ 1  2  3  4]
 [-2 -1  4  1]
 [ 3 -4 -5  6]
 [ 1  2  3  4]]

```

det(b) = 0.0

5.2 Ma trận nghịch đảo



VINBIGDATA VINGROUP



Academy
Vietnam

Ma trận nghịch đảo của ma trận vuông M ký hiệu M^{-1}

$M * M^{-1} = I$ (ma trận đơn vị)

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 5 & 6 & 0 \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} -24 & 18 & 5 \\ 20 & -15 & -4 \\ -5 & 4 & 1 \end{bmatrix}$$

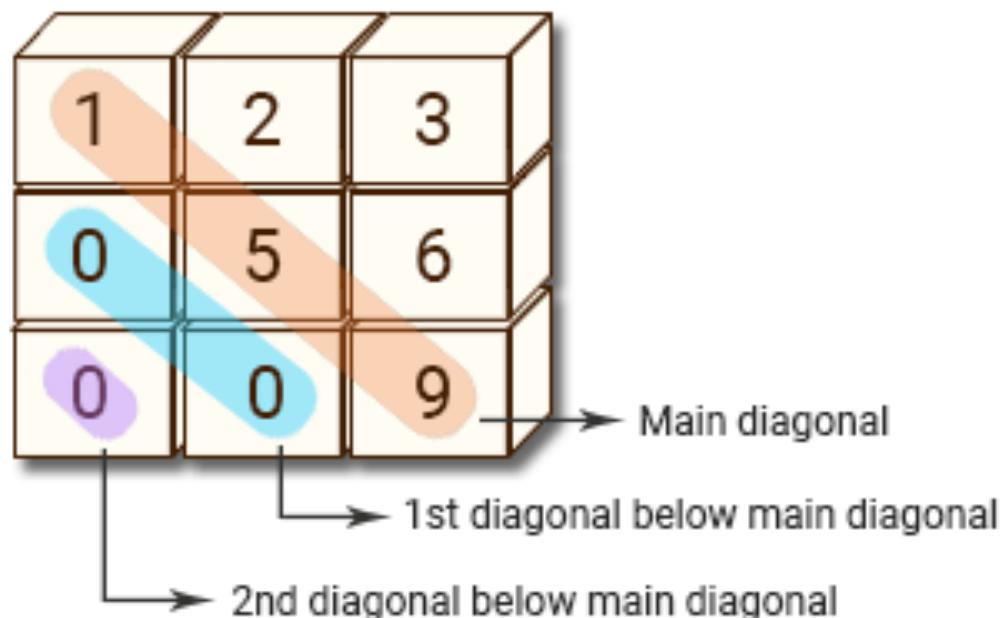
`np.linalg.inv(m)`: Tìm ma trận nghịch đảo của ma trận m

`det(m) = 0`: Không tồn tại ma trận nghịch đảo

a) Lấy phần tử trên đường chéo

a.**diagonal()**: trả về vector chứa các phần tử nằm trên đường chéo chính của ma trận a.

a.**diagonal(k)**: trả về vector chứa các phần tử nằm cách đường chéo chính của ma trận a, k phần tử ($k > 0$ trên đường chéo chính, $k < 0$ dưới đường chéo chính)





5.3 Đường chéo (2)

```
[79 98 60 84 47 28 10 48 83 43]
[59 55 80 82 30 52 70 56 77 91]
[ 6 15 17 62 21 64 89 31 69  1]
[20 50   6 77 62 10 96 54 51 89]
[72 16 56 62 77 30 23   3 77  4]
[73 68 71 70 80 20 78 70 90 58]
[ 7 48 14 78 26 99 69 13 91 21]
[ 4 35 57 20 31   3 73 16 25 14]
[89 73 27 32   2 83 71 55 17 34]
[95 15 67 75 86 47 36 96 72 92]
```

```
1 #Lấy các phần tử nằm trên đường chéo chính
2 #của ma trận a 1 phần tử
3 d_A1 = A.diagonal(1)
4 print(d_A1)
```

```
[98 80 62 62 30 78 13 25 34]
```

```
1 #Lấy các phần tử trên đường chéo chính của
2 #ma trận vuông A
3 d_A = A.diagonal()
4 print(d_A)
```

```
[79 55 17 77 77 20 69 16 17 92]
```

```
1 #Lấy các phần tử nằm dưới đường chéo chính
2 #của ma trận a 4 phần tử
3 d_A1 = A.diagonal(-4)
4 print(d_A1)
```

```
[72 68 14 20   2 47]
```



5.3 Đường chéo (3)

b) Ma trận tam giác (triu | tril)

`np.triu(m) | np.tril(m)`: trả về ma trận tam giác trên | dưới của ma trận m.

`np.triu(m, k)`: trả về ma trận trên của ma trận m cách đường chéo chính k phần tử (k = 0 (default), k<0 bên dưới đường chéo chính, k>0 bên trên đường chéo chính)

79	98	60	84	47	28	10	48	83	43
59	55	80	82	30	52	70	56	77	91
6	15	17	62	21	64	89	31	69	1
20	50	6	77	62	10	96	54	51	89
72	16	56	62	77	30	23	3	77	4
73	68	71	70	80	20	78	70	90	58
7	48	14	78	26	99	69	13	91	21
4	35	57	20	31	3	73	16	25	14
89	73	27	32	2	83	71	55	17	34
95	15	67	75	86	47	36	96	72	92

```
1 #Ma trận tam giác trên của
2 #ma trận A
3 d_A1 = np.triu(A)
4 print(d_A1)
```

```
[[ 79  98  60  84  47  28  10  48  83  43]
 [  0  55  80  82  30  52  70  56  77  91]
 [  0  0  17  62  21  64  89  31  69  1]
 [  0  0  0  77  62  10  96  54  51  89]
 [  0  0  0  0  77  30  23  3  77  4]
 [  0  0  0  0  0  20  78  70  90  58]
 [  0  0  0  0  0  0  69  13  91  21]
 [  0  0  0  0  0  0  0  16  25  14]
 [  0  0  0  0  0  0  0  0  17  34]
 [  0  0  0  0  0  0  0  0  0  92]]
```



5.3 Đường chéo (4)

```
1 #Tạo ma trận là các phần tử trên đường chéo chính
2 #của ma trận vuông A cách đường chéo chính
3 #về phía trên 2 đường
4 d_A1 = np.triu(A,2)
5 print(d_A1)
```

```
[[ 0  0  60  84  47  28  10  48  83  43]
 [ 0  0  0  82  30  52  70  56  77  91]
 [ 0  0  0  0  21  64  89  31  69  1]
 [ 0  0  0  0  0  10  96  54  51  89]
 [ 0  0  0  0  0  0  23  3  77  4]
 [ 0  0  0  0  0  0  0  70  90  58]
 [ 0  0  0  0  0  0  0  0  91  21]
 [ 0  0  0  0  0  0  0  0  0  14]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]]
```

```
1 #Tạo ma trận là các phần tử trên đường chéo chính
2 #của ma trận vuông A cách đường chéo chính
3 #về phía dưới 3 đường
4 d_A1 = np.triu(A,-3)
5 print(d_A1)
```

```
[[79 98 60 84 47 28 10 48 83 43]
 [59 55 80 82 30 52 70 56 77 91]
 [ 6 15 17 62 21 64 89 31 69 1]
 [20 50  6 77 62 10 96 54 51 89]
 [ 0 16 56 62 77 30 23 3 77 4]
 [ 0  0 71 70 80 20 78 70 90 58]
 [ 0  0  0 78 26 99 69 13 91 21]
 [ 0  0  0  0 31  3 73 16 25 14]
 [ 0  0  0  0  0 83 71 55 17 34]
 [ 0  0  0  0  0  0 36 96 72 92]]
```



5.3 Đường chéo (5)

c) Vết của ma trận

Trace of a Matrix

Suppose $T \in \mathcal{L}(V)$, $\mathbf{F} = \mathbf{C}$, and we choose a basis of V corresponding to the Decomposition Theorem. Then trace T equals the sum of the diagonal entries of that matrix.

Definition: **trace of a matrix**

The *trace* of a square matrix A , denoted $\text{trace } A$, is defined to be the sum of the diagonal entries of A .

Example: Suppose

$$A = \begin{pmatrix} 3 & -1 & -2 \\ 3 & 2 & -3 \\ 1 & 2 & 0 \end{pmatrix}.$$

Then

$$\begin{aligned} \text{trace } A &= 3 + 2 + 0 \\ &= 5. \end{aligned}$$

Trace of AB equals trace of BA

If A and B are square matrices of the same size, then

$$\text{trace}(AB) = \text{trace}(BA).$$

```
1 #Tính trace của ma trận vuông A
2 trace_A = A.trace()
3 print('Trace of Matrix A: ',trace_A)
```

Trace of Matrix A: 519

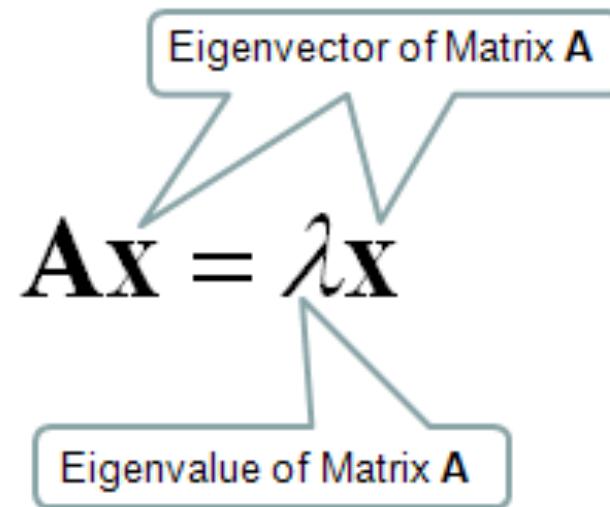
```
1 #Cách 2: Tính trace của ma trận vuông A
2 trace_A = A.diagonal().sum()
3 print('Trace of Matrix A: ',trace_A)
```

Trace of Matrix A: 519



d) Vector riêng(Eigenvector), giá trị riêng (Eigenvalue)

Một số λ và một vector khác 0 x thỏa mãn



được gọi lần lượt là *giá trị riêng* và *vector riêng* của A

```
1 #Tìm giá trị riêng, vector riêng của vector A
2 A = np.array([(3, 4, -2),
   (1, 4, -1),
   (2, 6, -1)])
3
4 eigenvalues, eigenvectors = np.linalg.eig(A)
5
6 print('Eigenvalues: ', eigenvalues)
7 print('Eigenvectors: \n', eigenvectors)
```

Eigenvalues: [3. 2. 1.]

Eigenvectors:

```
[[ -4.08248290e-01  2.67561446e-15  7.07106781e-01]
 [-4.08248290e-01  4.47213595e-01 -3.33066907e-16]
 [-8.16496581e-01  8.94427191e-01  7.07106781e-01]]
```

Thực hành 3



6.Phép toán với 2 ma trận

10. Phép toán trên 2 ma trận



VINBIGDATA
VINGROUP



Academy
Vietnam

$$\begin{matrix} \textcolor{orange}{\square} & \textcolor{orange}{\square} \\ \vdots & \vdots \\ \textcolor{orange}{\square} & \textcolor{orange}{\square} \end{matrix} = \begin{matrix} \textcolor{orange}{\square} & \textcolor{orange}{\square} \\ \vdots & \vdots \\ \textcolor{orange}{\square} & \textcolor{orange}{\square} \end{matrix}$$

Equality of
Matrices

Addition of
Matrices

$$\begin{matrix} \textcolor{red}{\square} & \textcolor{red}{\square} \\ \vdots & \vdots \\ \textcolor{red}{\square} & \textcolor{red}{\square} \end{matrix} + \begin{matrix} \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \vdots & \vdots \\ \textcolor{blue}{\square} & \textcolor{blue}{\square} \end{matrix} = \begin{matrix} \textcolor{purple}{\square} & \textcolor{purple}{\square} \\ \vdots & \vdots \\ \textcolor{purple}{\square} & \textcolor{purple}{\square} \end{matrix}$$

$$\begin{matrix} \textcolor{purple}{\square} & \textcolor{purple}{\square} \\ \vdots & \vdots \\ \textcolor{purple}{\square} & \textcolor{purple}{\square} \end{matrix} - \begin{matrix} \textcolor{red}{\square} & \textcolor{red}{\square} \\ \vdots & \vdots \\ \textcolor{red}{\square} & \textcolor{red}{\square} \end{matrix} = \begin{matrix} \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \vdots & \vdots \\ \textcolor{blue}{\square} & \textcolor{blue}{\square} \end{matrix}$$

Subtraction of
Matrices

Multiplication of Matrices

$$\begin{pmatrix} 1 & 1 \end{pmatrix} + \begin{pmatrix} 2 & 2 \end{pmatrix} + \begin{pmatrix} 3 & 3 \end{pmatrix} = \begin{pmatrix} 1 \end{pmatrix}$$

codeforme.com

$$\begin{matrix} 1 & 2 & 3 \\ \hline 1 & 2 & 3 \end{matrix} \times \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} = \begin{matrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{matrix}$$

$$\textcolor{gray}{\square} \times \begin{matrix} \textcolor{green}{\square} & \textcolor{green}{\square} \\ \vdots & \vdots \\ \textcolor{green}{\square} & \textcolor{green}{\square} \end{matrix} = \begin{matrix} \textcolor{green}{\square} & \textcolor{green}{\square} \\ \vdots & \vdots \\ \textcolor{green}{\square} & \textcolor{green}{\square} \end{matrix}$$

Matrix
Multiplied by
a Scalar

Power of a
Matrix

$$\left(\begin{matrix} \textcolor{green}{\square} & \textcolor{green}{\square} \\ \vdots & \vdots \\ \textcolor{green}{\square} & \textcolor{green}{\square} \end{matrix} \right)^{\textcolor{red}{\square}} = \begin{matrix} \textcolor{darkgreen}{\square} & \textcolor{darkgreen}{\square} \\ \vdots & \vdots \\ \textcolor{darkgreen}{\square} & \textcolor{darkgreen}{\square} \end{matrix}$$



6. Phép toán trên 2 ma trận

a) So sánh 2 ma trận

`np.equal(a,b) | ==:` trả về ma trận (T|F) so sánh từng phần tử của ma trận a và ma trận b theo vị trí.

Lưu ý: ma trận a, b có cùng kích thước (m,n)

Matrix a:

```
[[ 9  4 19  1 18]
 [15 11  1  9 14]
 [17  8  4 10 13]]
```

Matrix b:

```
[[ 6  4  9 12  4]
 [ 3  6 11 14 10]
 [ 1  6  5 12  2]]
```

```
1 #1) So sánh 2 ma trận
2 equal_ab = np.equal(a,b)
3 #hoặc equal_ab = a==b
4
5 print(equal_ab)
```

```
[[False True False False False]
 [False False False False False]
 [False False False False False]]
```



6. Phép toán trên 2 ma trận

b) Cộng, trừ 2 ma trận

np.add(a,b) | +: trả về ma trận có các phần tử là tổng của phần tử của ma trận a và ma trận b.

np.subtract(a,b) | - : trả về ma trận có các phần tử là hiệu của phần tử ma trận a và ma trận b

Lưu ý: ma trận a, b có cùng kích thước (m,n)

```
1 #Phép cộng 2 ma trận
2 sum_ab = np.add(a,b)
3 #hoặc sum_ab = a + b
4 print (sum_ab)
```

```
[[15  8 28 13 22]
 [18 17 12 23 24]
 [18 14  9 22 15]]
```

```
1 #Phép trừ 2 ma trận
2 sub_ab = np.subtract(a,b)
3 #hoặc sub_ab = a - b
4 print (sub_ab)
```

```
[[ 3   0  10 -11  14]
 [ 12  5 -10  -5  4]
 [ 16  2  -1  -2 11]]
```

6. Phép toán trên 2 ma trận

c) Nhân 2 ma trận

`np.dot(a,b)` | `@`: trả về ma trận kết quả là tích của 2 ma trận a,b

Lưu ý: ma trận a có kích thước (m,n)
ma trận c có kích thước (n,k)
ma trận ac có kích thước (m,k)

```
1 #Tích của 2 vector:  
2 vector_a = np.random.randint(1,20,10)  
3 vector_b = np.random.randint(1,20,10)  
4 #Thực hiện tính tích của 2 vector  
5 #Kết quả trả về một số  
6 vector_ab = vector_a @ vector_b  
7  
8 print('Vector a:\n',vector_a)  
9 print('Vector b:\n',vector_b)  
10 print('Tích của hai vector:\n',vector_ab)
```

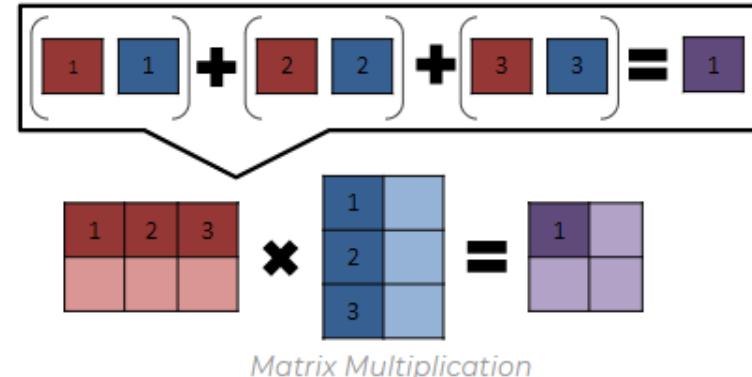
Vector a:
[7 15 18 6 3 13 8 11 2 4]

Vector b:
[4 1 15 12 13 16 17 6 9 14]

Tích của hai vector:
908

Matrix a:
[[9 4 19 1 18]
 [15 11 1 9 14]
 [17 8 4 10 13]]

Matrix c:
[[13 8 9 13]
 [17 11 4 3]
 [13 7 2 18]
 [12 1 7 2]
 [1 13 6 4]]



```
1 #3) Tích của 2 ma trận:  
2 multi_ac = np.dot(a,c)  
3 #hoặc multi_ac1 = a@c  
4 print(multi_ac)
```

[[462 484 250 545]
 [517 439 328 320]
 [542 431 341 389]]

Thực hành 4

7. Hạng của ma trận, ma trận chuyển vị



7.1 Hạng của ma trận A

Hạng của ma trận là cấp cao nhất của định thức con khác 0 của ma trận đó.

Hạng của ma trận A kí hiệu $\text{rank}(A)$ hoặc $r(A)$

- + Ma trận 0 có hạng bằng 0
- + Ma trận A cấp $m \times n$ thì $0 \leq r(A) \leq \min(m,n)$
- + Ma trận A vuông cấp n :
 - Nếu $\det(A) \neq 0$ thì $r(A) = n$
 - Nếu $\det(A) = 0$ thì $r(A) < n$

`np.linalg.matrix_rank(A)`: Tính hạng của ma trận A



7.1 Hạng của ma trận A

- Find the rank and nullity of the matrix

$$A = \begin{bmatrix} -1 & 2 & 0 & 4 & 5 & -3 \\ 3 & -7 & 2 & 0 & 1 & 4 \\ 2 & -5 & 2 & 4 & 6 & 1 \\ 4 & -9 & 2 & -4 & -4 & 7 \end{bmatrix}$$

Solution.

The reduced row-echelon form of A is

$$\begin{bmatrix} 1 & 0 & -4 & -28 & -37 & 13 \\ 0 & 1 & -2 & -12 & -16 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Since there are two nonzero rows, the row space and column space are both two-dimensional, so $\text{rank}(A)=2$.

```

1 A=np.array([(-1, 2, 0, 4, 5,-3),
2             ( 3,-7, 2, 0, 1, 4),
3             ( 2,-5, 2, 4, 6, 1),
4             ( 4,-9, 2,-4,-4, 7)])
5 #Tìm hạng của ma trận A
6 rank_a = np.linalg.matrix_rank(A)
7 print(A)
8 print('Rank(A) = ', rank_a)
```

```

[[ -1  2  0  4  5 -3]
 [ 3 -7  2  0  1  4]
 [ 2 -5  2  4  6  1]
 [ 4 -9  2 -4 -4  7]]
Rank(A) = 2
```

```

1 #Hạng của ma trận θ
2 A_θ = np.zeros((4,5))
3 print(A_θ)
4 rank = np.linalg.matrix_rank(A_θ)
5 print('Rank(A_θ) = ', rank)
```

```

[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
Rank(A_θ) = 0
```

Ma trận B1:

```

[[ 1  2  3  4]
 [-2 -1  4  1]
 [ 3 -4 -5  6]
 [ 1  2  3  4]]
```

Ma trận B2:

```

[[ 1  3  1  4]
 [ 3  9  5 15]
 [ 0  2  1  1]
 [ 0  4  2  3]]
```

$\det(B1) = 0.0$

$\text{Rank}(B1) = 3$

$\det(B2) = -4.0$

$\text{Rank}(B2) = 4$

7.2 Ma trận chuyển vị



VINBIGDATA
VINGROUP



Academy
Vietnam

Chuyển vị của ma trận $m \times n$ A là ma trận $n \times m$ A^T tạo ra bằng cách chuyển hàng thành cột và cột thành hàng:

A.T: Tìm ma trận chuyển vị của ma trận A

```
1 A=np.array([(-1, 2, 0, 4, 5, -3),  
2             ( 3, -7, 2, 0, 1, 4),  
3             ( 2, -5, 2, 4, 6, 1),  
4             ( 4, -9, 2, -4,-4, 7)])  
5 #Tìm ma trận chuyển vị của A  
6 A_T = A.T  
7 print('Ma trận A:\n',A)  
8 print('Ma trận chuyển vị của A:\n',A_T)
```

Ma trận A:

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

Ma trận chuyển vị của A:

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

```
1 B =np.array([( 1, 3, 1, 4),  
2                 ( 3, 9, 5,15),  
3                 ( 0, 2, 1, 1),  
4                 ( 0, 4, 2, 3)])  
5 #Tìm ma trận chuyển vị của B  
6 B_T = B.T  
7 print('Ma trận B:\n',B)  
8 print('Ma trận chuyển vị của B:\n',B_T)
```

Ma trận B:

```
[[ 1  3  1  4]  
[ 3  9  5 15]  
[ 0  2  1  1]  
[ 0  4  2  3]]
```

Ma trận chuyển vị của B:

```
[[ 1  3  0  0]  
[ 3  9  2  4]  
[ 1  5  1  2]  
[ 4 15  1  3]]
```

Thực hành 5



Q & A
Thank you!

Bài 6: LẬP TRÌNH PYTHON CƠ BẢN (Phân tích và xử lý dữ liệu với Pandas - 01)

AI Academy Vietnam

Nội dung bài 6

1. Giới thiệu Pandas
2. Tạo đối tượng cơ bản trong Pandas
 - Series
 - Dataframe
3. Đọc dữ liệu từ các nguồn khác nhau
4. Quan sát và truy xuất dữ liệu trong DataFrame
5. Replacing Values, Rename Columns
6. Lọc dữ liệu trong DataFrame
7. Xác định các tham số thống kê: Sum, Cumsum, Min, Max, Mean, Median, Std
8. Giá trị duy nhất (Unique)
9. Phân tích Time series data (Tiếp cận từ bài toán thực tế)

Pandas



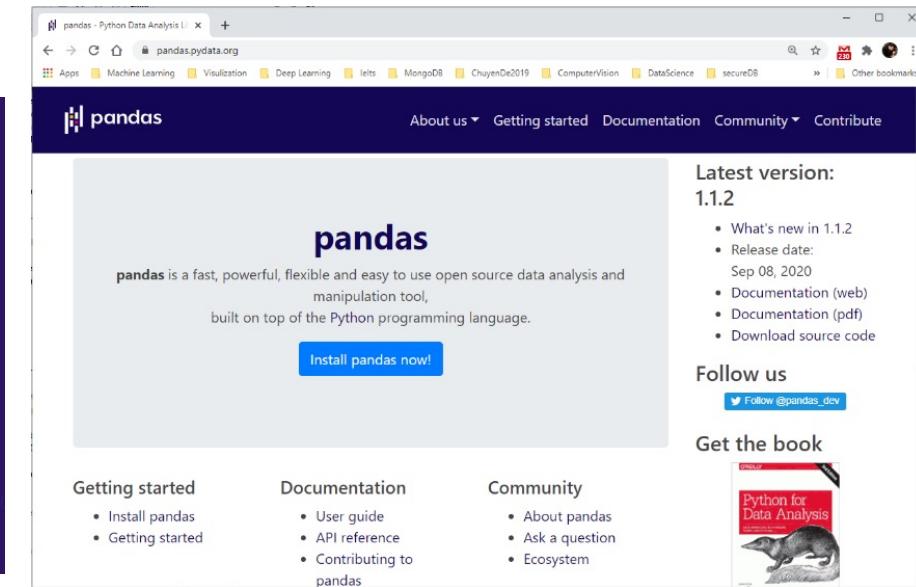
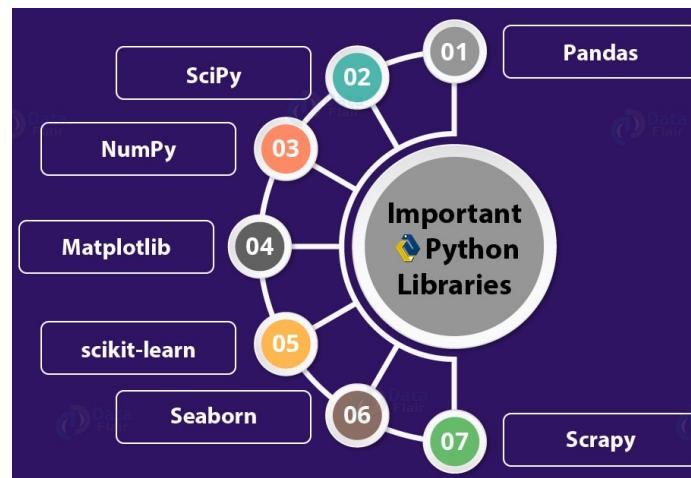
1. Giới thiệu Pandas

1. Giới thiệu

Pandas là một thư viện mã nguồn mở được xây dựng dựa trên NumPy, sử dụng để thao tác và phân tích dữ liệu. Với Pandas chúng ta có thể:

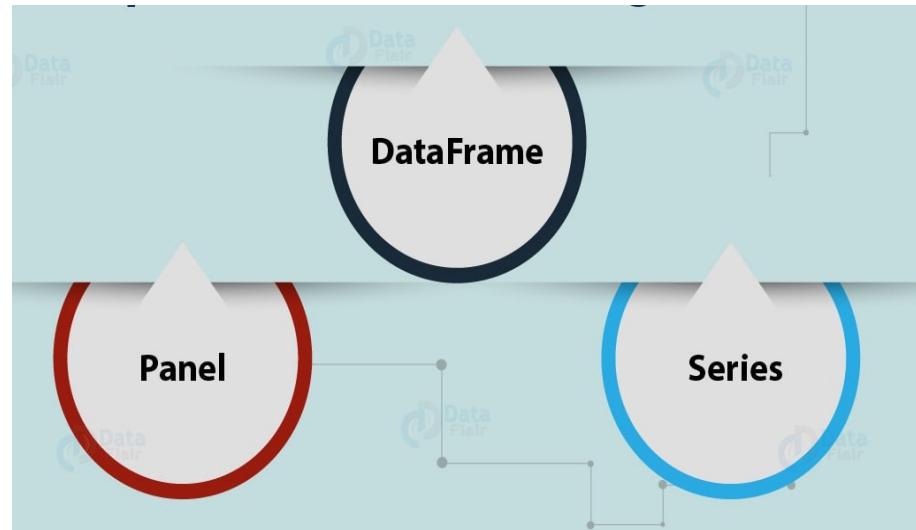
- Xử lý tập dữ liệu khác nhau về định dạng: chuỗi thời gian, bảng không đồng nhất, ma trận dữ liệu
- Import dữ liệu từ nhiều nguồn khác nhau như CSV, DB/SQL...
- Xử lý vô số phép toán cho tập dữ liệu: subsetting, slicing, filtering, merging, groupBy, re-ordering, and re-shaping,..
- Xử lý dữ liệu mượt mà theo mong muốn.
- Xử lý, phân tích dữ liệu tốt như mô hình hóa và thống kê.
- Tích hợp tốt với các thư viện khác của python.

<https://pandas.pydata.org/>



1. Giới thiệu Pandas

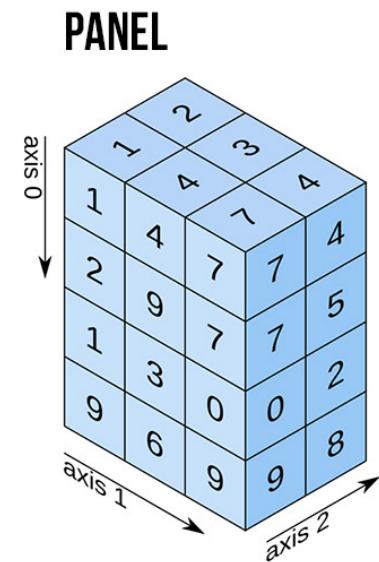
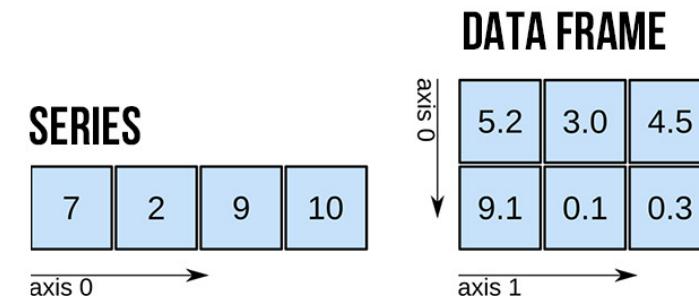
Pandas làm việc thông qua 3 đối tượng Series, DataFrame, Panel



```
1 #Kiểm tra phiên bản của thư viện Pandas
2 import pandas as pd
3 print('Version Pandas: ',pd.__version__)
```

Version Pandas: 1.1.1

Trong ba kiểu dữ liệu, DataFrame là kiểu dữ liệu được sử dụng rộng rãi nhất.

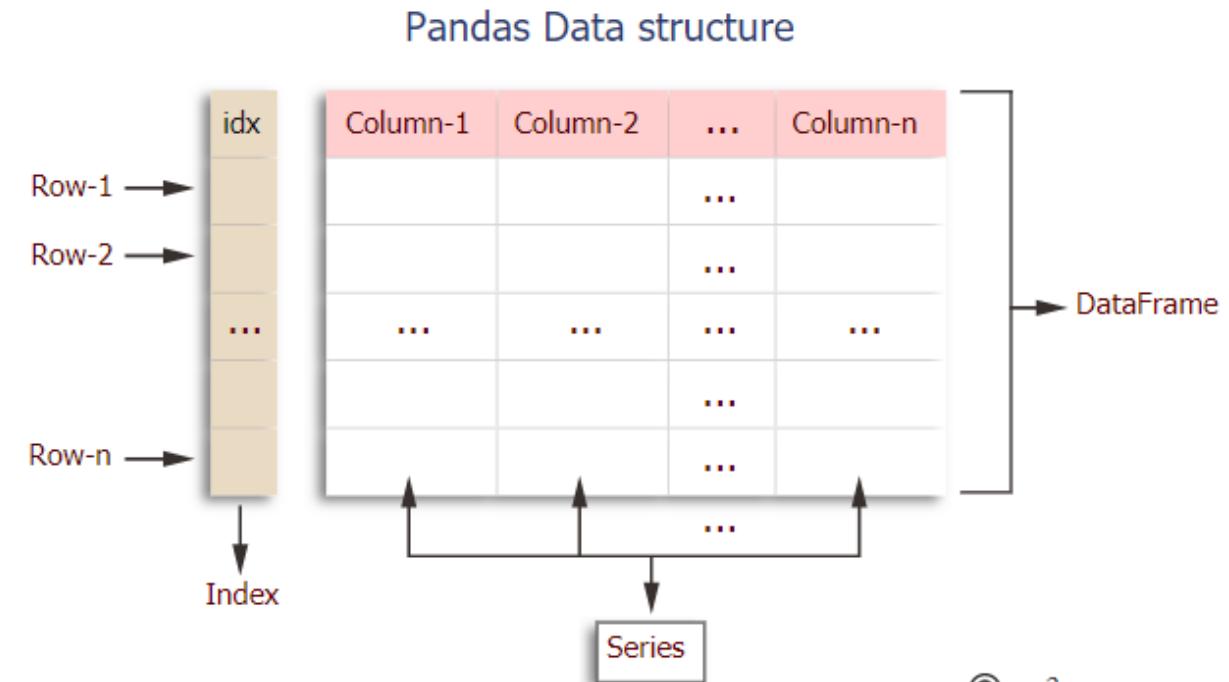
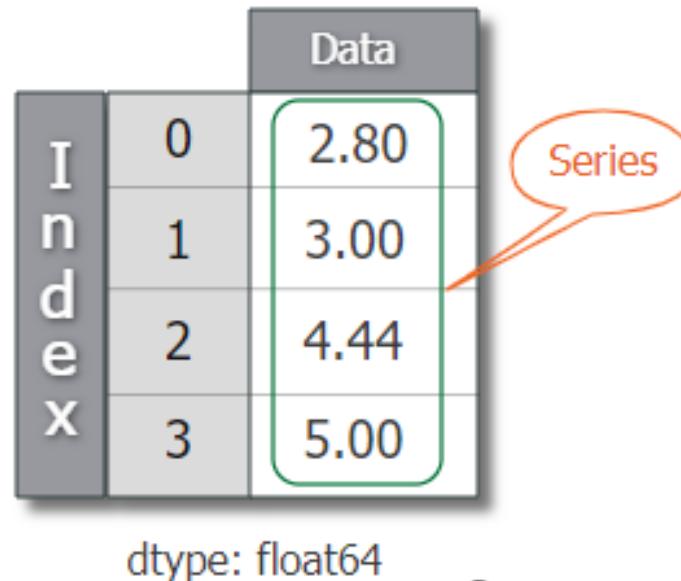




2. Series, DataFrame trong Pandas

2.1 Series

- **Series** là mảng một chiều (1D) giống như kiểu vector trong Numpy, hay như một cột của một bảng, nhưng nó bao gồm thêm một bảng đánh index.



2.1 Series

- Tạo Series sử dụng phương thức;
 - pd.Series(data, index, dtype, name)

```
1 #Tạo một đối tượng series
2 #index mặc định đánh số từ 0
3 data = pd.Series([2.8, 3, 4.44, 5])
4 data
```

```
0    2.80
1    3.00
2    4.44
3    5.00
dtype: float64
```

```
1 #Mỗi một đối tượng series bao gồm 2 thành phần
2 #1. Values
3 #2. index
4
5 print('Values:', data.values)
6 print('Indices:', data.index)
```

```
Values: [2.8 3. 4.44 5. ]
Indices: RangeIndex(start=0, stop=4, step=1)
```

```
1 #Tạo một đối tượng series với index thiết lập
2 data = pd.Series([1.25, 2, 3.5, 4.75, 8.0],
3                  index=['a', 'b', 'c', 'd', 'k'])
4 data
```

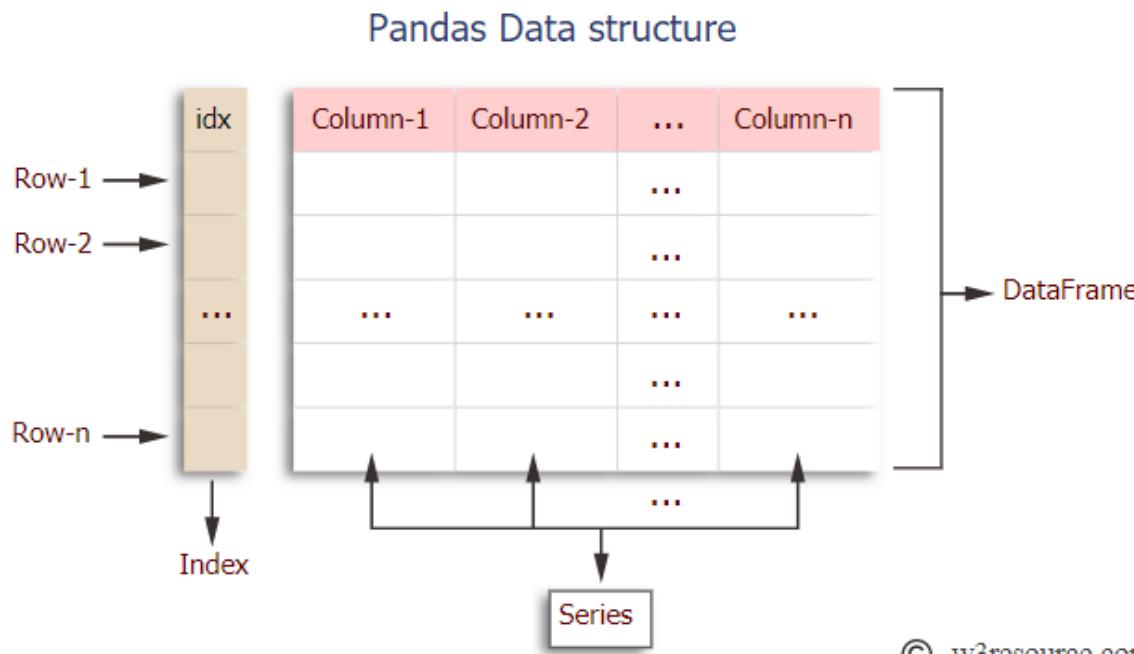
```
a    1.25
b    2.00
c    3.50
d    4.75
k    8.00
dtype: float64
```

```
1 print('Values:', data.values)
2 print('Indices:', data.index)
```

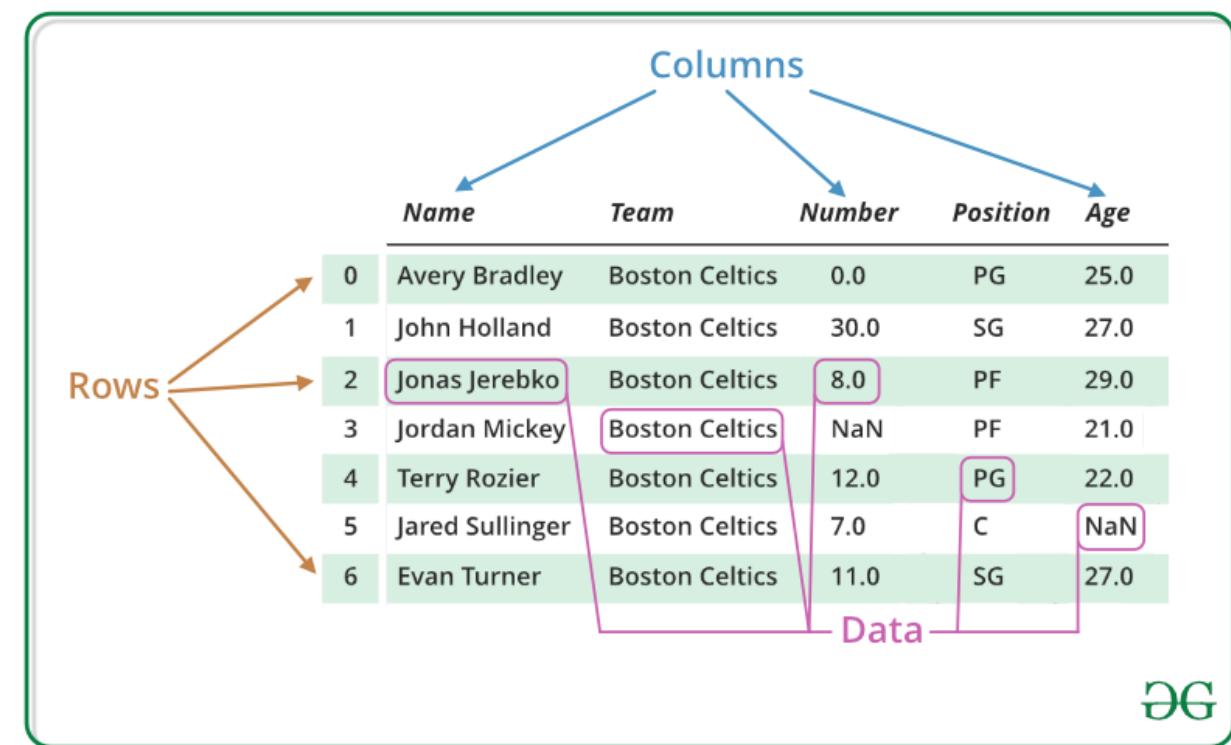
```
Values: [0.25 0.5 0.75 1. ]
Indices: Index(['a', 'b', 'c', 'd'], dtype='object')
```

2.2 DataFrame

DataFrame: Cấu trúc dạng bảng 2D, kích thước có thể thay đổi được. Dữ liệu một cột là đồng nhất nhưng có thể không đồng nhất giữa các cột



Columns



The diagram shows a DataFrame with columns labeled 'Name', 'Team', 'Number', 'Position', and 'Age'. The rows are indexed from 0 to 6. Arrows point from the word 'Rows' to the first three rows (0, 1, 2) and from the word 'Columns' to the last three columns ('Number', 'Position', 'Age'). Specific cells are highlighted with pink boxes: 'Boston Celtics' in row 2, column 'Team'; '8.0' in row 2, column 'Number'; 'PF' in row 2, column 'Position'; and '29.0' in row 2, column 'Age'. Other cells like 'NaN' and 'PG' are also highlighted.

	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Rows

Columns

Data

2.2 DataFrame

- **Tạo DataFrame sử dụng phương thức;**

- `pd.DataFrame(data, index, columns,dtype)`

```
1 #Tạo một DataFrame từ một biến Dict
2 #Chỉ số được tạo mặc định từ 0
3 data_dict = {
4     'apples': [3, 2, 0, 1],
5     'oranges': [0, 3, 7, 2]}
6
7 purchases = pd.DataFrame(data_dict)
8 purchases
```

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

```
1 #Tạo DataFrame với index thiết lập
2 purchases = pd.DataFrame(data_dict,
3                           index=['June', 'Robert', 'Lily', 'David'])
4 purchases
```

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

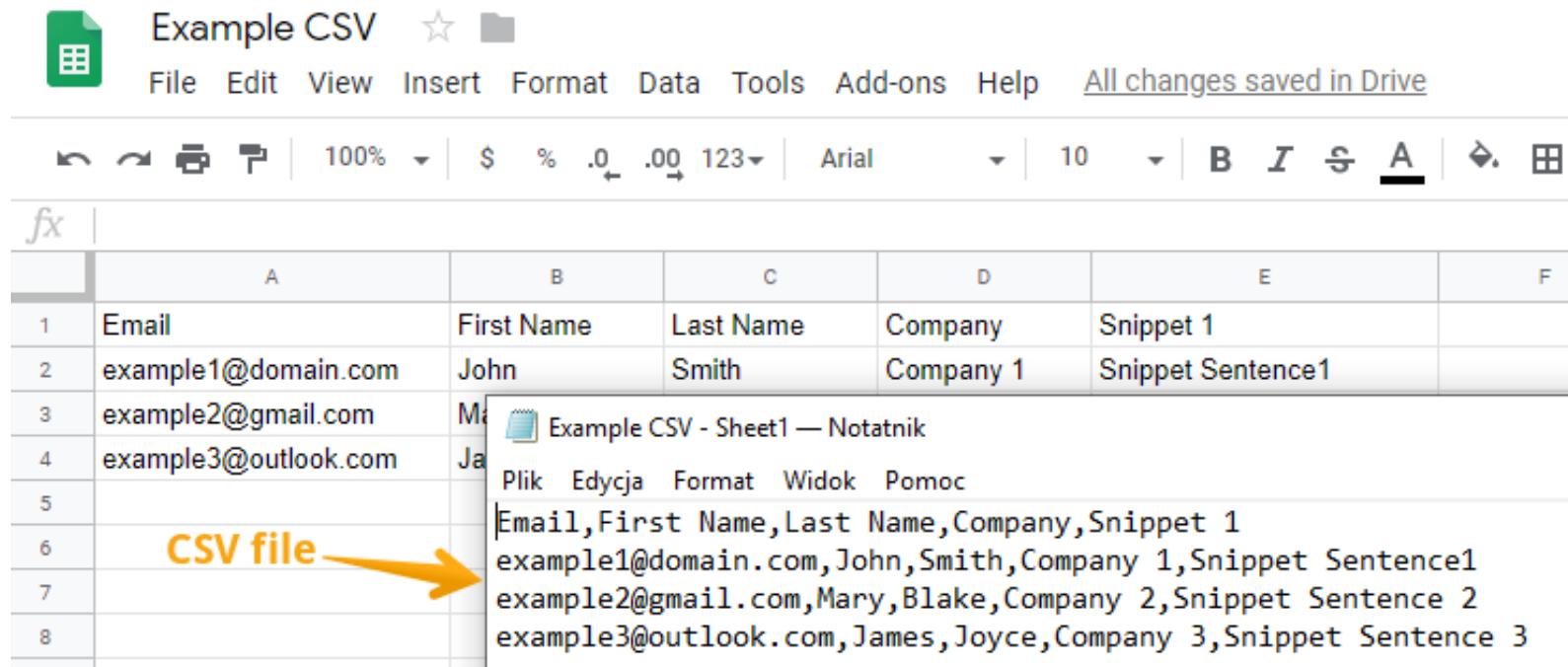
3. Đọc dữ liệu từ các nguồn khác nhau (CSV, Text, Excel)

3.1 Đọc file CSV, Text

- CSV là một định dạng dữ liệu văn bản đơn giản có tên đầy đủ là Comma Separated Values. Với định dạng CSV này, các giá trị được chia tách với nhau bởi các dấu phẩy. Định dạng CSV phổ biến bởi vì chúng có tính tương thích cao, dễ dàng di chuyển từ phần mềm này sang phần mềm khác để sử dụng mà không lo gặp các xung đột.
- Tài liệu CSV cũng làm một trong những tài liệu phổ biến trên thế giới với khả năng lưu trữ nhỏ nhẹ.



What Is A CSV File?



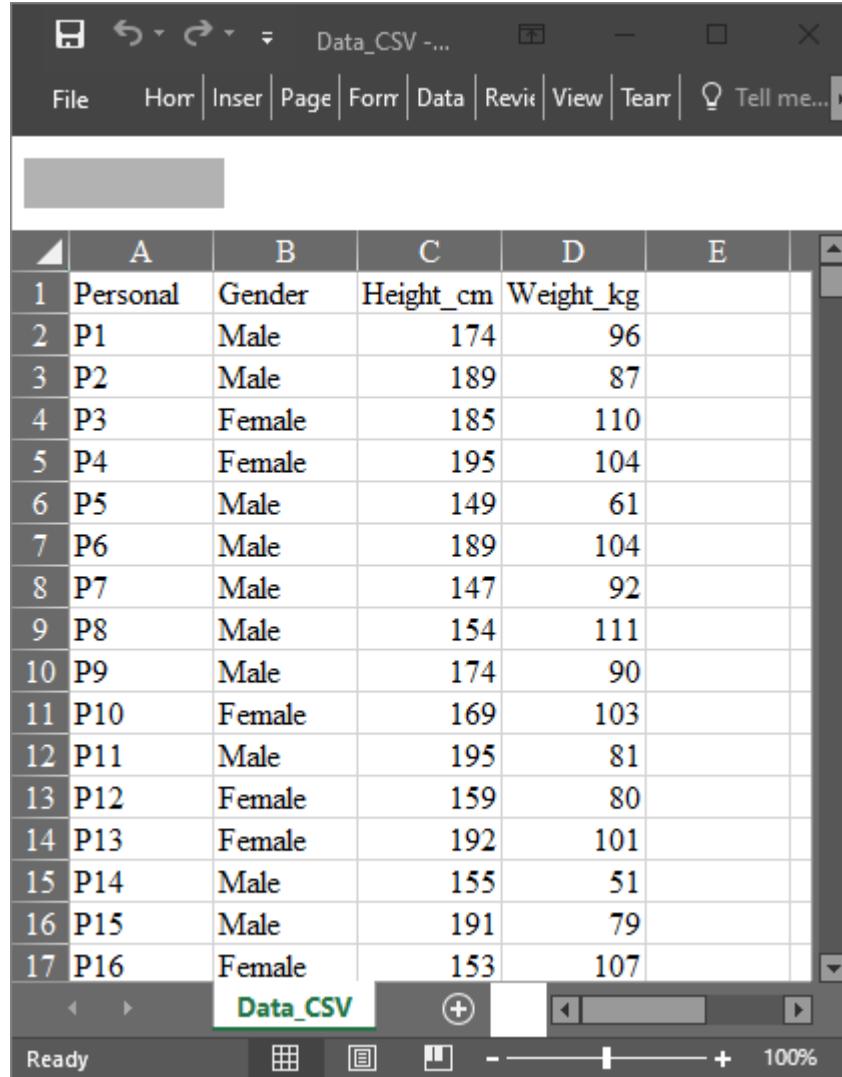
	A	B	C	D	E	F
1	Email	First Name	Last Name	Company	Snippet 1	
2	example1@domain.com	John	Smith	Company 1	Snippet Sentence1	
3	example2@gmail.com					
4	example3@outlook.com					
5						
6	CSV file					
7						
8						

CSV file →

```
Example CSV - Sheet1 — Notatnik
Plik Edycja Format Widok Pomoc
Email,First Name,Last Name,Company,Snippet 1
example1@domain.com,John,Smith,Company 1,Snippet Sentence1
example2@gmail.com,Mary,Blake,Company 2,Snippet Sentence 2
example3@outlook.com,James,Joyce,Company 3,Snippet Sentence 3
```

3.1 Đọc file CSV, Text

Sử dụng phương thức `read_csv()` đọc dữ liệu từ file .CSV



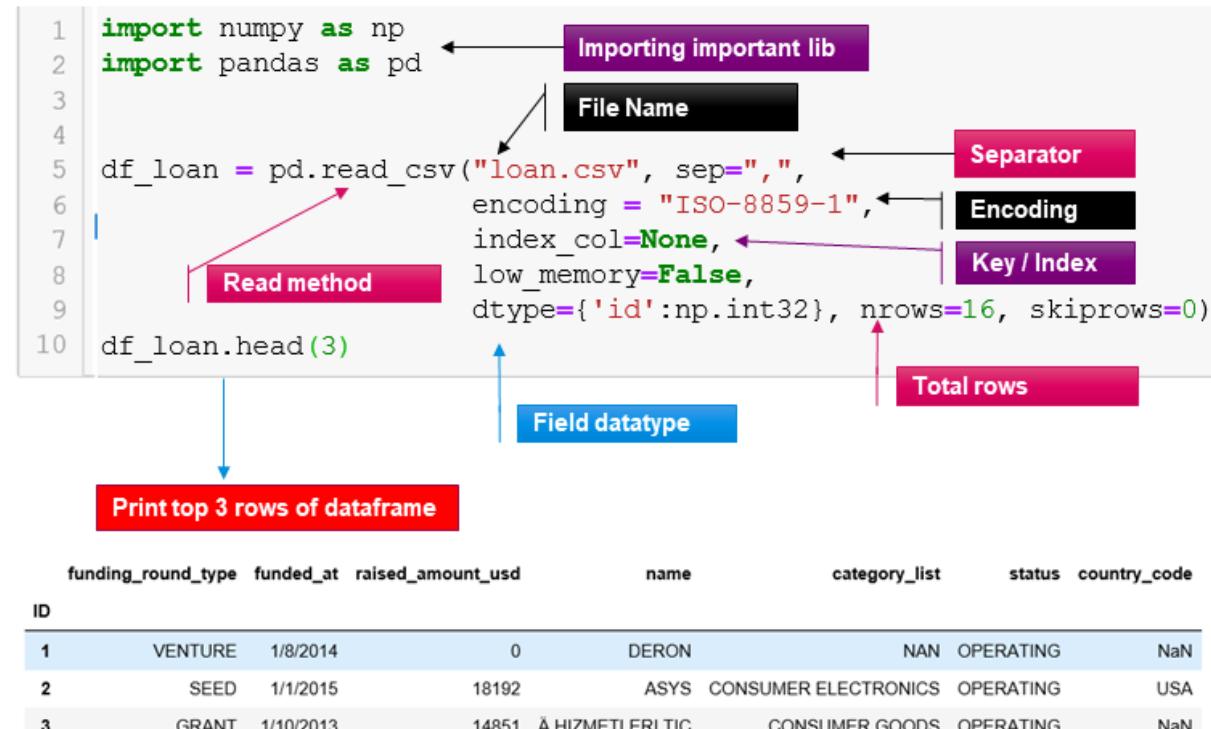
	A	B	C	D	E
1	Personal	Gender	Height_cm	Weight_kg	
2	P1	Male	174	96	
3	P2	Male	189	87	
4	P3	Female	185	110	
5	P4	Female	195	104	
6	P5	Male	149	61	
7	P6	Male	189	104	
8	P7	Male	147	92	
9	P8	Male	154	111	
10	P9	Male	174	90	
11	P10	Female	169	103	
12	P11	Male	195	81	
13	P12	Female	159	80	
14	P13	Female	192	101	
15	P14	Male	155	51	
16	P15	Male	191	79	
17	P16	Female	153	107	

```
1 import pandas as pd
2 path = 'Data_Excersice\CSV\Data_CSV.csv'
3 #Sử dụng phương thức read_csv
4 data = pd.read_csv(path)
5 #Hiển thị thông tin biến Data
6 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Personal    500 non-null    object 
 1   Gender      500 non-null    object 
 2   Height_cm   500 non-null    int64  
 3   Weight_kg   500 non-null    int64  
dtypes: int64(2), object(2)
memory usage: 15.8+ KB
```

3.1 Đọc file CSV, Text

Sử dụng phương thức `read_csv()` có rất nhiều tham số khác nhau để thiết lập cách thức đọc file .csv



The diagram shows a Python code snippet for reading a CSV file:

```
1 import numpy as np
2 import pandas as pd
3
4 df_loan = pd.read_csv("loan.csv", sep=",",
5 encoding = "ISO-8859-1",
6 index_col=None,
7 low_memory=False,
8 dtype={'id':np.int32}, nrows=16, skiprows=0)
9
10 df_loan.head(3)
```

Annotations explain the parameters:

- Importing important lib**: Points to the imports of numpy and pandas.
- File Name**: Points to the file path "loan.csv".
- Separator**: Points to the separator ", ".
- Encoding**: Points to the encoding "ISO-8859-1".
- Key / Index**: Points to the index_col=None parameter.
- Total rows**: Points to the nrows=16 parameter.
- Field datatype**: Points to the dtype parameter.
- Print top 3 rows of dataframe**: Points to the df_loan.head(3) call.

The resulting DataFrame output is:

ID	funding_round_type	funded_at	raised_amount_usd	name	category_list	status	country_code
1	VENTURE	1/8/2014	0	DERON	NAN	OPERATING	NaN
2	SEED	1/1/2015	18192	ASYS	CONSUMER ELECTRONICS	OPERATING	USA
3	GRANT	1/10/2013	14851	À HİZMETLERİ TİC	CONSUMER GOODS	OPERATING	NaN

Arguments

```
read_csv(filepath_or_buffer, sep=',',
delimiter=None, header='infer',
names=None, index_col=None, usecols=None,
squeeze=False, prefix=None,
mangle_dupe_cols=True, dtype=None,
engine=None, converters=None,
true_values=None, false_values=None,
skipinitialspace=False, skiprows=None,
nrows=None, na_values=None,
keep_default_na=True, na_filter=True,
verbose=False, skip_blank_lines=True,
parse_dates=False,
infer_datetime_format=False,
keep_date_col=False, date_parser=None,
dayfirst=False, iterator=False,
chunksize=None, compression='infer',
thousands=None, decimal=',',
lineterminator=None, quotechar='"',
quoting=0, escapechar=None, comment=None,
encoding=None, dialect=None,
tupleize_cols=False,
error_bad_lines=True,
warn_bad_lines=True, skipfooter=0,
skip_footer=0, doublequote=True,
delim_whitespace=False,
as_recarray=False, compact_ints=False,
use_unsigned=False, low_memory=True,
buffer_lines=None, memory_map=False,
float_precision=None)
```



3.1 Đọc file CSV, Text

Vd1: sử dụng tham số index_col để thiết lập cột index khi đọc file csv

```
1 #Sử dụng phương thức read_csv()
2 #Tham số: Thiết lập cột index là cột Personal
3 data1 = pd.read_csv(path,
4                     index_col=0) index sẽ là mảng định
5 data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 500 entries, P1 to P500
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Gender       500 non-null    object  
 1   Height_cm    500 non-null    int64  
 2   Weight_kg    500 non-null    int64  
dtypes: int64(2), object(1)
memory usage: 15.6+ KB
```

```
1 #Hiển thị dữ liệu 5 dòng đầu tiên
2 data1.head()
```

	Personal	Gender	Height_cm	Weight_kg
0	P1	Male	174	96
1	P2	Male	189	87
2	P3	Female	185	110
3	P4	Female	195	104
4	P5	Male	149	61

3.1 Đọc file CSV, Text

Vd2: Thiết lập tham số chỉ đọc 100 dòng đầu tiên và dữ liệu trong 2 cột Height_cm, Weight_kg

```
1 #Sử dụng phương thức read_csv()
2 #Thiết Lập số hàng, cột muốn đọc dữ Liệu
3 data2 = pd.read_csv(path,
4                     nrows=100,
5                     usecols=['Height_cm', 'Weight_kg'])
6 data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          -----          ----- 
 0   Height_cm    100 non-null    int64  
 1   Weight_kg    100 non-null    int64  
dtypes: int64(2)
memory usage: 1.7 KB
```

```
1 #Hiển thị dữ liệu 5 dòng đầu tiên
2 data2.head()
```

	Height_cm	Weight_kg
0	174	96
1	189	87
2	185	110
3	195	104
4	149	61

3.1 Đọc file CSV, Text

Vd3: Thiết lập tham số đọc dữ liệu từ dòng thứ 5 trở đi, và đặt lại tên của từng cột dữ liệu thành ['ID','Sex','H(cm)',W(kg)]

```
1 #Thiết lập tham số đọc dữ liệu từ dòng thứ 5 trong file
2 #và đặt lại tên của các cột dữ liệu
3 data3 = pd.read_csv(path,
4                     names=['ID','Sex','H(cm)','W(kg)'],
5                     skiprows=5)
6 data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 496 entries, 0 to 495
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype  
---  -- 
 0   ID        496 non-null    object 
 1   Sex       496 non-null    object 
 2   H(cm)    496 non-null    int64  
 3   W(kg)    496 non-null    int64  
dtypes: int64(2), object(2)
memory usage: 15.6+ KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data3.head()
```

	ID	Sex	H(cm)	W(kg)
0	P5	Male	149	61
1	P6	Male	189	104
2	P7	Male	147	92
3	P8	Male	154	111
4	P9	Male	174	90

3.1 Đọc file CSV, Text

Vd4: Đọc dữ liệu lưu trữ trong file Text vào biến DataFrame cũng sử dụng phương thức `read_csv()`

```
1 #Đọc dữ liệu trong file txt_Data_Diamonds.txt:  
2 df_Diamonds = pd.read_csv('Data_Excercise/txt_Data_Diamonds.txt',  
3                             names=['Weight(carat)', 'Price(USD)'],  
4                             sep='\t', #mặc định sep=','  
5                             header=None)  
6  
7 df_Diamonds
```

	Weight(carat)	Price(USD)
0	0.23	484
1	0.31	942
2	0.20	345
3	1.02	4459

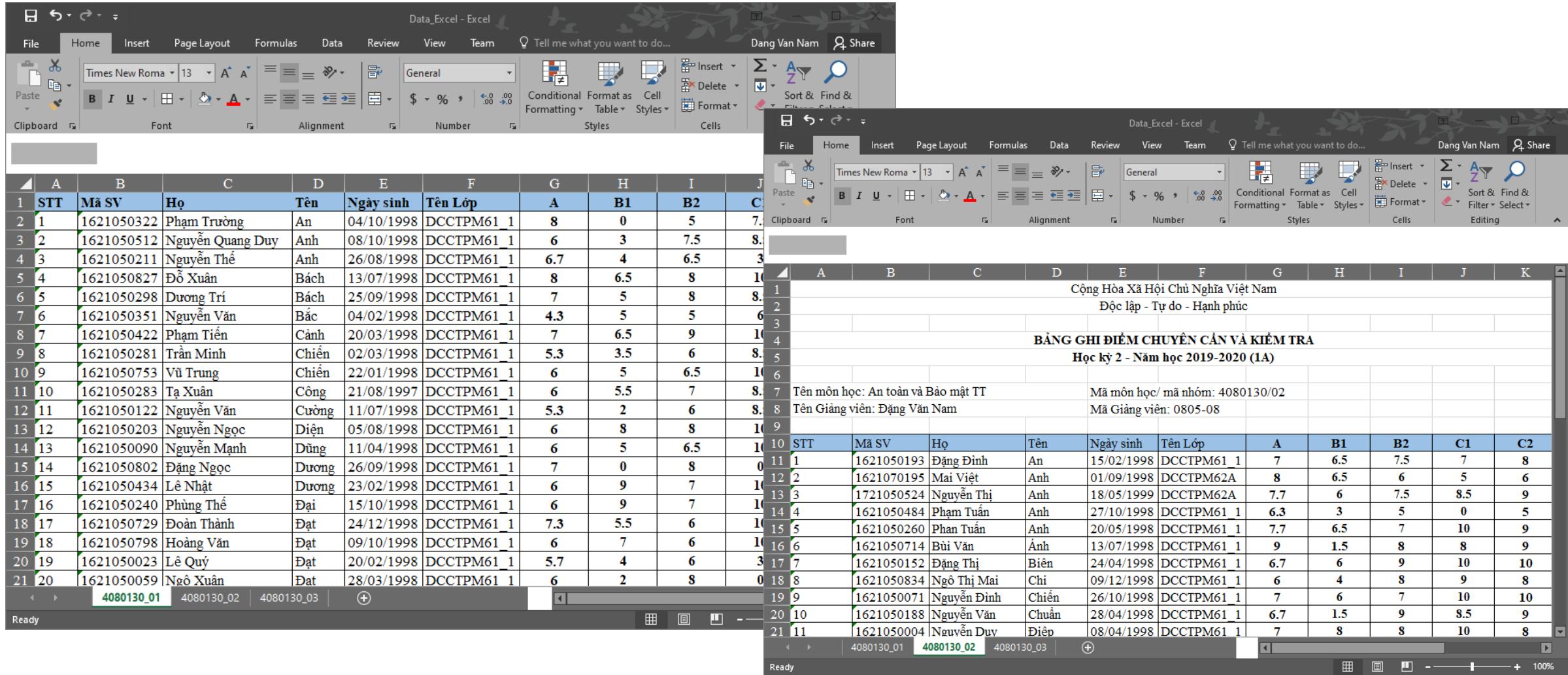
Thực hành 1



3.2 Đọc dữ liệu từ file Excel

3.2 Đọc file Excel

- File dữ liệu Excel demo gồm 3 sheet:



The screenshot shows three Excel sheets side-by-side:

- Data_Excel**: A table with columns: STT, Mã SV, Họ, Tên, Ngày sinh, Tên Lớp, A, B1, B2, C1, C2. Data includes student records like Phạm Trường An (04/10/1998, DCC TPM61_1) and Nguyễn Ngọc Mạnh (11/04/1998, DCC TPM61_1).
- Sheet1**: Contains text: "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam", "Độc lập - Tự do - Hạnh phúc", and two tables:
 - BÀNG GHI ĐIỂM CHUYÊN CẨN VÀ KIỂM TRA**
Học kỳ 2 - Năm học 2019-2020 (1A)
 - Tên môn học: An toàn và Bảo mật TT
Mã môn học/ mã nhóm: 4080130/02
 - Tên Giảng viên: Đặng Văn Nam
Mã Giảng viên: 0805-08
- Sheet2**: Contains student records from the first sheet, likely a copy or a different view of the same data.

3.2 Đọc file Excel

- Sử dụng phương thức `pd.read_excel()` để đọc dữ liệu từ file excel.
 - Lưu ý 2 tham số `sheetname=""` xác định sheet muốn đọc dữ liệu (Mặc định là sheet đầu tiên)

```
1 import pandas as pd
2 path_excel = 'Data_Excersice\Data_Excel.xlsx'
3 #Đọc dữ liệu từ file excel
4 data_ex = pd.read_excel(path_excel)
5 data_ex.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66 entries, 0 to 65
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   STT         66 non-null    int64  
 1   Mã SV       66 non-null    int64  
 2   Họ          66 non-null    object  
 3   Tên          66 non-null    object  
 4   Ngày sinh   66 non-null    object  
 5   Tên Lớp     66 non-null    object  
 6   A            66 non-null    float64 
 7   B1           66 non-null    float64 
 8   B2           66 non-null    float64 
 9   C1           66 non-null    float64 
 10  C2           66 non-null    float64 
dtypes: float64(5), int64(2), object(4)
memory usage: 5.8+ KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data_ex.head()
```

	STT	Mã SV	Họ	Tên	Ngày sinh	Tên Lớp	A	B1	B2	C1	C2
0	1	1621050322	Phạm Trường	An	04/10/1998	DCCTPM61_1	8.0	0.0	5.0	7.5	8.0
1	2	1621050512	Nguyễn Quang Duy	Anh	08/10/1998	DCCTPM61_1	6.0	3.0	7.5	8.5	9.0
2	3	1621050211	Nguyễn Thế	Anh	26/08/1998	DCCTPM61_1	6.7	4.0	6.5	3.0	5.0
3	4	1621050827	Đỗ Xuân	Bách	13/07/1998	DCCTPM61_1	8.0	6.5	8.0	10.0	9.0
4	5	1621050298	Dương Trí	Bách	25/09/1998	DCCTPM61_1	7.0	5.0	8.0	8.5	9.0

3.2 Đọc file Excel

- Một vài tham số quan trọng trong phương thức `pd.read_excel()` để đọc dữ liệu từ file excel.

Argument	Description
io	A string containing the pathname of the given Excel file.
sheet_name	The Excel sheet name, or sheet number, of the data you want to import. The sheet number can be an integer where 0 is the first sheet, 1 is the second, etc. If a list of sheet names/numbers are given, then the output will be a dictionary of DataFrames. The default is to read all the sheets and output a dictionary of DataFrames.
header	Row number to use for the list of column labels. The default is 0, indicating that the first row is assumed to contain the column labels. If the data does not have a row of column labels, None should be used.
names	A separate Python list input of column names. This option is None by default. This option is the equivalent of assigning a list of column names to the columns attribute of the output DataFrame.
index_col	Specifies which column should be used for row indices. The default option is None, meaning that all columns are included in the data, and a range of numbers is used as the row indices.
usecols	An integer, list of integers, or string that specifies the columns to be imported into the DataFrame. The default is to import all columns. If a string is given, then Pandas uses the standard Excel format to select columns (e.g. "A:C,F,G" will import columns A, B, C, F, and G).
skiprows	The number of rows to skip at the top of the Excel sheet. Default is 0. This option is useful for skipping rows in Excel that contain explanatory information about the data below it.

3.2 Đọc file Excel

- Sử dụng phương thức `pd.read_excel()` với một số tham số cơ bản.

```
1 #Ví dụ:  
2 #Đọc dữ liệu tại sheet đầu tiên,  
3 #Chỉ lấy dữ liệu cột Mã SV và các cột điểm  
4 #Thiết lập cột đầu tiên làm index  
5 data_ex1 = pd.read_excel(path_excel,  
6                           sheet_name=0,  
7                           usecols=[1,6,7,8,9,10],  
8                           index_col=0)  
9 data_ex1.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 66 entries, 1621050322 to 1621050013  
Data columns (total 5 columns):  
 #   Column  Non-Null Count  Dtype     
---  --  -----  --  
 0   A       66 non-null    float64  
 1   B1      66 non-null    float64  
 2   B2      66 non-null    float64  
 3   C1      66 non-null    float64  
 4   C2      66 non-null    float64  
dtypes: float64(5)  
memory usage: 3.1 KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên  
2 data_ex1.head()
```

	A	B1	B2	C1	C2
Mã SV					
1621050322	8.0	0.0	5.0	7.5	8.0
1621050512	6.0	3.0	7.5	8.5	9.0
1621050211	6.7	4.0	6.5	3.0	5.0
1621050827	8.0	6.5	8.0	10.0	9.0
1621050298	7.0	5.0	8.0	8.5	9.0

3.2 Đọc file Excel

- Sử dụng phương thức `pd.read_excel()` với một số tham số cơ bản.
 - Đọc dữ liệu sheet 2 ['4080130_02'], từ dòng 9.

```

1 #Ví dụ 3:
2 #Đọc dữ liệu tại sheet 2, từ dòng 9
3 data_ex3 = pd.read_excel(path_excel,
4                         sheet_name='4080130_02',
5                         skiprows=9)
6 data_ex3.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39 entries, 0 to 38
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
--- 
 0   STT          39 non-null    int64  
 1   Mã SV        39 non-null    int64  
 2   Họ           39 non-null    object  
 3   Tên          39 non-null    object  
 4   Ngày sinh   39 non-null    object  
 5   Tên Lớp     39 non-null    object  
 6   A             39 non-null    float64 
 7   B1            39 non-null    float64 
 8   B2            39 non-null    float64 
 9   C1            39 non-null    float64 
 10  C2            39 non-null    float64 
dtypes: float64(5), int64(2), object(4)
memory usage: 3.5+ KB

```

```

1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data_ex3.head()

```

STT	Mã SV	Họ	Tên	Ngày sinh	Tên Lớp	A	B1	B2	C1	C2
0	1 1621050193	Đặng Đình	An	15/02/1998	DCCTPM61_1	7.0	6.5	7.5	7.0	8.0
1	2 1621070195	Mai Việt	Anh	01/09/1998	DCCTPM62A	8.0	6.5	6.0	5.0	6.0
2	3 1721050524	Nguyễn Thị	Anh	18/05/1999	DCCTPM62A	7.7	6.0	7.5	8.5	9.0
3	4 1621050484	Phạm Tuấn	Anh	27/10/1998	DCCTPM61_1	6.3	3.0	5.0	0.0	5.0
4	5 1621050260	Phan Tuấn	Anh	20/05/1998	DCCTPM61_1	7.7	6.5	7.0	10.0	9.0

3.2 Đọc file Excel

- Sử dụng phương thức `pd.read_excel()` với một số tham số cơ bản.
 - Đọc dữ liệu sheet 3 ['4080130_03'], không có dòng header

```
1 #Ví dụ 4
2 #Đọc dữ liệu từ sheet: '4080130_03'
3 #Dữ liệu không chứa dòng header
4 data_ex4 = pd.read_excel(path_excel,
5                           sheet_name='4080130_03',
6                           header=None)
7 data_ex4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39 entries, 0 to 38
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype  
---  -- 
 0   0          39 non-null    int64  
 1   1          39 non-null    int64  
 2   2          39 non-null    object  
 3   3          39 non-null    object  
 4   4          39 non-null    object  
 5   5          39 non-null    object  
 6   6          39 non-null    float64 
 7   7          39 non-null    float64 
 8   8          39 non-null    float64 
 9   9          39 non-null    float64 
 10  10         39 non-null    float64 
dtypes: float64(5), int64(2), object(4)
memory usage: 3.5+ KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data_ex4.head()
```

	0	1	2	3	4	5	6	7	8	9	10
0	1	1621050041	Đào Tuấn	Anh	22/10/1998	DCCTPM61_1	6.7	9.0	5.5	8.5	8.0
1	2	1621050262	Vũ Thị Lan	Anh	26/09/1998	DCCTPM61_1	6.7	7.0	9.0	8.5	6.0
2	3	1621050083	Trịnh Như	Bình	06/04/1998	DCCTPM61_1	7.3	8.5	9.5	10.0	9.0
3	4	1621050113	Trần Văn	Cường	19/06/1998	DCCTPM61_1	5.7	5.0	6.0	10.0	5.0
4	5	1621050384	Nguyễn Sỹ	Dũng	02/10/1998	DCCTPM61_1	7.0	0.0	7.5	8.5	9.0

3.2 Đọc file Excel

- Sử dụng phương thức `pd.read_excel()` với một số tham số cơ bản.
 - Đọc dữ liệu sheet 3 ['4080130_03'],
 - Không có dòng header
 - Chỉ lấy dữ liệu cột 1,6,7,8,9,10
 - Đặt tên cho các cột lần lượt là ['Mã SV', 'A', 'B1','B2','C1','C2']
 - Thiết lập cột đầu tiên làm Index

```
5 data_ex41 = pd.read_excel(path_excel,
6                             sheet_name='4080130_03',
7                             header=None,
8                             usecols=[1,6,7,8,9,10],
9                             names=['Mã SV', 'A', 'B1', 'B2', 'C1', 'C2'],
10                            index_col=0)
11 data_ex41.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 39 entries, 1621050041 to 1621050034
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  --   --   --   --   -- 
 0   A       39 non-null    float64
 1   B1      39 non-null    float64
 2   B2      39 non-null    float64
 3   C1      39 non-null    float64
 4   C2      39 non-null    float64
dtypes: float64(5)
memory usage: 1.8 KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data_ex41.head()
```

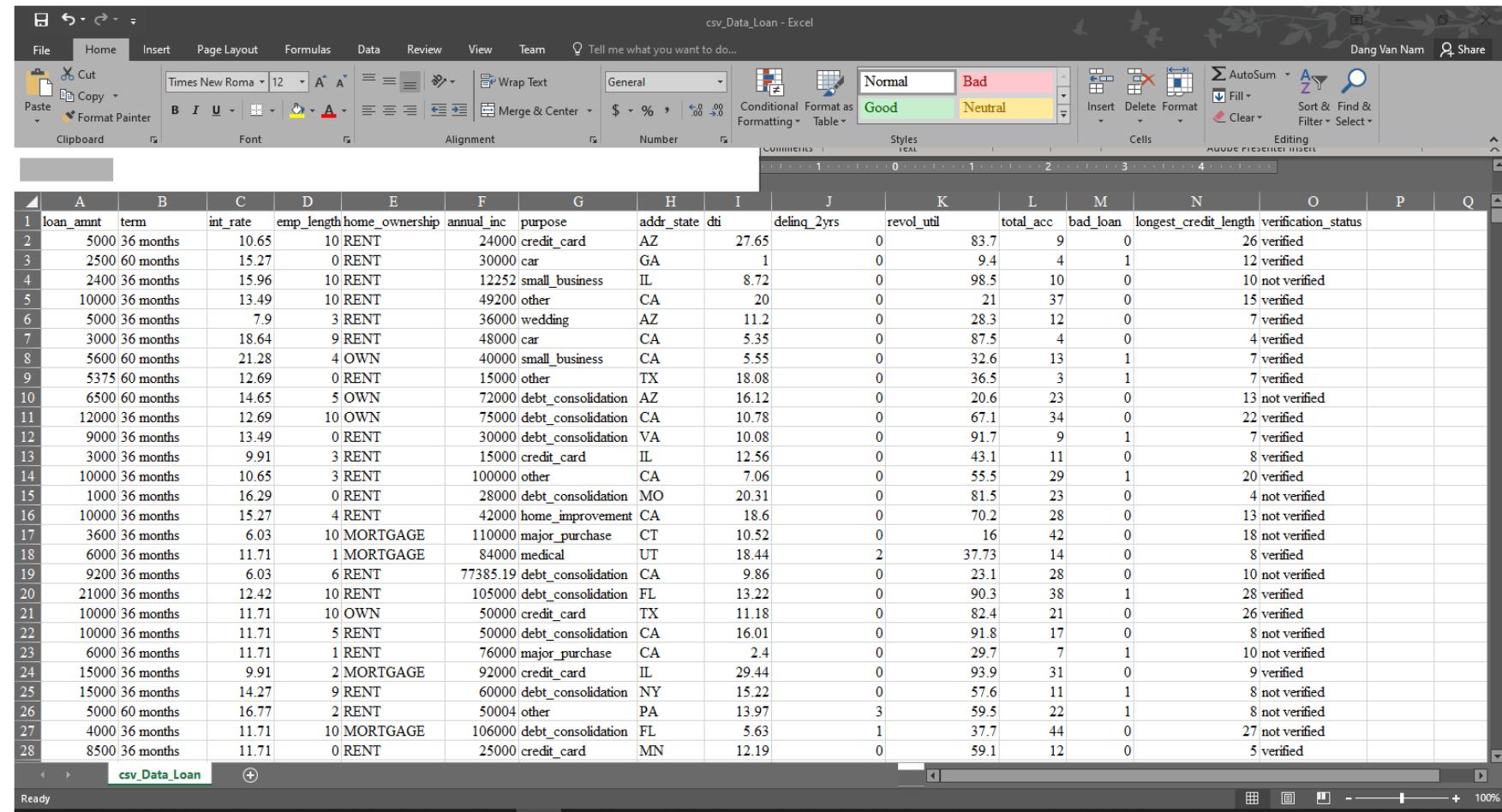
	A	B1	B2	C1	C2
Mã SV					
1621050041	6.7	9.0	5.5	8.5	8.0
1621050262	6.7	7.0	9.0	8.5	6.0
1621050083	7.3	8.5	9.5	10.0	9.0
1621050113	5.7	5.0	6.0	10.0	5.0
1621050384	7.0	0.0	7.5	8.5	9.0

Thực hành 2

4. Quan sát và truy cập dữ liệu trong DataFrame

4.1 Quan sát dữ liệu

- Đọc file dữ liệu mẫu: **csv_Data_loan**
- Đây là file dữ liệu cho biết thông tin về các khoản vay cho các mục đích khác nhau của người dùng Mỹ.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
1	loan_amnt	term	int_rate	emp_length	home_ownership	annual_inc	purpose	addr_state	dti	delinq_2yrs	revol_util	total_acc	bad_loan	longest_credit_length	verification_status			
2	5000	36 months	10.65	10 RENT		24000	credit_card	AZ	27.65	0	83.7	9	0		26	verified		
3	2500	60 months	15.27	0 RENT		30000	car	GA		1	0	9.4	4	1		12	verified	
4	2400	36 months	15.96	10 RENT		12252	small_business	IL	8.72	0	98.5	10	0		10	not verified		
5	10000	36 months	13.49	10 RENT		49200	other	CA	20	0	21	37	0		15	verified		
6	5000	36 months	7.9	3 RENT		36000	wedding	AZ	11.2	0	28.3	12	0		7	verified		
7	3000	36 months	18.64	9 RENT		48000	car	CA	5.35	0	87.5	4	0		4	verified		
8	5600	60 months	21.28	4 OWN		40000	small_business	CA	5.55	0	32.6	13	1		7	verified		
9	5375	60 months	12.69	0 RENT		15000	other	TX	18.08	0	36.5	3	1		7	verified		
10	6500	60 months	14.65	5 OWN		72000	debt_consolidation	AZ	16.12	0	20.6	23	0		13	not verified		
11	12000	36 months	12.69	10 OWN		75000	debt_consolidation	CA	10.78	0	67.1	34	0		22	verified		
12	9000	36 months	13.49	0 RENT		30000	debt_consolidation	VA	10.08	0	91.7	9	1		7	verified		
13	3000	36 months	9.91	3 RENT		15000	credit_card	IL	12.56	0	43.1	11	0		8	verified		
14	10000	36 months	10.65	3 RENT		100000	other	CA	7.06	0	55.5	29	1		20	verified		
15	1000	36 months	16.29	0 RENT		28000	debt_consolidation	MO	20.31	0	81.5	23	0		4	not verified		
16	10000	36 months	15.27	4 RENT		42000	home_improvement	CA	18.6	0	70.2	28	0		13	not verified		
17	3600	36 months	6.03	10 MORTGAGE		110000	major_purchase	CT	10.52	0	16	42	0		18	not verified		
18	6000	36 months	11.71	1 MORTGAGE		84000	medical	UT	18.44	2	37.73	14	0		8	verified		
19	9200	36 months	6.03	6 RENT		77385.19	debt_consolidation	CA	9.86	0	23.1	28	0		10	not verified		
20	21000	36 months	12.42	10 RENT		105000	debt_consolidation	FL	13.22	0	90.3	38	1		28	verified		
21	10000	36 months	11.71	10 OWN		50000	credit_card	TX	11.18	0	82.4	21	0		26	verified		
22	10000	36 months	11.71	5 RENT		50000	debt_consolidation	CA	16.01	0	91.8	17	0		8	not verified		
23	6000	36 months	11.71	1 RENT		76000	major_purchase	CA	2.4	0	29.7	7	1		10	not verified		
24	15000	36 months	9.91	2 MORTGAGE		92000	credit_card	IL	29.44	0	93.9	31	0		9	verified		
25	15000	36 months	14.27	9 RENT		60000	debt_consolidation	NY	15.22	0	57.6	11	1		8	not verified		
26	5000	60 months	16.77	2 RENT		50004	other	PA	13.97	3	59.5	22	1		8	not verified		
27	4000	36 months	11.71	10 MORTGAGE		106000	debt_consolidation	FL	5.63	1	37.7	44	0		27	not verified		
28	8500	36 months	11.71	0 RENT		25000	credit_card	MN	12.19	0	59.1	12	0		5	verified		

4.1 Quan sát dữ liệu

- **df.info()** : Hiển thị thông tin chi tiết biến DataFrame
- **df.head(n)**: Hiển thị n dòng đầu tiên của biến df (default = 5)
- **df.tail(n)** : Hiển thị n dòng cuối cùng biến df (default = 5)
- **df.shape** : Hiển thị kích thước (rows x columns) của biến df
- **df.columns**: Tên các cột trong biến df
- **df.isnull()** : Kiểm tra dữ liệu rỗng trong biến df
- **df.isnull().sum()** : Tính tổng các dòng dữ liệu null trong df
- **df.count()** : Tổng số dòng dữ liệu không null trong df
- **df.size** : Số phần tử của biến df (=rows x columns)
- **df.dtypes** : Kiểu dữ liệu của từng columns trong df



4.1 Quan sát dữ liệu

- **df.describe()** : Một số đặc trưng thống kê của biến df
 - Tham số include =‘O’: thống kê các cột có kiểu dữ liệu Object
 - Tham số include=‘all’: Thống kê tất cả các cột trong df

```
1 #Quan sát một số đặc trưng thống kê của df
2 #Thống kê các cột dữ liệu Object
3 df_loan.describe(include='O')
```

	term	home_ownership	purpose	addr_state	verification_status
count	163987	163987	163987	163987	163987
unique	2	6	14	50	2
top	36 months	MORTGAGE	debt_consolidation	CA	verified
freq	129950	79714	93261	28702	104832

4.2 Truy cập dữ liệu

- `df[['Col1', 'Col2', 'Col3']]`: Chỉ truy cập dữ liệu của các cột có tên **Col1**, **Col2**, **Col3** trong dataframe df

```
1 #Truy xuất dữ liệu theo cột
2 #Lấy dữ liệu của một cột
3 df_state = df_loan[['addr_state']]
4 df_state.head()
```

	addr_state
0	AZ
1	GA
2	IL
3	CA
4	AZ

```
1 #Truy xuất dữ liệu theo cột
2 #Chỉ lấy dữ liệu của 3 cột: loan_amnt, int_rate, purpose
3 df_loan1 = df_loan[['loan_amnt','int_rate','purpose']]
4 df_loan1.head()
```

	loan_amnt	int_rate	purpose
0	5000	10.65	credit_card
1	2500	15.27	car
2	2400	15.96	small_business
3	10000	13.49	other
4	5000	7.90	wedding

4.2 Truy cập dữ liệu

- `df.iloc[[index_row],[index_col]]`: Truy cập tới dữ liệu của hàng và cột qua **chỉ số index_row, index_col (tương tự như với Numpy)**

```
1 #Sử dụng .iloc truy xuất dữ liệu như với Numpy  
2 #Truy xuất 10 dòng dữ liệu từ [10 --> 20) tất cả các cột  
3 df_loan.iloc[10:20,:]
```

	loan_amnt	term	int_rate	emp_length	home_ownership	annual_inc	purpose	addr_state	dti	delinq_2yrs	revol_util	total_acc	bad_loan	...
10	9000	36 months	13.49	0.0	RENT	30000.00	debt_consolidation	VA	10.08	0.0	91.70	9.0	1	
11	3000	36 months	9.91	3.0	RENT	15000.00	credit_card	IL	12.56	0.0	43.10	11.0	0	
12	10000	36 months	10.65	3.0	RENT	100000.00	other	CA	7.06	0.0	55.50	29.0	1	
13	1000	36 months	16.29	0.0	RENT	28000.00	debt_consolidation	MO	20.31	0.0	81.50	23.0	0	
14	10000	36 months	15.27	4.0	RENT	42000.00	home_improvement	CA	18.60	0.0	70.20	28.0	0	
15	3600	36 months	6.03	10.0	MORTGAGE	110000.00	major_purchase	CT	10.52	0.0	16.00	42.0	0	

4.2 Truy cập dữ liệu

- `df.loc[[name_index],[name_col]]`: Truy cập tới dữ liệu của hàng và cột qua **name_index, name_column**

```
1 #Truy cập từ dòng 20 đến dòng 25 của df
2 #chỉ lấy dữ liệu 4 cột: loan_amnt, home_ownership, purpose, addr_state
3 df_loan.loc[20:25,['loan_amnt','home_ownership','purpose','addr_state']]
```

	loan_amnt	home_ownership	purpose	addr_state
20	10000	RENT	debt_consolidation	CA
21	6000	RENT	major_purchase	CA
22	15000	MORTGAGE	credit_card	IL
23	15000	RENT	debt_consolidation	NY
24	5000	RENT	other	PA
25	4000	MORTGAGE	debt_consolidation	FL

4.2 Truy cập dữ liệu

Type	Notes
df[val]	Select single column or sequence of columns from the DataFrame; special case conveniences: boolean array (filter rows), slice (slice rows), or boolean DataFrame (set values based on some criterion)
df.loc[val]	Selects single row or subset of rows from the DataFrame by label
df.loc[:, val]	Selects single column or subset of columns by label
df.loc[val1, val2]	Select both rows and columns by label
df.iloc[where]	Selects single row or subset of rows from the DataFrame by integer position
df.iloc[:, where]	Selects single column or subset of columns by integer position
df.iloc[where_i, where_j]	Select both rows and columns by integer position
df.at[label_i, label_j]	Select a single scalar value by row and column label
df.iat[i, j]	Select a single scalar value by row and column position (integers)
reindex method	Select either rows or columns by labels
get_value, set_value methods	Select single value by row and column label



5. Replacing Values, Rename columns

5.1 Replacing Values

- Thay thế 1 giá trị trong Dataframe, thực hiện tương tự như với Numpy. Sử dụng `.loc`; `.iloc` để xác định phần tử cần cập nhật, thay đổi giá trị

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ
5	3000	RENT	car	CA
6	5600	OWN	small_business	CA
7	5375	RENT	other	TX
8	6500	OWN	debt_consolidation	AZ
9	12000	OWN	debt_consolidation	CA
10	9000	RENT	debt_consolidation	VA

```
1 #Thay thế giá trị purpose: credit_card--> wedding
2 #của index đầu tiên
3 df_new.loc[0, 'purpose'] = 'wedding'
4 df_new
```

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	wedding	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA

```
1 #Thay thế giá trị thuộc tính loan_amnt: 2400 --> 8800
2 #của index = 2
3 df_new.iloc[2,0] = 8800
4 df_new
```

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	wedding	AZ
1	2500	RENT	car	GA
2	8800	RENT	small_business	IL
3	10000	RENT	other	CA

5.1 Replacing Values

- **df.replace():** Thay thế các giá trị trong toàn bộ DataFrame. (tham số inplace=True|False áp dụng thay đổi cho dataframe hiện tại hay không?).

loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	credit_card
1	2500	RENT	car
2	2400	RENT	small_business
3	10000	RENT	other
4	5000	RENT	wedding
5	3000	RENT	car
6	5600	OWN	small_business
7	5375	RENT	other
8	6500	OWN	debt_consolidation
9	12000	OWN	debt_consolidation
10	9000	RENT	debt_consolidation

```
1 #Khi muốn thay đổi áp dụng lên DataFrame hiện tại
2 #Thiết lập tham số inplace=True
3 df_new.replace({'RENT':'MORTGAGE',
4                 'car':'small_business'}, inplace=True)
5 df_new
```

loan_amnt	home_ownership	purpose	addr_state
0	5000	MORTGAGE	credit_card
1	2500	MORTGAGE	small_business
2	2400	MORTGAGE	small_business

02

01

```
1 #Thay thế nhiều giá trị trong DataFrame
2 #RENT --> MORTGAGE
3 #car --> small_business
4 df_new.replace({'RENT':'MORTGAGE',
5                 'car':'small_business'})
```

loan_amnt	home_ownership	purpose	addr_state
0	5000	MORTGAGE	wedding
1	2500	MORTGAGE	small_business
2	8800	MORTGAGE	small_business
3	10000	MORTGAGE	other
4	5000	MORTGAGE	wedding
5	3000	MORTGAGE	small_business
6	5600	OWN	small_business
7	5375	MORTGAGE	other
8	6500	OWN	debt_consolidation
9	12000	OWN	debt_consolidation
10	9000	MORTGAGE	debt_consolidation

5.1 Replacing Values

- **df.replace():** Thay thế các giá trị theo từng cột (tham số inplace=True|False áp dụng thay đổi cho dataframe hiện tại hay không?).

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ
5	3000	RENT	car	CA
6	5600	OWN	small_business	CA
7	5375	RENT	other	TX
8	6500	OWN	debt_consolidation	AZ
9	12000	OWN	debt_consolidation	CA
10	9000	RENT	debt_consolidation	VA

```
1 #Thay thế tên viết tắt bằng tên đầy đủ.  
2 state_name={'AZ':'Arizona',  
3             'GA':'Georgia',  
4             'IL':'Illinois',  
5             'CA':'California',  
6             'TX':'Texas',  
7             'VA':'Virgrinia'}  
8 #Trong cột addr_state  
9 df_new['addr_state'].replace(state_name,inplace=True)  
10 df_new
```

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	credit_card	Arizona
1	2500	RENT	car	Georgia
2	2400	RENT	small_business	Illinois
3	10000	RENT	other	California
4	5000	RENT	wedding	Arizona
5	3000	RENT	car	California

5.2 Rename Columns

- df.rename(): thay đổi tên cột trong DataFrame

```
1 #Muốn áp dụng thay đổi vào trực tiếp biến df, sử dụng inplace=True
2 df_new.rename(columns={'loan_amnt':'Số tiền vay',
3                     'home_ownership':'Tình trạng nhà ở',
4                     'purpose': 'Mục đích vay tiền',
5                     'addr_state':'Địa chỉ'}, inplace=True)
6 df_new.head()
```

01

	Số tiền vay	Tình trạng nhà ở	Mục đích vay tiền	Địa chỉ
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ

```
1 #Đổi tên cột sang viết hoa
2 df_new.rename(str.upper, axis='columns')
```

02

	SÓ TIỀN VAY	TÌNH TRẠNG NHÀ Ở	MỤC ĐÍCH VAY TIỀN	ĐỊA CHỈ
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ

Thực hành 3

6. Filter Data

6. Filter Data

- Để lọc dữ liệu trong DataFrame có thể sử dụng nhiều cách khác nhau

```
1 #Lọc danh sách người giới tính nam
2 #Cách 1:
3 df_male1 = df_bmi[df_bmi.Gender=='Male']
4 df_male1.head(2)
```

	Personal	Gender	Height_cm	Weight_kg
0	P1	Male	174	96
1	P2	Male	189	87

01

```
1 #Cách 2: sử dụng phương thức query
2 df_male2 = df_bmi.query('Gender=="Male"')
3 df_male2.head(2)
```

	Personal	Gender	Height_cm	Weight_kg
0	P1	Male	174	96
1	P2	Male	189	87

02

```
1 #Cách 3: sử dụng iloc
2 df_male3 = df_bmi.loc[(df_bmi.Gender=="Male")]
3 df_male3.head(2)
```

	Personal	Gender	Height_cm	Weight_kg
0	P1	Male	174	96
1	P2	Male	189	87

03

6. Filter Data

- Sử dụng toán tử & (and) - | (or) - ~ (not) để kết hợp nhiều điều kiện trong khi lọc dữ liệu

```

1 #Kết hợp nhiều tiêu chí Lọc dữ Liệu
2 #lọc người có giới tính Femal và cân nặng dưới 70kg
3 df_p1 = df_bmi[(df_bmi.Gender =='Female') & (df_bmi.Weight_kg<70)]
4 df_p1

```

	Personal	Gender	Height_cm	Weight_kg
24	P25	Female	172	67
25	P26	Female	151	64
32	P33	Female	195	65
51	P52	Female	176	54

01

```

1 #Kết hợp nhiều tiêu chí tìm kiếm
2 #lọc người có chiều cao > 195 cm hoặc cân nặng > 150kg
3 df_p2 = df_bmi[(df_bmi.Height_cm >195) | (df_bmi.Weight_kg>150)]
4 df_p2

```

	Personal	Gender	Height_cm	Weight_kg
28	P29	Female	163	159
29	P30	Male	179	152
34	P35	Female	157	153
36	P37	Female	197	114

02

```

1 # toán tử ~ - Not
2 df_p3 = df_bmi[~(df_bmi.Weight_kg<155)]
3 df_p3

```

	Personal	Gender	Height_cm	Weight_kg
28	P29	Female	163	159
65	P66	Female	179	158

03

6. Filter Data

- Sử dụng phương thức `.isin()` để kết lọc dữ liệu theo một tập hợp

```
1 #Lọc ra những người có cân nặng bằng 150, 155 và 160kg
2 # phương thức isin (tương tự như in)
3 df_p4 = df_bmi[df_bmi.Weight_kg.isin([150,155,160])]
4 df_p4
```

	Personal	Gender	Height_cm	Weight_kg
102	P103	Male	161	155
106	P107	Male	166	160
123	P124	Female	184	160
134	P135	Female	171	155
135	P136	Female	183	150



7. Tính toán đặc trưng thống kê trong DataFrame

7. Đặc trưng thống kê

- Sử dụng phương thức `.max()`, `min()`, `sum()`, `mean()`, `median()`, `cumsum()`, `std()` để tính các đặc trưng thống kê cho DataFrame hoặc theo từng cột.

```

1 #tìm Max, Min của thuộc tính cân nặng
2 w_max = df_bmi['Weight_kg'].max()
3 w_min = df_bmi['Weight_kg'].min()
4 print('Cân nặng lớn nhất:',w_max, '(kg)')
5 print('Cân nặng nhỏ nhất:',w_min, '(kg)')

```

Cân nặng lớn nhất: 160 (kg)

Cân nặng nhỏ nhất: 50 (kg)

```

1 #tìm Mean, Median của chiều cao
2 h_mean = df_bmi['Height_cm'].mean()
3 h_median = df_bmi['Height_cm'].median()
4 print('Chiều cao trung bình:',h_mean, '(cm)')
5 print('Trung vị:',h_median, '(cm)')

```

Chiều cao trung bình: 169.944 (cm)

Trung vị: 170.5 (cm)

```

1 #tìm độ Lệch chuẩn của chiều cao, cân nặng
2 h_std = df_bmi['Height_cm'].std()
3 w_std = df_bmi['Weight_kg'].std()
4 print('sdt của chiều cao:', h_std)
5 print('sdt của cân nặng:', w_std)

```

sdt của chiều cao: 16.37526067959376

sdt của cân nặng: 32.38260746964435

8. Xác định giá trị duy nhất (Unique)

8. Unique

- **df.unique():** liệt kê danh sách các giá trị khác nhau trong một cột dữ liệu của DataFrame.
- **df.value_counts():** Tính tổng số theo từng giá trị khác nhau trong một cột dữ liệu của DataFrame. Kết quả là một đối tượng series.

```

1 #Xác định giá trị duy nhất trong một cột
2 df_bmi['Gender'].unique()

```

```
array(['Male', 'Female'], dtype=object)
```

```

1 #Thống kê số Lượng theo giá trị duy nhất
2 unique_gender = df_bmi['Gender'].value_counts()
3 unique_gender

```

```

Female    255
Male     245
Name: Gender, dtype: int64

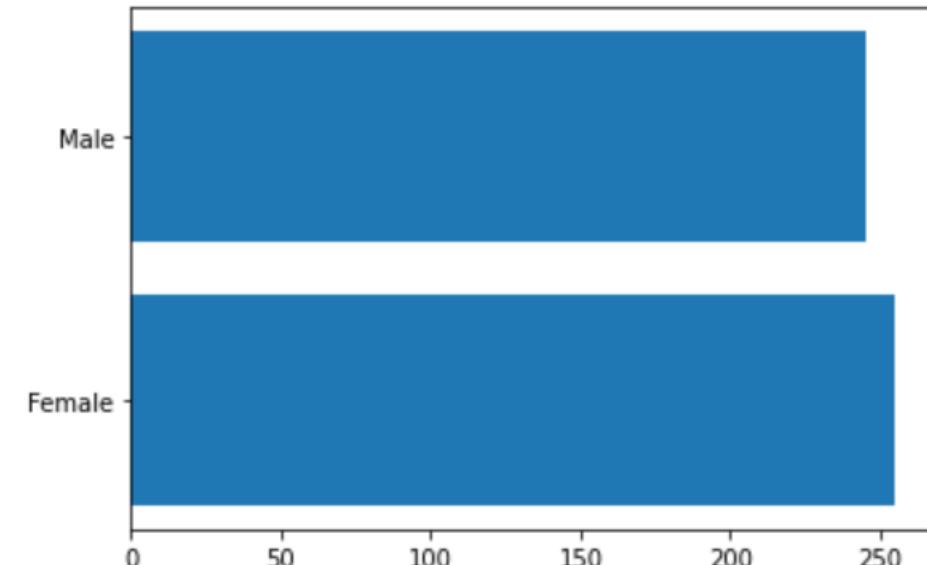
```

```

1 #Vẽ đồ thị thể hiện kết quả
2 plt.barh(unique_gender.index, unique_gender.values)

```

<BarContainer object of 2 artists>



Thực hành 4



9. Phân tích dữ liệu Time series (Tiếp cận từ bài toán thực tế)

Mô tả bài toán

Fremont Bridge

[Trang web](#) [Chỉ đường](#) [Lưu](#)

4,4 ★★★★★ 127 đánh giá trên Google

Cầu ở Portland, Oregon

Được dịch từ tiếng Anh - Cầu Fremont là cây cầu vòm bằng thép bắc qua sông Willamette nằm ở Portland, Oregon, Hoa Kỳ. Nó mang giao thông Interstate 405 và US 30 giữa trung tâm thành phố và Bắc Portland, nơi giao nhau với Xa lộ liên tiểu bang 5.

[Wikipedia \(tiếng Anh\)](#)

[Xem mô tả gốc](#) ▾

Địa chỉ: Stadium Fwy, Portland, OR 97232, Hoa Kỳ

Chiều cao: 116 m

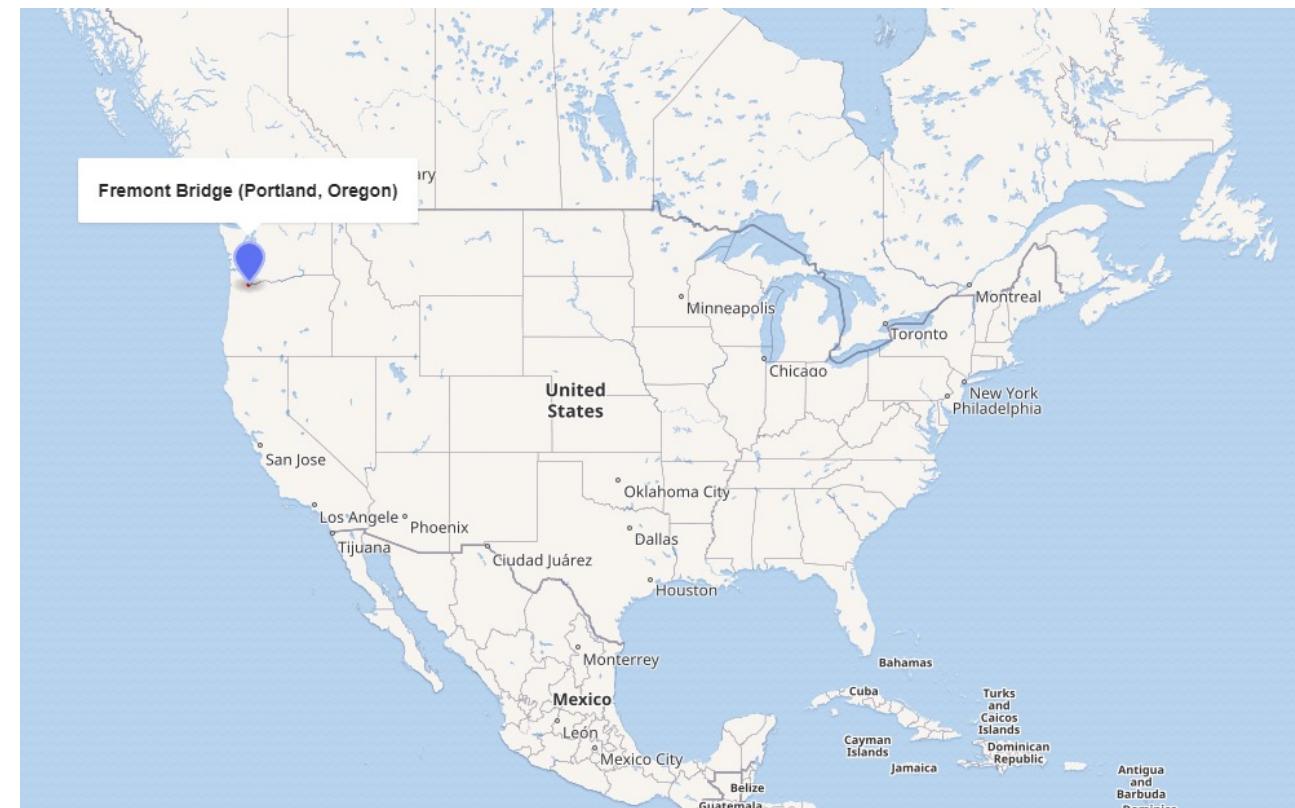
Bắt đầu xây dựng: 1968

Tổng chiều dài: 656 m

Khoảng hở bên dưới: 53 m



- + Người ta lắp đặt thiết bị để đếm số lượng xe đạp đi qua chiều phía đông và phía tây của cây cầu Fremont Bridge theo từng giờ.
- + Chi tiết: <https://data.seattle.gov/Transportation/Fremont-Bridge-Bicycle-Counter/65db-xm6k>





Tập dữ liệu

- Tập dữ liệu là số lượng xe đạp đi qua cây cầu Fremont Bridge. Dữ liệu này được thu thập tự động thông qua các cảm biến ở 2 lối đi bộ ở phía đông và phía tây của cây cầu. Số lượng xe đạp được tổng hợp theo từng giờ.
- Tập dữ liệu bao gồm 4 cột:
 - Date: Thời gian (ngày - giờ): 10/03/2012 12:00:00 AM (Kiểu thời gian)
 - Fremont Bridge Total: Tổng số xe đạp đi theo cả 2 lối đông và tây (Kiểu số nguyên)
 - Fremont Bridge East Sidewalk: Số xe đạp đi qua lối phía đông của cầu tương ứng với thời gian (Kiểu số nguyên)
 - Fremont Bridge West Sidewalk: Số xe đạp đi qua lối phía tây của cầu tương ứng với thời gian (Kiểu số nguyên)



Date	Fremont Bridge Total	Fremont Bridge East Sidewalk	Fremont Bridge West Sidewalk
10/03/2012 12:00:00 AM	13	4	9
10/03/2012 01:00:00 AM	10	4	6
10/03/2012 02:00:00 AM	2	1	1
10/03/2012 03:00:00 AM	5	2	3
10/03/2012 04:00:00 AM	7	6	1
10/03/2012 05:00:00 AM	31	21	10
10/03/2012 06:00:00 AM	155	105	50
10/03/2012 07:00:00 AM	352	257	95
10/03/2012 08:00:00 AM	437	291	146
10/03/2012 09:00:00 AM	276	172	104
10/03/2012 10:00:00 AM	118	72	46
10/03/2012 11:00:00 AM	42	10	32
10/03/2012 12:00:00 PM	76	35	41
10/03/2012 01:00:00 PM	90	42	48
10/03/2012 02:00:00 PM	128	77	51
10/03/2012 03:00:00 PM	164	72	92
10/03/2012 04:00:00 PM	315	133	182

Mục tiêu

- Phân tích dữ liệu chuỗi thời gian (Time Series Data) sử dụng Pandas.
- Kết hợp với các biểu đồ để tìm ra được những Insight ẩn chứa trong tập dữ liệu.

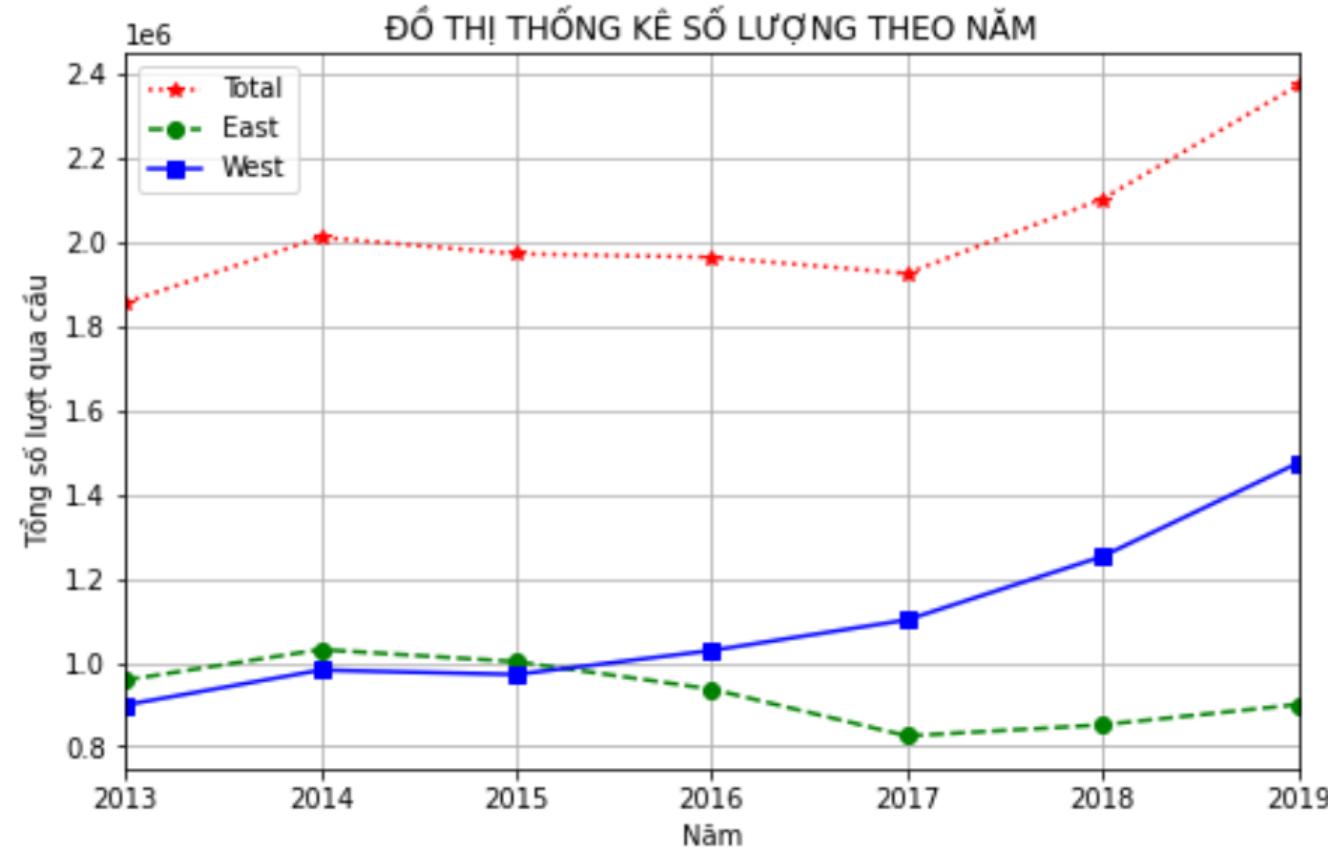
INSIGHT



Kết Quả

• 1) Tùy biến đồ thống kê tổng số xe đạp qua cầu theo năm ta thấy:

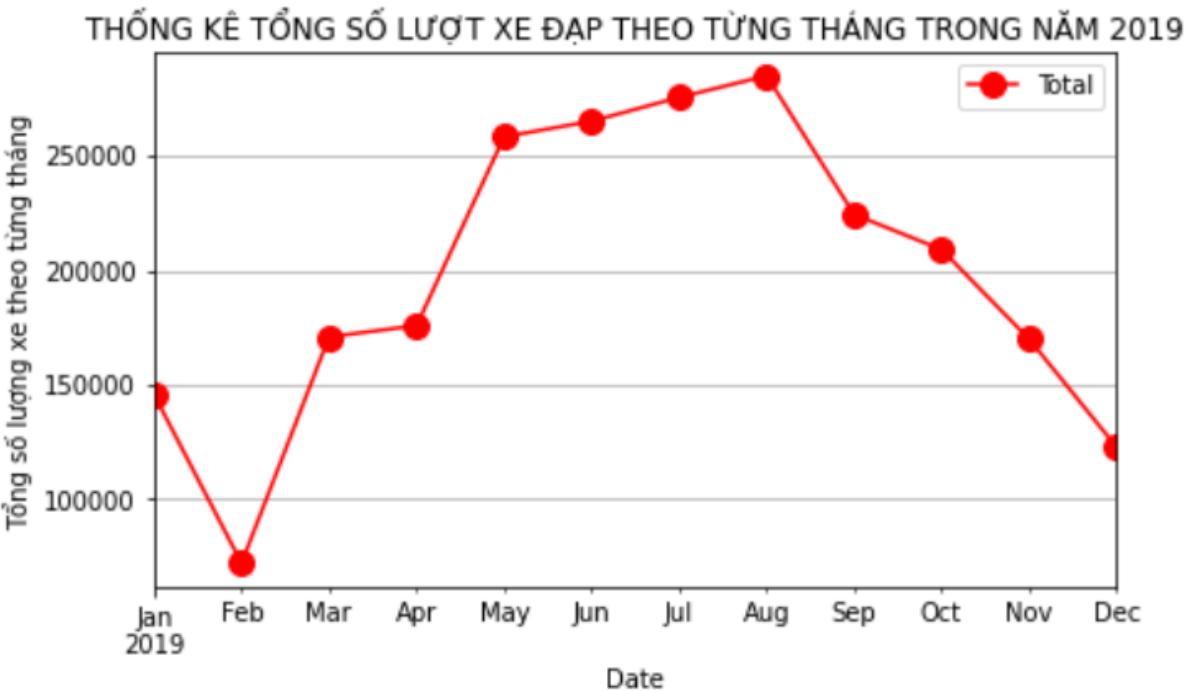
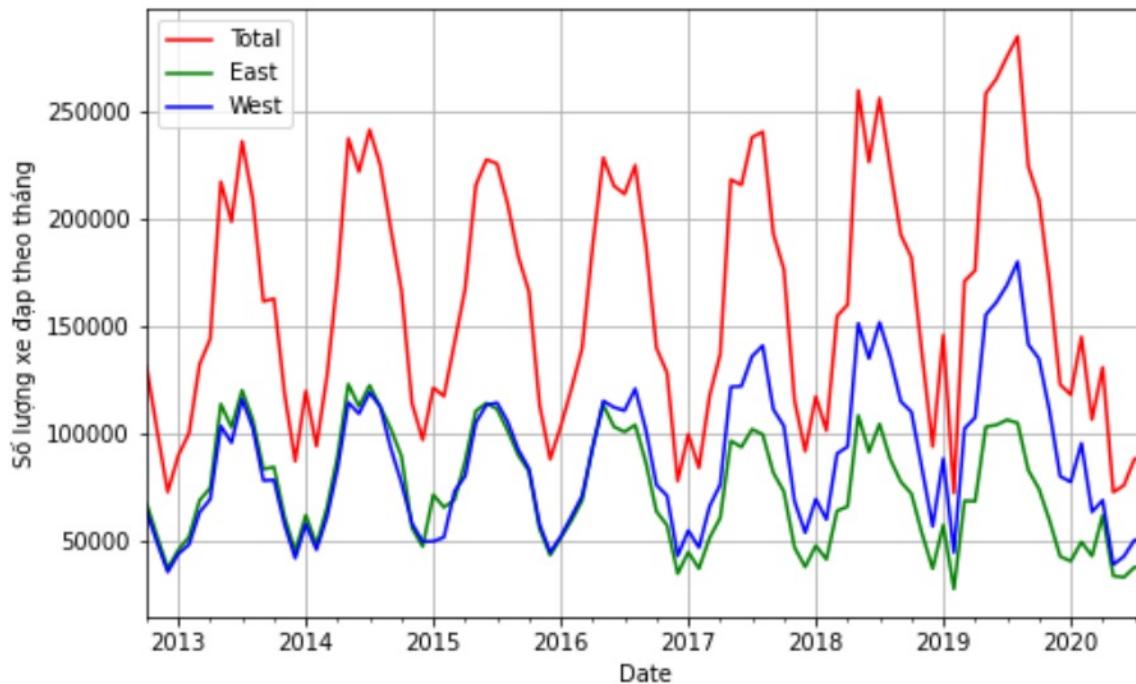
- Số lượng người đi xe đạp qua cầu Fremont có xu hướng tăng lên theo từng năm, những năm gần đây tăng nhanh.
- Lượt xe đạp qua lối đi phía tây nhiều hơn lối đi phía đông, và cũng có xu hướng tăng nhanh trong những năm gần đây.



Kết Quả

• 2) Từ biểu đồ thể hiện lượng xe đạp qua cầu theo tháng ta thấy:

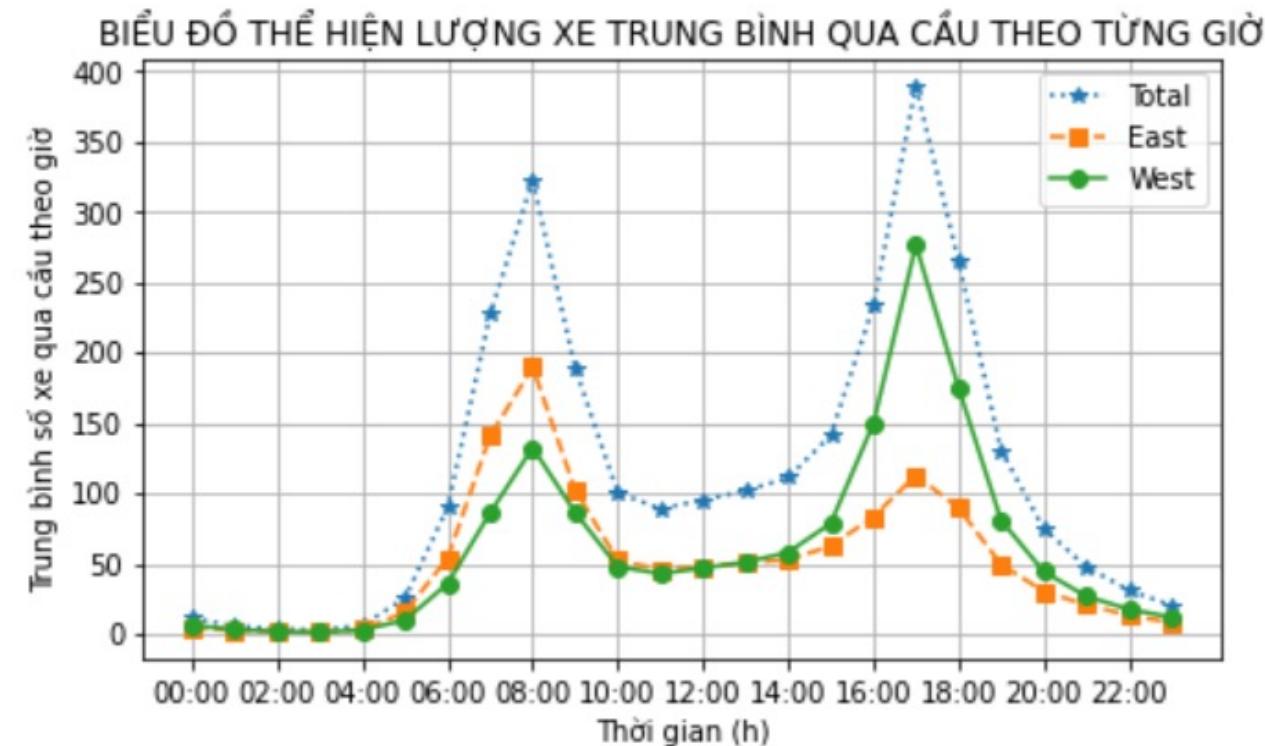
- Mọi người đạp xe nhiều hơn vào các tháng mùa hè và ít hơn vào các tháng mùa đông (4 tháng có số lượng người đạp xe nhiều nhất: 5, 6, 7 và 8)
- Dữ liệu chuỗi thời gian về lượng xe đạp qua cầu có tính xu hướng (tăng dần) và tính thời vụ (số lượng nhiều hơn vào các tháng mùa hè và ít hơn vào các tháng mùa đông)



Kết Quả

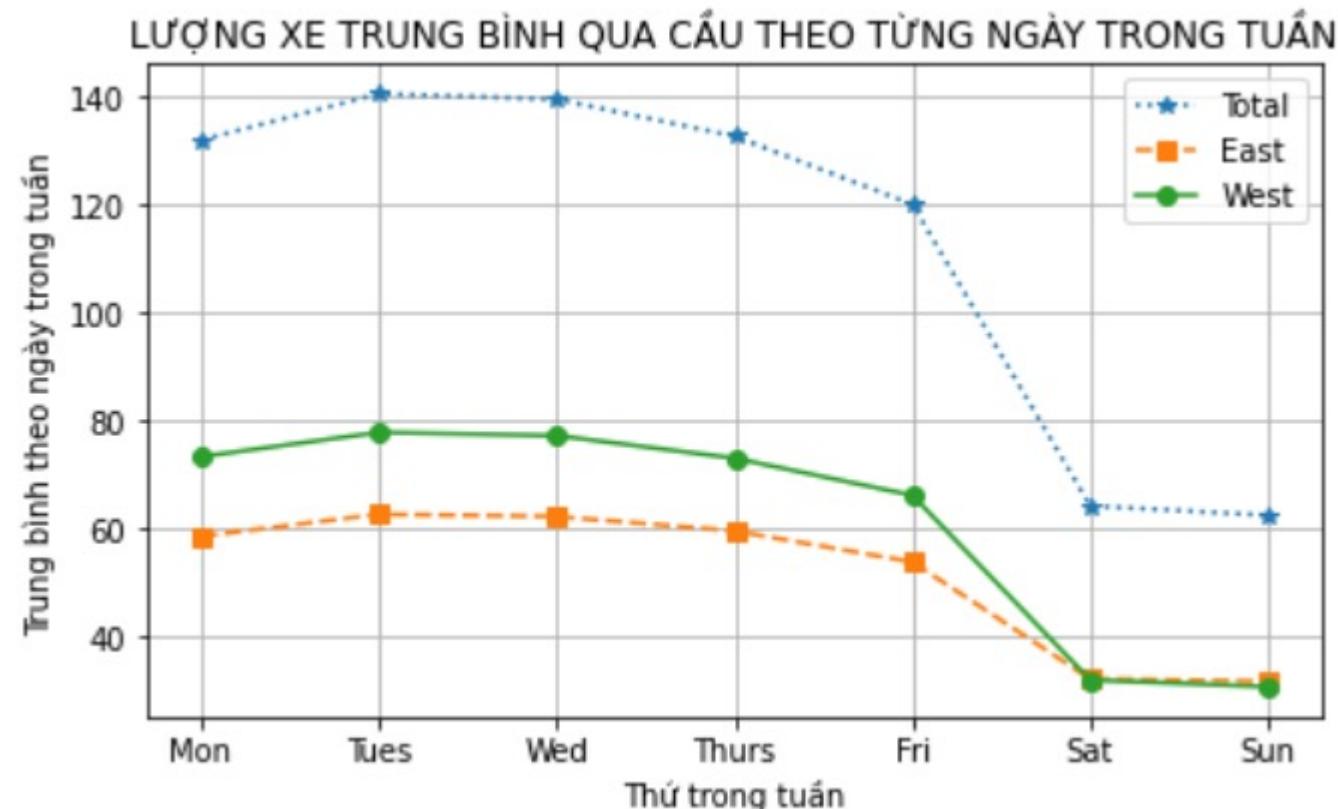
- 3) Tùy biến đồ thể hiện lượng xe đạp qua cầu trung bình theo giờ ta thấy:**

- Lượng người đi xe đạp qua cầu chủ yếu tập trung vào thời điểm 7,8,9h buổi sáng | 16, 17, 18h buổi chiều
- Lượng người đi nhiều nhất vào thời điểm 8(h) sáng - 17h chiều.
- Thời điểm buổi sáng lượng người đi qua cầu làn phía Đông (East) Lớn hơn làn phía Tây (Đi từ bên ngoài bào trung tâm thành phố Seattle) | Buổi chiều lượng người đi qua cầu làn phía Tây (West) lớn hơn (đi ra khỏi trung tâm thành phố).



Kết Quả

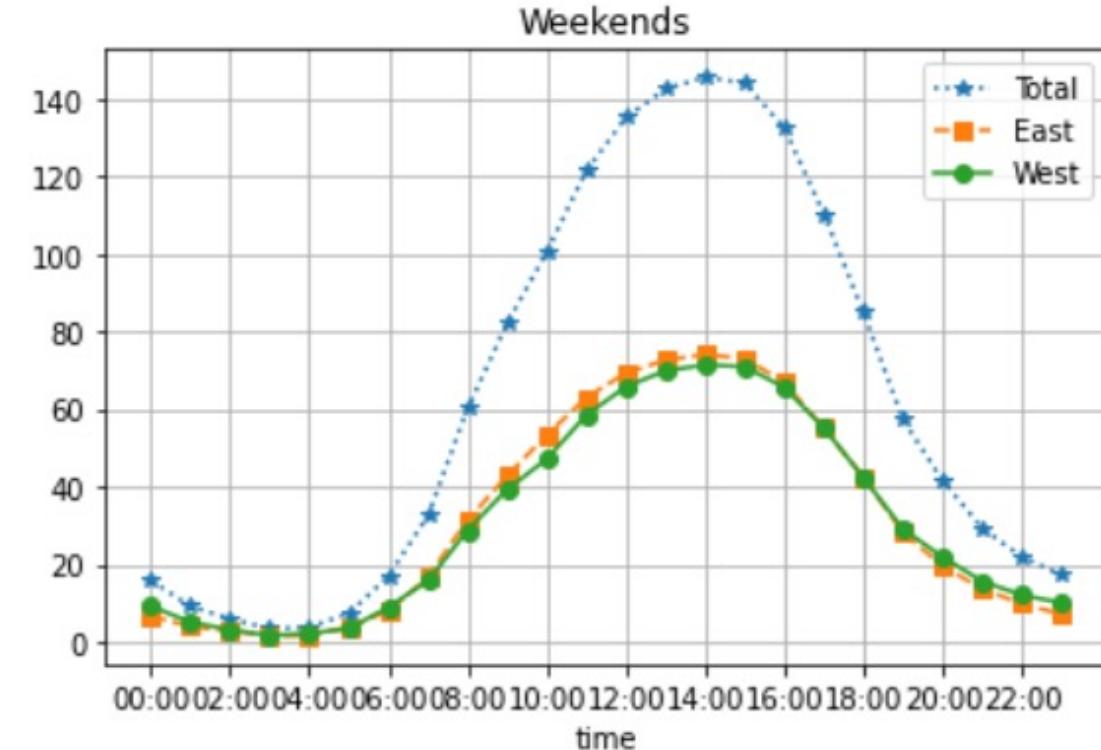
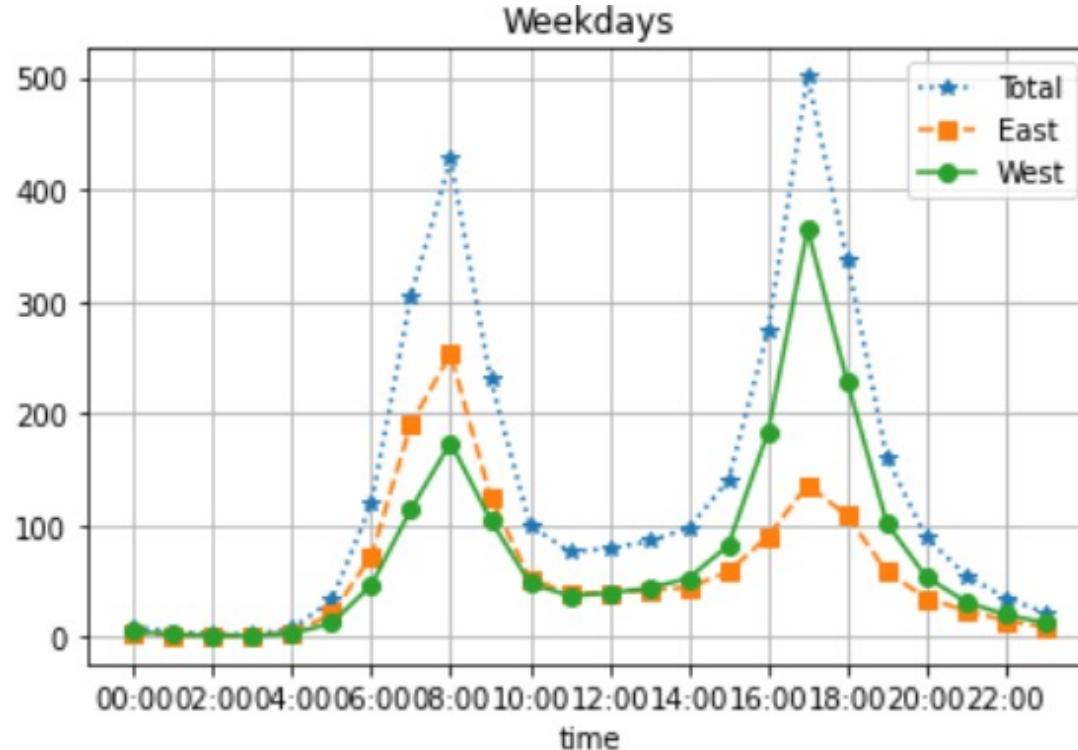
- 4) **Tùy biến đồ thị thể hiện lượng xe đạp qua cầu trung bình theo ngày trong tuần:**
 - Lượng người đi xe đạp qua cầu chủ yếu vào các ngày làm việc trong tuần [thứ 2 --> thứ 6]; Cuối tuần [Thứ 7, CN] lượng người đi qua cầu giảm đi đáng kể. Lượng người đi qua cầu ngày làm việc gấp đôi ngày cuối tuần.



Kết Quả

- 5) **Tùy biến đồ thị thể hiện lượng xe đạp qua cầu trung bình theo các ngày trong tuần và các ngày cuối tuần theo từng giờ ta thấy:**

- Vào các ngày làm việc trong tuần lượng người đi xe đạp qua cầu chủ yếu tập trung vào thời điểm 7,8,9h buổi sáng | 16, 17, 18h buổi chiều. Lượng người đi nhiều nhất vào thời điểm 8(h) sáng - 17h chiều.
- Vào các ngày cuối tuần, người đi xe đạp chủ yếu qua cầu trong thời gian từ 12-16h



Thực hành 5



Q & A
Thank you!

Bài 7:

LẬP TRÌNH PYTHON CƠ BẢN

(Phân tích và xử lý dữ liệu với Pandas - 02)

AI Academy Vietnam

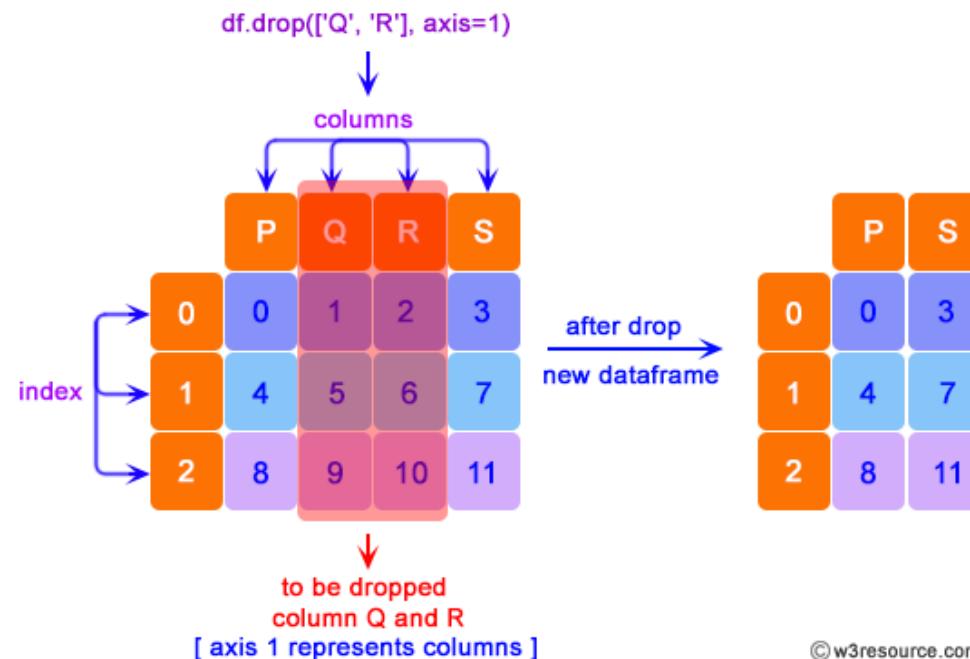
1. Xóa cột/hàng trong Dataframe
2. Xử lý hàng trùng lặp (Duplicate rows)
3. Sắp xếp trong DataFrame (Sort)
4. Nhóm các hàng trong DataFrame dựa vào giá trị
5. Áp dụng hàm cho các phần tử trong DataFrame (Apply)
6. Trộn các DataFrame (Merging)
7. Ghép nối các DataFrame (Concatenating)
8. Phát hiện và xử lý giá trị khuyết thiếu (Missing)
9. Phát hiện và xử lý giá trị ngoại biên (Outliers)
10. Phân tích dữ liệu bán hàng (Tiếp cận từ bài toán thực tế)



1. Xóa cột/hàng trong DataFrame

1.1 Xóa cột

- `df.drop([column_name], axis=1 | 'columns', inplace=True | Flase)`: Để xóa 1 cột hoặc nhiều cột trong một DataFrame.
- **Lưu ý khi sử dụng tham số `inplace = True` → Áp dụng thay đổi cho chính DataFrame hiện tại**



```

1 #Xử dụng .drop(axis=1/columns) để xóa cột
2 #Xóa một số cột trong df_loan
3 df_loan1 = df_loan.drop(['annual_inc',
4                           'dti','delinq_2yrs',
5                           'revol_util',
6                           'longest_credit_length',
7                           'verification_status'], axis=1)
8 df_loan1.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163987 entries, 0 to 163986
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   loan_amnt        163987 non-null   int64  
 1   term              163987 non-null   object  
 2   int_rate          163987 non-null   float64 
 3   emp_length        158183 non-null   float64 
 4   home_ownership    163987 non-null   object  
 5   purpose            163987 non-null   object  
 6   addr_state         163987 non-null   object  
 7   total_acc          163958 non-null   float64 
 8   bad_loan           163987 non-null   int64  
dtypes: float64(3), int64(2), object(4)
memory usage: 11.3+ MB

```

1.1 Xóa cột

- df.drop(df.columns[index], axis=1 | columns): Để xóa 1 cột hoặc nhiều cột trong một DataFrame trong trường hợp DataFrame không có tên cột, sử dụng chỉ số cột.**

```

1 #Xóa cột trong một DataFrame sử dụng chỉ số cột
2 df_loan2 = df_loan.drop(df_loan.columns[[5,8,9,10,13,14]],
3                         axis='columns')
4 df_loan2.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163987 entries, 0 to 163986
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   loan_amnt        163987 non-null   int64  
 1   term              163987 non-null   object  
 2   int_rate          163987 non-null   float64 
 3   emp_length        158183 non-null   float64 
 4   home_ownership    163987 non-null   object  
 5   purpose            163987 non-null   object  
 6   addr_state         163987 non-null   object  
 7   total_acc          163958 non-null   float64 
 8   bad_loan           163987 non-null   int64  
dtypes: float64(3), int64(2), object(4)
memory usage: 11.3+ MB

```

1.2 Xóa hàng

Xóa hàng theo index:

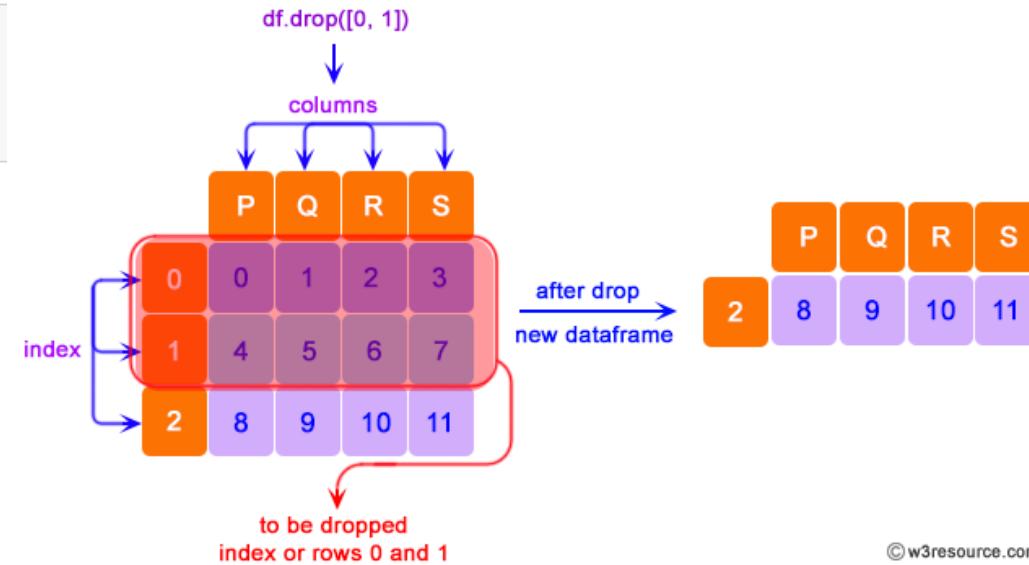
- **df.drop([index rows], axis=0):** Để xóa 1 hàng hoặc nhiều hàng trong một DataFrame theo index của hàng.
- Tham số axis = 0 (Default)

```

1 #Xóa hàng trong một DataFrame
2 #Xóa hàng có index: 3,9
3 df_loan2.drop([3,9], inplace=True)
4 df_loan2.head(10)

```

	loan_amnt	term	int_rate	emp_length	home_ownership	purpose	addr_state	total_acc	bad_loan
0	5000	36 months	10.65	10.0	RENT	credit_card	AZ	9.0	0
1	2500	60 months	15.27	0.0	RENT	car	GA	4.0	1
2	2400	36 months	15.96	10.0	RENT	small_business	IL	10.0	0
4	5000	36 months	7.90	3.0	RENT	wedding	AZ	12.0	0
5	3000	36 months	18.64	9.0	RENT	car	CA	4.0	0
6	5600	60 months	21.28	4.0	OWN	small_business	CA	13.0	1
7	5375	60 months	12.69	0.0	RENT	other	TX	3.0	1
8	6500	60 months	14.65	5.0	OWN	debt_consolidation	AZ	23.0	0
10	9000	36 months	13.49	0.0	RENT	debt_consolidation	VA	9.0	1
11	3000	36 months	9.91	3.0	RENT	credit_card	IL	11.0	0



©w3resource.com

1.2 Xóa hàng

Xóa hàng theo điều kiện (filter data):

```

1 #Loại bỏ tất cả các dòng dữ liệu có addr_state = CA
2 df_loan3 = df_loan1[df_loan1.addr_state != 'CA']
3 df_loan3

```

	loan_amnt	term	int_rate	emp_length	home_ownership	purpose	addr_state	total_acc	bad_loan
0	5000	36 months	10.65	10.0	RENT	credit_card	AZ	9.0	0
1	2500	60 months	15.27	0.0	RENT	car	GA	4.0	1
2	2400	36 months	15.96	10.0	RENT	small_business	IL	10.0	0
4	5000	36 months	7.90	3.0	RENT	wedding	AZ	12.0	0
7	5375	60 months	12.69	0.0	RENT	other	TX	3.0	1
...
163982	15000	60 months	12.39	3.0	MORTGAGE	credit_card	OK	34.0	0
163983	20000	36 months	14.99	10.0	OWN	home_improvement	VA	18.0	0
163984	12825	36 months	17.14	6.0	MORTGAGE	debt_consolidation	TX	24.0	0
163985	27650	60 months	21.99	0.0	RENT	credit_card	NY	20.0	0
163986	17000	60 months	15.99	10.0	MORTGAGE	debt_consolidation	PA	28.0	0

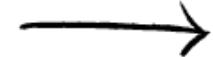
135285 rows × 9 columns

2. Xử lý hàng trùng lặp

2. Xử lý các hàng trùng lặp



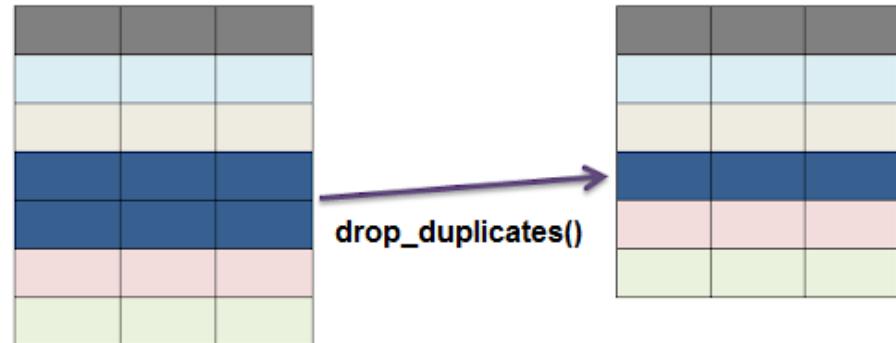
	Pet	Color	Eyes
0	Cat	Brown	Black
1	Dog	Golden	Black
2	Dog	Golden	Black
3	Dog	Golden	Brown
4	Cat	Black	Green



	Pet	Color	Eyes
0	Cat	Brown	Black
1	Dog	Golden	Black
2	Dog	Golden	Black
3	Dog	Golden	Brown
4	Cat	Black	Green

Drop duplicates

Drop Duplicate Pandas



- `df.duplicated()`: để tìm kiếm các hàng trùng lặp

2. Xử lý các hàng trùng lặp

- `df.drop_duplicates()`: với các tham số mặc định sẽ thực hiện
 - Xóa hết các hàng dữ liệu trùng lặp nhau ở tất cả các cột
 - Giữ lại hàng đầu tiên trùng lặp

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
3	jodha	23	55.0
4	raghu	23	31.0
5	Cathrine	24	77.0
6	Alisa	26	85.0
7	Bobby	24	63.0
8	Bobby	22	42.0
9	Alisa	26	85.0
10	raghu	23	31.0
11	Cathrine	24	NaN

```
1 #Trường hợp 1:  
2 #Sử dụng df.drop_duplicates() với các tham số mặc định  
3 #--> giữ lại hàng trùng lặp đầu tiên  
4 df1 = df.drop_duplicates()  
5 df1
```

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
5	Cathrine	24	77.0
7	Bobby	24	63.0
8	Bobby	22	42.0
11	Cathrine	24	NaN

2. Xử lý các hàng trùng lặp

- **df.drop_duplicates(keep='last'):**

- Xóa hết các hàng dữ liệu trùng lặp nhau ở tất cả các cột
- Giữ lại hàng trùng lặp cuối cùng

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
3	jodha	23	55.0
4	raghu	23	31.0
5	Cathrine	24	77.0
6	Alisa	26	85.0
7	Bobby	24	63.0
8	Bobby	22	42.0
9	Alisa	26	85.0
10	raghu	23	31.0
11	Cathrine	24	Nan

```
1 #Trường hợp 2:  
2 #Sử dụng df.drop_duplicates()  
3 #với các tham số keep='Last'  
4 #Giữ Lại các hàng trùng Lặp cuối cùng  
5 df2=df.drop_duplicates(keep='last')  
6 df2
```

	Name	Age	Score
3	jodha	23	55.0
5	Cathrine	24	77.0
7	Bobby	24	63.0
8	Bobby	22	42.0
9	Alisa	26	85.0
10	raghu	23	31.0
11	Cathrine	24	NaN

2. Xử lý các hàng trùng lặp

- **df.drop_duplicates(keep=False):**

- Xóa hết các hàng dữ liệu trùng lặp nhau ở tất cả các cột, chỉ giữ lại các hàng dữ liệu không trùng lặp

	Name	Age	Score	
0	Alisa	26	85.0	3
1	raghu	23	31.0	2
2	jodha	23	55.0	1
3	jodha	23	55.0	
4	raghu	23	31.0	2
5	Cathrine	24	77.0	
6	Alisa	26	85.0	3
7	Bobby	24	63.0	
8	Bobby	22	42.0	
9	Alisa	26	85.0	3
10	raghu	23	31.0	2
11	Cathrine	24	NaN	

```
1 #Trường hợp 3:  
2 #Sử dụng df.drop_duplicates()  
3 #với các tham số keep=False  
4 #Xóa hết các hàng trùng Lặp khỏi df  
5 df3=df.drop_duplicates(keep=False)  
6 df3
```

	Name	Age	Score
5	Cathrine	24	77.0
7	Bobby	24	63.0
8	Bobby	22	42.0
11	Cathrine	24	NaN

2. Xử lý các hàng trùng lặp

- df.drop_duplicates([name columns], keep='first' | 'last' | False):
 - Xóa các hàng dữ liệu trùng lặp nhau ở các cột được chỉ định

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
3	jodha	23	55.0
4	raghu	23	31.0
5	Cathrine	24	77.0
6	Alisa	26	85.0
7	Bobby	24	63.0
8	Bobby	22	42.0
9	Alisa	26	85.0
10	raghu	23	31.0
11	Cathrine	24	NaN

01

```

1 #Trường hợp 4:
2 #Sử dụng df.drop_duplicates()
3 #Loại bỏ các hàng trùng nhau theo cột Name
4 df4=df.drop_duplicates(['Name'],keep='first')
5 df4

```

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
5	Cathrine	24	77.0
7	Bobby	24	63.0

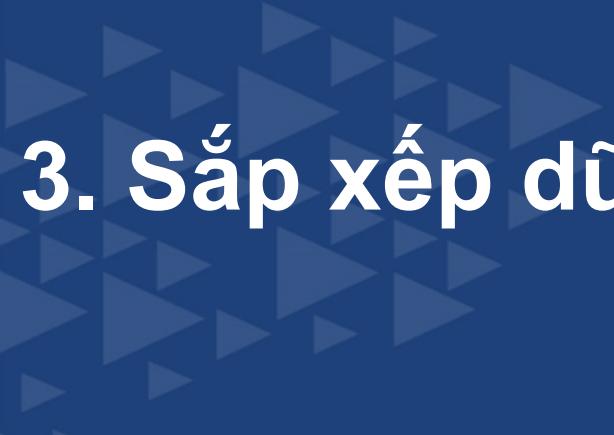
```

1 #Trường hợp 5:
2 #Sử dụng df.drop_duplicates()
3 #Loại bỏ các hàng trùng nhau theo cột Name, Age
4 df5=df.drop_duplicates(['Name','Age'],
5 keep='first')
6 df5

```

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
5	Cathrine	24	77.0
7	Bobby	24	63.0
8	Bobby	22	42.0

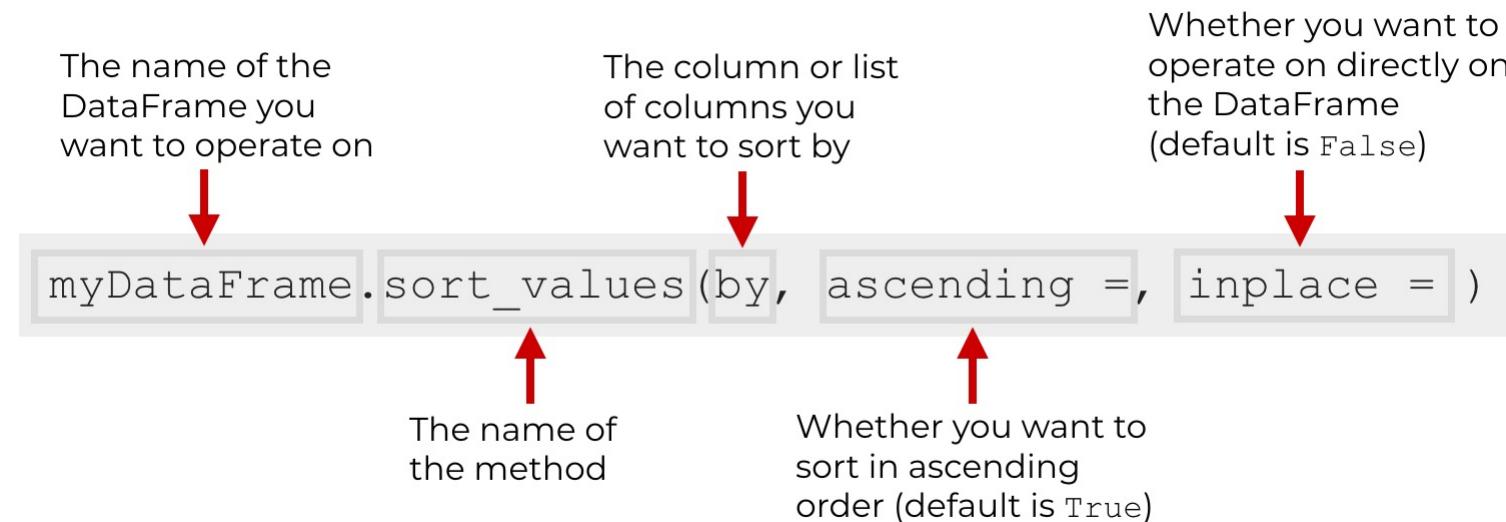
02



3. Sắp xếp dữ liệu trong DataFrame

3. Sắp xếp dữ liệu

- **df.sort_values():** sắp xếp dữ liệu trong DataFrame theo giá trị của các cột, tăng dần (**ascending = True**)-default hoặc giảm dần (**ascending=False**)
- Lưu ý khi sử dụng tham số **inplace = True**



- **df.sort_index():** sắp xếp dữ liệu trong DataFrame theo index

3. Sắp xếp dữ liệu

- Ví dụ:

	Name	Age	Score
0	Alisa	26	89
1	Bobby	27	87
2	Cathrine	25	67
3	Madonna	24	55
4	Rocky	31	47
5	Sebastian	27	72
6	Jaqluine	25	76
7	Rahul	33	79
8	David	42	44
9	Andrew	32	92
10	Ajay	51	99
11	Teresa	47	69
12	Madonna	38	73

```
1 #Trường hợp 1:  
2 #Sắp xếp dữ liệu Dataframe theo cột Score  
3 #Mặc định là sắp xếp tăng dần  
4 df.sort_values(by='Name')
```

	Name	Age	Score
10	Ajay	51	99
0	Alisa	26	89
9	Andrew	32	92
1	Bobby	27	87
2	Cathrine	25	67
8	David	42	44
6	Jaqluine	25	76
3	Madonna	24	55
12	Madonna	38	73

01

3. Sắp xếp dữ liệu

- Ví dụ:

	Name	Age	Score
0	Alisa	26	89
1	Bobby	27	87
2	Cathrine	25	67
3	Madonna	24	55
4	Rocky	31	47
5	Sebastian	27	72
6	Jaqluine	25	76
7	Rahul	33	79
8	David	42	44
9	Andrew	32	92
10	Ajay	51	99
11	Teresa	47	69
12	Madonna	38	73

1 #Trường hợp 2:
2 #Sắp xếp dữ liệu Dataframe theo cột Score
3 #Giá trị giảm dần
4 df.sort_values(by='Score', ascending=False)

	Name	Age	Score
10	Ajay	51	99
9	Andrew	32	92
0	Alisa	26	89
1	Bobby	27	87
7	Rahul	33	79
6	Jaqluine	25	76
12	Madonna	38	73
5	Sebastian	27	72

02

3. Sắp xếp dữ liệu

- Trường hợp sắp xếp nhiều cột, sẽ thực hiện sắp xếp theo thứ tự các cột từ trái sang phải:

	Name	Age	Score
0	Alisa	26	89
1	Bobby	27	87
2	Cathrine	25	67
3	Madonna	24	55
4	Rocky	31	47
5	Sebastian	27	72
6	Jaqluine	25	76
7	Rahul	33	79
8	David	42	44
9	Andrew	32	92
10	Ajay	51	99
11	Teresa	47	69
12	Madonna	38	73

```
1 #Trường hợp 3:  
2 #Sắp xếp dữ liệu Dataframe theo cột Name, Score  
3 #Giá trị tăng dần  
4 df.sort_values(by=['Name', 'Score'])
```

	Name	Age	Score
10	Ajay	51	99
0	Alisa	26	89
9	Andrew	32	92
1	Bobby	27	87
2	Cathrine	25	67
8	David	42	44
6	Jaqluine	25	76
3	Madonna	24	55
12	Madonna	38	73

03

3. Sắp xếp dữ liệu

- Ví dụ:

	Name	Age	Score
0	Alisa	26	89
1	Bobby	27	87
2	Cathrine	25	67
3	Madonna	24	55
4	Rocky	31	47
5	Sebastian	27	72
6	Jaqluine	25	76
7	Rahul	33	79
8	David	42	44
9	Andrew	32	92
10	Ajay	51	99
11	Teresa	47	69
12	Madonna	38	73

```
1 #Trường hợp 4:  
2 #Sắp xếp dữ liệu Dataframe theo cột Name, Score  
3 #Giá trị cột Name tăng dần  
4 #Giá trị cột Score giảm dần  
5 df.sort_values(by=['Name','Score'],  
6 ascending=[True,False])
```

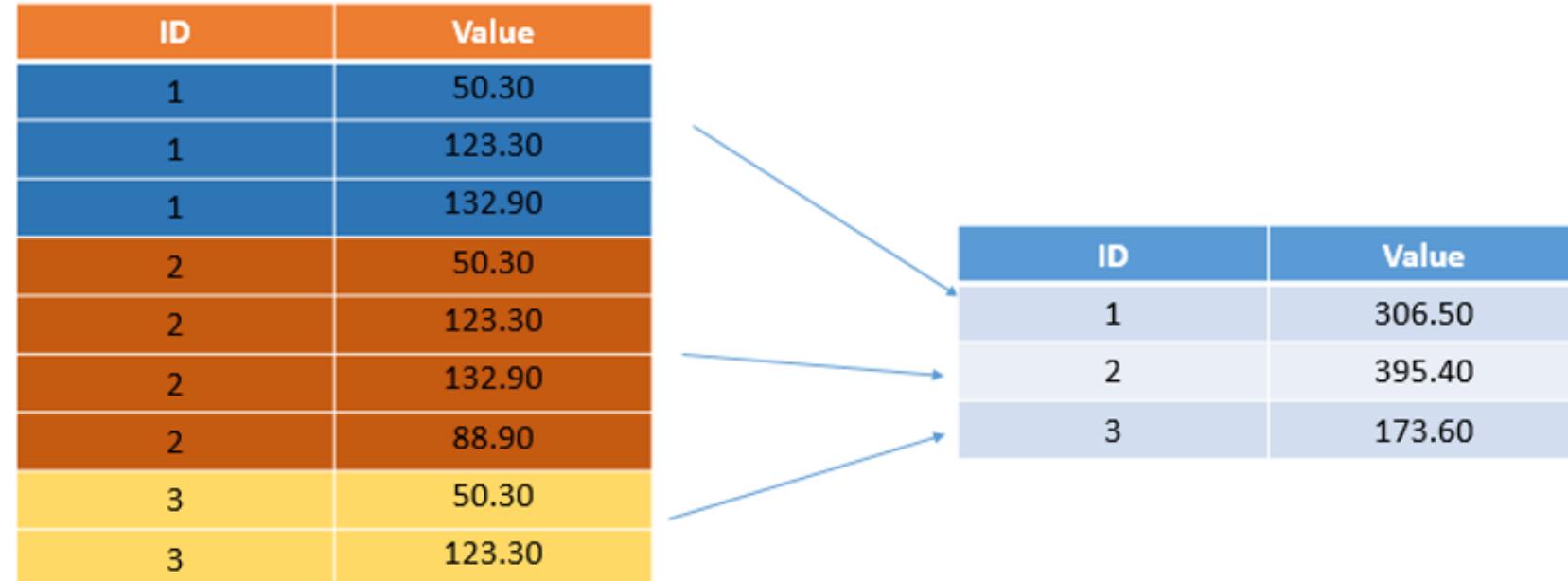
	Name	Age	Score
10	Ajay	51	99
0	Alisa	26	89
9	Andrew	32	92
1	Bobby	27	87
2	Cathrine	25	67
8	David	42	44
6	Jaqluine	25	76
12	Madonna	38	73
3	Madonna	24	55
7	Rahul	33	79
4	Rocky	31	47

04

4. Nhóm dữ liệu (groupby)

Groupby values

- df.groupby(): Gom nhóm các giá trị trong một DataFrame bởi các cột được chỉ định.
- Kết hợp với sum(), mean(), max(), min() để xác định các thông số theo từng nhóm.



The diagram illustrates the process of grouping data. On the left, there is a large table with two columns: 'ID' and 'Value'. The data is as follows:

ID	Value
1	50.30
1	123.30
1	132.90
2	50.30
2	123.30
2	132.90
2	88.90
3	50.30
3	123.30

Three blue arrows point from this large table to three smaller tables on the right, each representing a group:

ID	Value
1	306.50
2	395.40
3	173.60

Groupby values

- Sử dụng phương thức groupby():

	Name	Exam	Subject	Score
0	Alisa	Semester 1	Mathematics	62
1	Bobby	Semester 1	Mathematics	47
2	Cathrine	Semester 1	Mathematics	55
3	Alisa	Semester 1	Science	74
4	Bobby	Semester 1	Science	31
5	Cathrine	Semester 1	Science	77
6	Alisa	Semester 2	Mathematics	85
7	Bobby	Semester 2	Mathematics	63
8	Cathrine	Semester 2	Mathematics	42
9	Alisa	Semester 2	Science	67
10	Bobby	Semester 2	Science	89
11	Cathrine	Semester 2	Science	81

```

1 #Trường hợp 1:
2 #Nhóm theo tên sinh viên (Name)
3 #Thực hiện tính điểm trung bình Score
4 df['Score'].groupby([df['Name']]).mean()

```

```

Name
Alisa      72.00
Bobby      57.50
Cathrine   63.75
Name: Score, dtype: float64

```

01

```

1 #Trường hợp 2:
2 #Nhóm dữ liệu theo tên sinh viên (Name)
3 #và Bài kiểm tra (Exam)
4 #sau đó thực hiện tính tổng
5 df['Score'].groupby([df['Name'],
6                      df['Exam']]).sum()

```

```

Name      Exam
Alisa    Semester 1  136
          Semester 2  152
Bobby    Semester 1  78
          Semester 2  152
Cathrine Semester 1  132
          Semester 2  123
Name: Score, dtype: int64

```

02

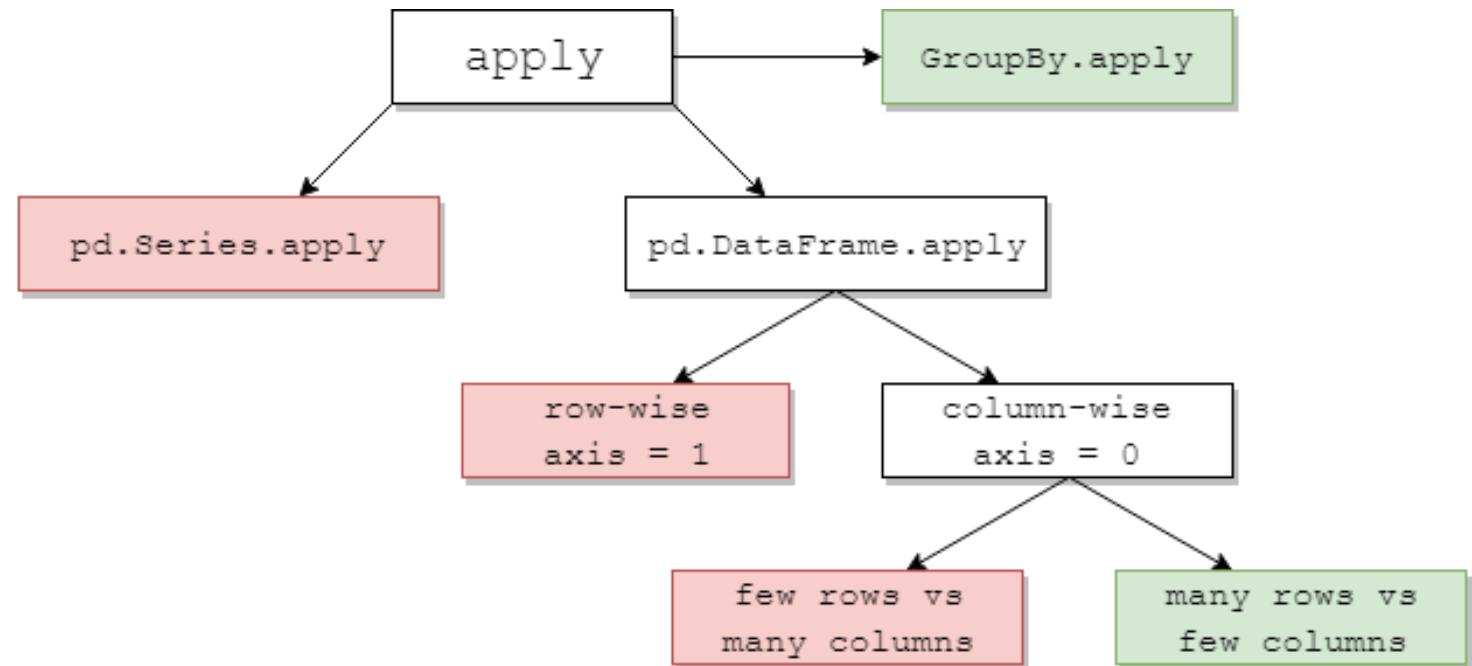
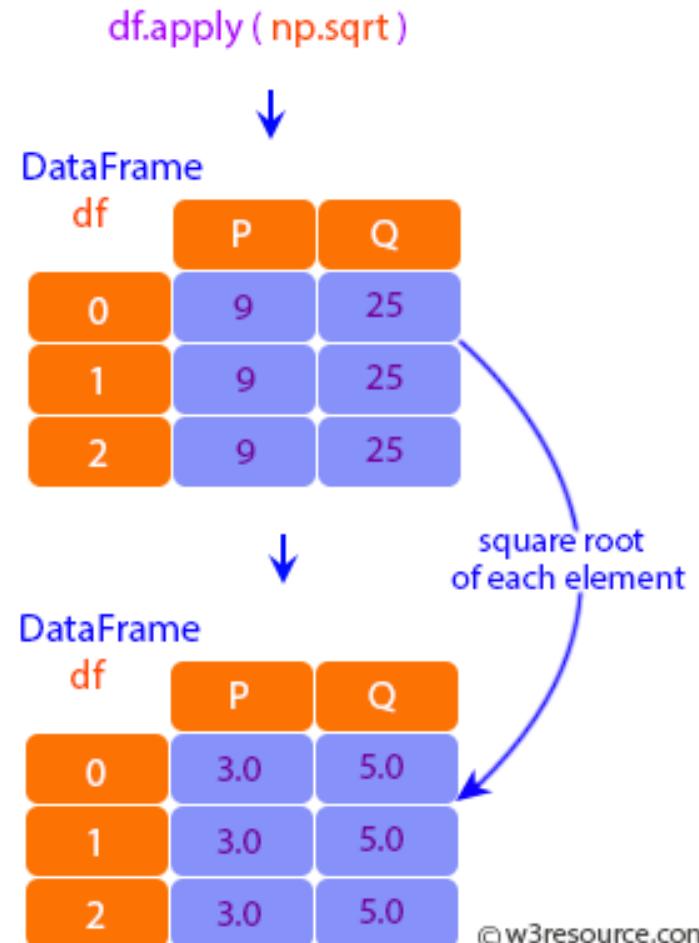
Thực hành 1

5. `apply(function)`



5 .apply(func)

- df.apply(func): Thực hiện thao tác func áp dụng cho từng cột riêng lẻ trong DataFrame, hoặc cho nhiều cột



5 .apply(func)

- Áp dụng hàm cho các phần tử trong một cột dữ liệu của DataFrame: Viết hoa các giá trị trong cột Name (3 Cách thực hiện)

	Name	Score_Math	Score_Science
0	william	66	89
1	Mason	57	87
2	ella	75	67
3	jackson	44	55
4	lincoln	31	47
5	aubrey	67	72
6	Hudson	85	76
7	christian	33	79
8	Sawyer	42	44
9	silas	62	92
10	Bennett	51	93
11	kingston	47	69

```

1 #Thực hiện: viết hoa tên học sinh
2 #Cách 1:
3 def uppercase(x):
4     return x.upper()
5
6 df[ 'Name' ] = df[ 'Name' ].apply(uppercase)
7 df

```

01

	Name	Score_Math	Score_Science
0	WILLIAM	66	89
1	MASON	57	87
2	ELLA	75	67

```

1 #Cách 2:
2 df[ 'Name' ] = df[ 'Name' ].apply(lambda x:x.upper())
3 df

```

02

	Name	Score_Math	Score_Science
0	WILLIAM	66	89
1	MASON	57	87
2	ELLA	75	67

```

1 #Cách 3:
2 df[ 'Name' ] = df[ 'Name' ].str.upper()
3 df

```

03

	Name	Score_Math	Score_Science
0	WILLIAM	66	89
1	MASON	57	87
2	ELLA	75	67

5 .apply(func)

- Áp dụng hàm cho các phần tử trong nhiều cột dữ liệu của DataFrame:
- Thực hiện tính điểm cho từng học sinh theo công thức:
 - **Point = (Score_Math*2 + Score_Science)/3**

	Name	Score_Math	Score_Science
0	william	66	89
1	Mason	57	87
2	ella	75	67
3	jackson	44	55
4	lincoln	31	47
5	aubrey	67	72
6	Hudson	85	76
7	christian	33	79
8	Sawyer	42	44
9	silas	62	92
10	Bennett	51	93
11	kingston	47	69

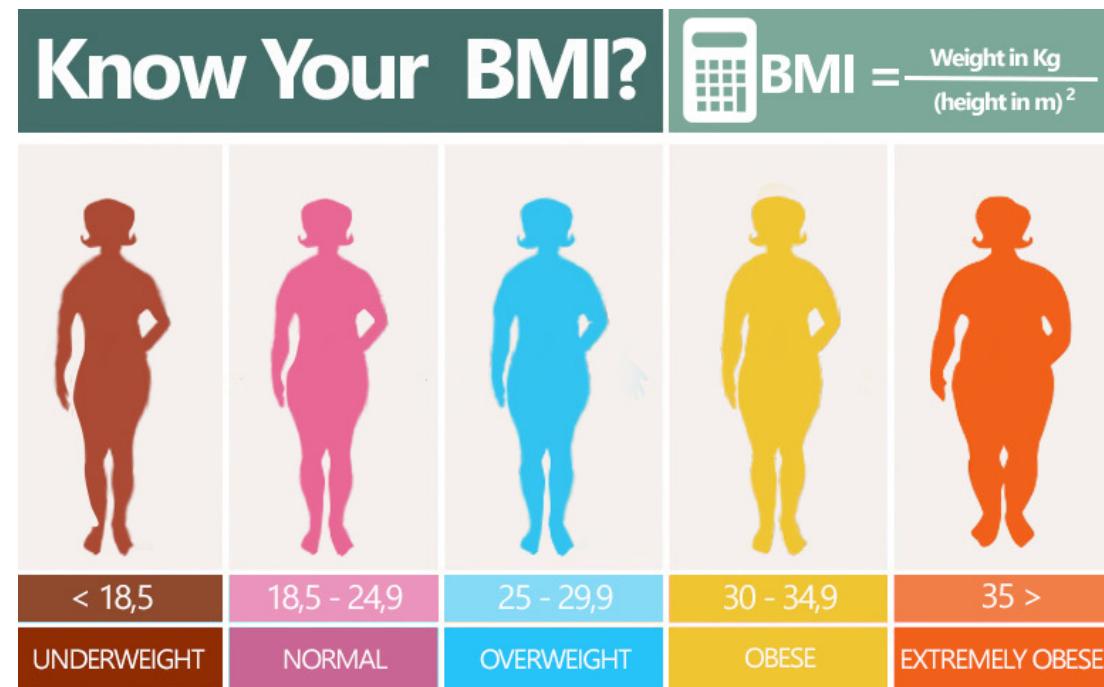
```
1 #Điểm trung bình = (score_math*2 + score_science)/3
2 #Viết hàm tính điểm trung bình
3 def mean_point(point1,point2):
4     return round((point1*2+point2)/3,1)

1 #Tạo một cột Point tính điểm của từng học sinh
2 df['Point'] = df.apply(lambda row: mean_point(row['Score_Math'],
3                                                 row['Score_Science']),
4                                                 axis=1)
5 df
```

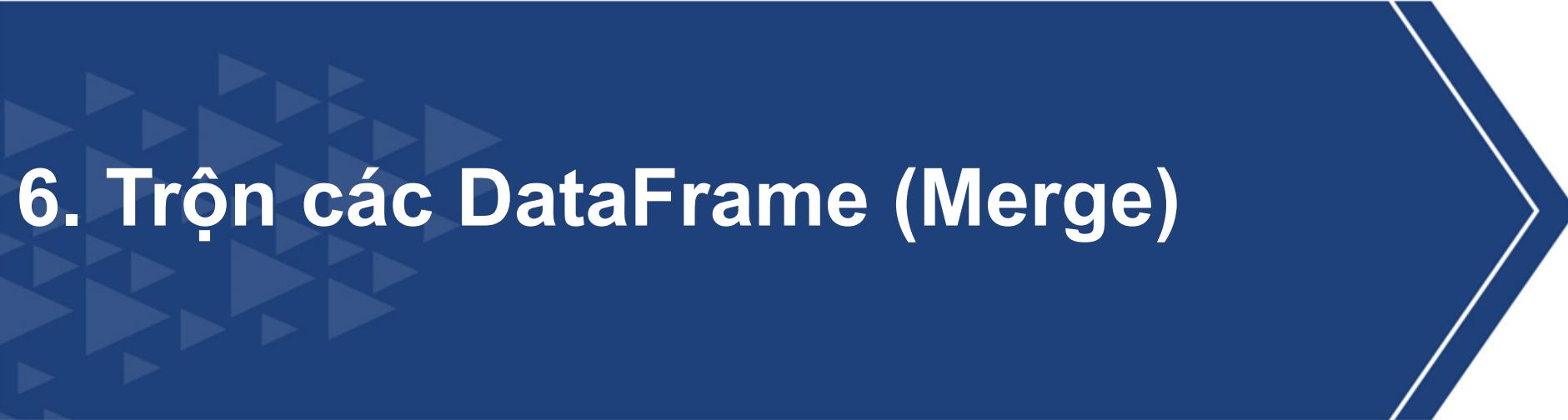
	Name	Score_Math	Score_Science	Point
0	WILLIAM	66	89	73.7
1	MASON	57	87	67.0
2	ELLA	75	67	72.3
3	JACKSON	44	55	47.7
4	LINCOLN	31	47	36.3

5 .apply(func)

- Áp dụng viết các hàm để tính chỉ số BMI, và phân loại dựa theo chỉ số tính được trên tập dữ liệu csv_Data_BMI.csv



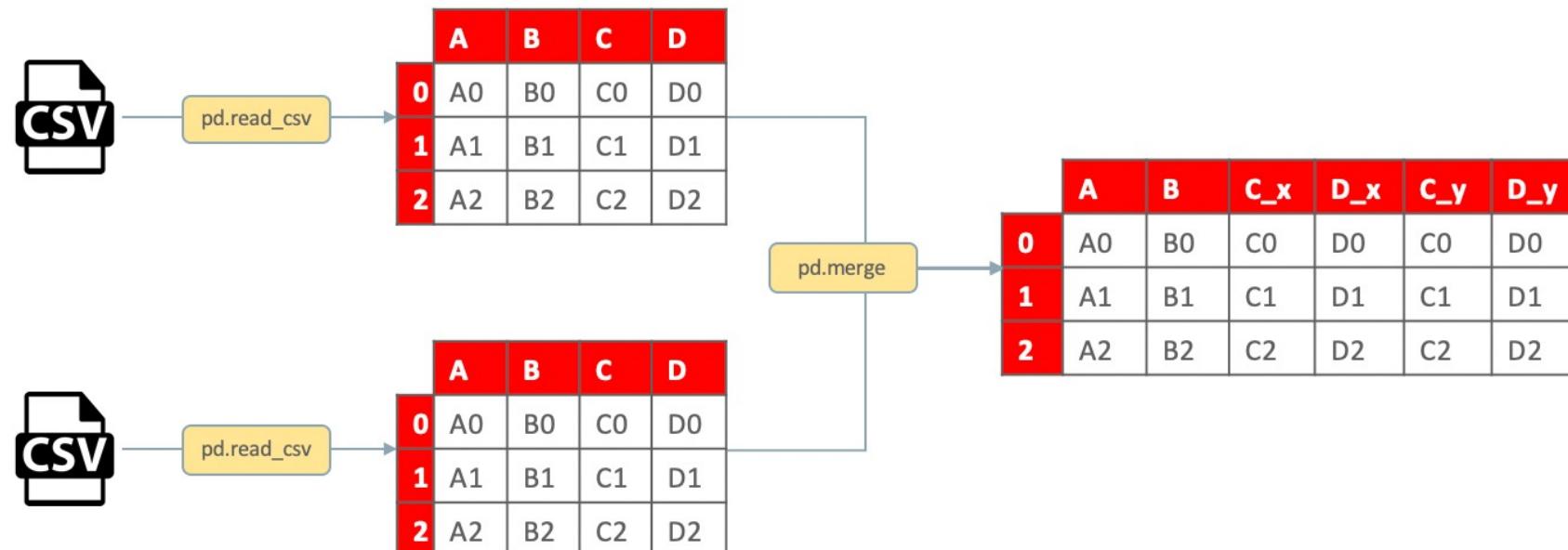
Gender	Height_cm	Weight_kg	BMI
Personal			
P1	Male	174	96 31.7
P2	Male	189	87 24.4
P3	Female	185	110 32.1
P4	Female	195	104 27.4
P5	Male	149	61 27.5



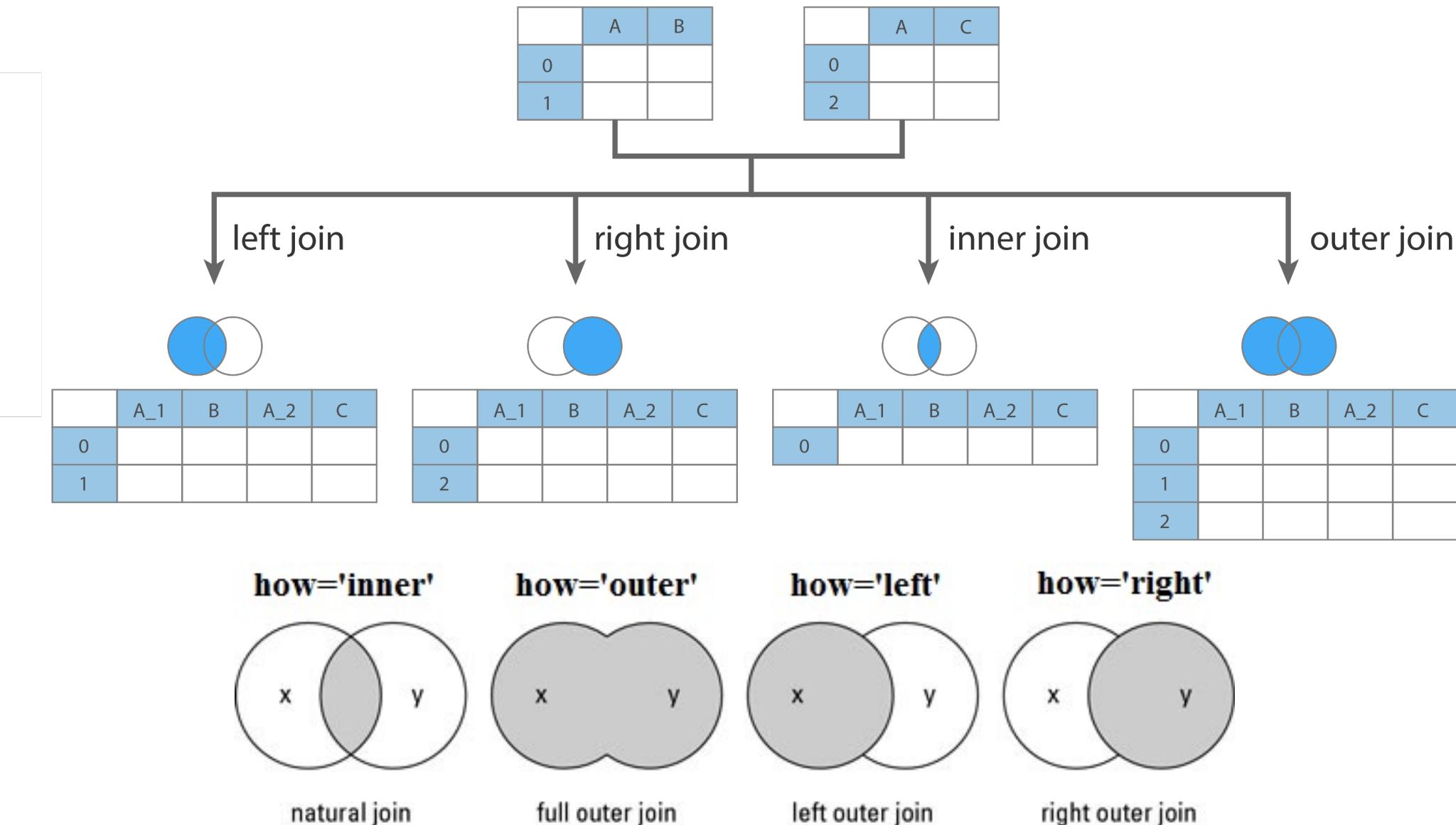
6. Trộn các DataFrame (Merge)

Trộn các DataFrame

- **pd.merge(left_df,right_df, on='key', how='left' | 'right' | 'inner' | 'outer')**: Thực hiện trộn 2 DataFrame lại với nhau.
 - Left_df: DataFrame 1
 - Right_df: DataFrame2
 - On: Tên cột dùng để nối dữ liệu giữa 2 DataFrame (tên cột phải có ở trong cả 2 DataFrame 1, 2)
 - How: Cách thức trộn dữ liệu [left, right, outer, inner (default)]



Trộn các DataFrame



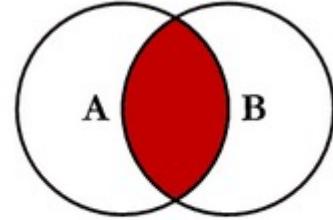
Trộn các DataFrame (inner)

	Customer_id	Product
0	1	Oven
1	2	Oven
2	3	Oven
3	4	Television
4	5	Television
5	6	Television

df1

	Customer_id	State
0	2	California
1	4	California
2	6	Texas
3	7	New York
4	8	Indiana

df2



Inner Join

```
1 #Trường hợp 1:  
2 #Inner join DataFrame  
3 inner_join_df= pd.merge(df1, df2,  
4                           on='Customer_id',  
5                           how='inner')  
6 inner_join_df
```

	Customer_id	Product	State
0	2	Oven	California
1	4	Television	California
2	6	Television	Texas

Trộn các DataFrame (outer)



VINBIGDATA VINGROUP

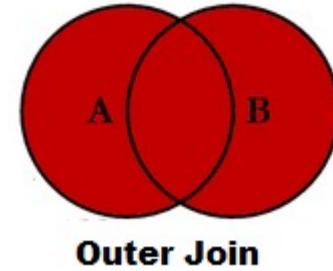
Academy Vietnam

Customer_id	Product
0	1 Oven
1	2 Oven
2	3 Oven
3	4 Television
4	5 Television
5	6 Television

df1

Customer_id	State
0	2 California
1	4 California
2	6 Texas
3	7 New York
4	8 Indiana

df2



Outer Join

```
1 #Trường hợp 2:  
2 #Outer join DataFrame  
3 inner_join_df = pd.merge(df1, df2,  
4                           on='Customer_id',  
5                           how='outer')  
6 inner_join_df
```

Customer_id	Product	State
0	1 Oven	NaN
1	2 Oven	California
2	3 Oven	NaN
3	4 Television	California
4	5 Television	NaN
5	6 Television	Texas
6	7 NaN	New York
7	8 NaN	Indiana

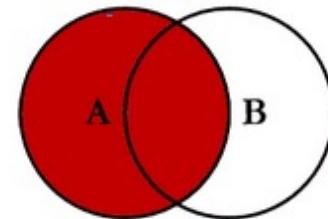
Trộn các DataFrame (left)

	Customer_id	Product
0	1	Oven
1	2	Oven
2	3	Oven
3	4	Television
4	5	Television
5	6	Television

df1

	Customer_id	State
0	2	California
1	4	California
2	6	Texas
3	7	New York
4	8	Indiana

df2



Left join

```
1 #Trường hợp 3:  
2 #Left join DataFrame  
3 inner_join_df= pd.merge(df1, df2,  
4                           on='Customer_id',  
5                           how='left')  
6 inner_join_df
```

	Customer_id	Product	State
0	1	Oven	NaN
1	2	Oven	California
2	3	Oven	NaN
3	4	Television	California
4	5	Television	NaN
5	6	Television	Texas

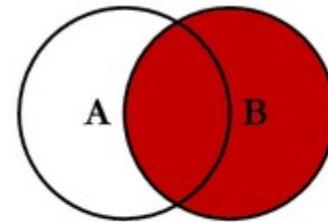
Trộn các DataFrame (right)

Customer_id	Product
0	1 Oven
1	2 Oven
2	3 Oven
3	4 Television
4	5 Television
5	6 Television

df1

Customer_id	State
0	2 California
1	4 California
2	6 Texas
3	7 New York
4	8 Indiana

df2



Right Join

```
1 #Trường hợp 4:  
2 #Right join DataFrame  
3 inner_join_df= pd.merge(df1, df2,  
4                           on='Customer_id',  
5                           how='right')  
6 inner_join_df
```

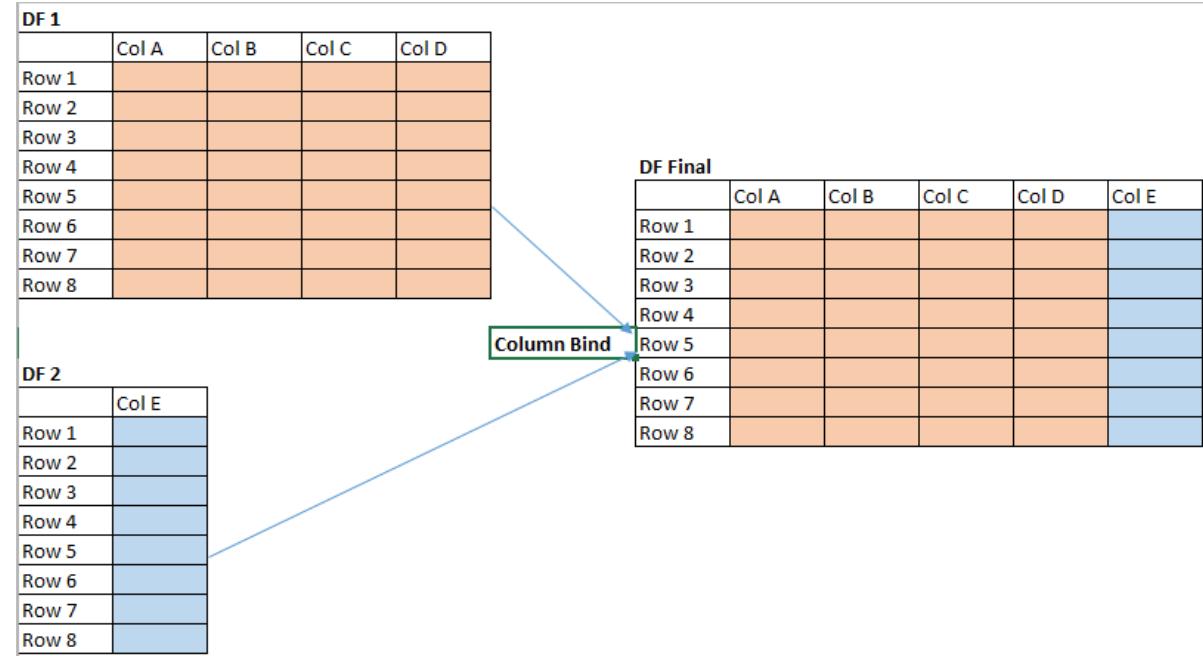
Customer_id	Product	State
0	2 Oven	California
1	4 Television	California
2	6 Television	Texas
3	7 NaN	New York
4	8 NaN	Indiana



7. Nối các DataFrame (concat, append)

Nối các DataFrame theo cột

- `pd.concat([df1,df2], axis=1, join='inner'|'outer')`: Thực hiện ghép nối các DataFrame lại với nhau theo cột



DF 1

	Col A	Col B	Col C	Col D
Row 1				
Row 2				
Row 3				
Row 4				
Row 5				
Row 6				
Row 7				
Row 8				

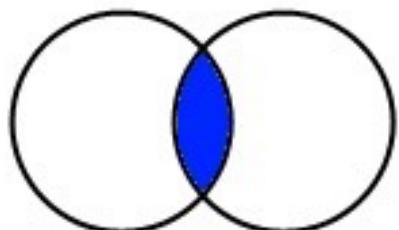
DF 2

	Col E
Row 1	
Row 2	
Row 3	
Row 4	
Row 5	
Row 6	
Row 7	
Row 8	

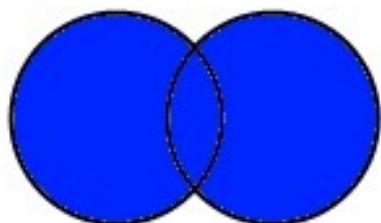
DF Final

	Col A	Col B	Col C	Col D	Col E
Row 1					
Row 2					
Row 3					
Row 4					
Row 5					
Row 6					
Row 7					
Row 8					

INNER JOIN



FULL OUTER JOIN



Nối các DataFrame theo cột

	Name	Score1	Score2
0	Alisa	62	89
1	Bobby	47	87
2	Cathrine	55	67
3	Madonna	74	55
4	Rocky	31	47
5	Sebastian	77	72
6	Jaqluine	85	76
7	Rahul	63	79
8	David	42	44

01

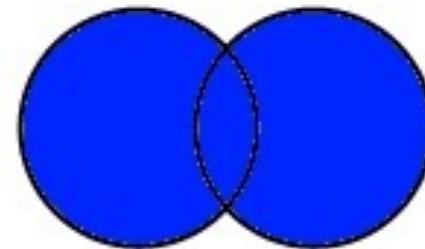
- **pd.concat([df1,df2], axis=1)**: sử dụng các tham số mặc định

```
1 #Trường hợp 1:  
2 #Mặc định join='outer'  
3 df_concat1 = pd.concat([df1, df2], axis=1)  
4 df_concat1
```

	Name	Score3
0	Alisa	56
1	Bobby	86
2	Cathrine	77
3	Madonna	45
4	Rocky	73
5	Sebastian	62
6	Jaqluine	74

02

FULL OUTER JOIN



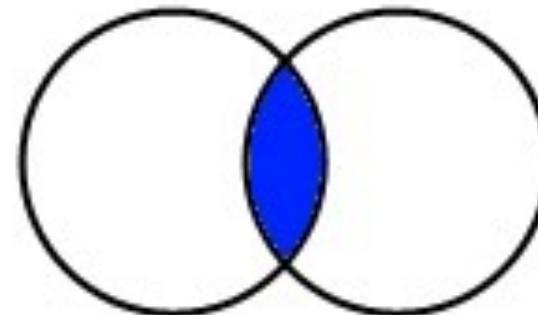
	Name	Score1	Score2	Name	Score3
0	Alisa	62	89	Alisa	56.0
1	Bobby	47	87	Bobby	86.0
2	Cathrine	55	67	Cathrine	77.0
3	Madonna	74	55	Madonna	45.0
4	Rocky	31	47	Rocky	73.0
5	Sebastian	77	72	Sebastian	62.0
6	Jaqluine	85	76	Jaqluine	74.0
7	Rahul	63	79	NaN	NaN
8	David	42	44	NaN	NaN

Nối các DataFrame theo cột

	Name	Score1	Score2
0	Alisa	62	89
1	Bobby	47	87
2	Cathrine	55	67
3	Madonna	74	55
4	Rocky	31	47
5	Sebastian	77	72
6	Jaqluine	85	76
7	Rahul	63	79
8	David	42	44

01

INNER JOIN



	Name	Score3
0	Alisa	56
1	Bobby	86
2	Cathrine	77
3	Madonna	45
4	Rocky	73
5	Sebastian	62
6	Jaqluine	74

02

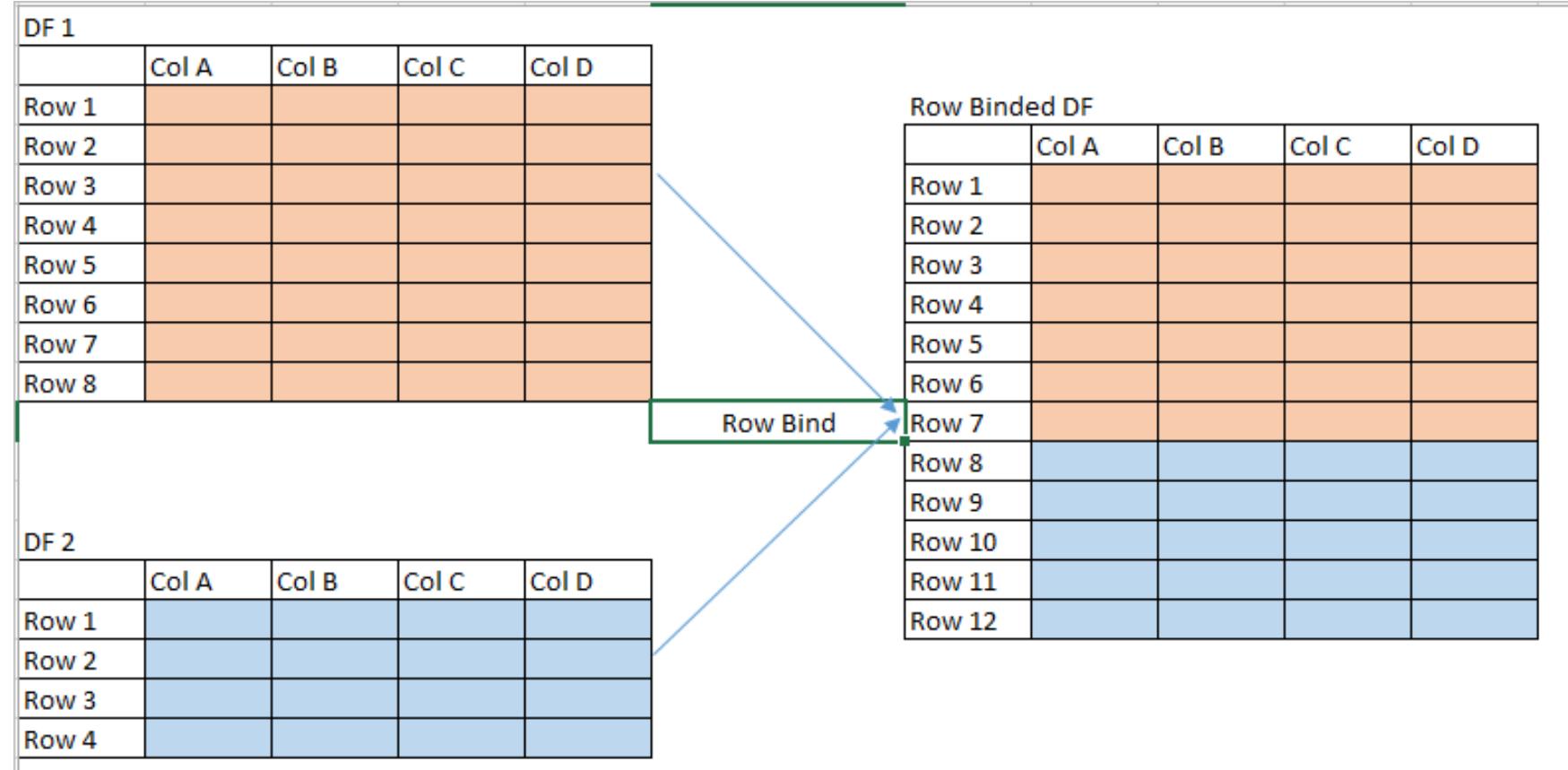
- **pd.concat([df1,df2], axis=1, join='inner'):**
Sử dụng tham số join

```
1 #Trường hợp 2:  
2 #Tham số join='inner'  
3 df_concat2 = pd.concat([df1, df2],  
4                         axis=1,  
5                         join='inner')  
6 df_concat2
```

	Name	Score1	Score2	Name	Score3
0	Alisa	62	89	Alisa	56
1	Bobby	47	87	Bobby	86
2	Cathrine	55	67	Cathrine	77
3	Madonna	74	55	Madonna	45
4	Rocky	31	47	Rocky	73
5	Sebastian	77	72	Sebastian	62
6	Jaqluine	85	76	Jaqluine	74

Nối các DataFrame theo hàng

- **pd.concat([df1,df2], axis=0, join='inner'|'outer', ignore_index=True|False)** hoặc
df1.append(df2): Thực hiện ghép nối các DataFrame lại với nhau theo hàng



Nối các DataFrame theo hàng



VINBIGDATA VINGROUP

Academy Vietnam

01

	Name	Score1	Score2	Score3
0	Alisa	62	89	56
1	Bobby	47	87	86
2	Cathrine	55	67	77
3	Madonna	74	55	45
4	Rocky	31	47	73

- **pd.concat([df1,df2]):**

```
1 #Trường hợp 1: sử dụng concat  
2 df_row = pd.concat([df1,df2])  
3 df_row
```

	Name	Score1	Score2	Score3
0	Alisa	62	89	56
1	Bobby	47	87	86
2	Cathrine	55	67	77
3	Madonna	74	55	45
4	Rocky	31	47	73
0	Andrew	32	92	67
1	Ajay	71	99	97
2	Teresa	57	69	68

Trường hợp các cột cùng tên:

02

	Name	Score1	Score2	Score3
0	Andrew	32	92	67
1	Ajay	71	99	97
2	Teresa	57	69	68

- **df1.append(df2):**

```
1 #Trường hợp 1: sử dụng append()  
2 df_row2 = df1.append(df2)  
3 df_row2
```

	Name	Score1	Score2	Score3
0	Alisa	62	89	56
1	Bobby	47	87	86
2	Cathrine	55	67	77
3	Madonna	74	55	45
4	Rocky	31	47	73
0	Andrew	32	92	67
1	Ajay	71	99	97
2	Teresa	57	69	68

Nối các DataFrame theo hàng



VINBIGDATA VINGROUP

Academy Vietnam

01

	Name	Score1	Score2	Score3
0	Alisa	62	89	56
1	Bobby	47	87	86
2	Cathrine	55	67	77
3	Madonna	74	55	45
4	Rocky	31	47	73

Trường hợp các cột khác tên:

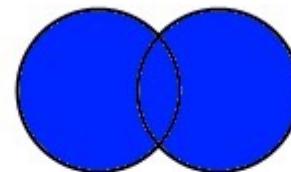
	Name	Score1	Score4	Score5
0	Jack	32	72	57
1	danny	71	91	72
2	vishwa	70	89	78

- pd.concat([df1,df3]) | df1.append(df3):

```
1 #Trường hợp các cột khác tên  
2 pd.concat([df1,df3])
```

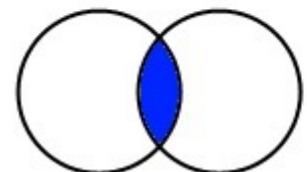
	Name	Score1	Score2	Score3	Score4	Score5
0	Alisa	62	89.0	56.0	NaN	NaN
1	Bobby	47	87.0	86.0	NaN	NaN
2	Cathrine	55	67.0	77.0	NaN	NaN
3	Madonna	74	55.0	45.0	NaN	NaN
4	Rocky	31	47.0	73.0	NaN	NaN
0	Jack	32	NaN	NaN	72.0	57.0
1	danny	71	NaN	NaN	91.0	72.0
2	vishwa	70	NaN	NaN	89.0	78.0

FULL OUTER JOIN



	Name	Score1
0	Alisa	62
1	Bobby	47
2	Cathrine	55
3	Madonna	74
4	Rocky	31
0	Jack	32
1	danny	71
2	vishwa	70

INNER JOIN



Thực hành 2



8. Phát hiện và xử lý giá trị khuyết thiếu

Phát hiện và xử lý missing data

- Vì nhiều lý do, dữ liệu có thể bị lỗi hoặc bị thiếu ở một số vị trí của một số feature.
- Hầu hết các thuật toán học máy không xử lý với dữ liệu bị thiếu, do đó cần phải được xử lý ở bước tiền xử lý dữ liệu.

Các nguyên nhân dẫn đến missing data:

- Khuyết ngẫu nhiên (Missing at Random – MAR):
- Khuyết hoàn toàn ngẫu nhiên (Missing Completely at Random – MCAR):
- Khuyết không ngẫu nhiên (Missing not at Random – MNAR):



	A	B	C	D	E	F	G
1	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
2	00 15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
3	01 15-9-2019		24.21	24.02	24.93	25.16	24.83
4	02 15-9-2019	25.05	23.73	23.89	24.79	24.8	24.55
5	03 15-9-2019	24.79	23.36	23.83		24.74	24.48
6	04 15-9-2019	24.59	23.05	23.69	24.82	24.8	24.38
7	05 15-9-2019	24.4		23.52	24.79	24.87	24.4
8	06 15-9-2019	24.38	22.79	23.68	25.1	24.71	24.41
9	07 15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
10	08 15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
11	09 15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
12	10 15-9-2019		29.97			27.68	27.53
13	11 15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
14	12 15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
15	13 15-9-2019	30.95		27.83	27.44	28	30.66
16	14 15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
17	15 15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
18	16 15-9-2019	30.8	30.2		26.45	27.29	29.13
19	17 15-9-2019	29.94	29.36	25.8	26.67	26.69	28.72
20	18 15-9-2019	28.53	27.48	24.82	25.92	25.81	27.46
21	19 15-9-2019	28.89	27.03	24.93	25.88	25.93	27.07
22	20 15-9-2019	28.06	26.41	24.7		25.97	26.75
23	21 15-9-2019	27.43	26.2	24.41	25.62	25.94	26.32
24	22 15-9-2019	26.98	25.79	24.17	25.6	25.9	26.29
25	23 15-9-2019	26.68	25.31	23.81	25.53	25.8	26.36
26							
27							

a.Phát hiện missing data

- Thống kê dữ liệu missing trong dataframe:
 - `df.isnull().sum()`

```

1 #Thống kê số lượng missing trong Data frame
2 #Theo từng cột
3 print('Số lượng missing data trong file dữ liệu:')
4 print(data_temp.isnull().sum())

```

Số lượng missing data trong file dữ liệu:

time	0
Ha Noi	2
Vinh	2
Da Nang	2
Nha Trang	3
Ho Chi Minh	0
Ca Mau	0
dtype: int64	

- Xây dựng hàm thống kê `missing_values()`

```

1 #Xây dựng hàm thống kê dữ liệu missing trong dataframe:
2 -----
3 #Đầu vào của hàm là 1 biến Dataframe
4 #Đầu ra bao gồm các thông số:
5 #Tổng số cột của file dữ liệu
6 #Tổng số cột có chứa dữ liệu missing
7 #Danh sách các cột chứa dữ liệu missing với 2 thông số:
8 #Tổng số giá trị missing tương ứng với cột đó
9 #Tỷ lệ % dữ liệu missing trên tổng số dữ liệu của cột
10 def missing_values(df):
11     mis_val = df.isnull().sum()
12     mis_val_percent = 100 * df.isnull().sum() / len(df)
13     mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
14     mis_val_table_ren_columns = mis_val_table.rename(
15         columns = {0 : 'Số giá trị Missing', 1 : 'Tỷ lệ % missing'})
16     mis_val_table_ren_columns = mis_val_table_ren_columns[
17         mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
18         'Tỷ lệ % missing', ascending=False).round(1)
19     print ("File dữ liệu bao gồm có: " + str(df.shape[1]) + " cột.\n"
20           "Có " + str(mis_val_table_ren_columns.shape[0]) +
21           " cột chứa missing values.")
22     return mis_val_table_ren_columns

```

a.Phát hiện missing data

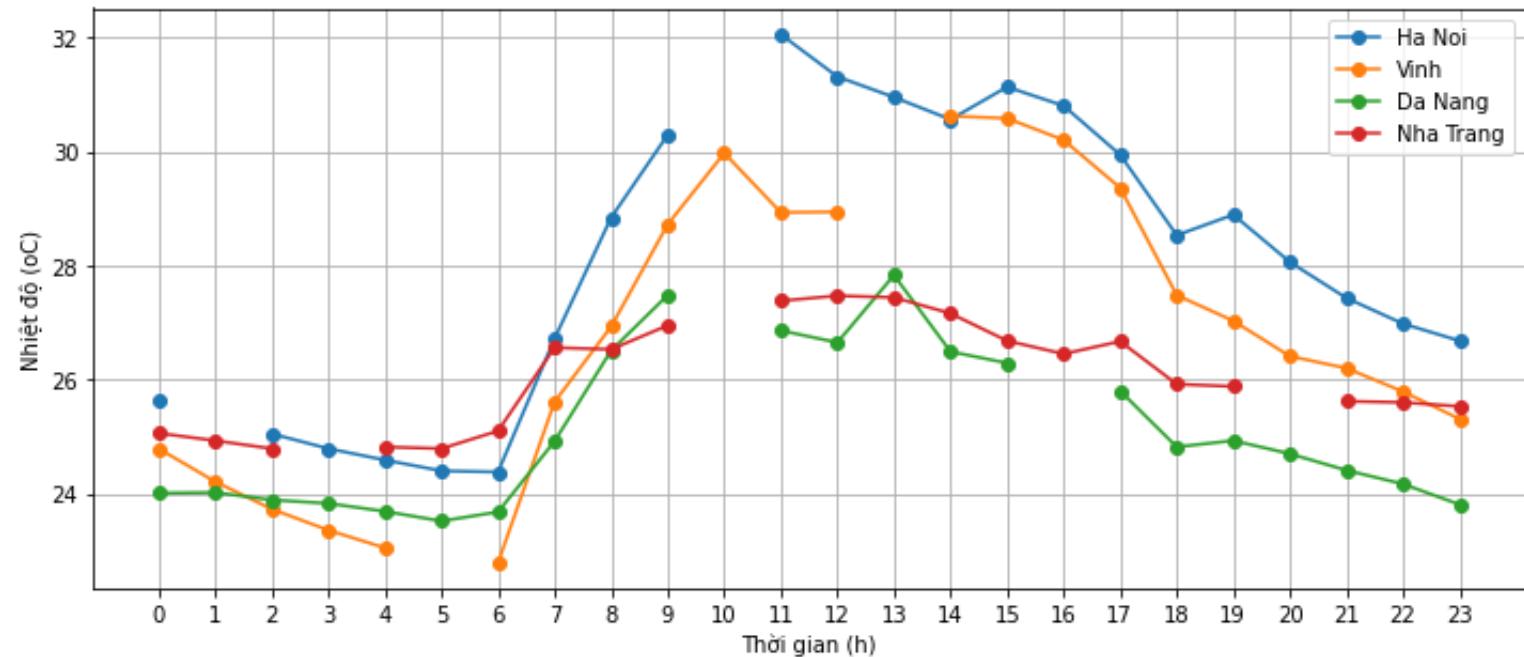
- Xây dựng hàm thống kê **missing_values()**

```
1 missing_values(data_temp)
```

File dữ liệu bao gồm có: 7 cột.

Có 4 cột chứa missing values.

Số giá trị Missing	Tỷ lệ % missing
Nha Trang	3
Ha Noi	2
Vinh	2
Da Nang	2



a. Phát hiện missing data

- Liệt kê các hàng chứa missing trong Dataframe:

```

1 #Liệt kê các dòng dữ liệu missing
2 #1. Liệt kê theo từng features:
3 data_temp[pd.isnull(data_temp['Ha Noi'])]

```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
1	01 15-9-2019	NaN	24.21	24.02	24.93	25.16	24.83
10	10 15-9-2019	NaN	29.97	NaN	NaN	27.68	27.53

```
1 data_temp[pd.isnull(data_temp['Nha Trang'])]
```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
3	03 15-9-2019	24.79	23.36	23.83	NaN	24.74	24.48
10	10 15-9-2019	NaN	29.97	NaN	NaN	27.68	27.53
20	20 15-9-2019	28.06	26.41	24.70	NaN	25.97	26.75

```

1 #2. Liệt kê các dòng missing của tất cả các features:
2 data_temp[data_temp.isnull().any(axis=1)]

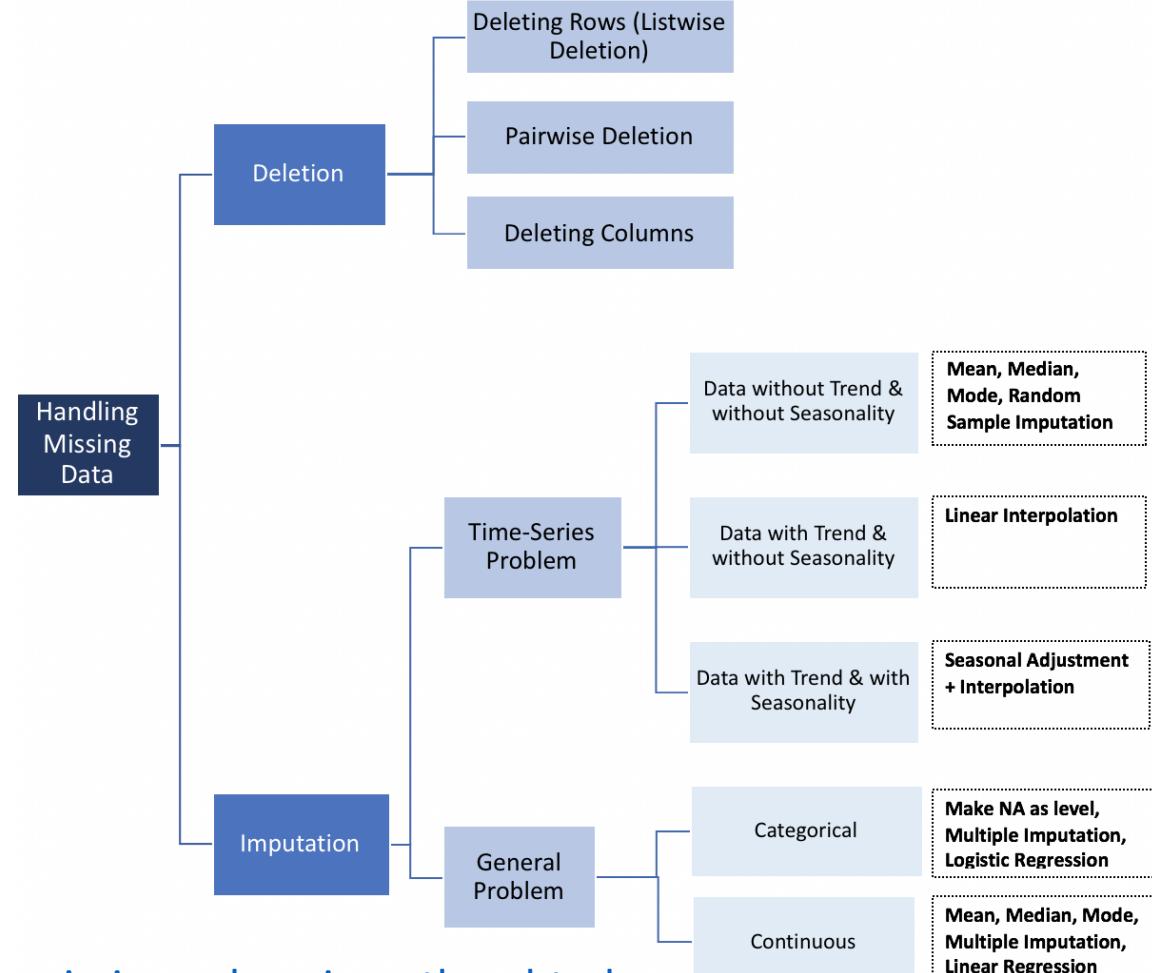
```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
1	01 15-9-2019	NaN	24.21	24.02	24.93	25.16	24.83
3	03 15-9-2019	24.79	23.36	23.83	NaN	24.74	24.48
5	05 15-9-2019	24.40	NaN	23.52	24.79	24.87	24.40
10	10 15-9-2019	NaN	29.97	NaN	NaN	27.68	27.53
13	13 15-9-2019	30.95	NaN	27.83	27.44	28.00	30.66
16	16 15-9-2019	30.80	30.20	NaN	26.45	27.29	29.13
20	20 15-9-2019	28.06	26.41	24.70	NaN	25.97	26.75

b.Xử lý missing data

- Để xử lý dữ liệu missing cần phải hiểu sâu sắc tập dữ liệu, việc lựa chọn phương pháp nào phụ thuộc vào từng bài toán cụ thể, một số phương pháp xử lý dữ liệu missing cơ bản:

1) Loại bỏ các missing (Deletion)



2) Thay thế các missing(Imputation)



b.Xử lý missing data

**df.dropna(axis=0) →
loại bỏ hàng**

```
1 #1) Phương pháp 1: Loại bỏ các dữ liệu missing (Deletion)
2
3 #Xóa toàn bộ các hàng chứa missing data: axis=0 -> xóa hàng
4 data_new = data_temp.dropna(axis=0, how='any')
5 #Kết quả sau khi loại bỏ các row chứa missing
6 print(data_new)
```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00 15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
2	02 15-9-2019	25.05	23.73	23.89	24.79	24.80	24.55
4	04 15-9-2019	24.59	23.05	23.69	24.82	24.80	24.38
6	06 15-9-2019	24.38	22.79	23.68	25.10	24.71	24.41
7	07 15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
8	08 15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
9	09 15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
11	11 15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
12	12 15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
14	14 15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
15	15 15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
17	17 15-9-2019	29.94	29.36	25.80	26.67	26.69	28.72
18	18 15-9-2019	28.53	27.48	24.82	25.92	25.81	27.46
19	19 15-9-2019	28.89	27.03	24.93	25.88	25.93	27.07
21	21 15-9-2019	27.43	26.20	24.41	25.62	25.94	26.32
22	22 15-9-2019	26.98	25.79	24.17	25.60	25.90	26.29
23	23 15-9-2019	26.68	25.31	23.81	25.53	25.80	26.36



b.Xử lý missing data

df.dropna(axis=1) →
loại bỏ cột

```
1 #1) Phương pháp 1: Loại bỏ các dữ liệu missing (Deletion)
2
3 #Xóa toàn bộ các cột chứa missing data: axis=1 -> xóa cột
4 data_new = data_temp.dropna(axis=1, how='any')
5 #Kết quả sau khi loại bỏ các cột chứa missing
6 print(data_new)
```

	time	Ho Chi Minh	Ca Mau
0	00 15-9-2019	25.48	24.97
1	01 15-9-2019	25.16	24.83
2	02 15-9-2019	24.80	24.55
3	03 15-9-2019	24.74	24.48
4	04 15-9-2019	24.80	24.38
5	05 15-9-2019	24.87	24.40
6	06 15-9-2019	24.71	24.41
7	07 15-9-2019	25.03	24.91
8	08 15-9-2019	25.75	25.85
9	09 15-9-2019	26.64	26.79
10	10 15-9-2019	27.68	27.53
11	11 15-9-2019	28.43	28.98
12	12 15-9-2019	28.29	29.24
13	13 15-9-2019	28.00	30.66
14	14 15-9-2019	27.67	30.97
15	15 15-9-2019	27.29	30.59
16	16 15-9-2019	27.29	29.13
17	17 15-9-2019	26.69	28.72

Các cột **Hà Nội, Vinh, Đà Nẵng, Nha Trang** có chứa dữ liệu missing đã bị loại bỏ

b.Xử lý missing data

df.fillna(value) → thay thế bằng một giá trị cố định

```

1 #PHƯƠNG PHÁP 2: Thay thế (Imputation)
2 #2.1) Thay thế các dữ liệu mất mát bằng một hằng số cố định
3 value = 25.0
4 #thay thế các giá trị missing bằng một giá trị cố định Value
5 data_new = data_temp.fillna(value)
6 print(data_new)

```

		time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00	15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
1	01	15-9-2019	25.00	24.21	24.02	24.93	25.16	24.83
2	02	15-9-2019	25.05	23.73	23.89	24.79	24.80	24.55
3	03	15-9-2019	24.79	23.36	23.83	25.00	24.74	24.48
4	04	15-9-2019	24.59	23.05	23.69	24.82	24.80	24.38
5	05	15-9-2019	24.40	25.00	23.52	24.79	24.87	24.40
6	06	15-9-2019	24.38	22.79	23.68	25.10	24.71	24.41
7	07	15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
8	08	15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
9	09	15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
10	10	15-9-2019	25.00	29.97	25.00	25.00	27.68	27.53
11	11	15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
12	12	15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
13	13	15-9-2019	30.95	25.00	27.83	27.44	28.00	30.66
14	14	15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
15	15	15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
16	16	15-9-2019	30.80	30.20	25.00	26.45	27.29	29.13

b.Xử lý missing data

df.fillna(method='pad') →
thay thế bằng giá trị liền trước

```

1 #PHƯƠNG PHÁP 2: Thay thế (Imputation)
2 #2.2)Thay thế các dữ liệu mất mát bằng giá trị liền trước của nó
3 data_new2 = data_temp.fillna(method='pad')
4 print(data_new2)

```

		time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00	15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
1	01	15-9-2019	25.65	24.21	24.02	24.93	25.16	24.83
2	02	15-9-2019	25.05	23.73	23.89	24.79	24.80	24.55
3	03	15-9-2019	24.79	23.36	23.83	24.79	24.74	24.48
4	04	15-9-2019	24.59	23.05	23.69	24.82	24.80	24.38
5	05	15-9-2019	24.40	23.05	23.52	24.79	24.87	24.40
6	06	15-9-2019	24.38	22.79	23.68	25.10	24.71	24.41
7	07	15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
8	08	15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
9	09	15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
10	10	15-9-2019	30.29	29.97	27.48	26.95	27.68	27.53
11	11	15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
12	12	15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
13	13	15-9-2019	30.95	28.94	27.83	27.44	28.00	30.66
14	14	15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
15	15	15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
16	16	15-9-2019	30.80	30.20	26.29	26.45	27.29	29.13

b.Xử lý missing data

df.fillna(method='bfill')

→ thay thế bằng giá trị
liền sau

```

1 #PHƯƠNG PHÁP 2: Thay thế (Imputation)
2 #2.3)Thay thế các dữ liệu mất mát bằng giá trị liền sau của nó
3 data_new3 = data_temp.fillna(method='bfill')
4 print(data_new3)

```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00 15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
1	01 15-9-2019	25.05	24.21	24.02	24.93	25.16	24.83
2	02 15-9-2019	25.05	23.73	23.89	24.79	24.80	24.55
3	03 15-9-2019	24.79	23.36	23.83	24.82	24.74	24.48
4	04 15-9-2019	24.59	23.05	23.69	24.82	24.80	24.38
5	05 15-9-2019	24.40	22.79	23.52	24.79	24.87	24.40
6	06 15-9-2019	24.38	22.79	23.68	25.10	24.71	24.41
7	07 15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
8	08 15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
9	09 15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
10	10 15-9-2019	32.05	29.97	26.86	27.38	27.68	27.53
11	11 15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
12	12 15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
13	13 15-9-2019	30.95	30.62	27.83	27.44	28.00	30.66
14	14 15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
15	15 15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
16	16 15-9-2019	30.80	30.20	25.80	26.45	27.29	29.13



b.Xử lý missing data

df.interpolate() → thay thế giá trị bằng giá trị trung bình của giá trị liền trước và liền sau

```
1 #PHƯƠNG PHÁP 2: Thay thế (Imputation)
2 #2.4)Xử Lý các giá trị missing theo phương pháp nội suy
3 #Sử dụng hàm interpolate để thay thế giá trị missing với tham số:
4 #Thuật toán nội suy: Tuyến tính (linear)
5 #Hướng nội suy: Tiến lên (forward)
6 data_new4 = data_temp.interpolate(method='linear', limit_direction ='forward')
7 print(data_new4)
```

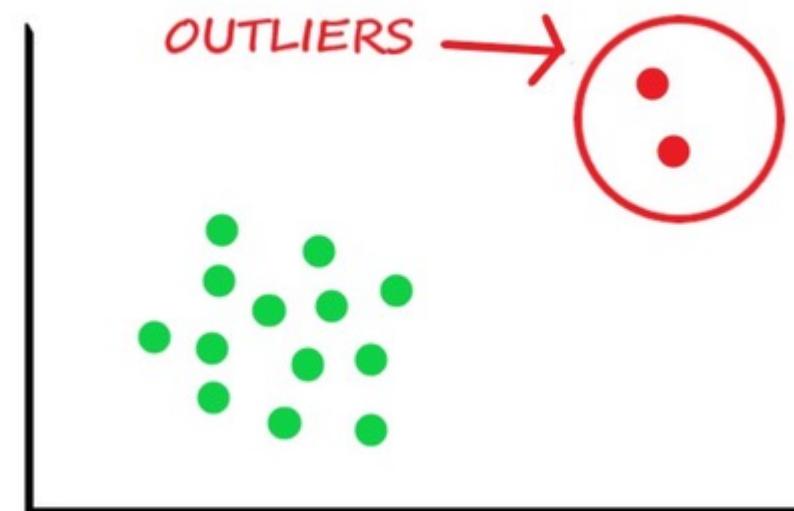
		time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00	15-9-2019	25.65	24.79	24.010	25.060	25.48	24.97
1	01	15-9-2019	25.35	24.21	24.020	24.930	25.16	24.83
2	02	15-9-2019	25.05	23.73	23.890	24.790	24.80	24.55
3	03	15-9-2019	24.79	23.36	23.830	24.805	24.74	24.48
4	04	15-9-2019	24.59	23.05	23.690	24.820	24.80	24.38
5	05	15-9-2019	24.40	22.92	23.520	24.790	24.87	24.40
6	06	15-9-2019	24.38	22.79	23.680	25.100	24.71	24.41
7	07	15-9-2019	26.72	25.61	24.920	26.560	25.03	24.91
8	08	15-9-2019	28.84	26.93	26.510	26.530	25.75	25.85
9	09	15-9-2019	30.29	28.72	27.480	26.950	26.64	26.79
10	10	15-9-2019	31.17	29.97	27.170	27.165	27.68	27.53
11	11	15-9-2019	32.05	28.93	26.860	27.380	28.43	28.98
12	12	15-9-2019	31.31	28.94	26.650	27.470	28.29	29.24
13	13	15-9-2019	30.95	29.78	27.830	27.440	28.00	30.66
14	14	15-9-2019	30.56	30.62	26.490	27.160	27.67	30.97
15	15	15-9-2019	31.13	30.58	26.290	26.680	27.29	30.59
16	16	15-9-2019	30.80	30.20	26.045	26.450	27.29	29.13



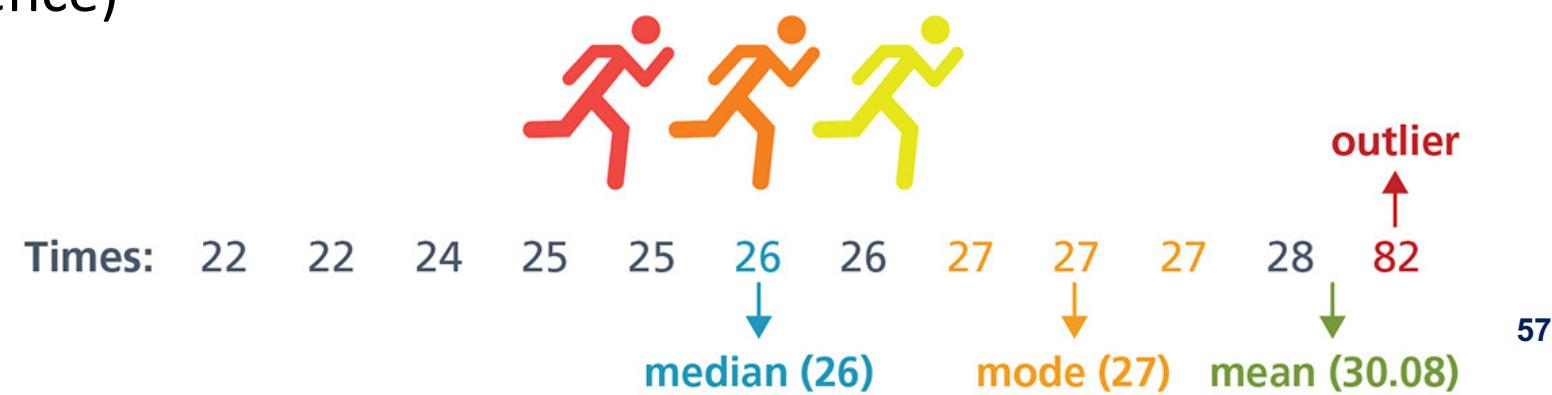
9. Phát hiện và xử lý ngoại lai

Giá trị ngoại lai (Outliers)

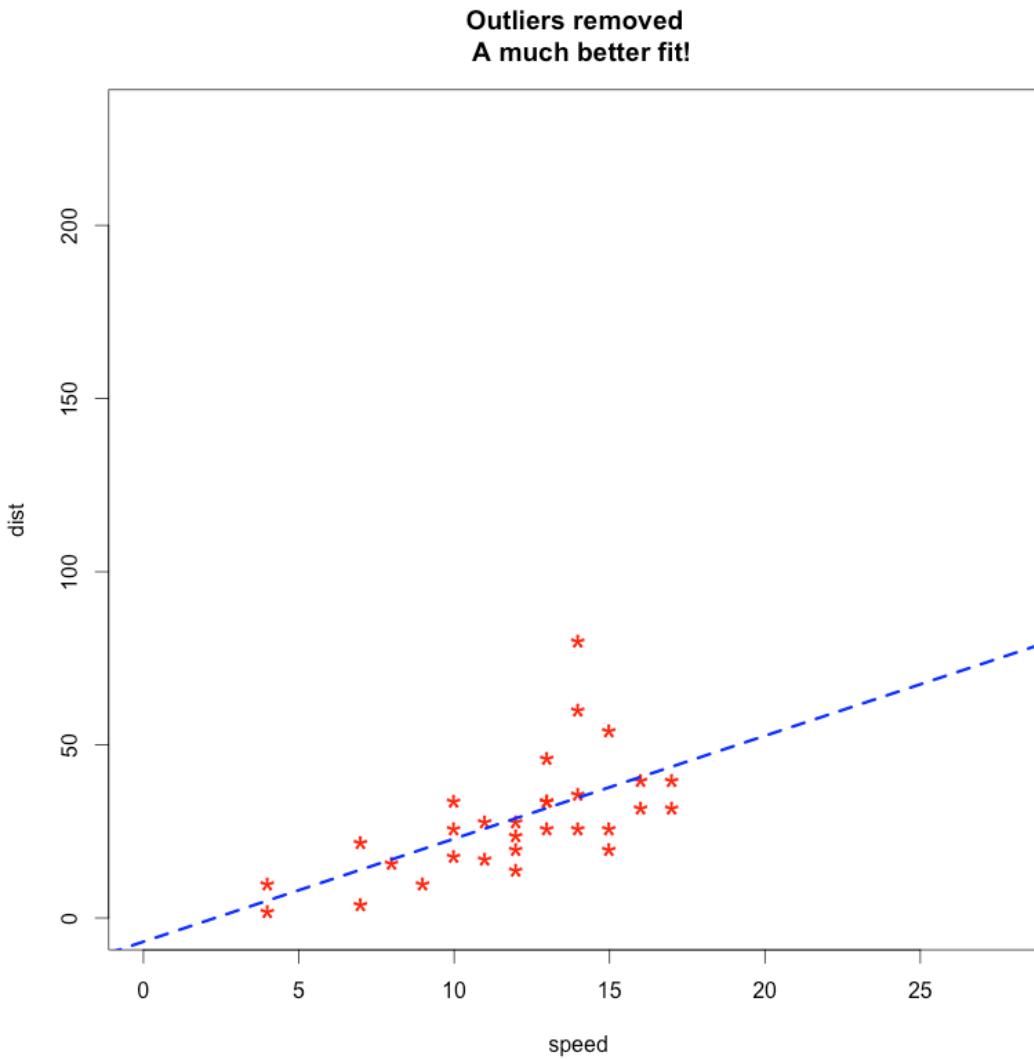
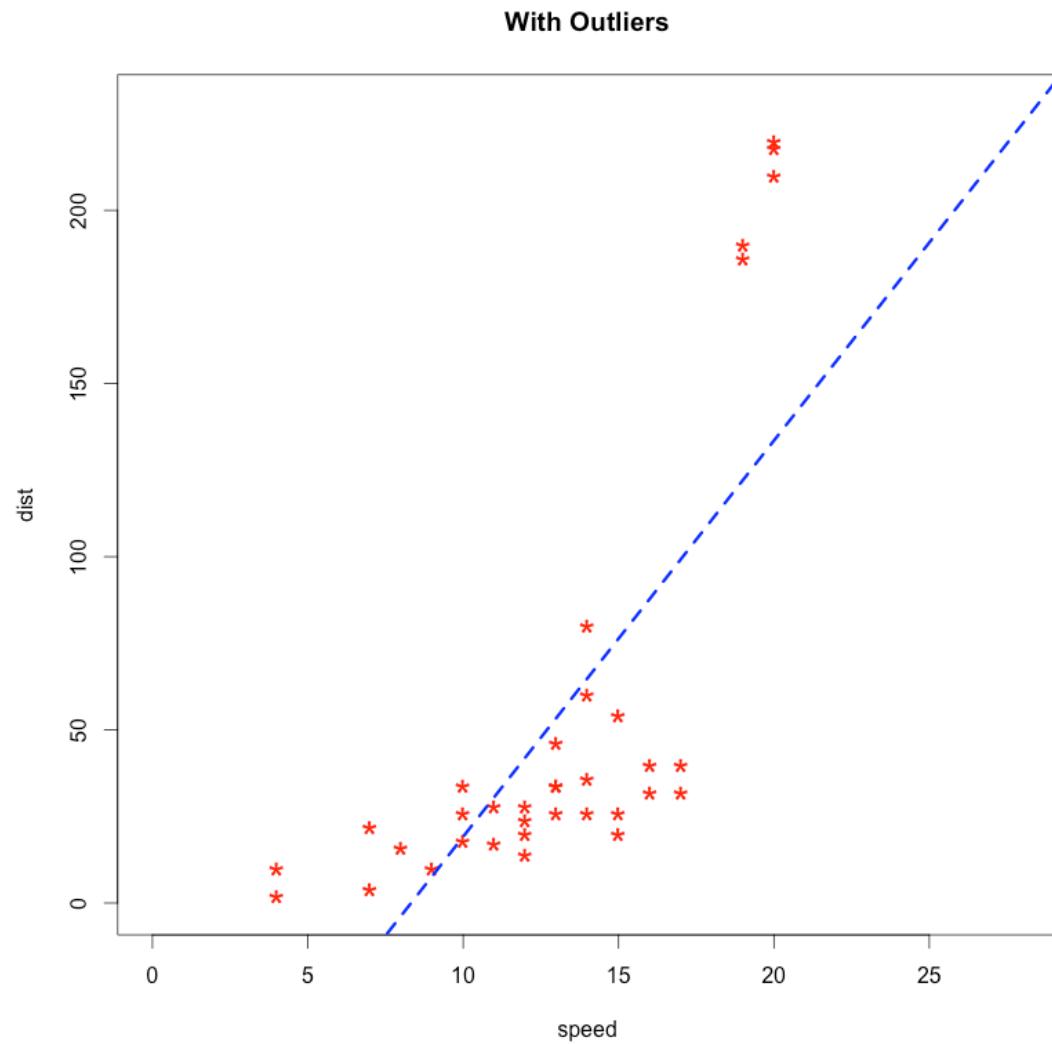
Một điểm ngoại lai là một điểm dữ liệu khác biệt đáng kể so với phần còn lại của tập dữ liệu. Ta thường xem các giá trị ngoại lai như là các mẫu dữ liệu đặc biệt, cách xa khỏi phần lớn dữ liệu khác trong tập dữ liệu



- Hệ thống phát hiện xâm nhập (Intrusion detection systems)
- Phát hiện gian lận tín dụng (Credit card fraud)
- Trong chuẩn đoán y tế (Medical diagnosis)
- Trong thực thi pháp luật (Law enforcement)
- Trong khoa học trái đất (Earth science)



Giá trị ngoại lai (Outliers)



Outliers

How to
identify?

How to
handle?

Box plot

Interquartile Range (IQR) based method

Standard Deviation based method

Histogram

Doing nothing

Deleting/Trimming

Winsorizing

Transformation

Binning

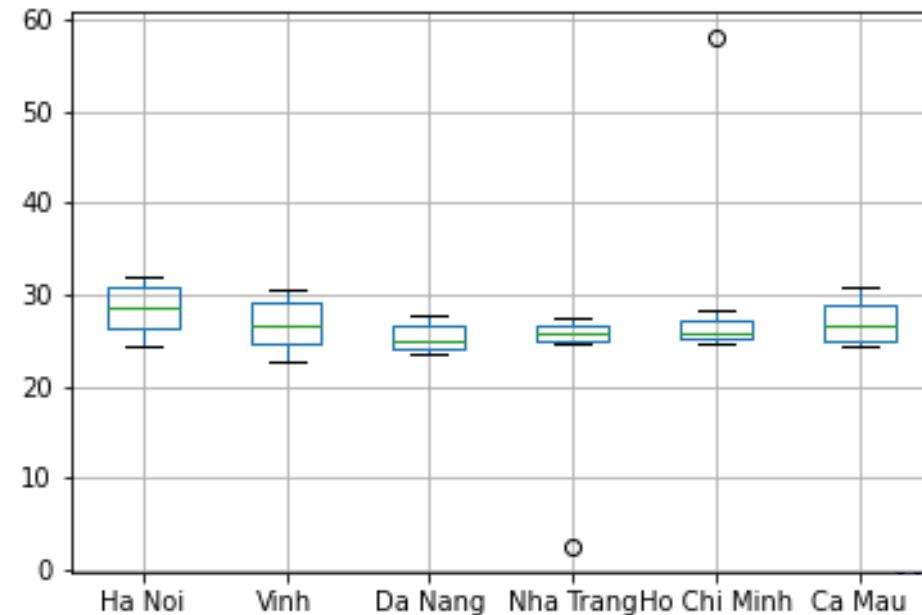
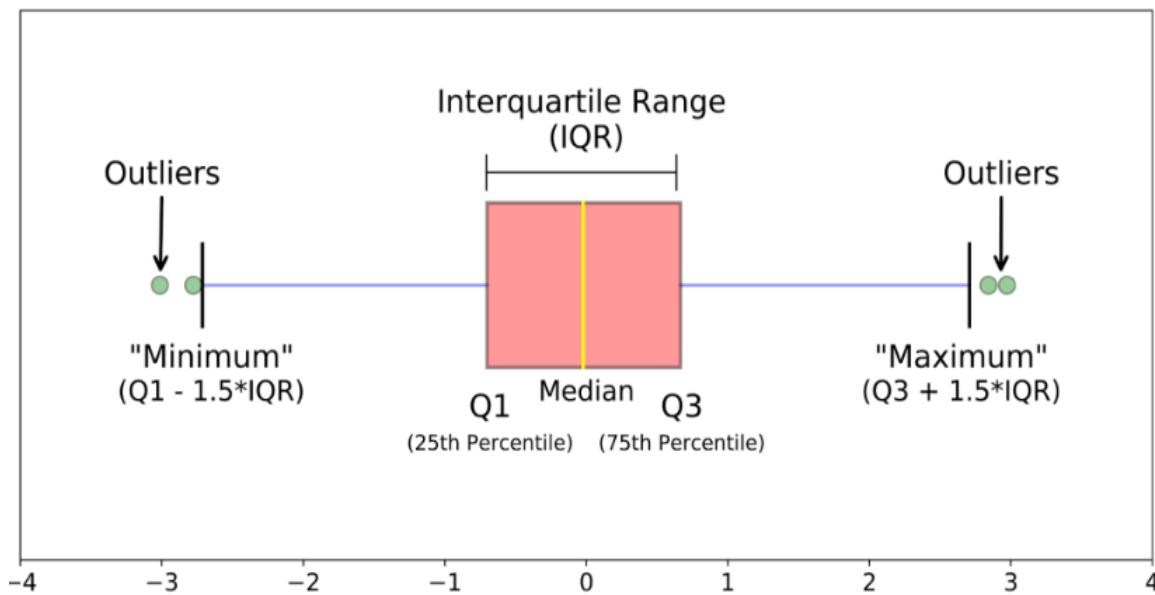
Use robust estimators

Imputing

Phát hiện Outliers

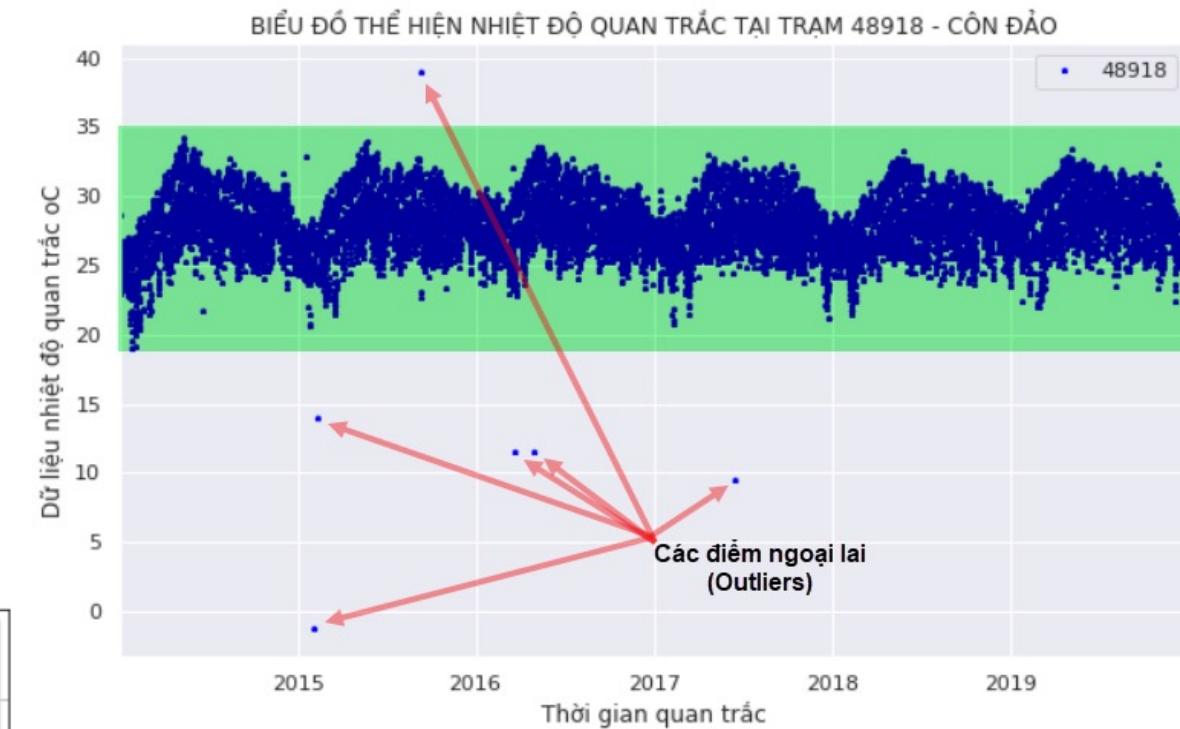
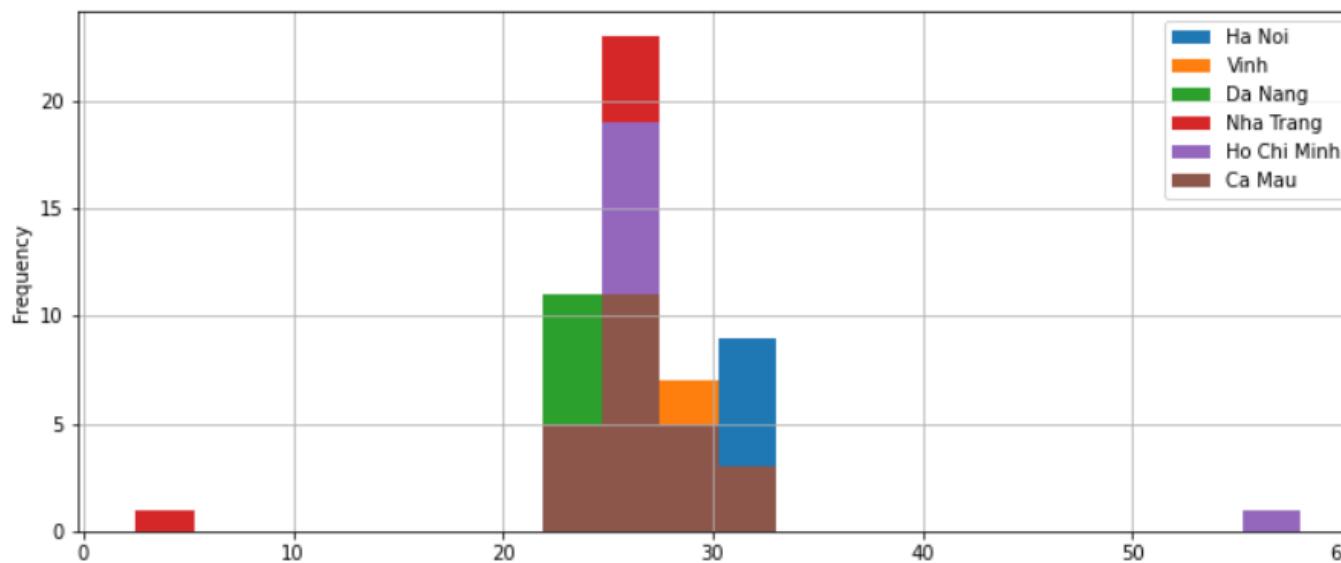
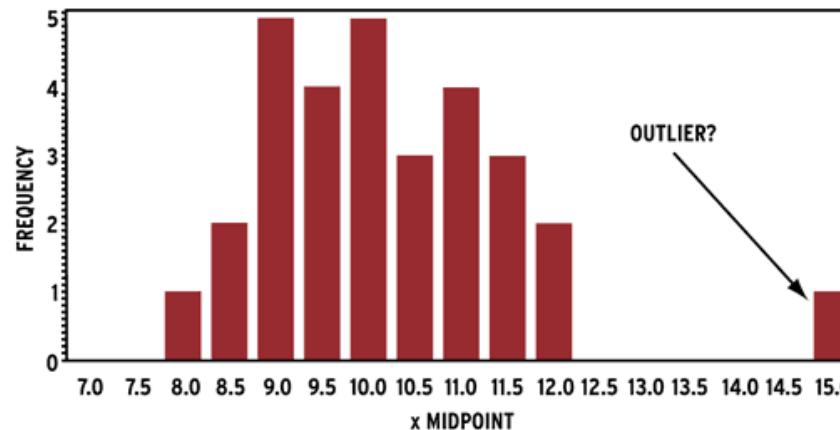
Biểu đồ Box-plot được sử dụng để đo khuynh hướng phân tán và xác định các giá trị ngoại lai của tập dữ liệu

- Giá trị bé nhất (**Minimum**) của tập dữ liệu được xác định bằng $Q1 - 1.5 * IQR$;
- Tứ phân vị thứ nhất (**Q1**) của tập dữ liệu
- Tứ phân vị thứ hai (**Q2**) chính là giá trị trung vị (Median) của tập dữ liệu
- Tứ phân vị thứ ba (**Q3**) của tập dữ liệu
- Giá trị lớn nhất (**Maximum**) của tập dữ liệu có giá trị bằng $Q3 + 1.5 * IQR$



Phát hiện Outliers

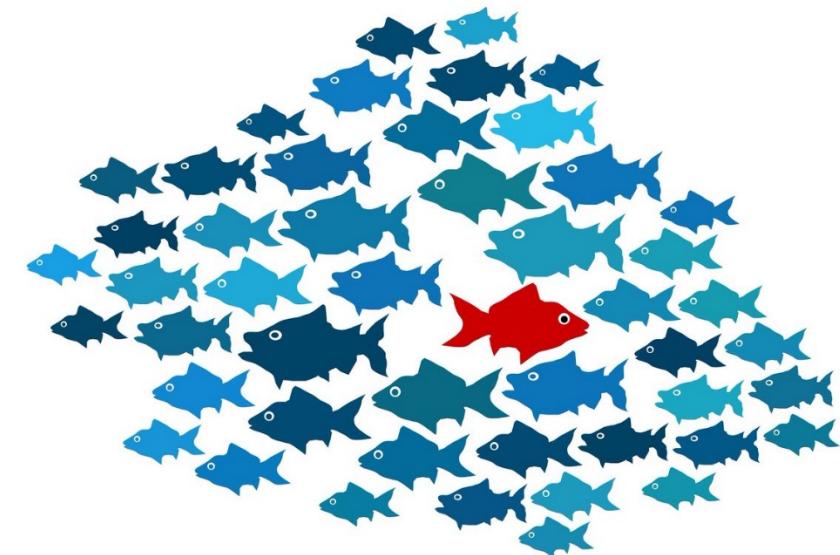
Sử dụng biểu đồ histogram;
Hoặc scatter (nhiều hơn 2 biến).



Xử lý Outliers

Không có một phương pháp, cách thức xử lý ngoại lai chung nào áp dụng cho tất cả các bài toán, các kiểu dữ liệu khác nhau.

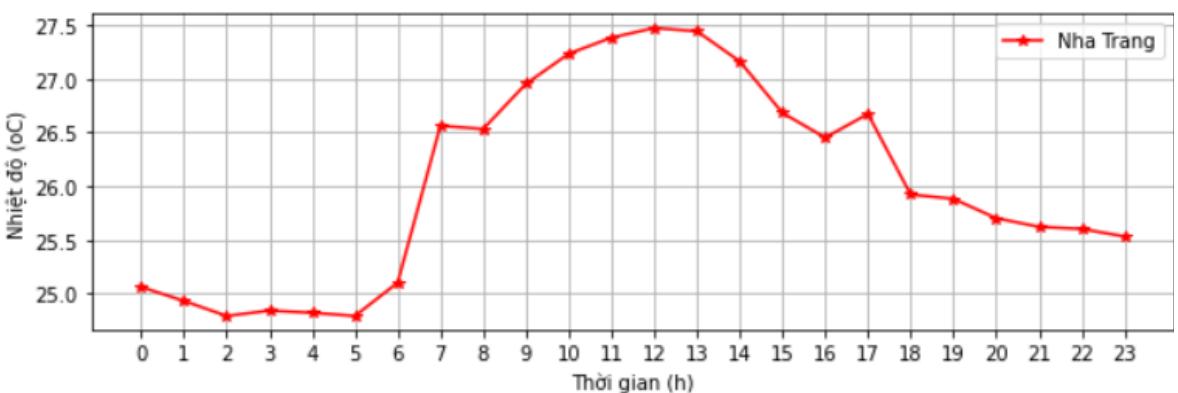
- Loại bỏ dòng dữ liệu chứa ngoại lai.
- Thay thế bằng một giá trị khác
- Thay thế giá trị ngoại lai về NAN (null) coi như là một điểm dữ liệu missing và xử lý như với dữ liệu missing



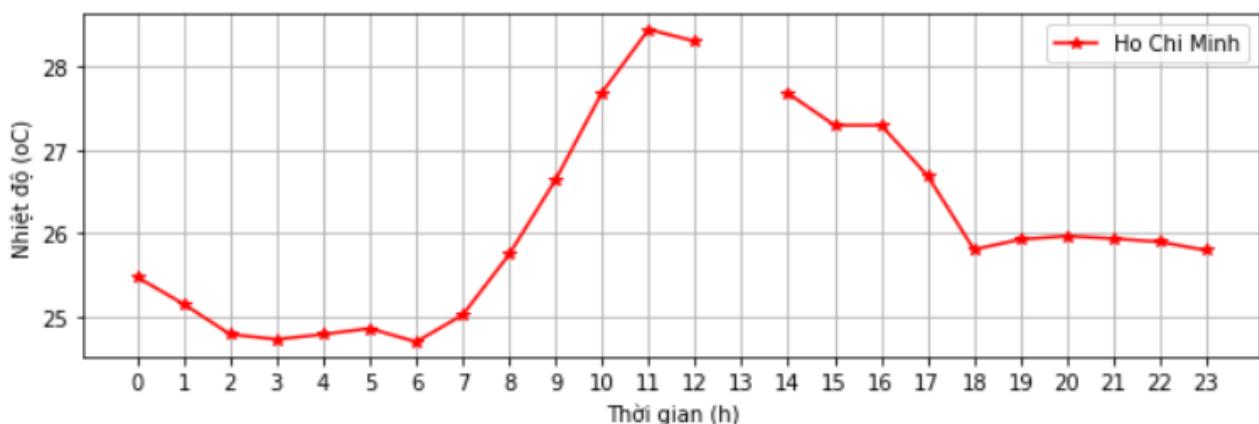
Để lựa chọn được phương pháp phù hợp cần có những hiểu biết sâu sắc về tập dữ liệu, về bài toán đang giải quyết, có thể sử dụng chỉ một phương pháp xử lý ngoại lai và/hoặc kết hợp cả 3 nhóm phương pháp đã chỉ ra ở trên để xử lý ngoại lai cho cùng một tập dữ liệu.

Xử lý Outliers

```
1 #Với giá trị ngoại lai trạm Nha Trang
2 #Xử lý bằng cách thay thế giá trị: 2.51 --> 25.1
3 x = np.arange(0,24)
4 plt.rcParams["figure.figsize"] = (10,3)
5 data_handling_outlier.loc[6,'Nha Trang'] = 25.10
6 data_handling_outlier[['Nha Trang']].plot(style='-*', color='red')
7 plt.xticks(x)
8 plt.xlabel('Thời gian (h)')
9 plt.ylabel('Nhiệt độ (oC)')
10 plt.grid(True)
11 plt.show()
```



```
1 #Với giá trị ngoại lai trạm Hồ Chí Minh
2 #Xử lý bằng cách chuyển về giá trị Null - coi như giá trị missing
3 data_handling_outlier.loc[13,'Ho Chi Minh'] = np.NaN
4 data_handling_outlier[['Ho Chi Minh']].plot(style='-*', color='red')
5 plt.xticks(x)
6 plt.xlabel('Thời gian (h)')
7 plt.ylabel('Nhiệt độ (oC)')
8 plt.grid(True)
9 plt.show()
```



Thực hành 3

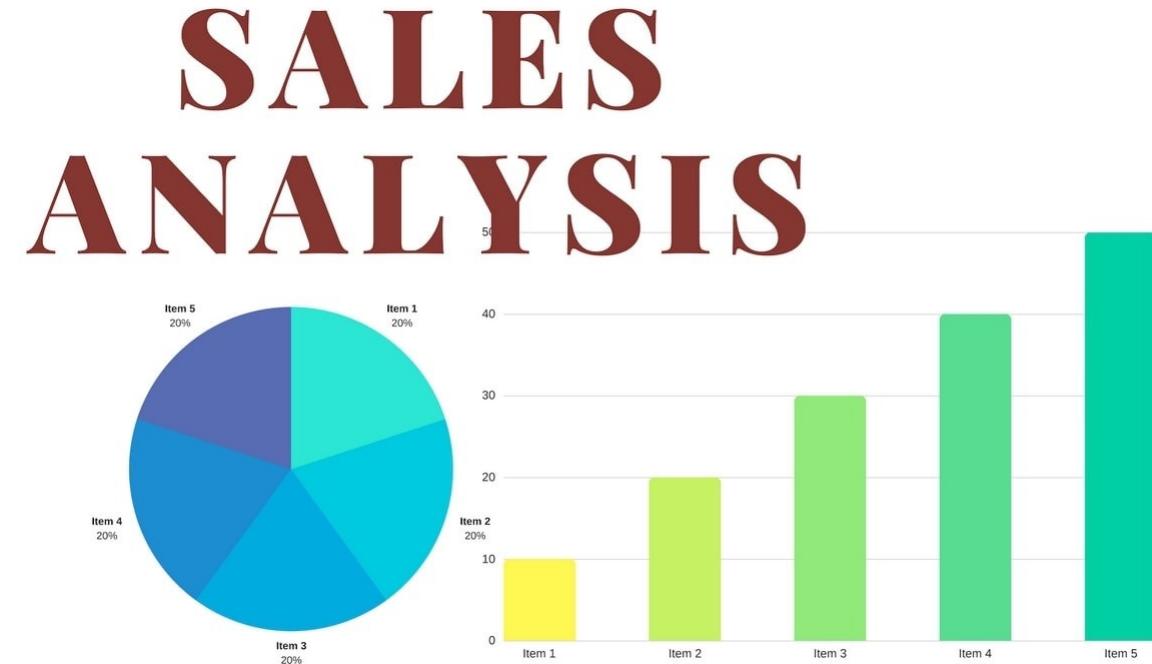
10. Phân tích dữ liệu bán hàng

(Tiếp cận từ bài toán thực tế)

Phân tích dữ liệu bán hàng

Dữ liệu bao gồm 12 file của một chuỗi cửa hàng kinh doanh thiết bị điện tử; Mỗi file dữ liệu lưu lại toàn bộ các đơn hàng đã bán được theo từng tháng tương ứng của năm 2019.

- * Dữ liệu tháng 1: Sales_January_2019.csv
- * Dữ liệu tháng 2: Sales_February_2019.csv
- * Dữ liệu tháng 3: Sales_March_2019.csv
- * Dữ liệu tháng 4: Sales_April_2019.csv
- * Dữ liệu tháng 5: Sales_May_2019.csv
- * Dữ liệu tháng 6: Sales_June_2019.csv
- * Dữ liệu tháng 7: Sales_July_2019.csv
- * Dữ liệu tháng 8: Sales_August_2019.csv
- * Dữ liệu tháng 9: Sales_September_2019.csv
- * Dữ liệu tháng 10: Sales_October_2019.csv
- * Dữ liệu tháng 11: Sales_November_2019.csv
- * Dữ liệu tháng 12: Sales_December_2019.csv



Phân tích dữ liệu bán hàng

Mỗi file dữ liệu bao gồm 6 thông tin:

1. Order ID: Mã đơn hàng
2. Product: Tên sản phẩm đã bán theo từng đơn hàng
3. Quantity Ordered: Số lượng sản phẩm bán
4. Price Each: Giá của mỗi sản phẩm
5. Order Date: Thời gian bán hàng
6. Purchase Address: Địa chỉ mua hàng

Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
141234	iPhone	1	700	01/22/19 21:25	944 Walnut St, Boston, MA 02215
141235	Lightning Charging Cable	1	14.95	01/28/19 14:15	185 Maple St, Portland, OR 97035
141236	Wired Headphones	2	11.99	01/17/19 13:33	538 Adams St, San Francisco, CA 94016
141237	27in FHD Monitor	1	149.99	01/05/19 20:33	738 10th St, Los Angeles, CA 90001
141238	Wired Headphones	1	11.99	01/25/19 11:59	387 10th St, Austin, TX 73301

Mục tiêu:

- Đọc dữ liệu từ nhiều file, Sử dụng các kỹ thuật làm sạch và chuẩn bị dữ liệu để phân tích
- Thực hiện phân tích tìm ra các thông tin có ích (insights) từ dữ liệu



Kết quả

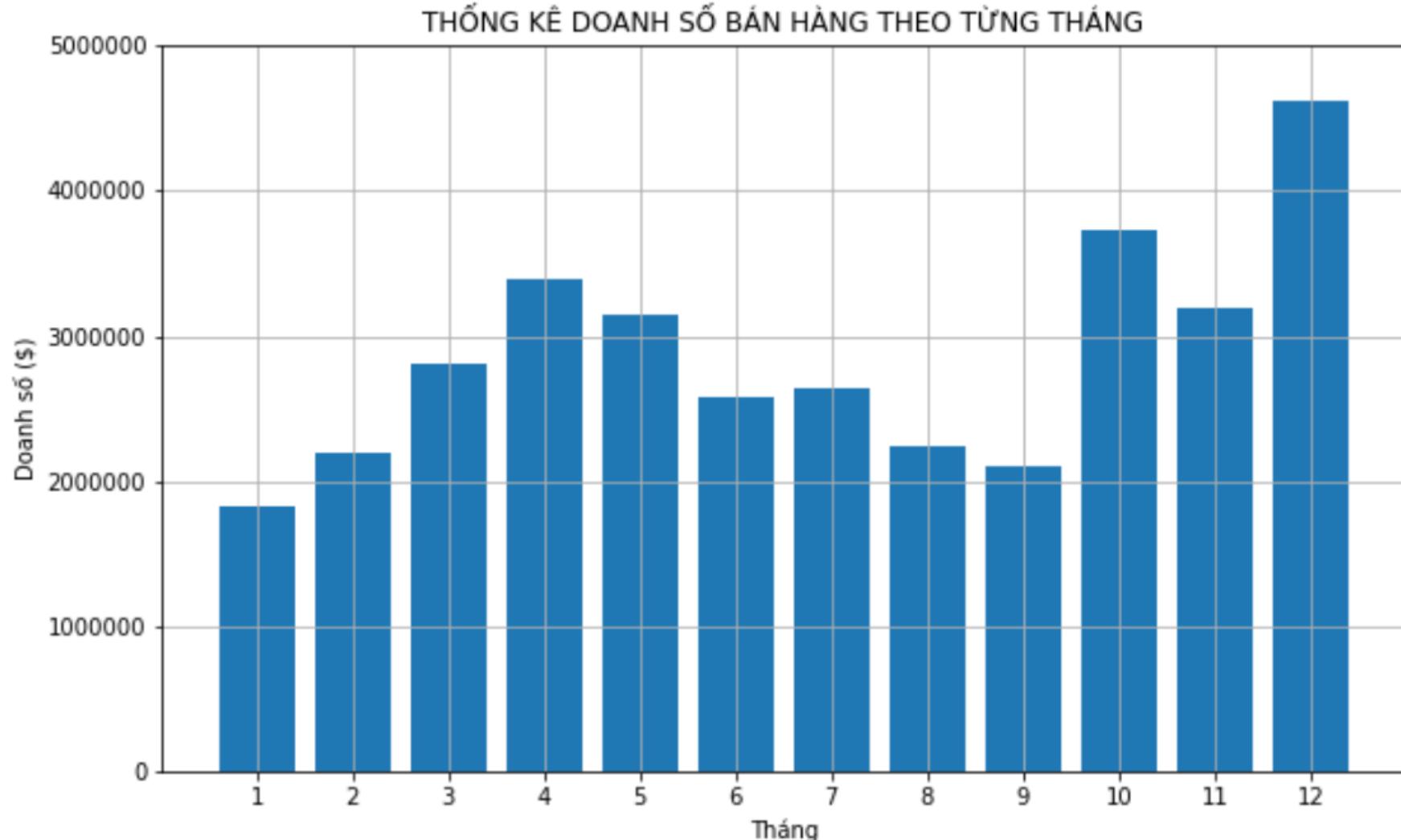
Tập dữ liệu sau khi được tiền chuẩn hóa và làm sạch:

Order ID	Product	Quantity	Price Each	Order Date	Purchase Address	Month	Hour	City	Sales
141234	iPhone	1	700	2019-01-22 21:25	944 Walnut St, Boston, MA 02215	1	21	Boston (MA)	700
141235	Lightning Charging Cable	1	14.95	2019-01-28 14:15	185 Maple St, Portland, OR 97035	1	14	Portland (OR)	14.95
141236	Wired Headphones	2	11.99	2019-01-17 13:33	538 Adams St, San Francisco, CA 94016	1	13	San Francisco (CA)	23.98
141237	27in FHD Monitor	1	149.99	2019-01-05 20:33	738 10th St, Los Angeles, CA 90001	1	20	Los Angeles (CA)	149.99
141238	Wired Headphones	1	11.99	2019-01-25 11:59	387 10th St, Austin, TX 73301	1	11	Austin (TX)	11.99
141239	AAA Batteries (4-pack)	1	2.99	2019-01-29 20:22	775 Willow St, San Francisco, CA 94016	1	20	San Francisco (CA)	2.99
141240	27in 4K Gaming Monitor	1	389.99	2019-01-26 12:16	979 Park St, Los Angeles, CA 90001	1	12	Los Angeles (CA)	389.99
141241	USB-C Charging Cable	1	11.95	2019-01-05 12:04	181 6th St, San Francisco, CA 94016	1	12	San Francisco (CA)	11.95
141242	Bose SoundSport Headphones	1	99.99	2019-01-01 10:30	867 Willow St, Los Angeles, CA 90001	1	10	Los Angeles (CA)	99.99
141243	Apple Airpods Headphones	1	150	2019-01-22 21:20	657 Johnson St, San Francisco, CA 94016	1	21	San Francisco (CA)	150
141244	Apple Airpods Headphones	1	150	2019-01-07 11:29	492 Walnut St, San Francisco, CA 94016	1	11	San Francisco (CA)	150
141245	Macbook Pro Laptop	1	1700	2019-01-31 10:12	322 6th St, San Francisco, CA 94016	1	10	San Francisco (CA)	1700
141246	AAA Batteries (4-pack)	3	2.99	2019-01-09 18:57	618 7th St, Los Angeles, CA 90001	1	18	Los Angeles (CA)	8.97
141247	27in FHD Monitor	1	149.99	2019-01-25 19:19	512 Wilson St, San Francisco, CA 94016	1	19	San Francisco (CA)	149.99
141248	Flatscreen TV	1	300	2019-01-03 21:54	363 Spruce St, Austin, TX 73301	1	21	Austin (TX)	300
141249	27in FHD Monitor	1	149.99	2019-01-05 17:20	440 Cedar St, Portland, OR 97035	1	17	Portland (OR)	149.99
141250	Vareebadd Phone	1	400	2019-01-10 11:20	471 Center St, Los Angeles, CA 90001	1	11	Los Angeles (CA)	400
141251	Apple Airpods Headphones	1	150	2019-01-24 8:13	414 Walnut St, Boston, MA 02215	1	8	Boston (MA)	150
141252	USB-C Charging Cable	1	11.95	2019-01-30 9:28	220 9th St, Los Angeles, CA 90001	1	9	Los Angeles (CA)	11.95
141253	AA Batteries (4-pack)	1	3.84	2019-01-17 0:09	385 11th St, Atlanta, GA 30301	1	0	Atlanta (GA)	3.84
141254	AAA Batteries (4-pack)	1	2.99	2019-01-08 11:51	238 Sunset St, Seattle, WA 98101	1	11	Seattle (WA)	2.99
141255	USB-C Charging Cable	1	11.95	2019-01-09 20:55	764 11th St, Los Angeles, CA 90001	1	20	Los Angeles (CA)	11.95
141256	Google Phone	1	600	2019-01-29 10:40	675 Washington St, Portland, OR 97035	1	10	Portland (OR)	600
141257	Apple Airpods Headphones	1	150	2019-01-12 18:51	338 Highland St, San Francisco, CA 94016	1	18	San Francisco (CA)	150

Kết quả

Câu hỏi 1: Tháng nào trong năm có doanh số bán hàng cao nhất - thấp nhất?

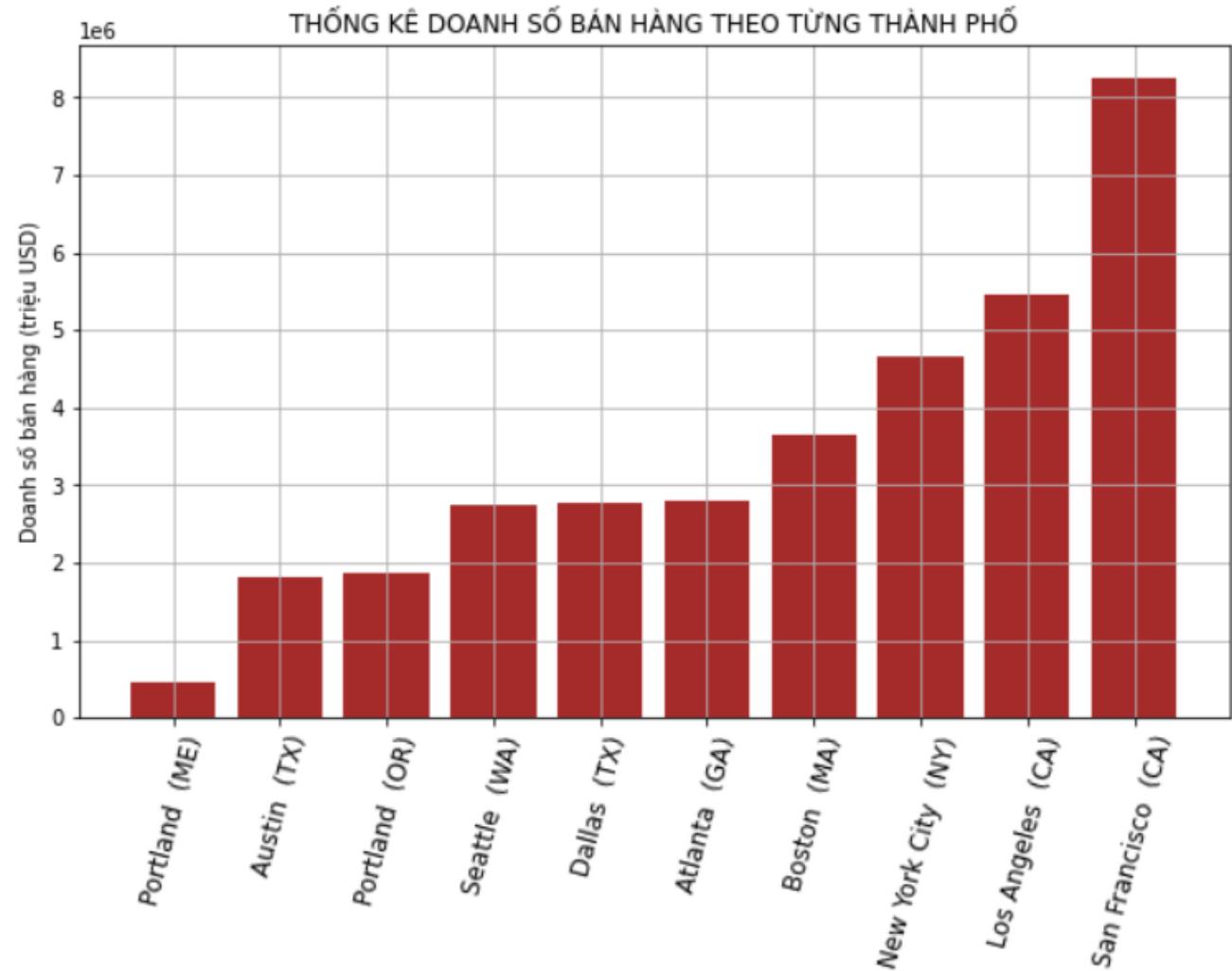
W
Insight!



Kết quả

Câu hỏi 2: Cửa hàng ở thành phố nào bán được hàng nhiều nhất?

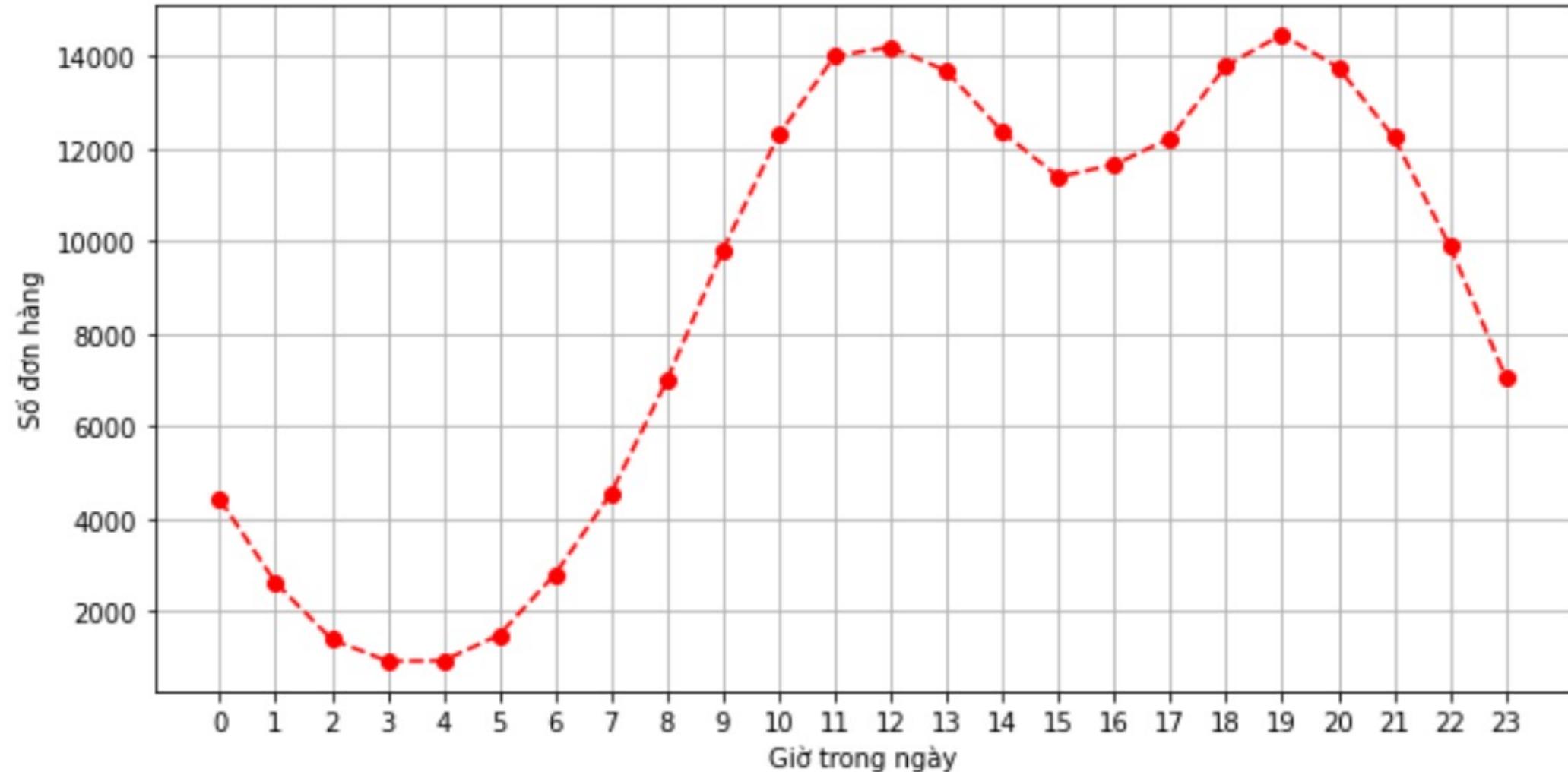
City	Sales
Portland (ME)	4.497583e+05
Austin (TX)	1.819582e+06
Portland (OR)	1.870732e+06
Seattle (WA)	2.747755e+06
Dallas (TX)	2.767975e+06
Atlanta (GA)	2.795499e+06
Boston (MA)	3.661642e+06
New York City (NY)	4.664317e+06
Los Angeles (CA)	5.452571e+06
San Francisco (CA)	8.262204e+06



W/ Insight!

Kết quả

Câu hỏi 3: Khách hàng mua sản phẩm thường tập trung vào khung giờ nào trong ngày?



Insight!

Câu hỏi 4: Trong các hóa đơn của Khách hàng, Sản phẩm nào thường được mua cùng với nhau?

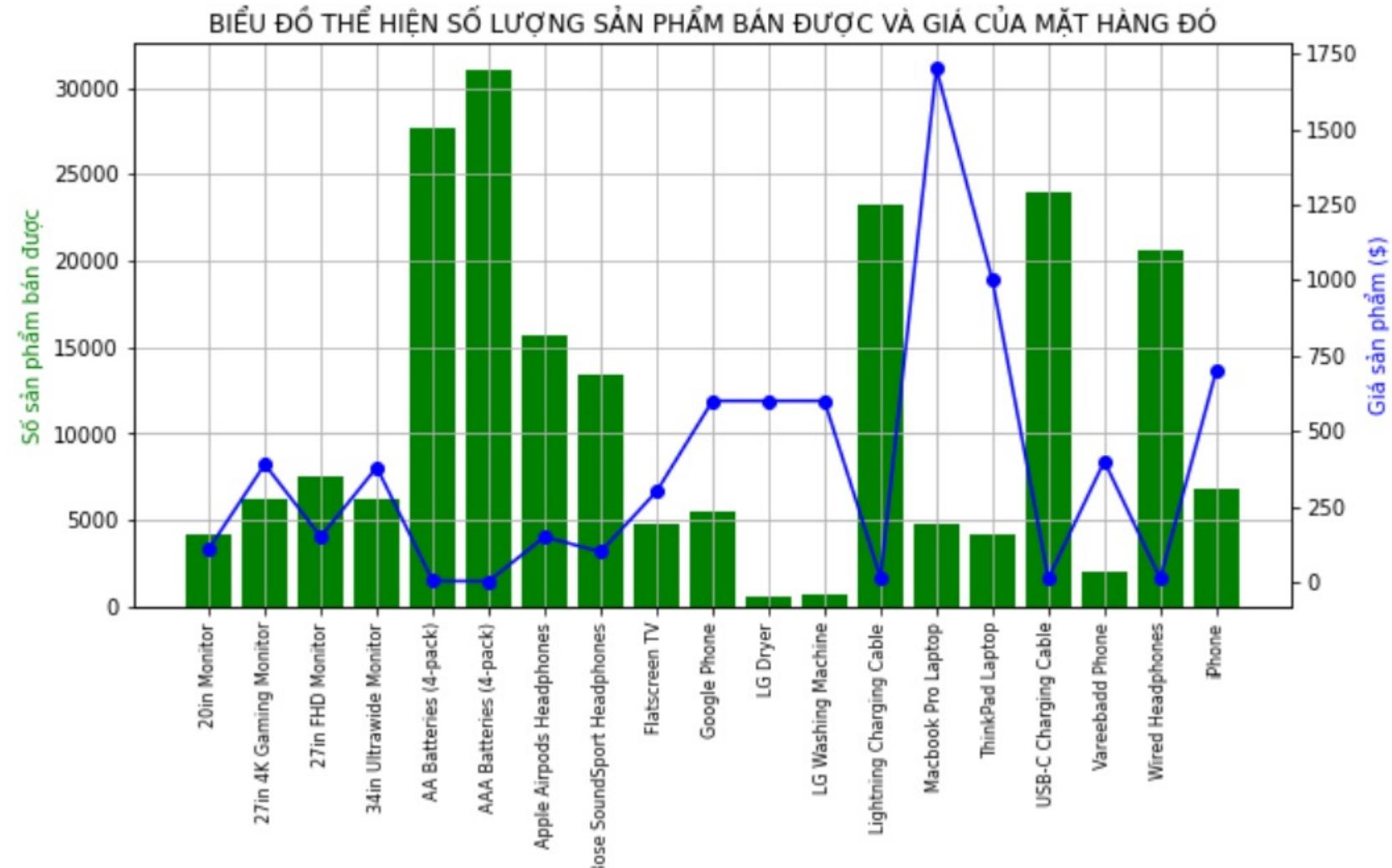
('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92
('Lightning Charging Cable', 'Apple Airpods Headphones') 81
('Vareebadd Phone', 'Bose SoundSport Headphones') 80
('USB-C Charging Cable', 'Bose SoundSport Headphones') 77
('Apple Airpods Headphones', 'Wired Headphones') 69
('Lightning Charging Cable', 'USB-C Charging Cable') 58



Kết quả

Câu hỏi 5: Sản phẩm nào của chuỗi cửa hàng bán được số lượng nhiều nhất?
Tại sao?

W
Insight!





Q & A
Thank you!



VINBIGDATA VINGROUP



AI
Academy
Vietnam

Bài 8: Trực quan hóa dữ liệu với Matplotlib

AI Academy Vietnam

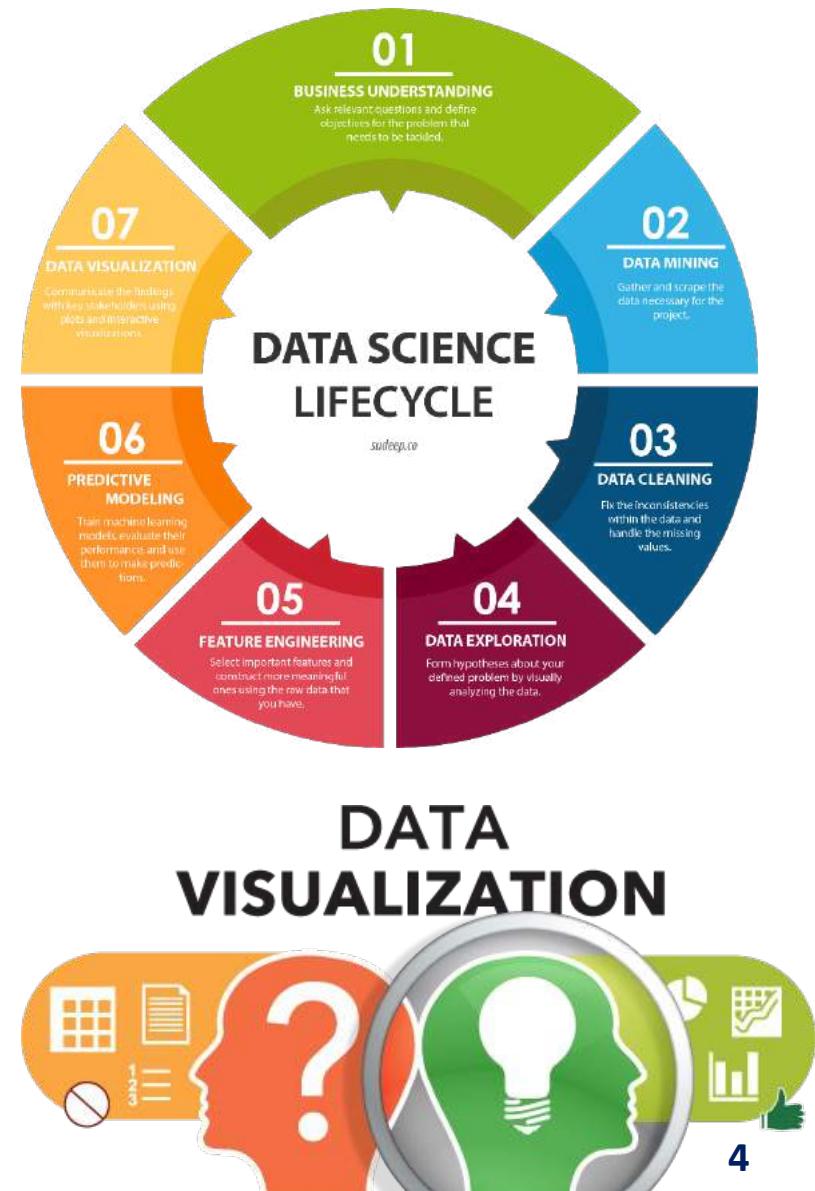
1. Tâm quan trọng của trực quan hóa dữ liệu
2. Một số lưu ý khi trực quan hóa dữ liệu
3. Một số thư viện trực quan hóa dữ liệu với Python
4. Biểu đồ Line chart
5. Biểu đồ Bar chart
6. Biểu đồ Pie chart
7. Biểu đồ Scatter plot
8. Biểu đồ Histogram plot
9. Biểu đồ Boxplot



1. Tầm quan trọng của trực quan hóa dữ liệu

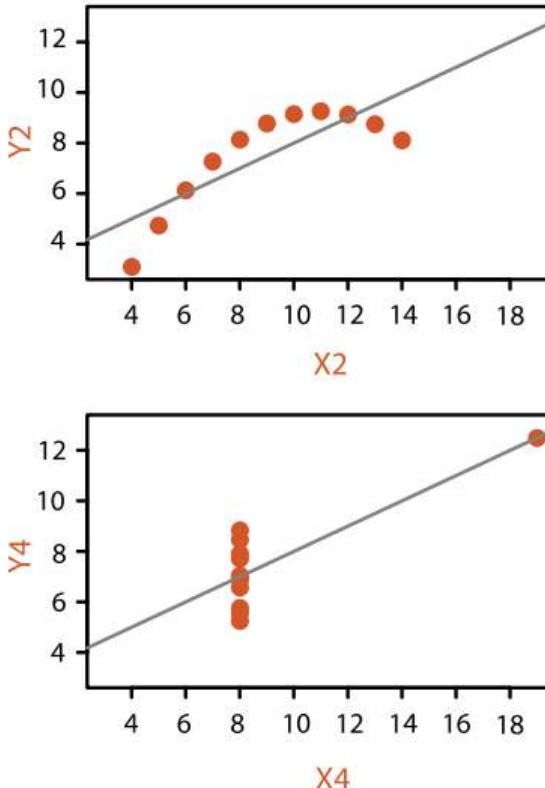
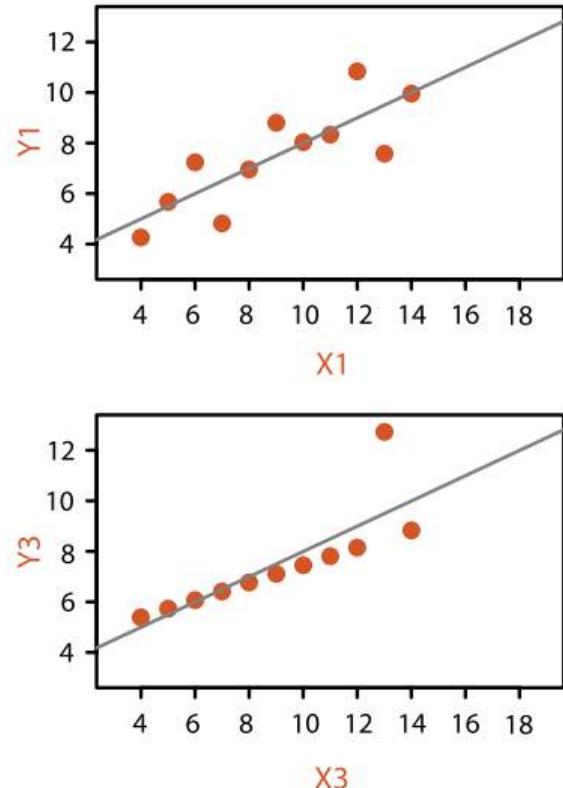
Trực quan hóa dữ liệu là gì?

- Trực quan hóa dữ liệu là việc biểu diễn đồ họa các thông tin trừu tượng nhằm 2 mục đích: Phân tích dữ liệu và truyền thông.
- Trực quan hóa dữ liệu là một công cụ mạnh mẽ để khám phá và trích rút các thông tin có giá trị (insight) từ tập dữ liệu.
- Bản chất của Trực quan hóa dữ liệu là sự trình bày dữ liệu theo định dạng hình ảnh hoặc đồ họa, từ đó truyền đạt thông tin rõ ràng và hiệu quả cho người dùng.
- Là yếu tố giao tiếp bằng hình ảnh của phân tích dữ liệu, giúp chuyển đổi dữ liệu thành thông tin và thông tin thành thông tin hữu ích.



Tầm quan trọng

	1		2		3		4	
	X	Y	X	Y	X	Y	X	Y
	10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
	8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
	13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
	9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
	11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
	14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
	6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
	4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
	12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
	7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
	5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89
Mean	9.0	7.5	9.0	7.5	9.0	7.5	9.0	7.5
Variance	10.0	3.75	10.0	3.75	10.0	3.75	10.0	3.75
Correlation	0.816		0.816		0.816		0.816	

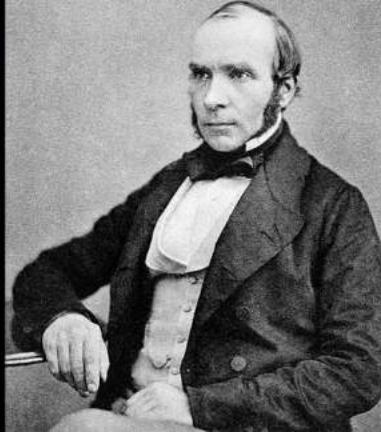
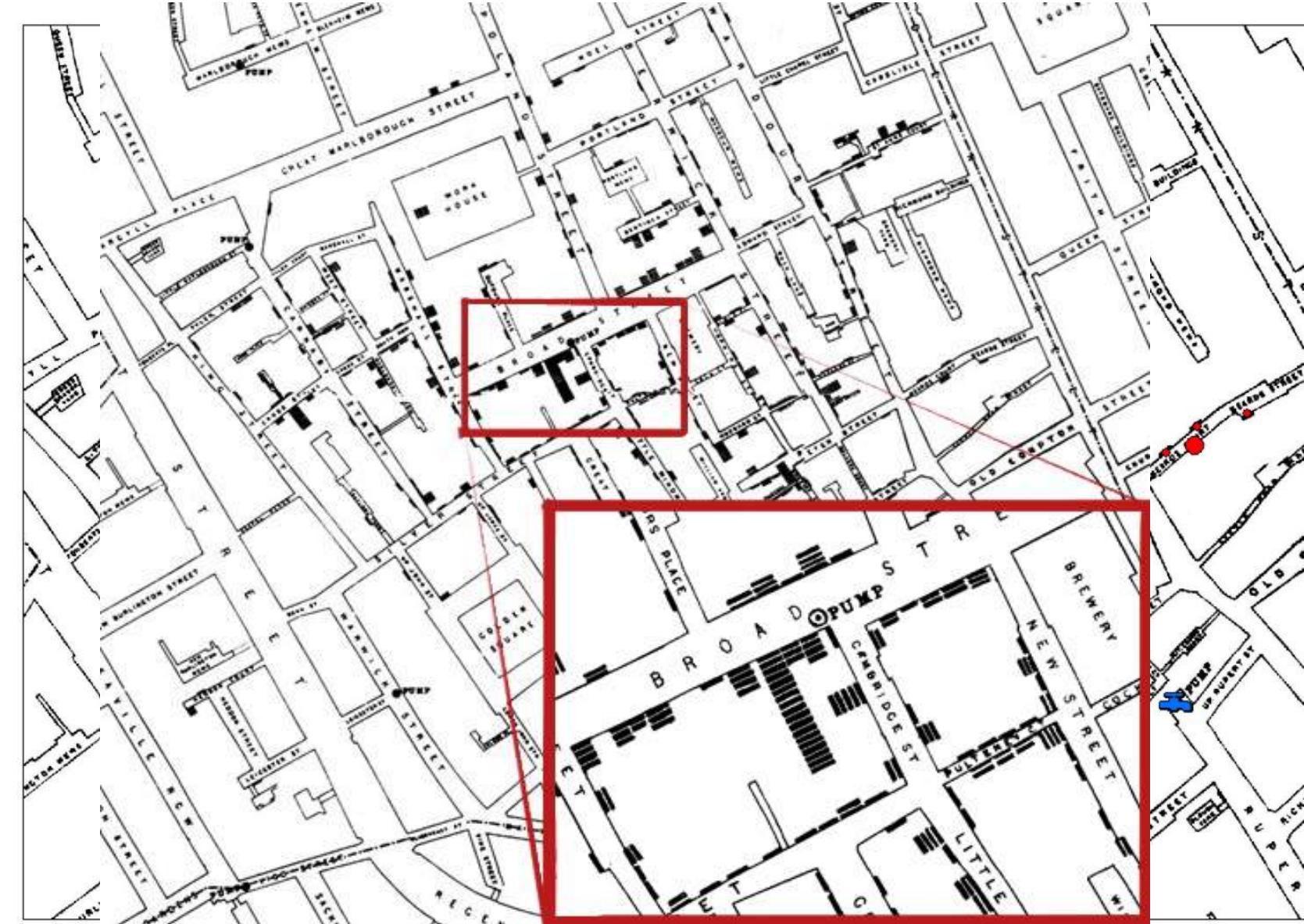


Tầm quan trọng



VINBIGDATA
VINGROUP

Academy
Vietnam



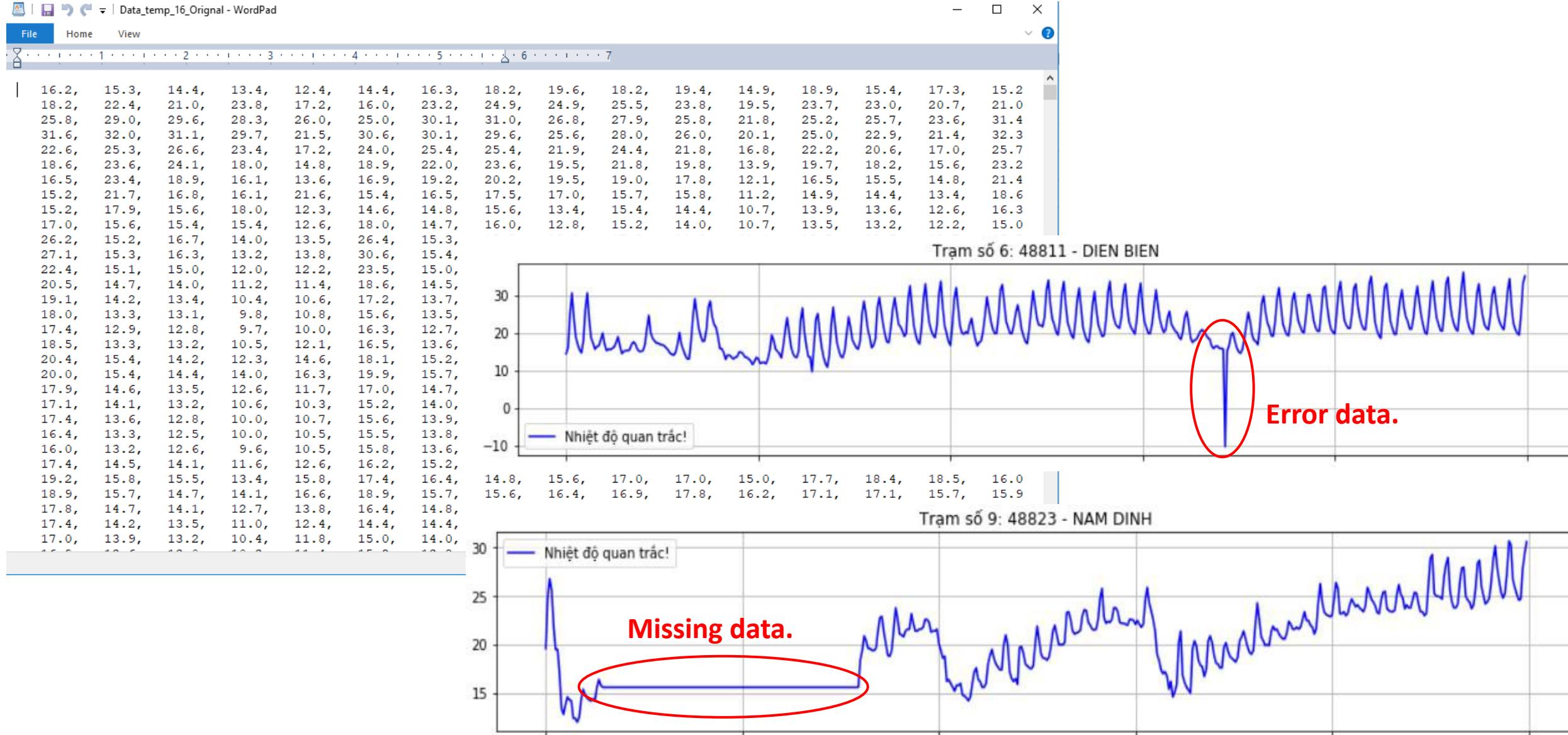
Dịch tả (London 1854), John Snow
600 người chết chỉ trong vài tuần...

Tầm quan trọng



VINBIGDATA VINGROUP

Academy Vietnam





THU CHI NGÂN SÁCH NHÀ NƯỚC 5T 2019

(Từ đầu năm đến thời điểm 15/5/2019)

TỔNG THU NGÂN SÁCH NHÀ NƯỚC



Tổng thu ngân sách Nhà nước bằng
39.2% so với dự toán năm 2019

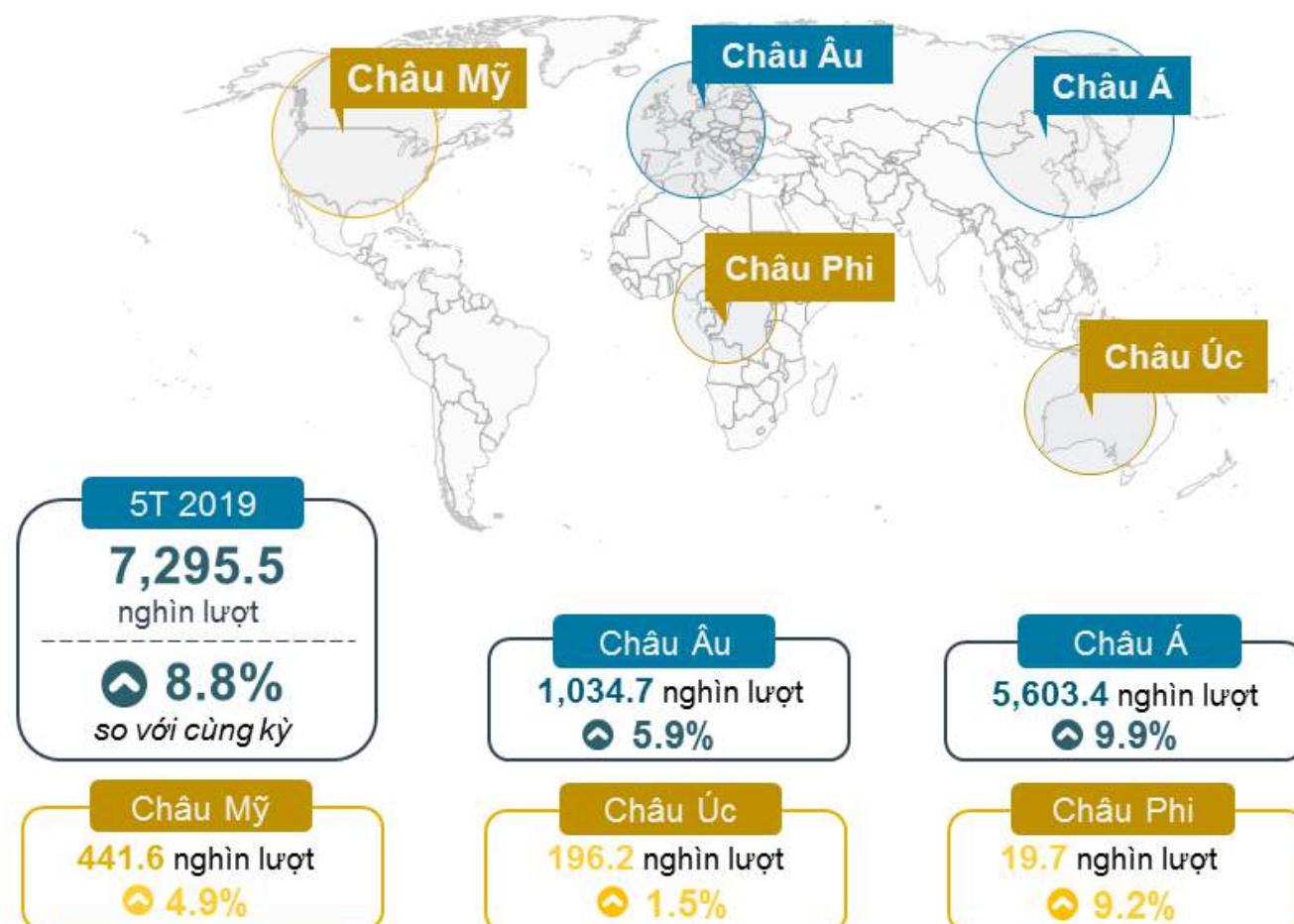
TỔNG CHI NGÂN SÁCH NHÀ NƯỚC



Tổng chi ngân sách Nhà nước bằng
29.8% so với dự toán năm 2019

Nguồn: Tổng cục Thống kê, Vietdata tổng hợp

KHÁCH QUỐC TẾ ĐẾN VIỆT NAM 5T 2019



THU HÚT VỐN ĐẦU TƯ NƯỚC NGOÀI

TOP 5

Khu vực có tổng vốn đăng ký
cấp mới nhiều nhất trong 5T 2019



Tính đến 20/5, tổng vốn đầu tư
đăng ký mới và tăng thêm đạt:

9.09 tỷ USD
so với cùng kỳ năm 2018 27.1%

Tổng vốn thực hiện trong 5T 2019

7.30 tỷ USD
so với cùng kỳ năm 2018 7.8%

1,363 dự án được **CẤP PHÉP MỚI**

6.46 tỷ USD **Tổng vốn đăng ký**

505 dự án **ĐIỀU CHỈNH TĂNG VDT**

2.63 tỷ USD **Tổng VĐK tăng thêm**

3,160 lượt **GÓP VỐN/ MUA CỔ PHẦN**

7.65 tỷ USD **Tổng giá trị góp vốn**

Nhóm ngành thu hút vốn FDI nhiều nhất 5T 2019

(số vốn đăng ký của các dự án được cấp phép mới)



Chế biến chế tạo
4.74 tỷ USD 73.5%



Kinh doanh BĐS
742.3 triệu USD 11.5%



Ngành khác
971.2 triệu USD 15.0%

“Một bức tranh bằng cả nghìn lời nói”

XUẤT NHẬP KHẨU HÀNG HÓA

XUẤT
KHẨU

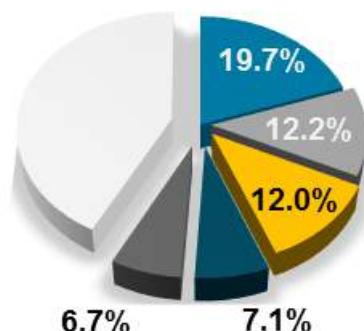
101.74
tỷ USD
↑6.7%

Tổng kim ngạch xuất khẩu
hàng hóa 5T 2019
so với cùng kỳ 2018

Thị trường xuất khẩu hàng hóa chính



Tỷ trọng một số mặt hàng XK chủ yếu



- Điện thoại/ Linh kiện
- Điện tử/ Máy tính/ Linh kiện
- Dệt, may
- Giày dép
- Máy móc/ Thiết bị/ Phụ tùng
- Khác

NHẬP
KHẨU

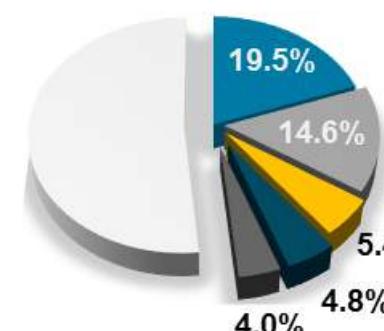
101.28
tỷ USD
↑10.3%

Tổng kim ngạch nhập khẩu
hàng hóa 5T 2019
so với cùng kỳ 2018

Thị trường nhập khẩu hàng hóa chính

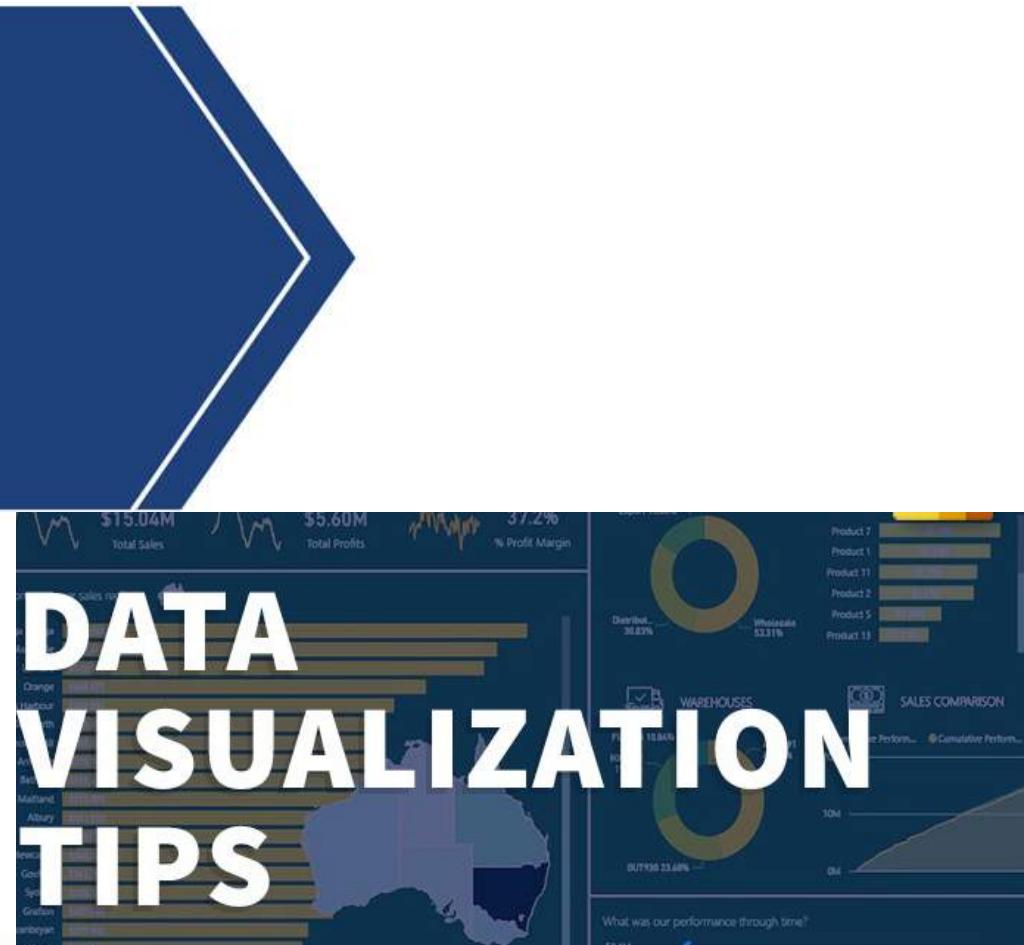


Tỷ trọng một số mặt hàng NK chủ yếu



- Điện tử/ Máy tính/ Linh kiện
- Máy móc/ Thiết bị/ Phụ tùng
- Vải
- Điện thoại/ Linh kiện
- Sắt thép
- Khác

3. Một số lưu ý khi trực quan hoá dữ liệu



<https://blog.csgsolutions.com/6-tips-for-creating-effective-data-visualizations>

Data Visualization Tips



VINBIGDATA VINGROUP

Academy
Vietnam



Data Visualization Tips

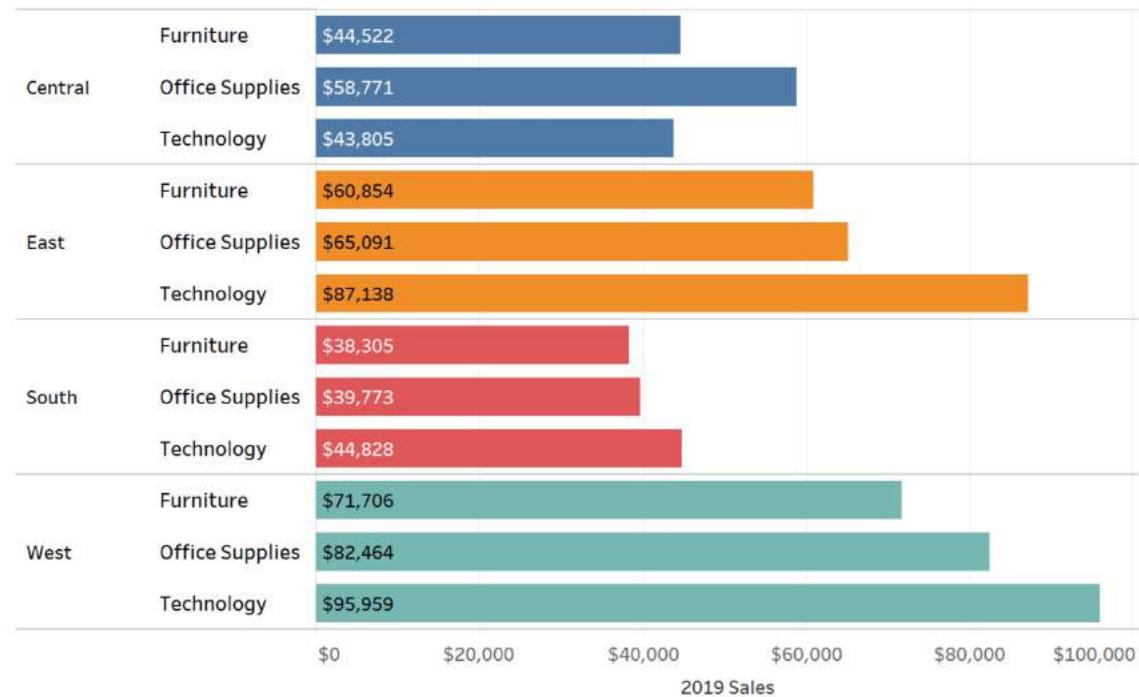
Tips 1: Xác định rõ mục đích và đối tượng cần truyền tải thông tin.

sales by region and category



! ineffective

sales by region and category

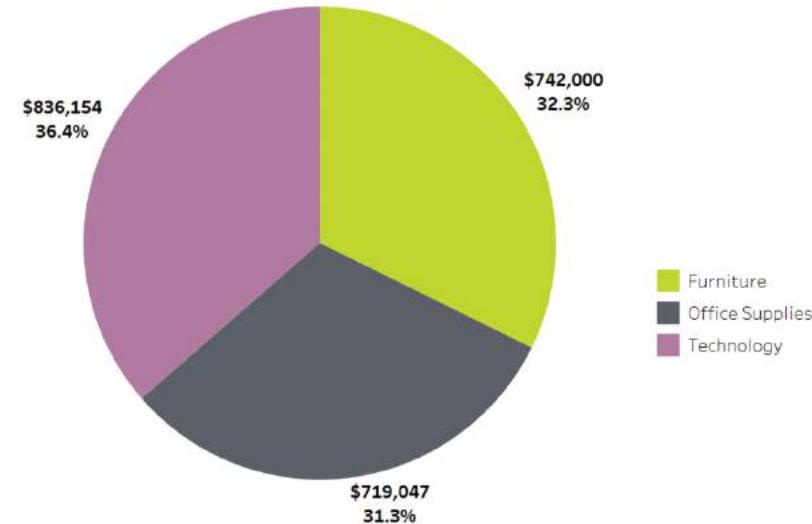


✓ effective

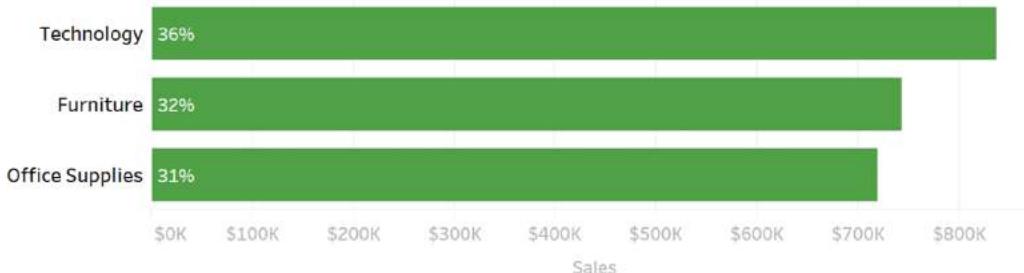
Data Visualization Tips

Tips 2: Lựa chọn loại biểu đồ phù hợp với mục đích của mình.

sales by product category



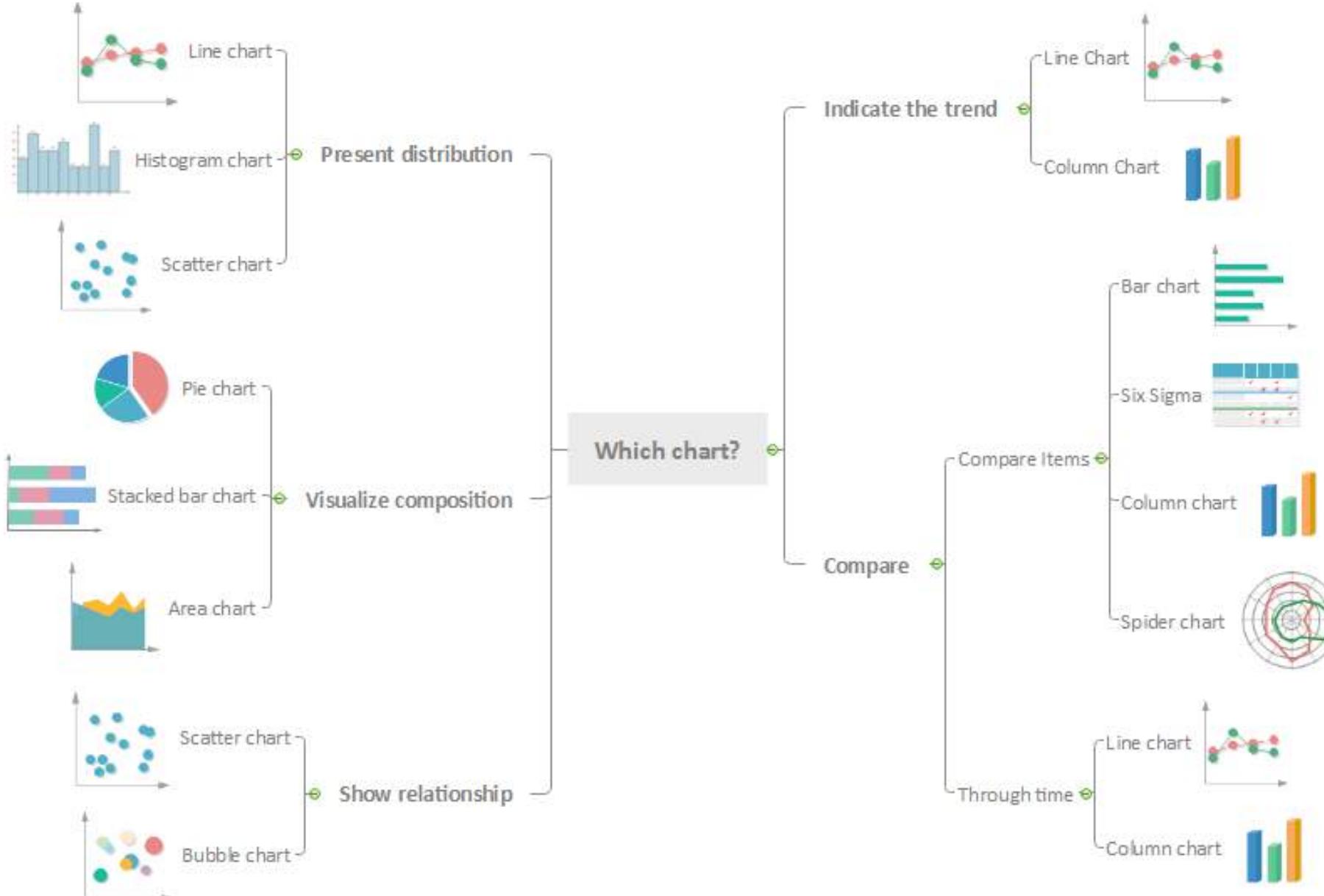
sales by product category



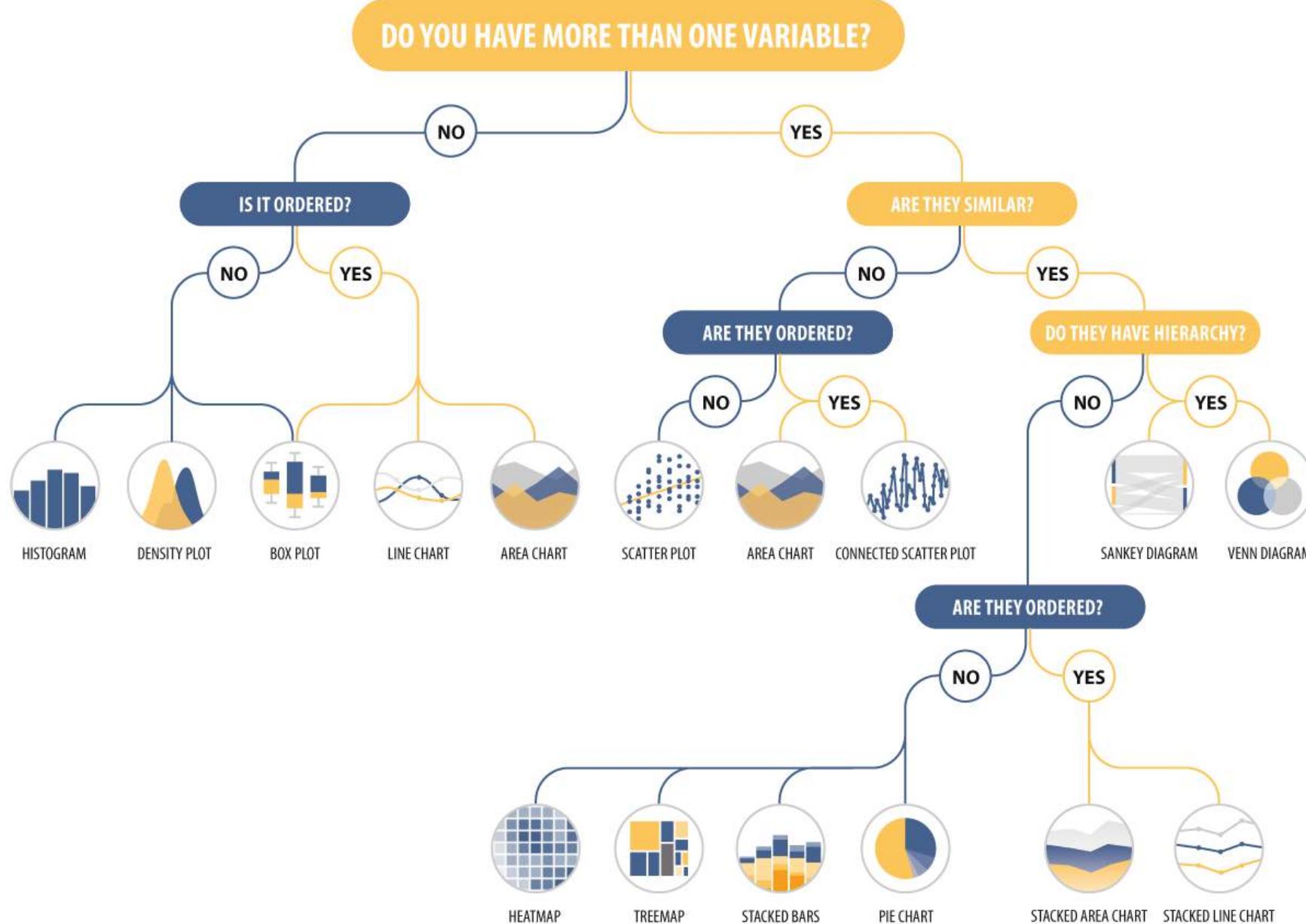
! ineffective

✓ effective

Data Visualization Tips



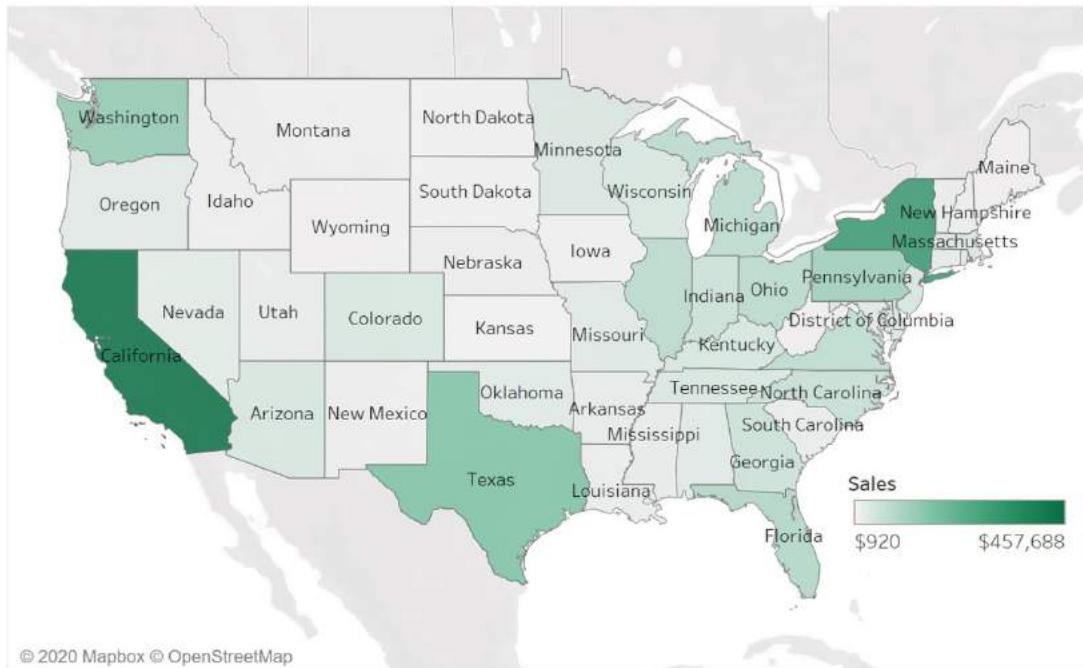
Data Visualization Tips



Data Visualization Tips

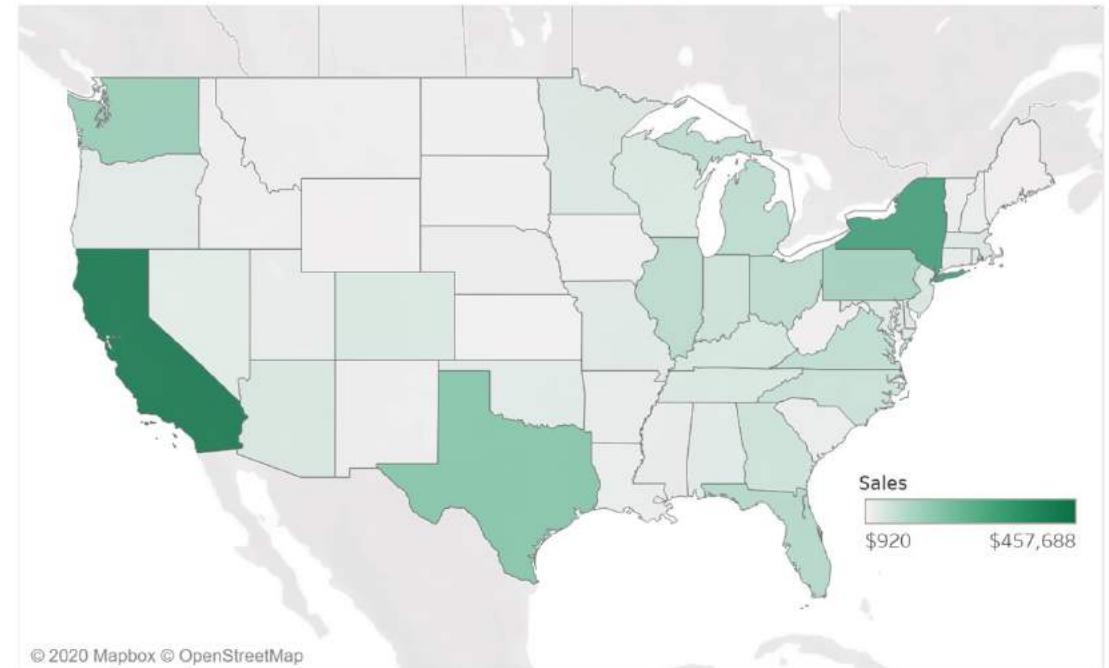
Tips 3: Đưa văn bản, nhãn vào biểu đồ hợp lý, tránh lộn xộn

total sales map



! ineffective

total sales map



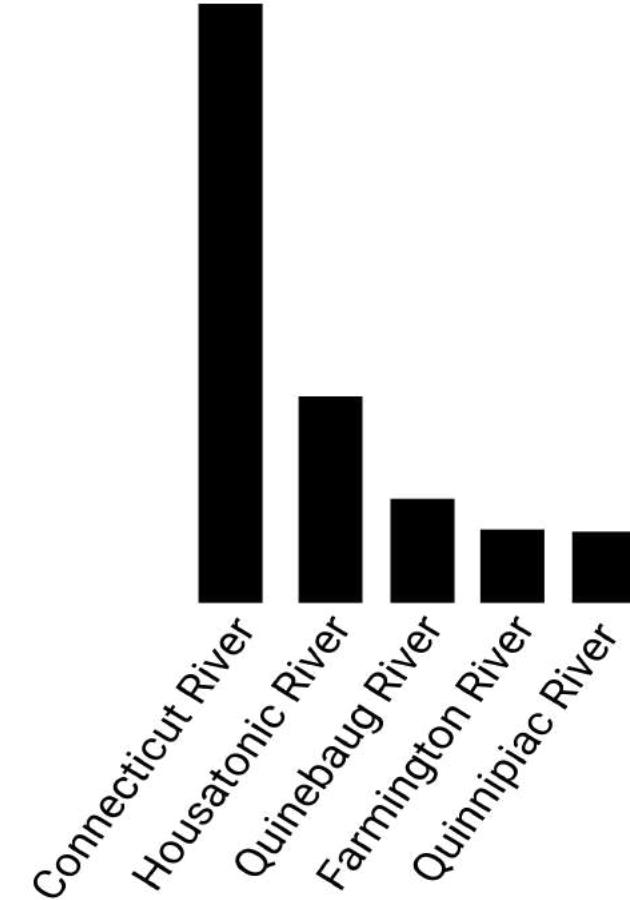
✓ effective

Data Visualization Tips

Tips 3: *Đưa văn bản, nhãn vào biểu đồ hợp lý, tránh lộn xộn (T)*



Good

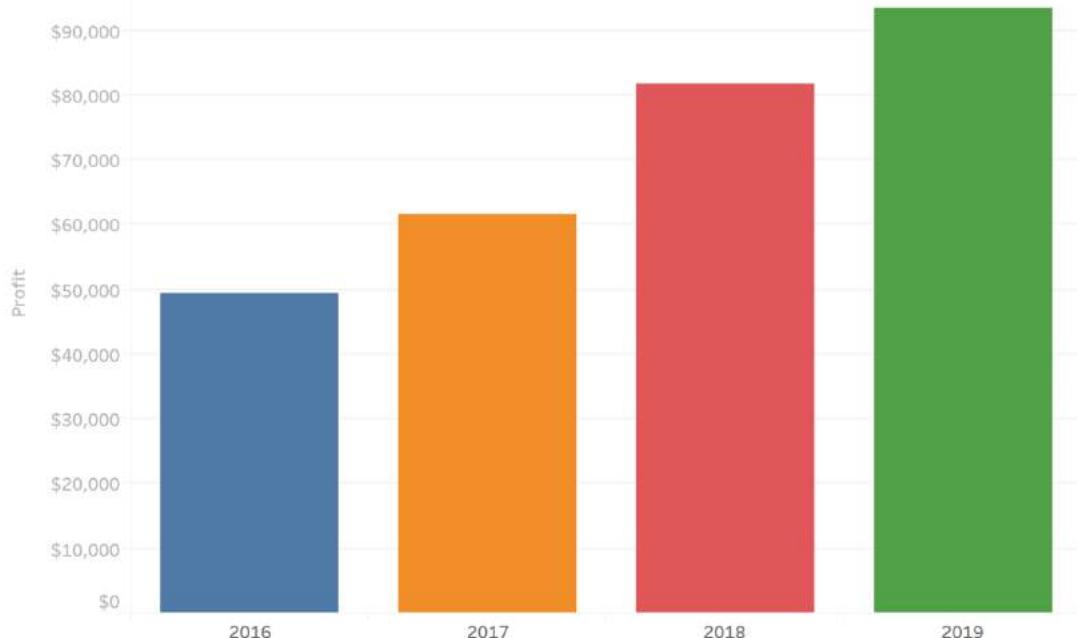


Could be better

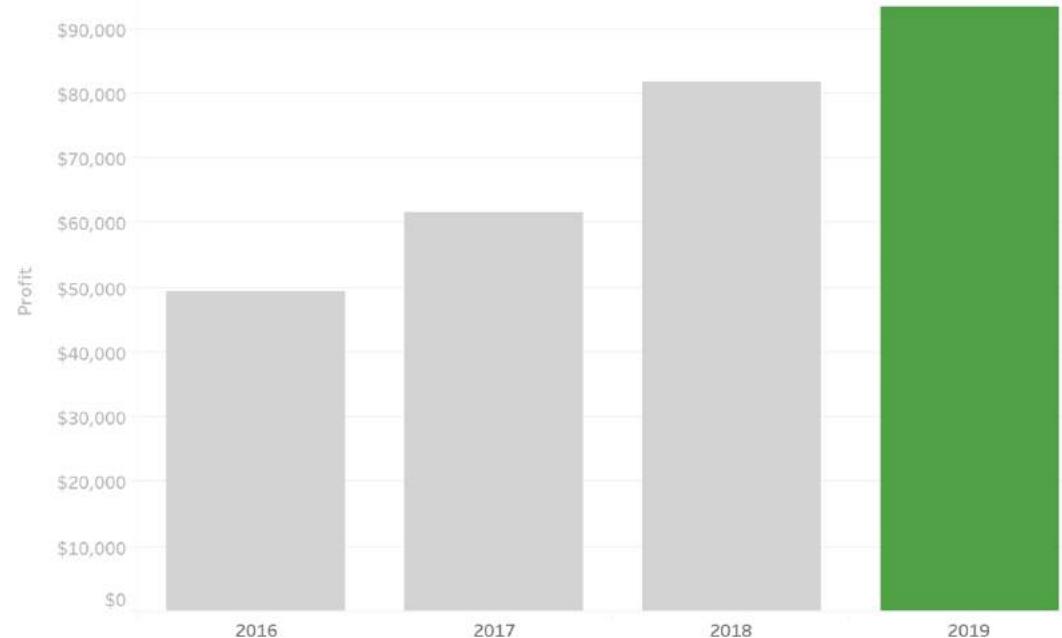
Data Visualization Tips

Tips 4: Sử dụng màu sắc hiệu quả để làm nổi bật các thông tin quan trọng.

profit by year



profit by year

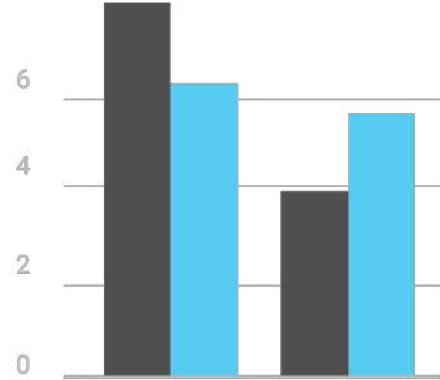


! ineffective

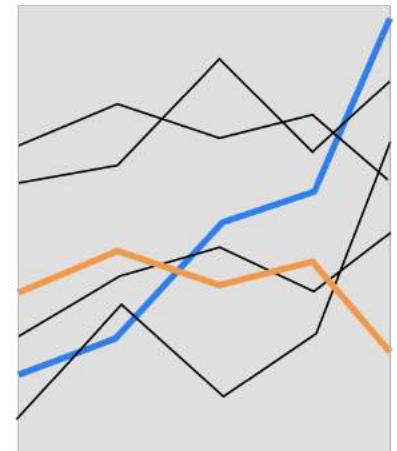
✓ effective

Data Visualization Tips

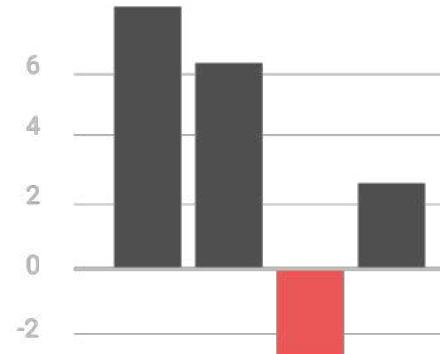
Tips 4: Sử dụng màu sắc hiệu quả để làm nổi bật các thông tin quan trọng (*t*).



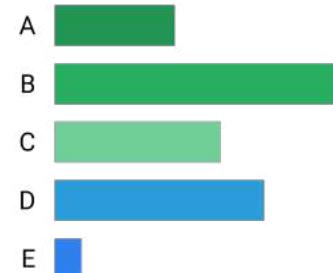
Good. Blue helps to distinguish between different data series



Good. Colored lines are distinguishable



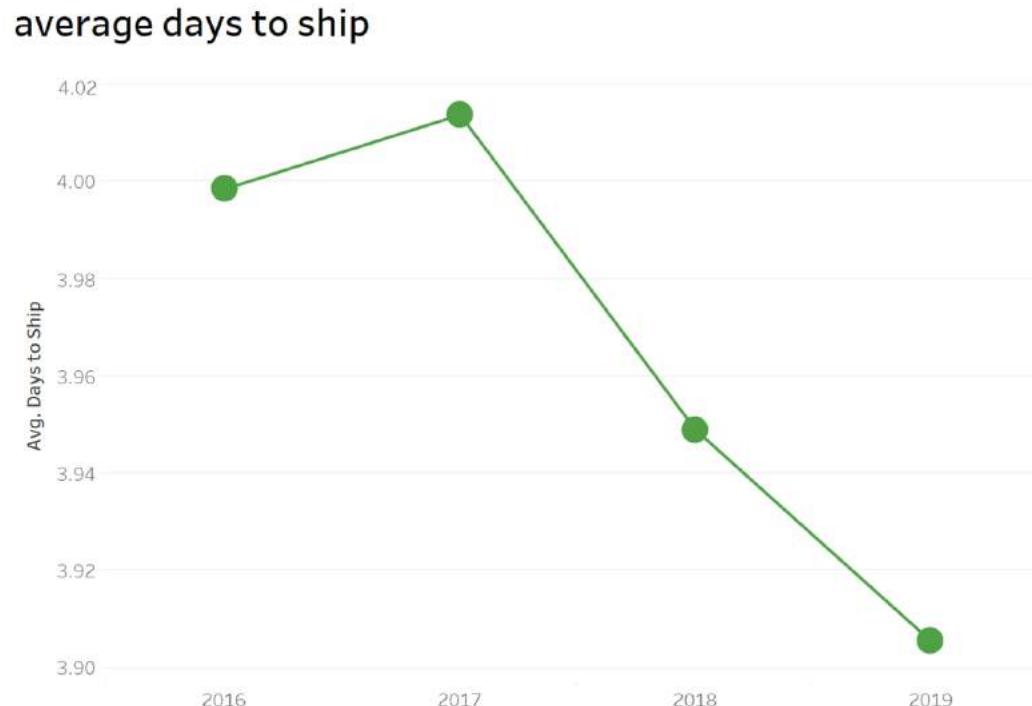
Fine. Red adds emphasis, but is not absolutely necessary



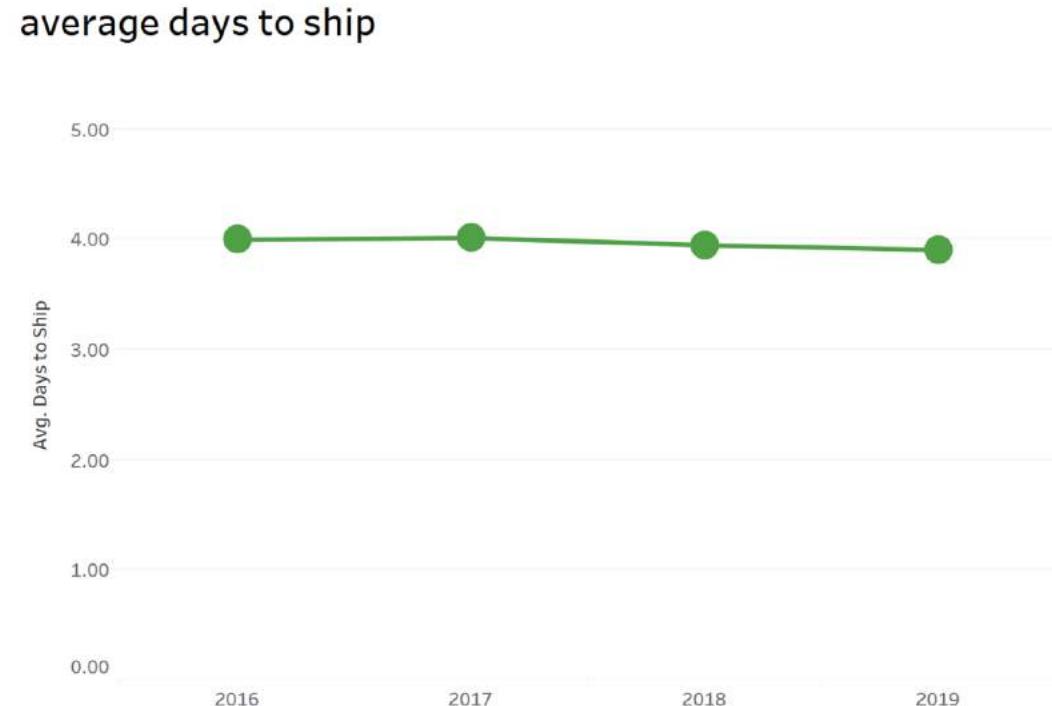
Bad. Colors are too similar, and are not needed for a separated bar chart

Data Visualization Tips

Tips 5: Tránh để người xem hình dung sai lệch dữ liệu.



! ineffective

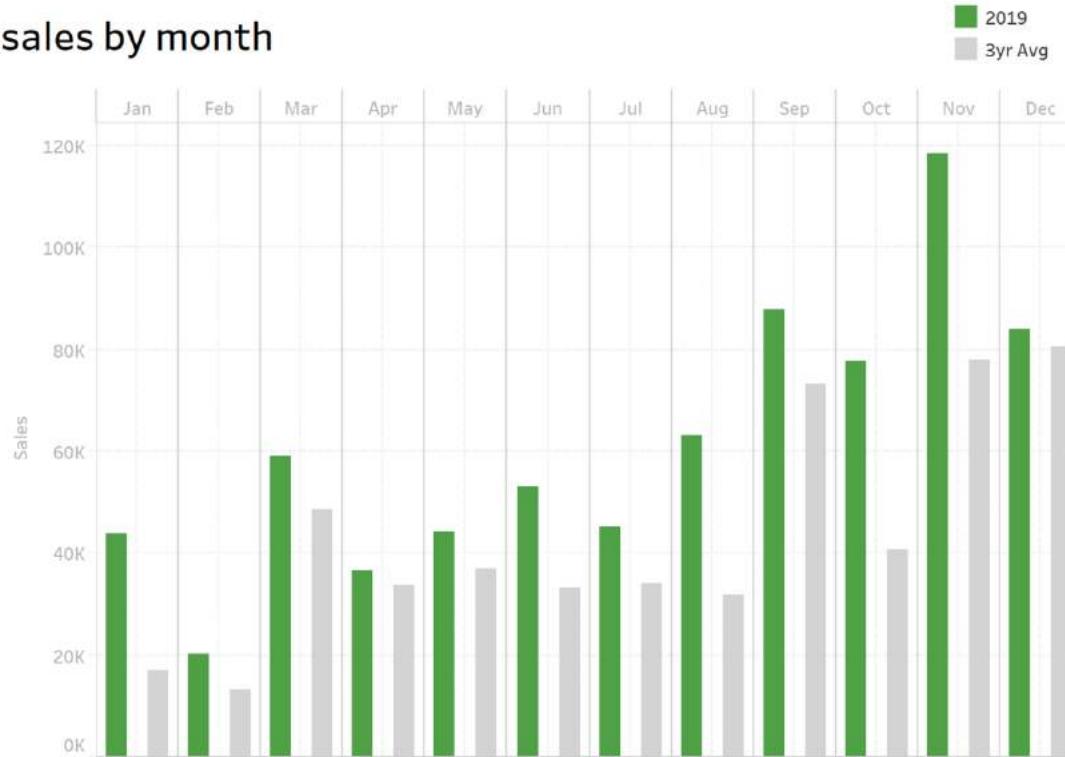


✓ effective

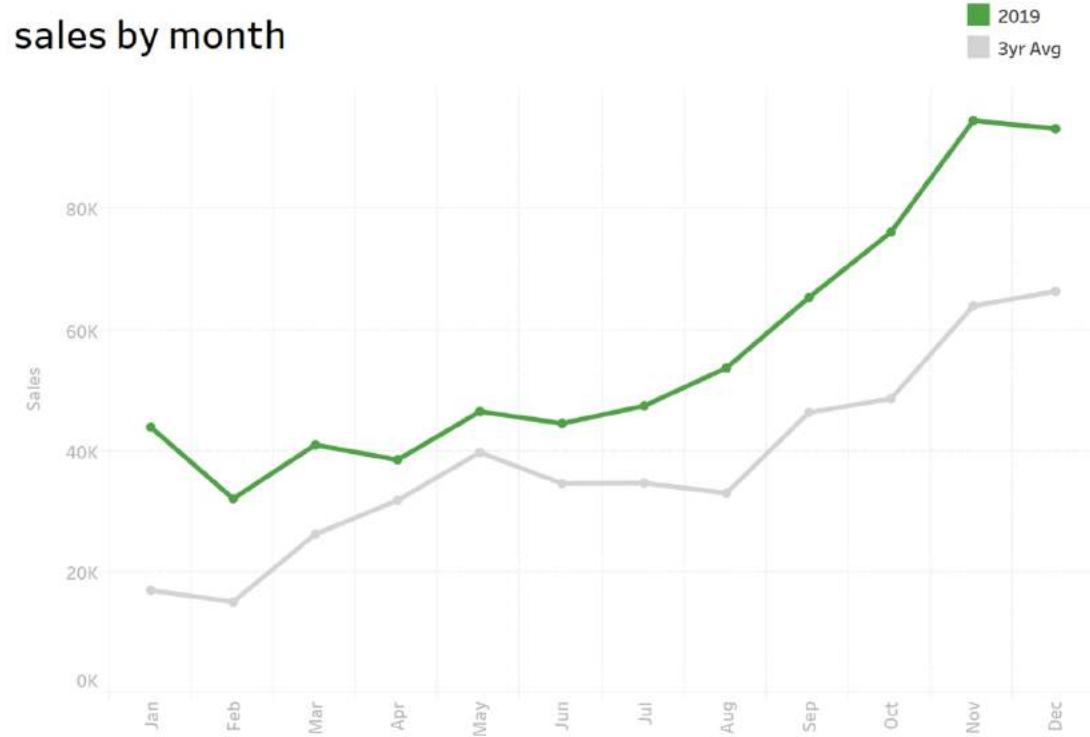
Data Visualization Tips

Tips 6: Sử dụng các biểu đồ càng đơn giản càng tốt.

sales by month



sales by month



! ineffective

✓ effective

Data Visualization Tips

Tips 7: Cân nhắc sắp xếp dữ liệu để đạt hiệu quả

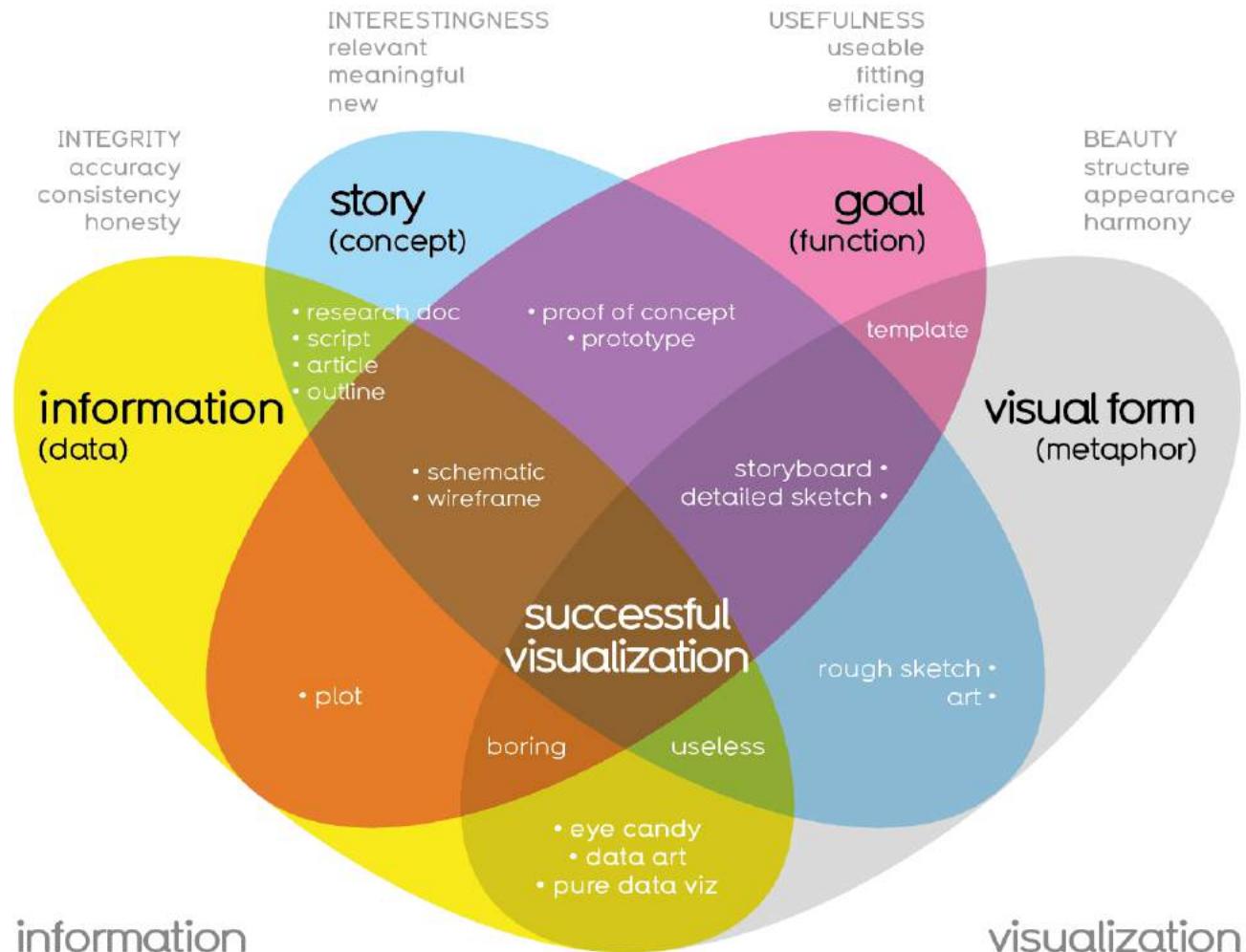


Good (Alphabetical)

Good (Values sorted)

Bad (Random order)

What Makes a Good Visualization?



3. Thư viện trực quan hóa với Python

Một số thư viện trực quan hóa



VINBIGDATA VINGROUP

Academy
Vietnam

Có nhiều thư viện mạnh mẽ để trực quan hóa dữ liệu với ngôn ngữ lập trình Python.

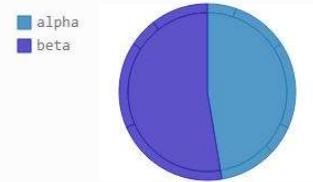
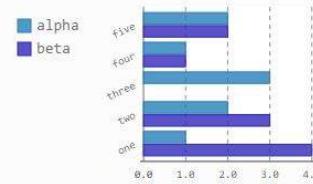
matplotlib

seaborn

plotly

Pygal

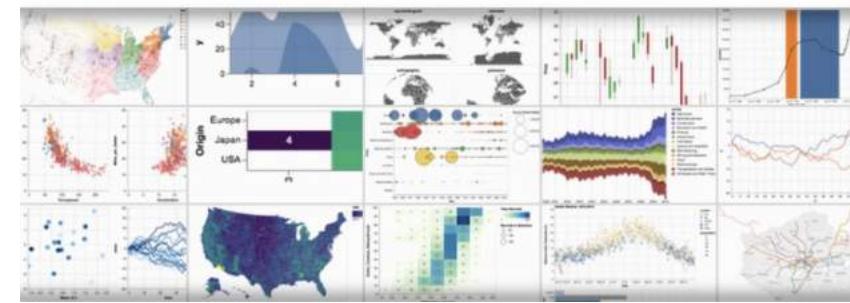
Sexy python charting



bokeh

Altair

Declarative Visualization in Python



- **Matplotlib** là thư viện dùng để vẽ đồ thị rất mạnh mẽ, có cú pháp tương tự như Matlab. Thư viện này được phát triển sớm nhất, 2003.
- Hỗ trợ nhiều loại biểu đồ, đặc biệt là các loại được sử dụng trong nghiên cứu hoặc kinh tế như biểu đồng đường, cột, tần suất (histograms), tương quan, scatterplots...
- Cấu trúc của Matplotlib gồm nhiều phần, phục vụ cho các mục đích sử dụng khác nhau. Trong đó module pyplot được sử dụng nhiều nhất, có cú pháp tương tự như Matlab.
- Matplotlib miễn phí và mã nguồn mở.

Tham khảo:

- + File: **CheatSheet-Matplotlib**
- + Link web: [**Matplotlib package!**](#)

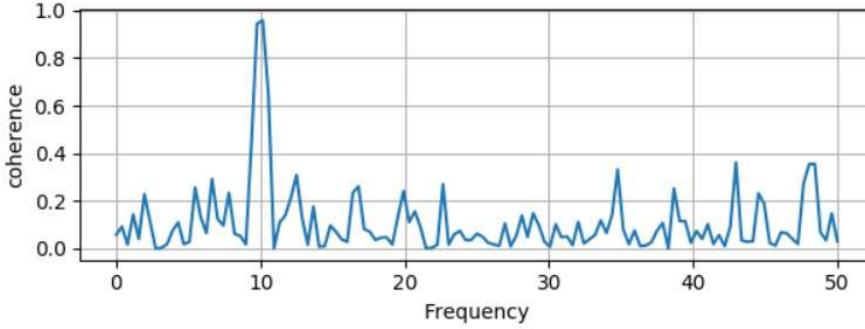
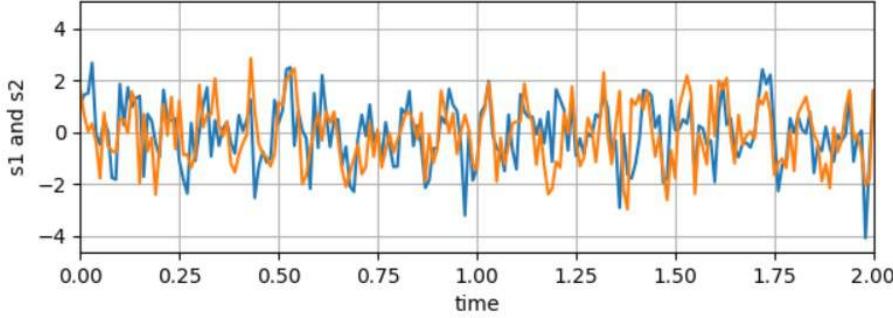


Thư viện matplotlib

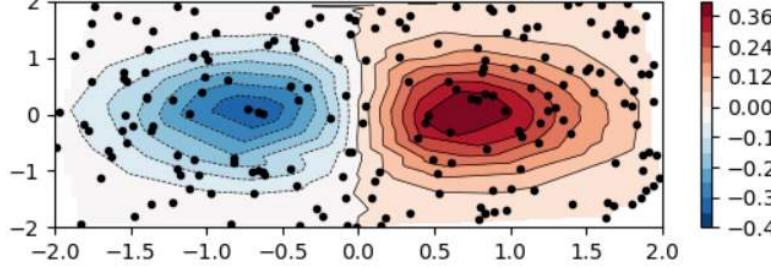


VINBIGDATA VINGROUP

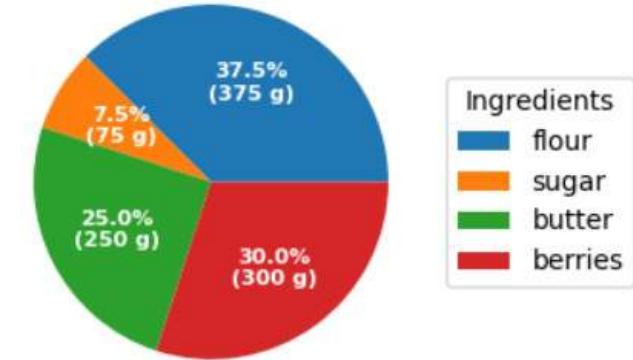
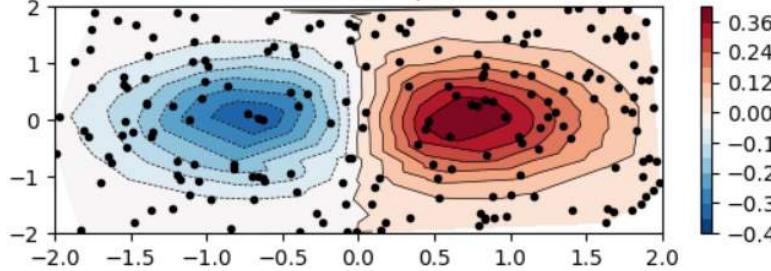
Academy
Vietnam



grid and contour (200 points, 20000 grid points)



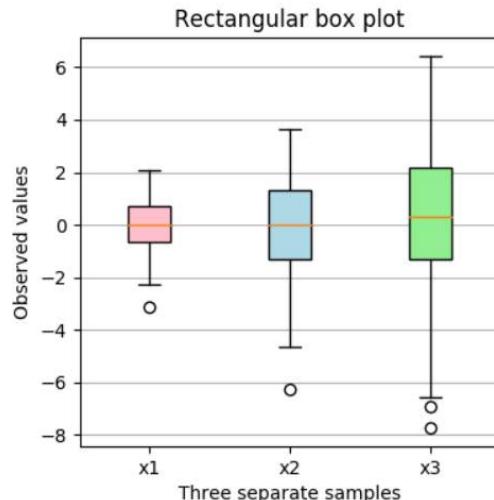
tricontour (200 points)



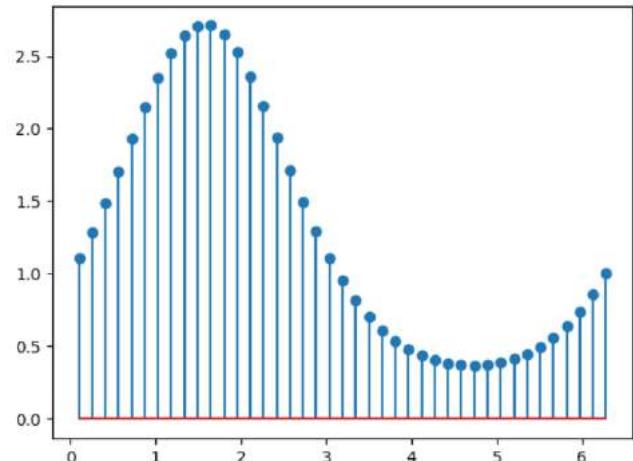
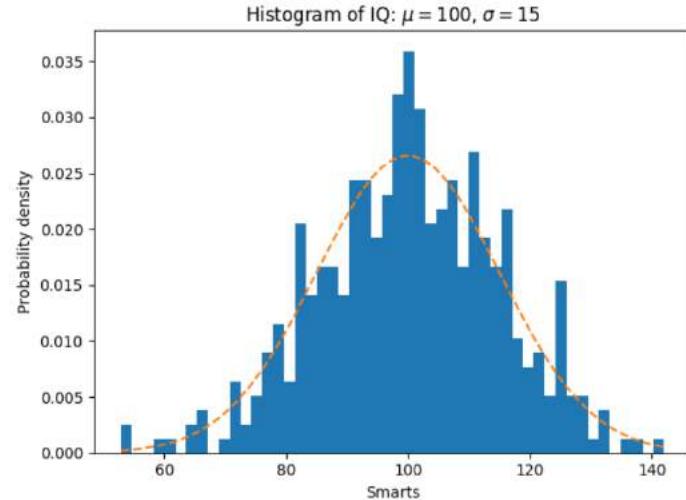
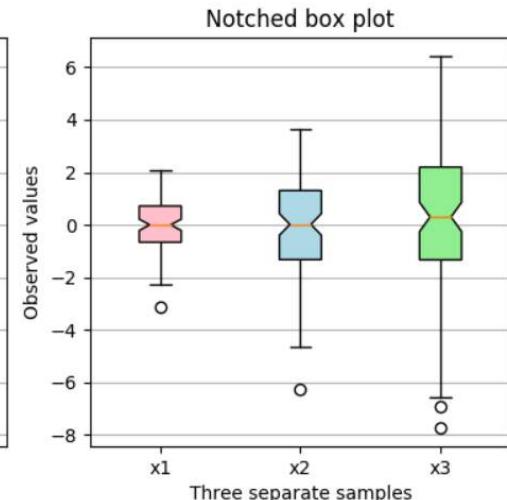
Ingredients

- flour
- sugar
- butter
- berries

matplotlib



Notched box plot



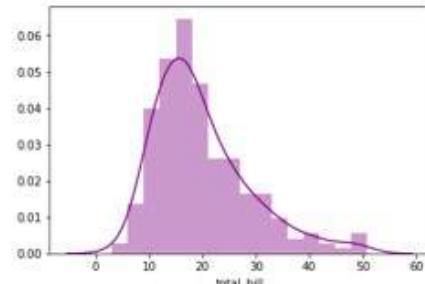
- **Seaborn** là một trong những thư viện mạnh mẽ, phổ biến trong việc trực quan hóa dữ liệu. Seaborn được coi là công cụ bổ sung cho Matplotlib (Mở rộng của Matplotlib).
- Seaborn bổ sung thêm các dạng biểu đồ mạnh mẽ, và nhiều tính năng mới, giúp cho việc trực quan hóa các biểu đồ, dữ liệu phức tạp trở nên dễ dàng hơn. Một số ưu điểm của Seaborn:
 1. Mục đích chính của Seaborn là làm cho việc trực quan hóa dữ liệu trở nên dễ dàng. Do đó, nó được xây dựng để tự động xử lý rất nhiều phép toán phức tạp ở phía sau.
 2. Seaborn hoạt động cực kỳ hiệu quả với cấu trúc dữ liệu của Pandas, đây là một thư viện Python được sử dụng rộng rãi để phân tích dữ liệu.
 3. Seaborn được xây dựng trên Matplotlib, là một thư viện trực quan hóa Python khác. Matplotlib cực kỳ linh hoạt. Seaborn cho phép chúng ta tận dụng tính linh hoạt để tránh được sự phức tạp trong quá trình xử lý dữ liệu.



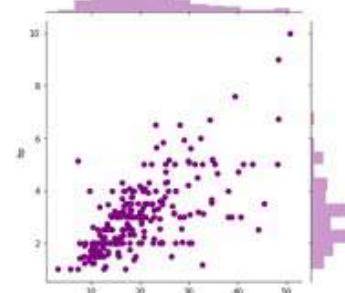
Tham khảo:

- + File: [CheatSheet-Seaborn](#)
- + Link web: [Seaborn package!](#) 31

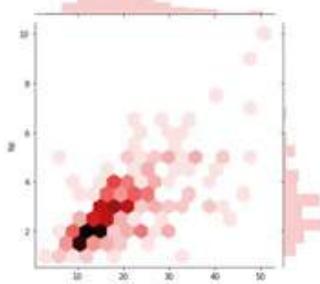
Seaborn Plots



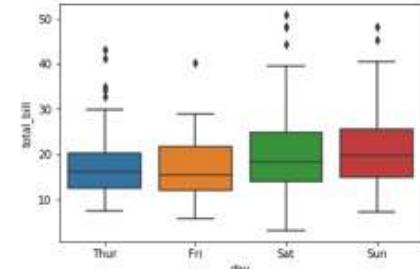
distplot



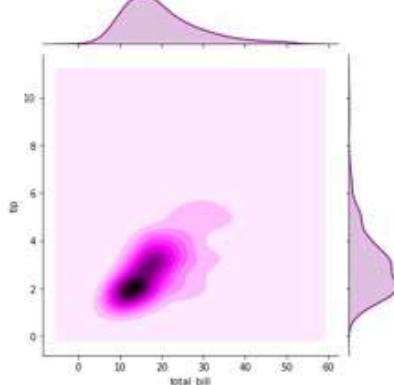
Jointplot



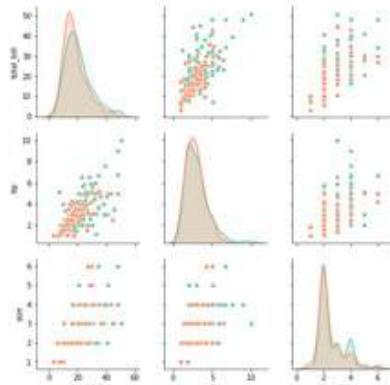
Hexplots



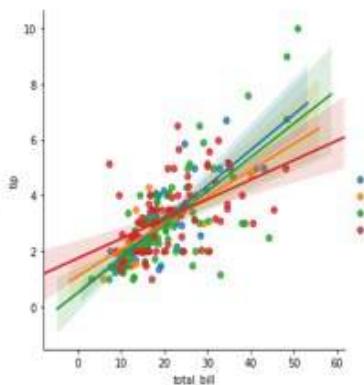
Boxplots



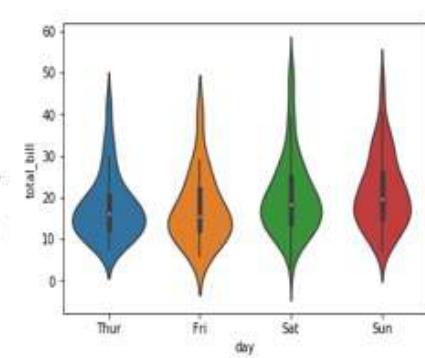
KDE Plot



Pair Plots



LM Plots



Violin Plots



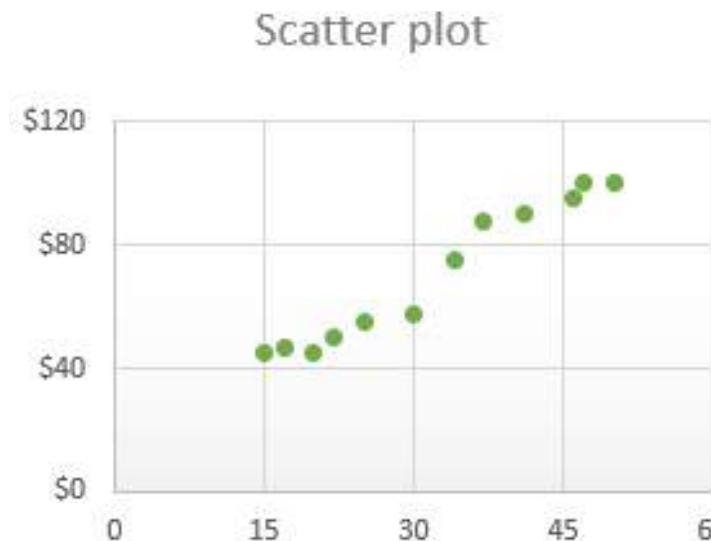
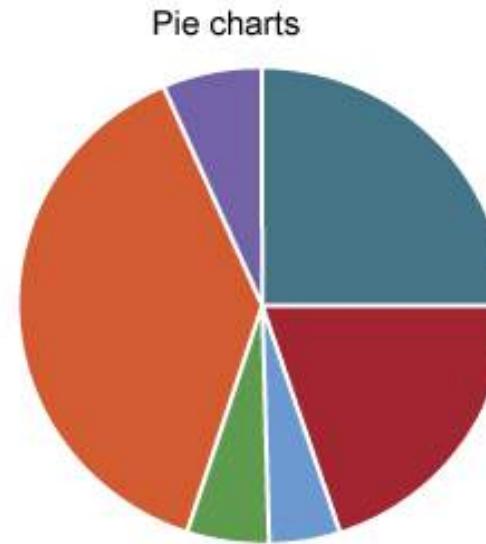
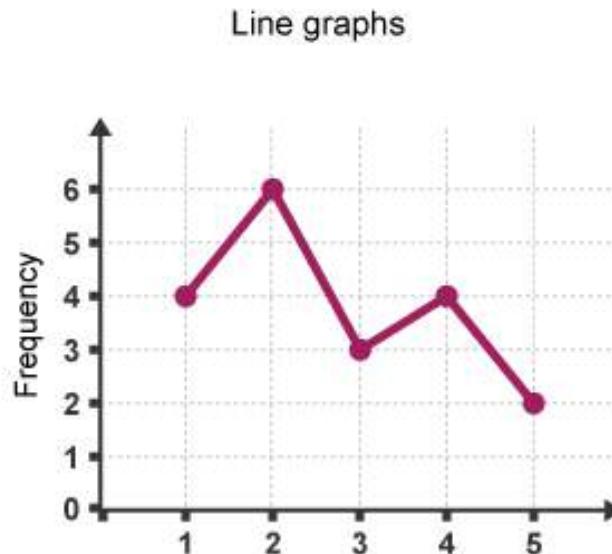
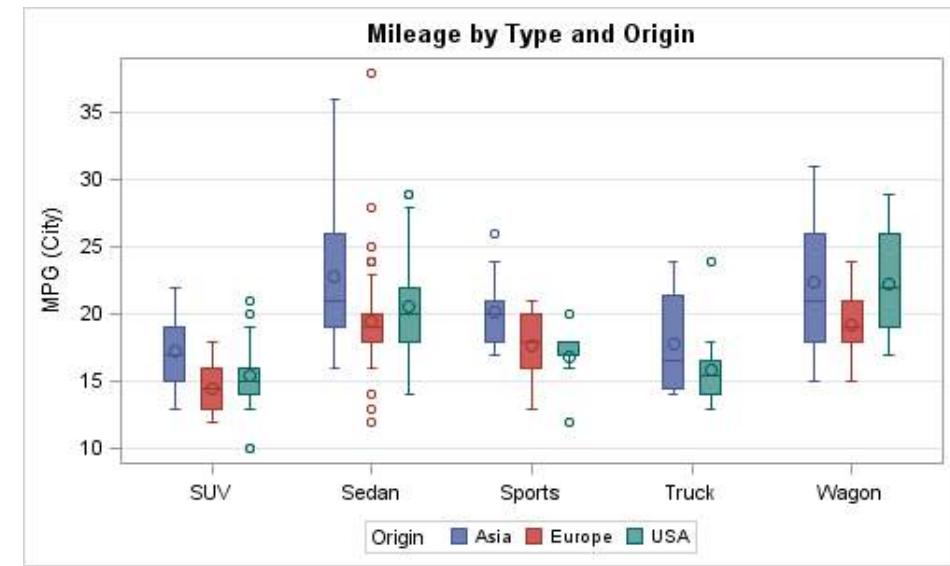
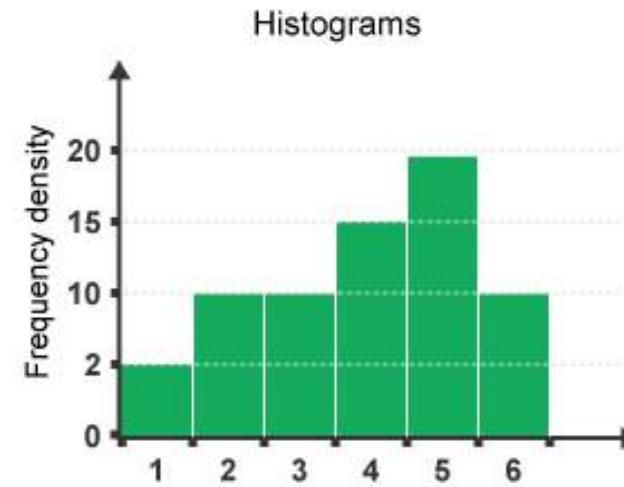
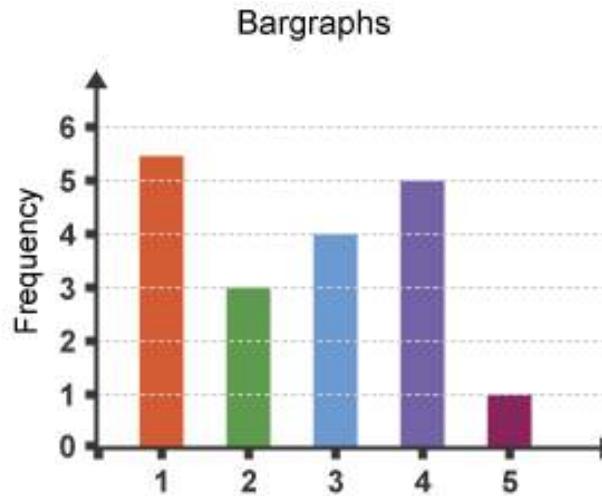
seaborn³²



MỘT SỐ BIỂU ĐỒ QUAN TRỌNG



Các dạng biểu đồ quan trọng

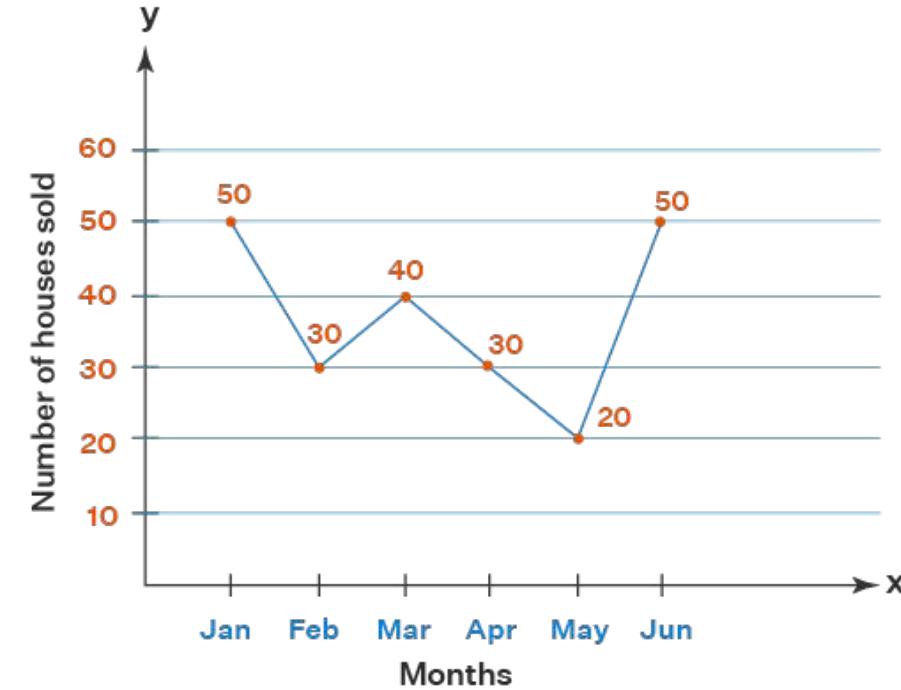
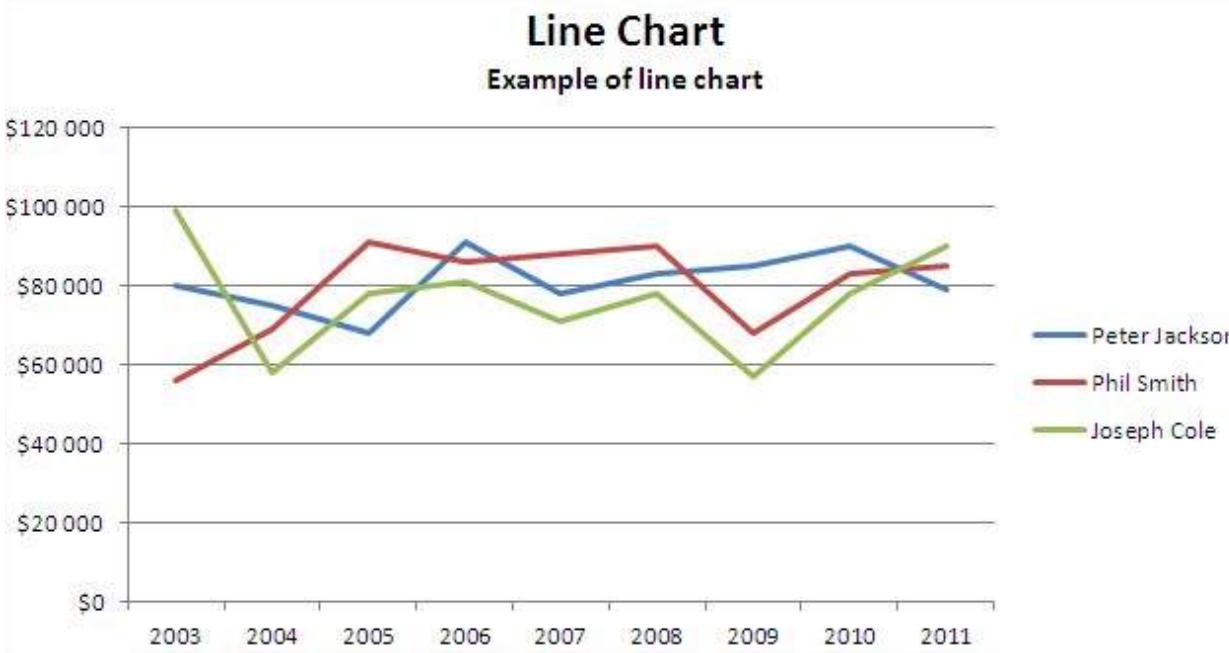




4. Biểu đồ đường (line chart)



Đồ thị dạng đường (line chart)



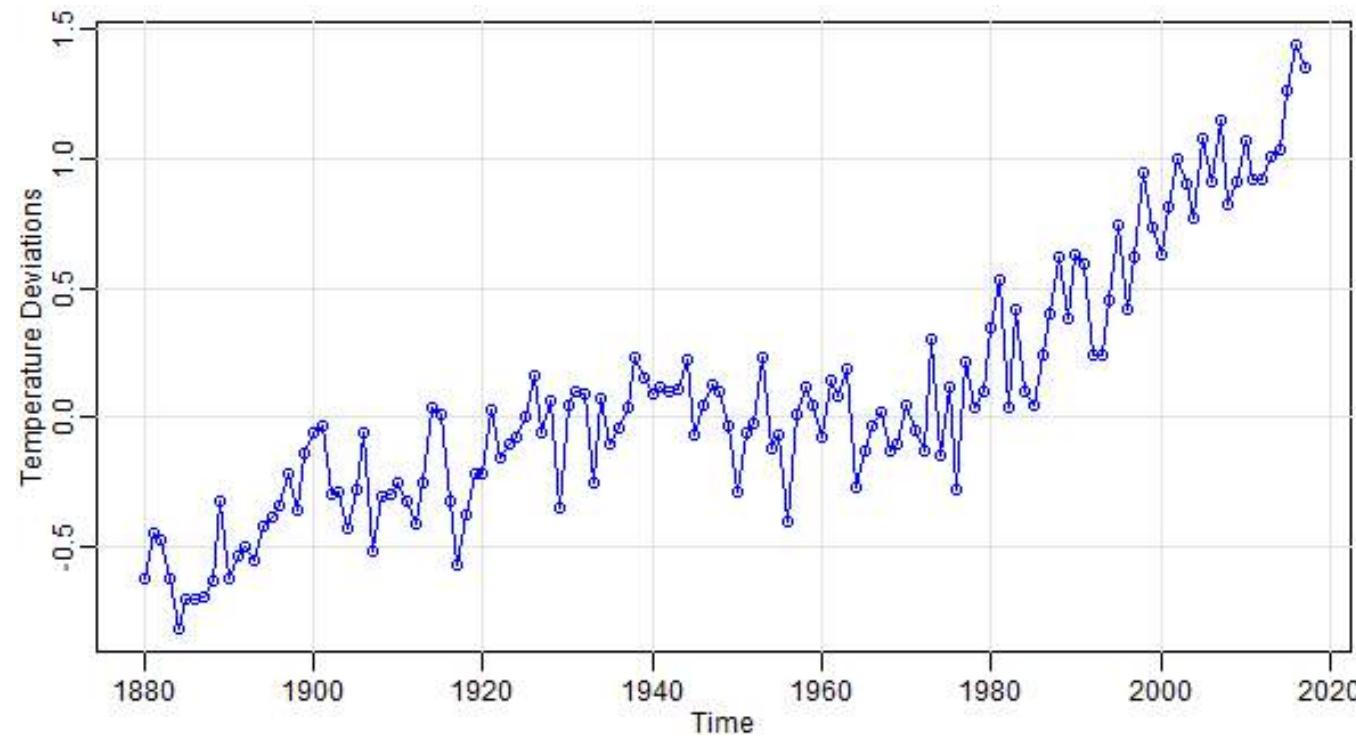
Line chart là một trong những dạng đồ thị phổ biến và hay được sử dụng trong thực tế.

- Khi muốn trình bày các dữ liệu liên mạch biểu đồ line chart là sự lựa chọn phù hợp. Các điểm trong biểu đồ đường được nối liền thành một đường, thể hiện mối quan hệ giữa các điểm đó. Thông thường các dòng dữ liệu sẽ liên quan đến các đơn vị đo lường thời gian như ngày, tháng, quý và năm.

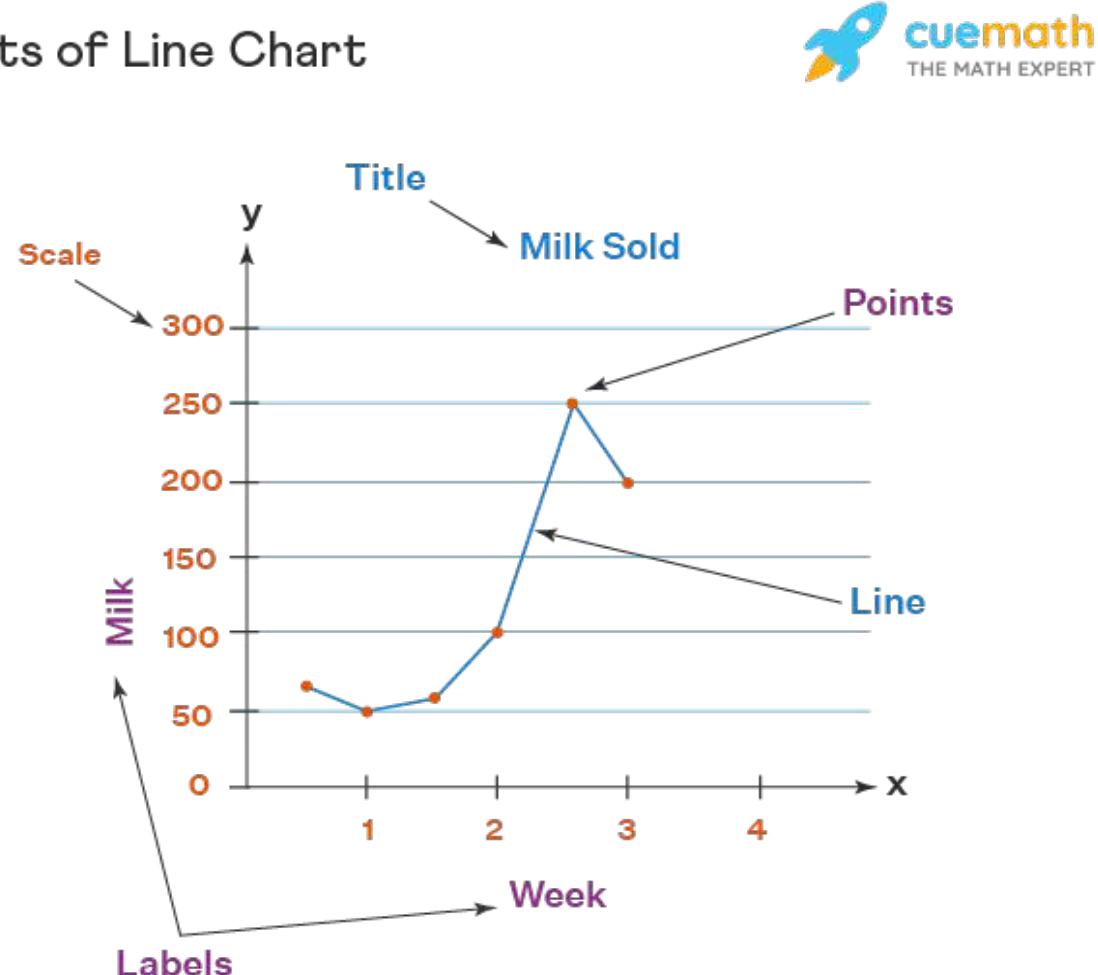


Đồ thị dạng đường (line chart)

Line chart giúp nhấn mạnh sự thay đổi trong dữ liệu của một biến vẽ trên trục x so với biến thứ 2 trên trục y.



Parts of Line Chart



Đồ thị dạng đường với Matplotlib



VINBIGDATA VINGROUP

Academy
Vietnam

Tập dữ liệu `gas_prices.csv`: Lưu trữ giá Gas của 10 nước trên thế giới trong
giai đoạn từ năm 1990 - 2008

```
1 data = pd.read_csv('Data_Visualize/gas_prices.csv')  
2 data
```

	Year	Australia	Canada	France	Germany	Italy	Japan	Mexico	South Korea	UK	USA
0	1990	NaN	1.87	3.63	2.65	4.59	3.16	1.00	2.05	2.82	1.16
1	1991	1.96	1.92	3.45	2.90	4.50	3.46	1.30	2.49	3.01	1.14
2	1992	1.89	1.73	3.56	3.27	4.53	3.58	1.50	2.65	3.06	1.13
3	1993	1.73	1.57	3.41	3.07	3.68	4.16	1.56	2.88	2.84	1.11
4	1994	1.84	1.45	3.59	3.52	3.70	4.36	1.48	2.87	2.99	1.11
5	1995	1.95	1.53	4.26	3.96	4.00	4.43	1.11	2.94	3.21	1.15
6	1996	2.12	1.61	4.41	3.94	4.39	3.64	1.25	3.18	3.34	1.23
7	1997	2.05	1.62	4.00	3.53	4.07	3.26	1.47	3.34	3.83	1.23
8	1998	1.63	1.38	3.87	3.34	3.84	2.82	1.49	3.04	4.06	1.06
9	1999	1.72	1.52	3.85	3.42	3.87	3.27	1.79	3.80	4.29	1.17
10	2000	1.94	1.86	3.80	3.45	3.77	3.65	2.01	4.18	4.58	1.51

Đồ thị dạng đường với Matplotlib

Cú pháp:

plt.plot(x, y, color, linestyle, linewidth, marker, markersize)

Trong đó:

- * X, Y – dữ liệu trục X, Y

Hàm pyplot.plot() còn có các tham số cơ bản sau:

- * Color (c): Màu của đường line
- * Linewidth (lw): Số thực - Độ rộng của đường đồ thị
- * linestyle (ls): Kiểu đường đồ thị
- * marker: Kiểu của điểm
- * markersize (ms): Số thực - Kích thước của điểm dữ liệu

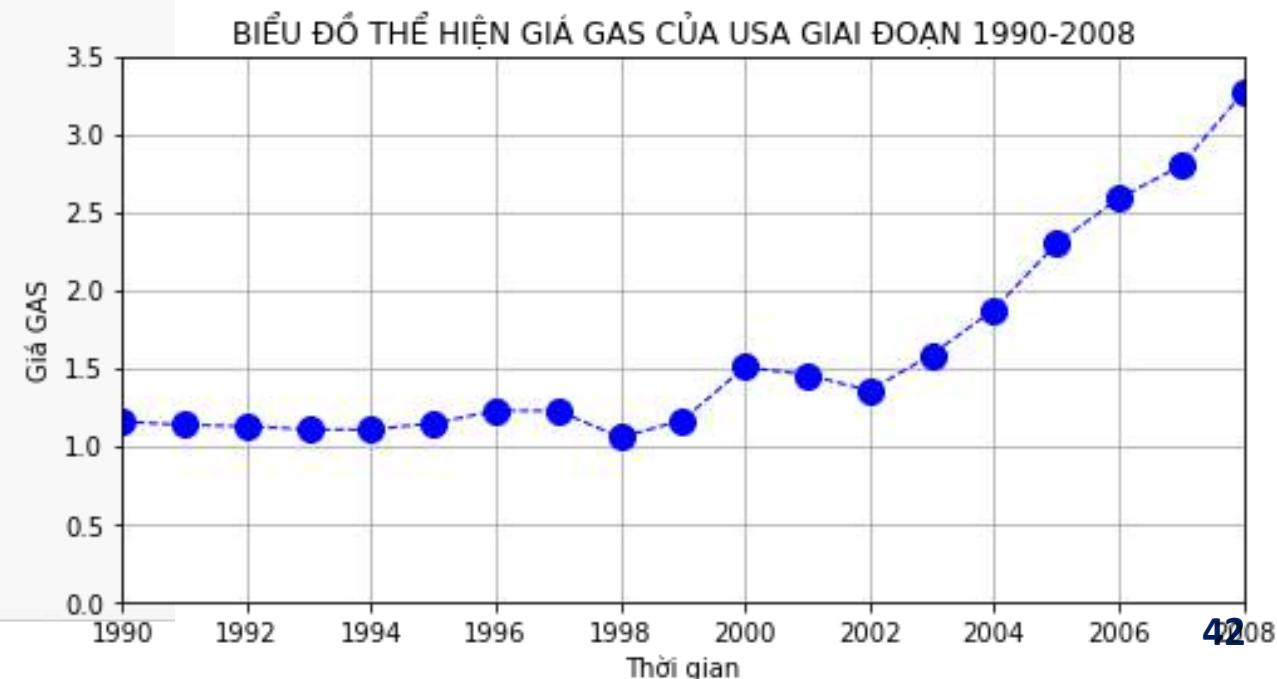
Các tham số color, marker, linestyle có thể được biểu diễn ở dạng '[color][marker][linestyle]', ví dụ: 'ro-' tương đương với color='r', marker='o', linestyle='-'.



a. Single line chart

Simple line chart

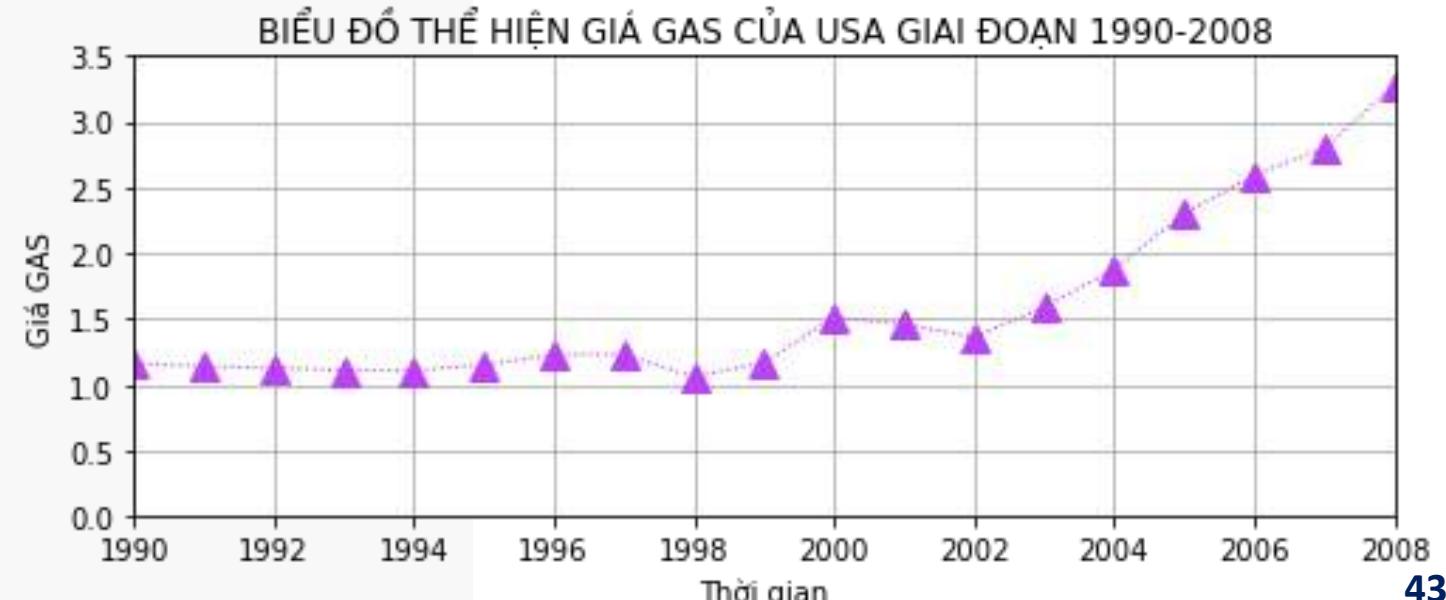
```
1 plt.figure(figsize = (8,4)) #Thiết lập kích thước biểu đồ
2
3 plt.plot(x,
4             y,
5             color='b',
6             linestyle='--',
7             linewidth=1.0,
8             marker='o',
9             markersize = 10) #Kích thước điểm
10
11 #Tiêu đề của đồ thị
12 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CỦA USA GIAI ĐOẠN 1990-2008')
13 #Nhãn cho trục X
14 plt.xlabel('Thời gian')
15 #Nhãn cho trục Y
16 plt.ylabel('Giá GAS')
17
18 #Setup giới hạn cho trục X:
19 plt.xlim(1990,2008)
20
21 #Setup giới hạn cho trục Y:
22 plt.ylim(0,3.5)
23
24 #Hiển thị lưới:
25 plt.grid()
26
27 plt.show()
```





Simple line chart

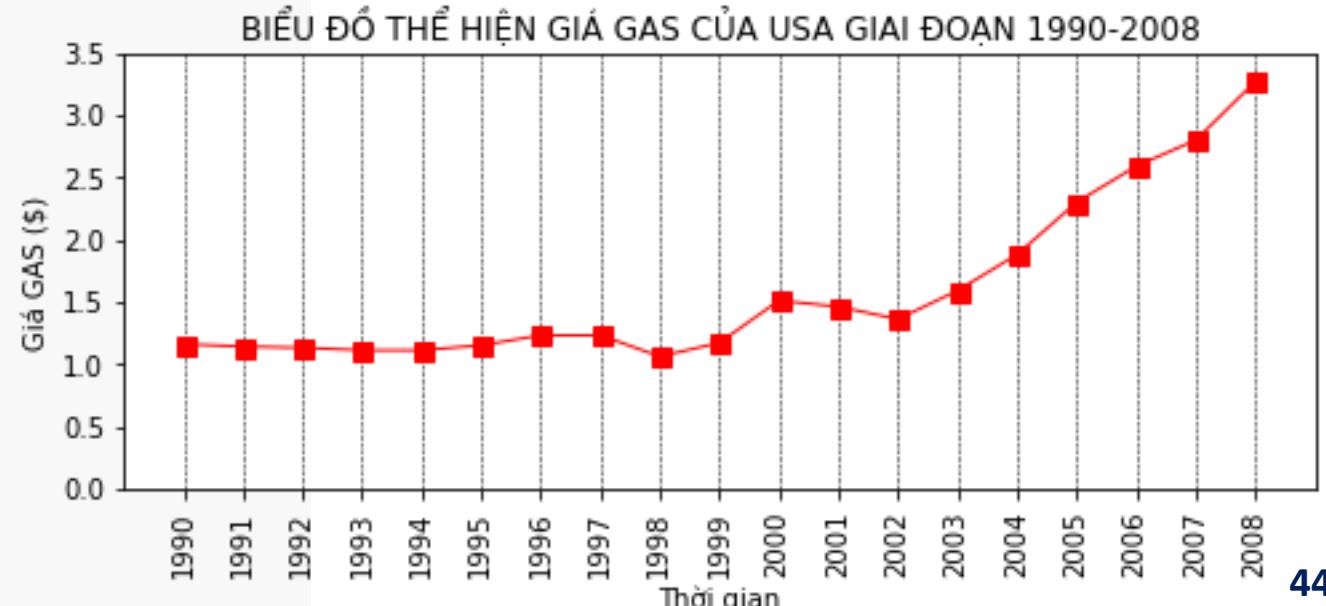
```
1 plt.figure(figsize = (8,3)) #Thiết lập kích thước biểu đồ
2
3 plt.plot(x,                      #Dữ liệu trục X
4           y,                      #Dữ liệu trục Y
5           c='#bc42f5',            #Màu của đường
6           ls=':',                #Kiểu đường
7           lw=1.0,                #Độ rộng của đường line
8           marker='^',            #Kiểu điểm
9           ms = 10)               #Kích thước điểm
10
11 #Tiêu đề của đồ thị
12 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CỦA USA GIAI ĐOẠN 1990-2008')
13 #Nhãn cho trục X
14 plt.xlabel('Thời gian')
15 #Nhãn cho trục Y
16 plt.ylabel('Giá GAS')
17
18 #Setup giới hạn cho trục X:
19 plt.xlim(1990,2008)
20
21 #Setup giới hạn cho trục Y:
22 plt.ylim(0,3.5)
23
24 #Hiển thị lưới:
25 plt.grid()
26
27 plt.show()
```



Simple line chart

```
1 plt.figure(figsize = (8,3)) #Thiết lập kích thước biểu đồ
2
3 plt.plot(x,                      #Dữ liệu trục X
4           y,                      #Dữ liệu trục Y
5           'r-s',                  #Độ rộng của đường line
6           linewidth=1.0,          #Độ rộng của đường line
7           markersize = 7)        #Kích thước điểm
8
9 #Tiêu đề của đồ thị
10 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CỦA USA GIAI ĐOẠN 1990-2008')
11 #Nhãn cho trục X
12 plt.xlabel('Thời gian')
13 #Nhãn cho trục Y
14 plt.ylabel('Giá GAS ($)')
15 #Setup giới hạn cho trục X:
16 plt.xlim(1989,2009)
17 #Setup giới hạn cho trục Y:
18 plt.ylim(0,3.5)
19 #Setup tick cho trục X:
20 plt.xticks(x,
21             rotation=90)
22 #Thiết lập lưới:
23 plt.grid(axis='x',
24           c='black',
25           ls='--',
26           lw=0.5)
27
28 plt.show()
```

Các tham số *color*, *marker*, *linestyle* có thể
được biểu diễn ở dạng
[color][marker][linestyle],
ví dụ: ‘ro-’ tương đương với *color='r'*,
marker='o', *linestyle='-'*.



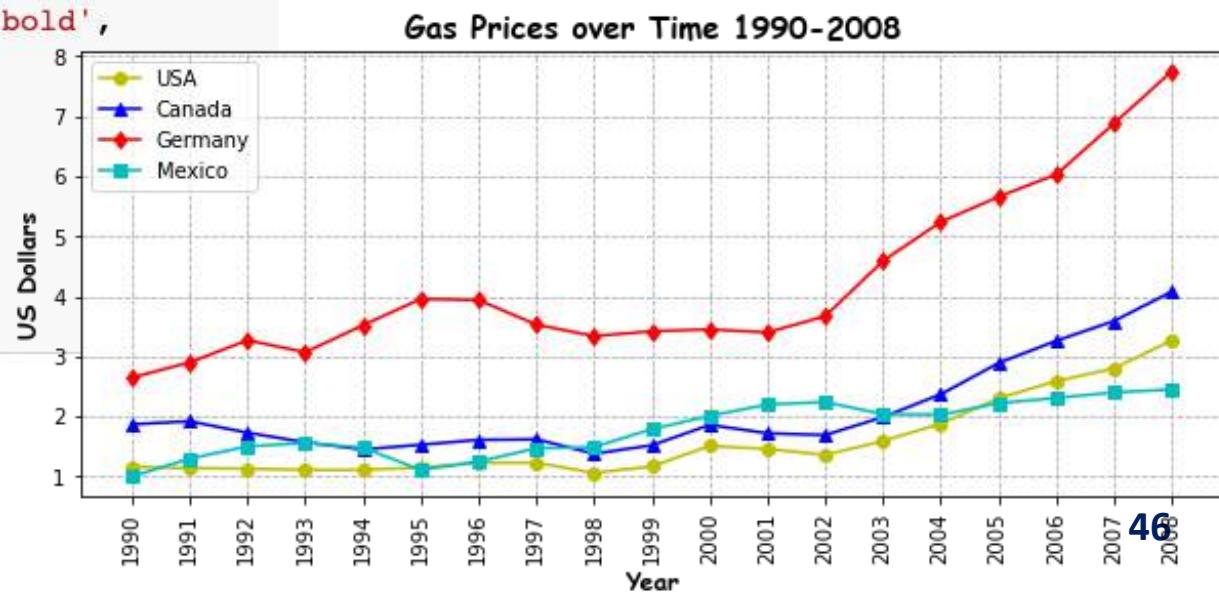


b. Multiple line chart

Multiple lines chart



```
1 plt.figure(figsize=(10,4))
2
3 #Vẽ multiple line:
4 plt.plot(x,y1,'y-o', label='USA')
5 plt.plot(x,y2,'b^-', label='Canada')
6 plt.plot(x,y3,'r-d', label='Germany')
7 plt.plot(x,y4,'c-s', label='Mexico')
8
9
10 plt.title('Gas Prices over Time 1990-2008',fontdict={'fontname':'Comic Sans MS',
11                               'fontweight':'bold',
12                               'fontsize':15})
13 plt.xlabel('Year',fontdict={'fontname':'Comic Sans MS',
14                               'fontweight':'bold',
15                               'fontsize':12})
16 plt.ylabel('US Dollars',fontdict={'fontname':'Comic Sans MS',
17                               'fontweight':'bold',
18                               'fontsize':12})
19 plt.xticks(x,rotation=90)
20 plt.grid(True,ls='--')
21
22 #Hiển thị chú thích trong biểu đồ:
23 plt.legend()
24
25 plt.show()
```





Multiple lines chart

Phương thức legend(): Hiển thị chú thích của biểu đồ Bao gồm các tham số chính:

1.loc: Xác định vị trí hiển thị của chú thích trong biểu đồ, gồm các tùy chọn sau:

- 'best' | 0
- 'upper right' | 1
- 'upper left' | 2
- 'lower left' | 3
- 'lower right' | 4
- 'right' | 5
- 'center left' | 6
- 'center right' | 7
- 'lower center' | 8
- 'upper center' | 9
- 'center' | 10

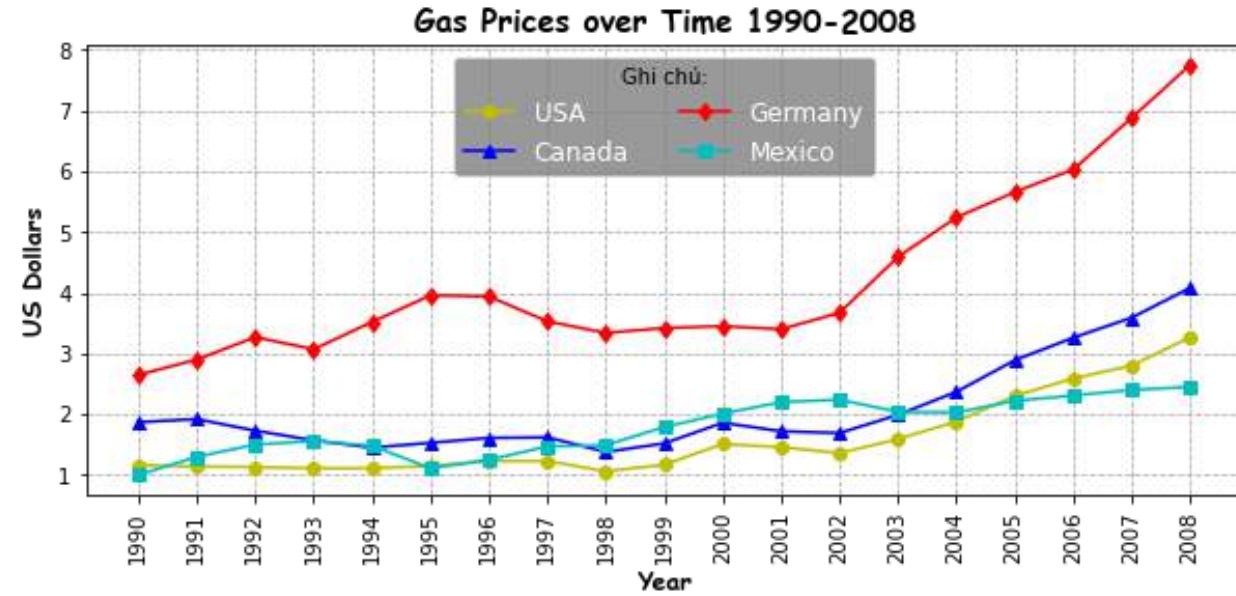
2.ncol: Số cột của chú thích (số nguyên, mặc định là 1)

3.fontsize: kích thước font chữ trong chú thích

4.labelcolor: Màu chữ trong chú thích (mặc định màu đen)

5.facecolor: Màu nền của ô chú thích (mặc định None)

6.title: Dòng tiêu đề trong chú thích



Multiple lines chart

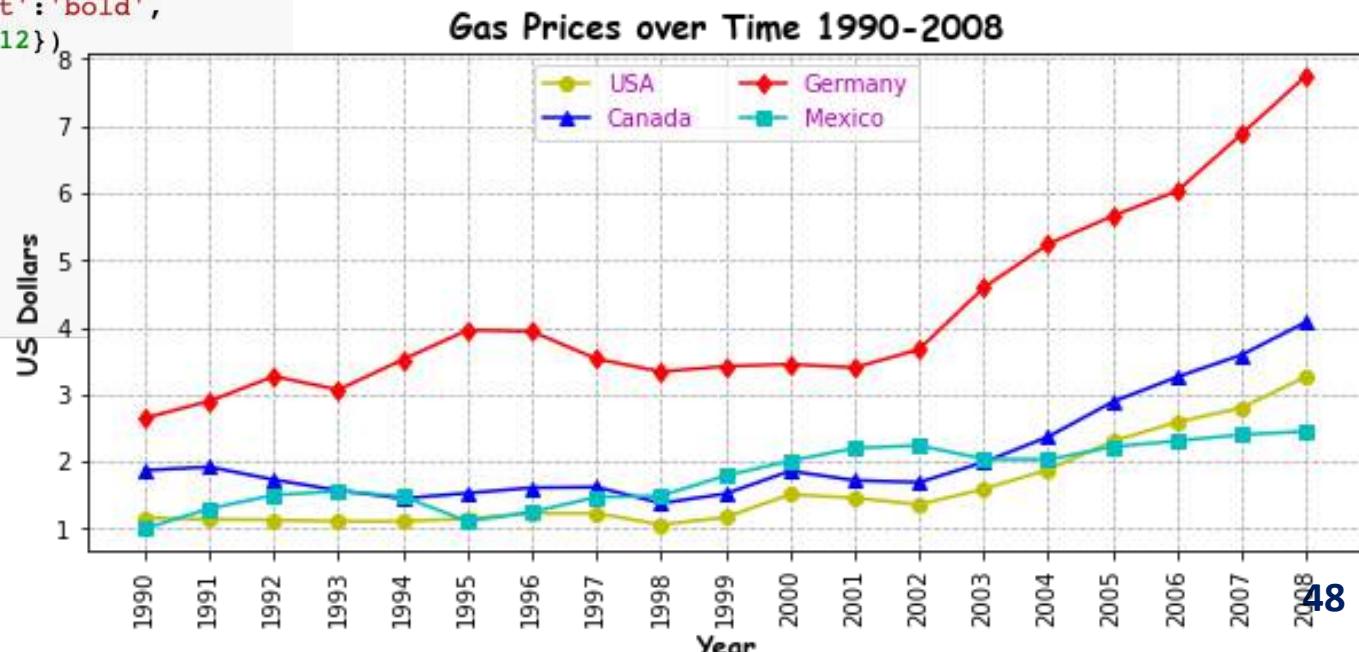
```
1 plt.figure(figsize=(10,4))
2
3 #Vẽ multiple line:
4 plt.plot(x,y1,'y-o', label='USA')
5 plt.plot(x,y2,'b-^', label='Canada')
6 plt.plot(x,y3,'r-d', label='Germany')
7 plt.plot(x,y4,'c-s', label='Mexico')
8
9 plt.title('Gas Prices over Time 1990-2008',fontdict={'fontname':'Comic Sans MS',
10                                'fontweight':'bold',
11                                'fontsize':15})
12 plt.xlabel('Year',fontdict={'fontname':'Comic Sans MS',
13                                'fontweight':'bold',
14                                'fontsize':12})
15 plt.ylabel('US Dollars',fontdict={'fontname':'Comic Sans MS',
16                                'fontweight':'bold',
17                                'fontsize':12})
18 plt.xticks(x,rotation=90)
19 plt.grid(True,ls='--')
20 plt.legend(loc = 9, ncol=2,labelcolor='m')
21 #Lưu đồ thị:
22 plt.savefig('Save_charts/Gas',dpi=300, format='png')
23 plt.savefig('Save_charts/Gas',dpi=500,format='pdf')
24
25 plt.show()
```

Lưu biểu đồ:

plt.savefig(fname, dpi, format)

Trong đó:

1. fname: đường dẫn lưu file
2. dpi: độ phân giải của đồ thị khi lưu (số pixel điểm ảnh trên mỗi inch)
3. format: định dạng file ('png', 'pdf',...)





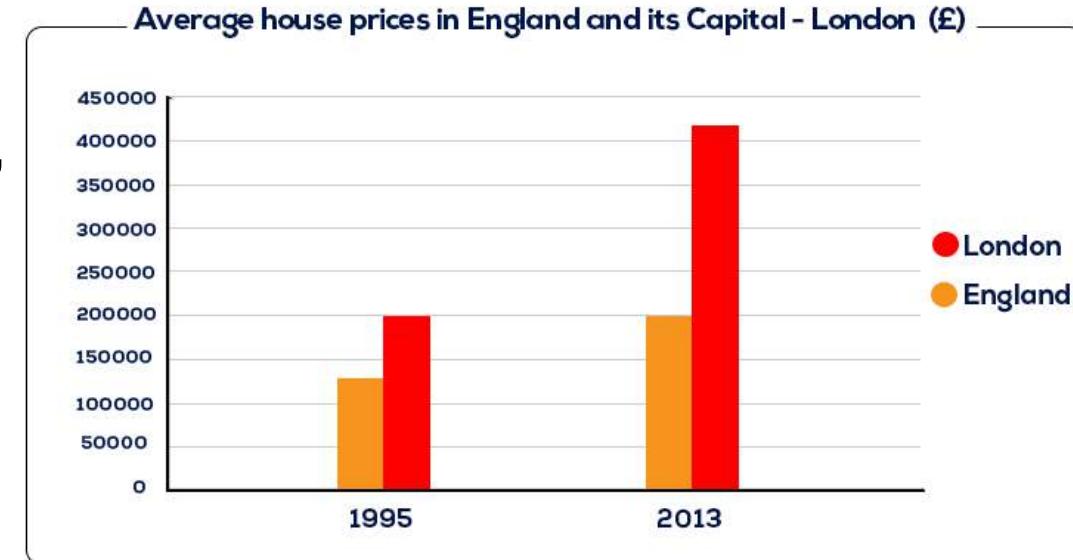
5. Biểu đồ thanh/cột (bar chart)

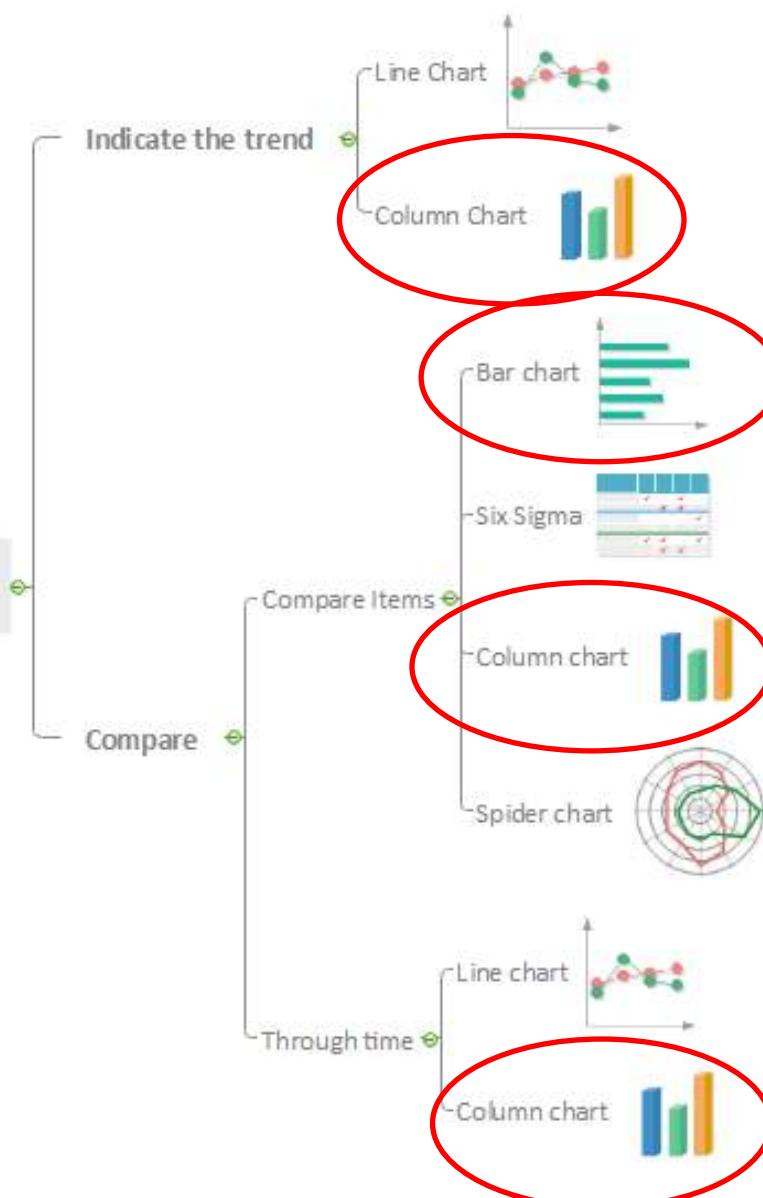
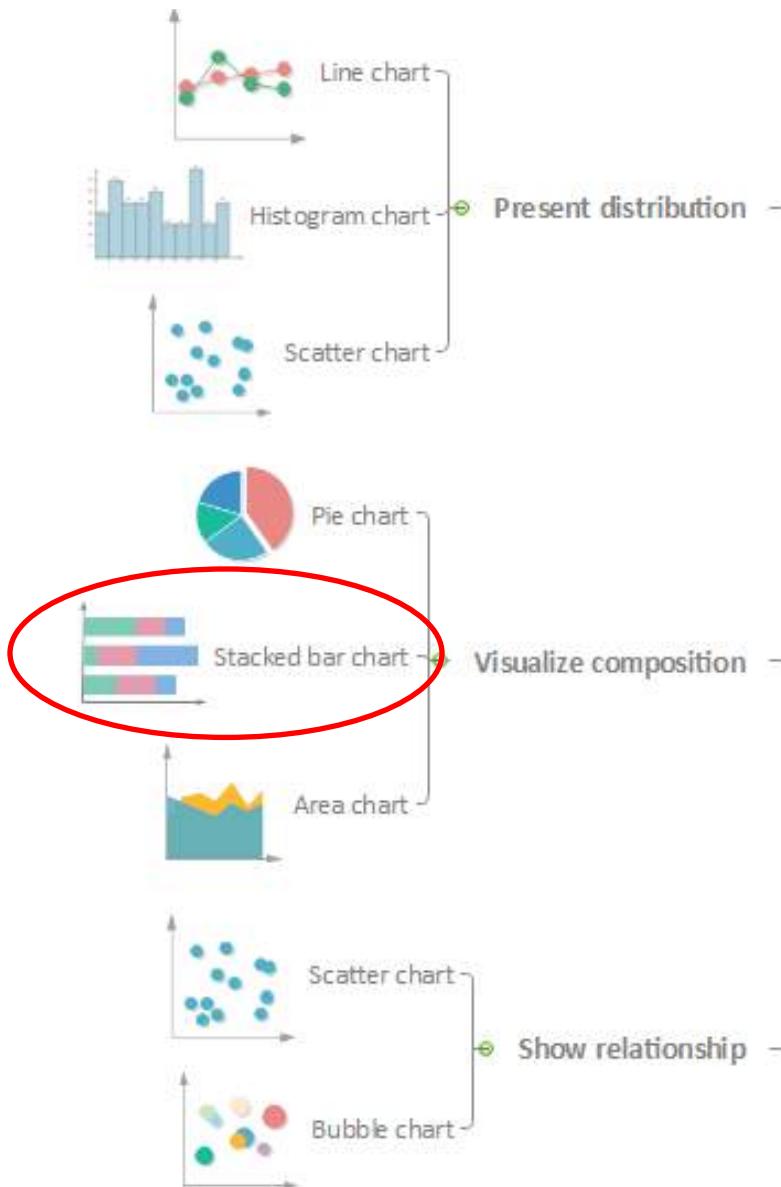
Biểu đồ Bar/Column chart

Bar chart cũng như line chart đây là một trong những dạng đồ thị phổ biến và hay được sử dụng trong thực tế.

Bar chart là một cách cụ thể để biểu diễn dữ liệu bằng cách sử dụng các thanh hình chữ nhật, trong đó chiều dài của mỗi thanh tỷ lệ với giá trị mà chúng đại diện. Nó là một cách biểu diễn đồ họa của dữ liệu bằng cách sử dụng các thanh có độ cao khác nhau.

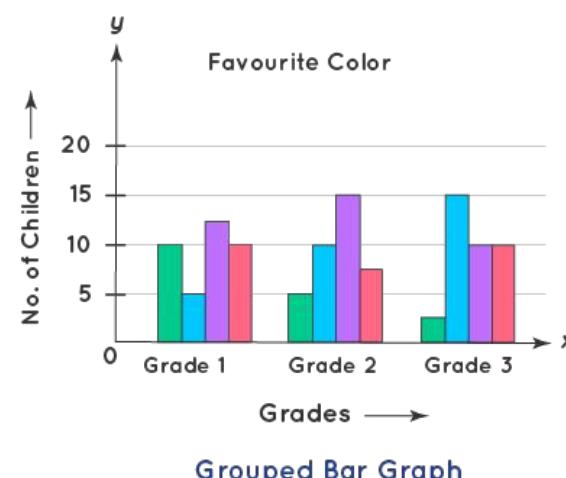
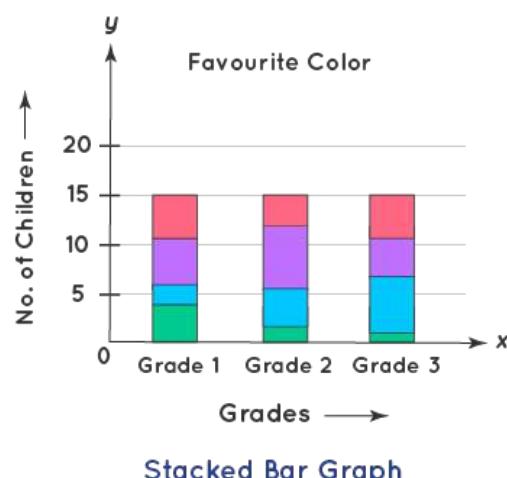
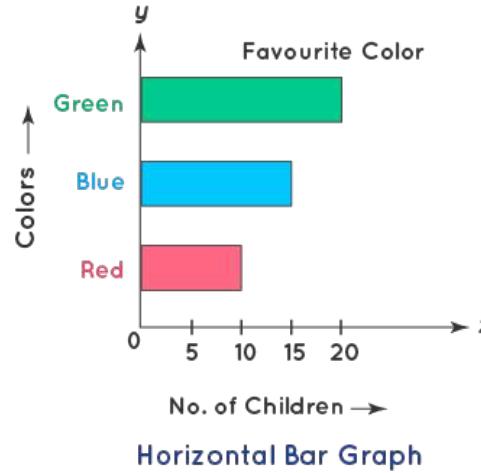
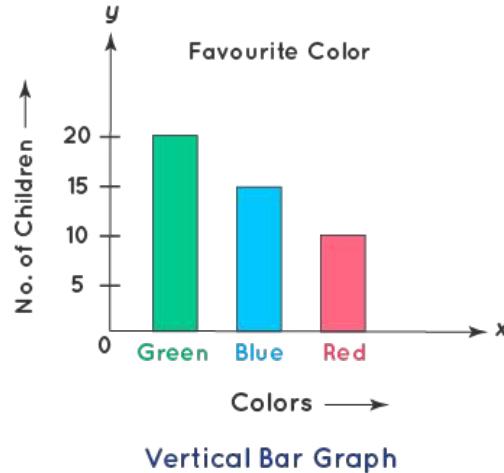
Bar chart là công cụ tuyệt vời để biểu diễn dữ liệu độc lập với nhau mà không cần theo bất kỳ thứ tự cụ thể nào khi biểu diễn.





Biểu đồ Bar/Column chart

Types of Bar Graph



Đặc điểm của Bar chart:

- Tất cả các thanh chữ nhật phải có chiều rộng bằng nhau và phải có khoảng trống giữa chúng bằng nhau.
- Các thanh hình chữ nhật có thể vẽ theo chiều ngang hoặc chiều dọc.
- Chiều cao của hình chữ nhật tương đương với giá trị của dữ liệu mà chúng đại diện.
- Các thanh hình chữ nhật phải nằm trên cùng một trực cơ sở.

Biểu đồ Bar chart với Matplotlib



VINBIGDATA VINGROUP

Academy
Vietnam

Tập dữ liệu **gas_prices.csv**: Lưu trữ giá Gas của 10 nước trên thế giới trong
giai đoạn từ năm 1990 - 2008



```
1 data = pd.read_csv('Data_Visualize/gas_prices.csv')  
2 data
```

	Year	Australia	Canada	France	Germany	Italy	Japan	Mexico	South Korea	UK	USA
0	1990	NaN	1.87	3.63	2.65	4.59	3.16	1.00		2.05	2.82
1	1991	1.96	1.92	3.45	2.90	4.50	3.46	1.30		2.49	3.01
2	1992	1.89	1.73	3.56	3.27	4.53	3.58	1.50		2.65	3.06
3	1993	1.73	1.57	3.41	3.07	3.68	4.16	1.56		2.88	2.84
4	1994	1.84	1.45	3.59	3.52	3.70	4.36	1.48		2.87	2.99
5	1995	1.95	1.53	4.26	3.96	4.00	4.43	1.11		2.94	3.21
6	1996	2.12	1.61	4.41	3.94	4.39	3.64	1.25		3.18	3.34
7	1997	2.05	1.62	4.00	3.53	4.07	3.26	1.47		3.34	3.83
8	1998	1.63	1.38	3.87	3.34	3.84	2.82	1.49		3.04	4.06
9	1999	1.72	1.52	3.85	3.42	3.87	3.27	1.79		3.80	4.29
10	2000	1.94	1.86	3.80	3.45	3.77	3.65	2.01		4.18	4.58
											1.51

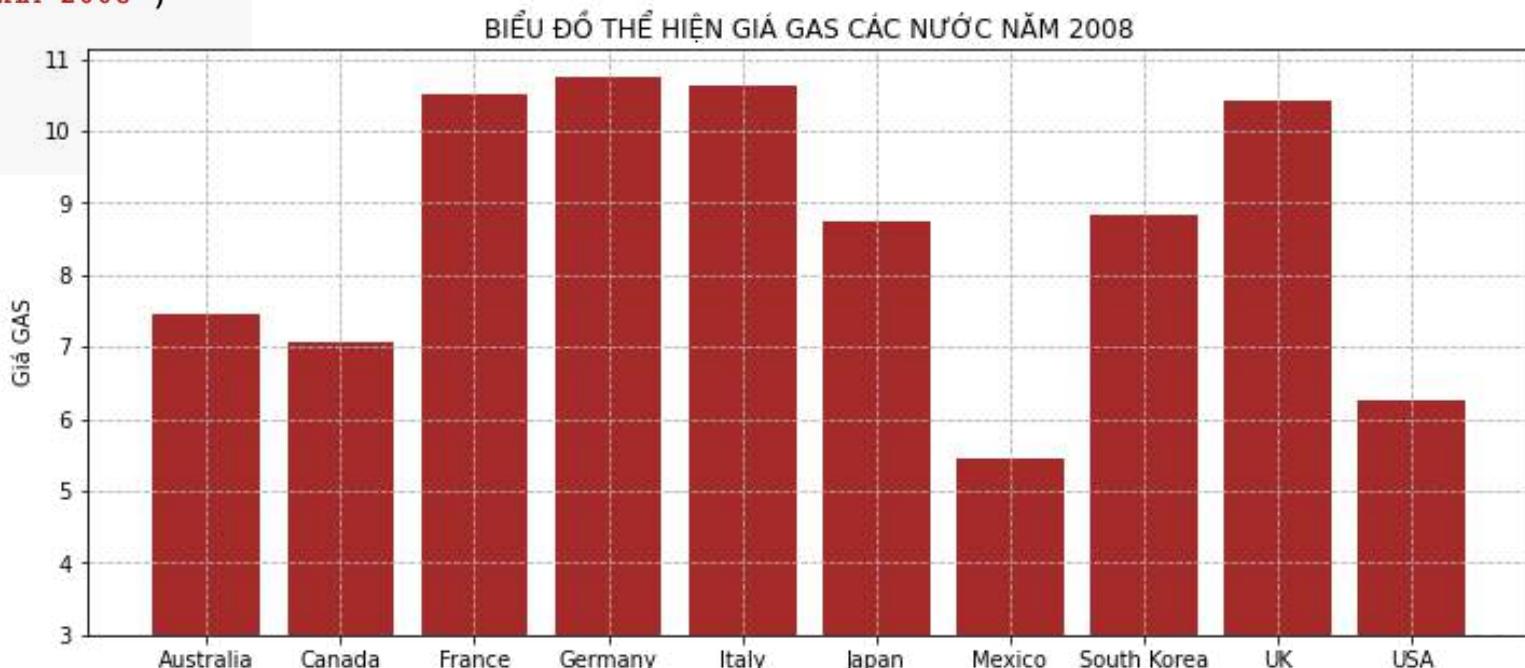


a. Vertical Bar chart

Biểu đồ cột dạng thẳng đứng

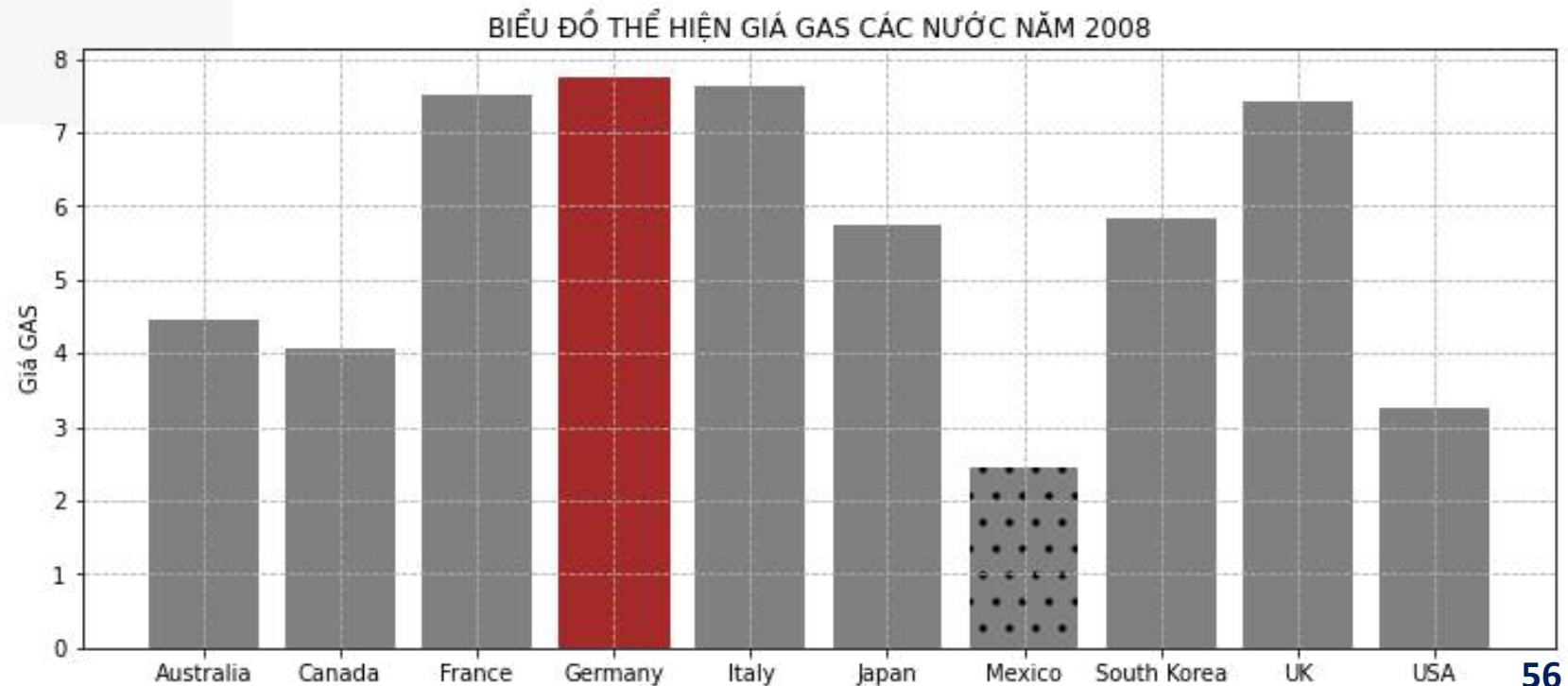
Cú pháp: plt.bar (labels, y)

```
1 plt.figure(figsize = (12,5)) #Thiết lập kích thước biểu đồ
2
3 #Vẽ biểu đồ cột:
4 plt.bar(labels,           #Nhãn của trục X
5         y_2008,          #Giá trị tương ứng với nhãn
6         color='brown',   #Màu của thanh
7         bottom=3,        #Giá trị bắt đầu của trục Y
8         width = 0.8)     #Chiều rộng của thanh
9
10 #Tiêu đề của đồ thị
11 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CÁC NƯỚC NĂM 2008')
12 plt.ylabel('Giá GAS')
13 plt.grid(ls='--')
14
15 plt.show()
```



Biểu đồ cột dạng thẳng đứng

```
1 #Làm nổi bật một thanh:  
2 plt.figure(figsize = (12,5))  
3  
4 bar = plt.bar(labels,y_2008,color='gray')  
5  
6 #Thay đổi màu sắc khác, tạo hatch cho thanh  
7 bar[3].set_color('brown')  
8 bar[6].set_hatch('.')  
9  
10 #Tiêu đề của đồ thị  
11 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CÁC NƯỚC NĂM 2008')  
12 plt.ylabel('Giá GAS')  
13 plt.grid(ls='--')  
14  
15 plt.show()
```





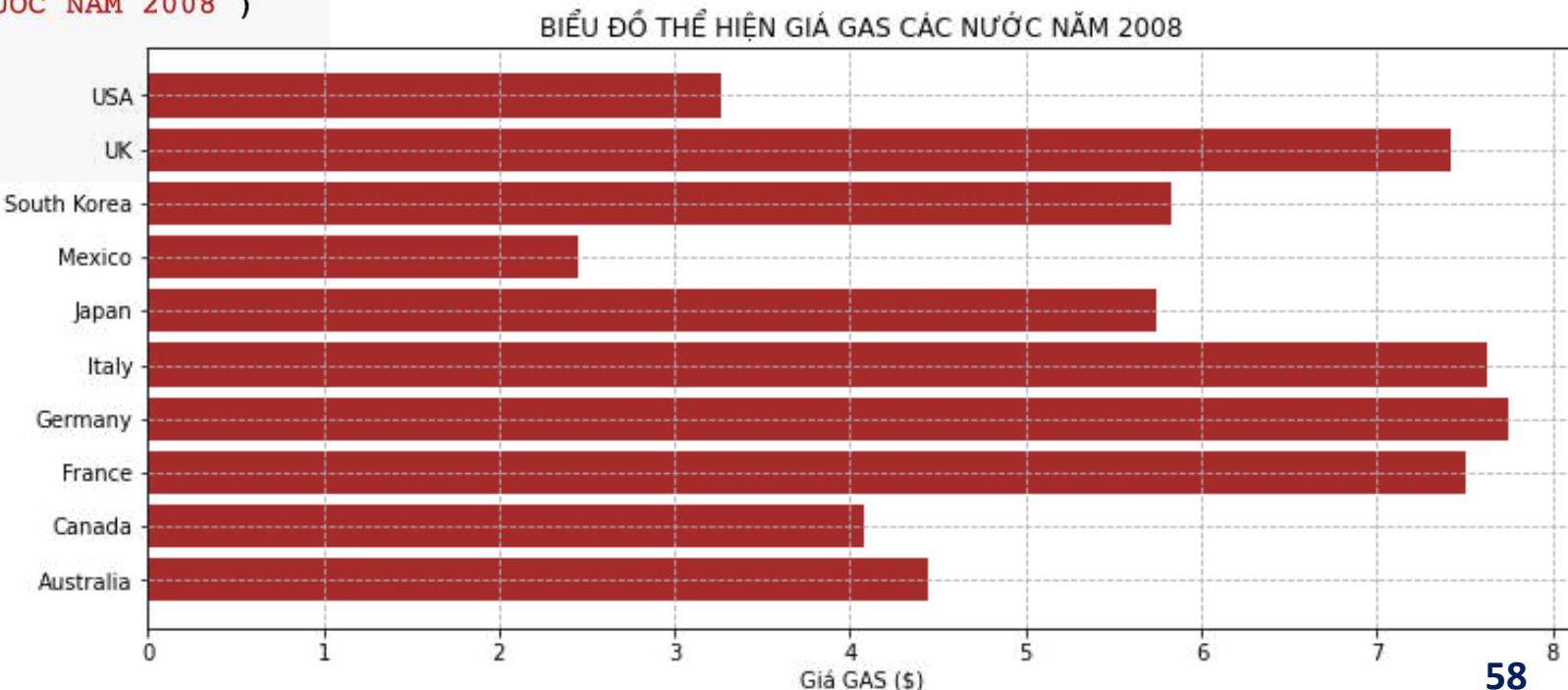
b. Horizontal Bar chart



Biểu đồ cột dạng nằm ngang

Cú pháp: plt.barh (labels, y)

```
1 plt.figure(figsize = (12,5)) #Thiết lập kích thước biểu đồ
2
3 plt.barh(labels,
4          y_2008,
5          color='brown',
6          height = 0.8) #Chiều rộng của thanh
7
8 #Tiêu đề của đồ thị
9 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CÁC NƯỚC NĂM 2008')
10 plt.xlabel('Giá GAS ($)')
11 plt.grid(ls='--')
12
13 plt.show()
```

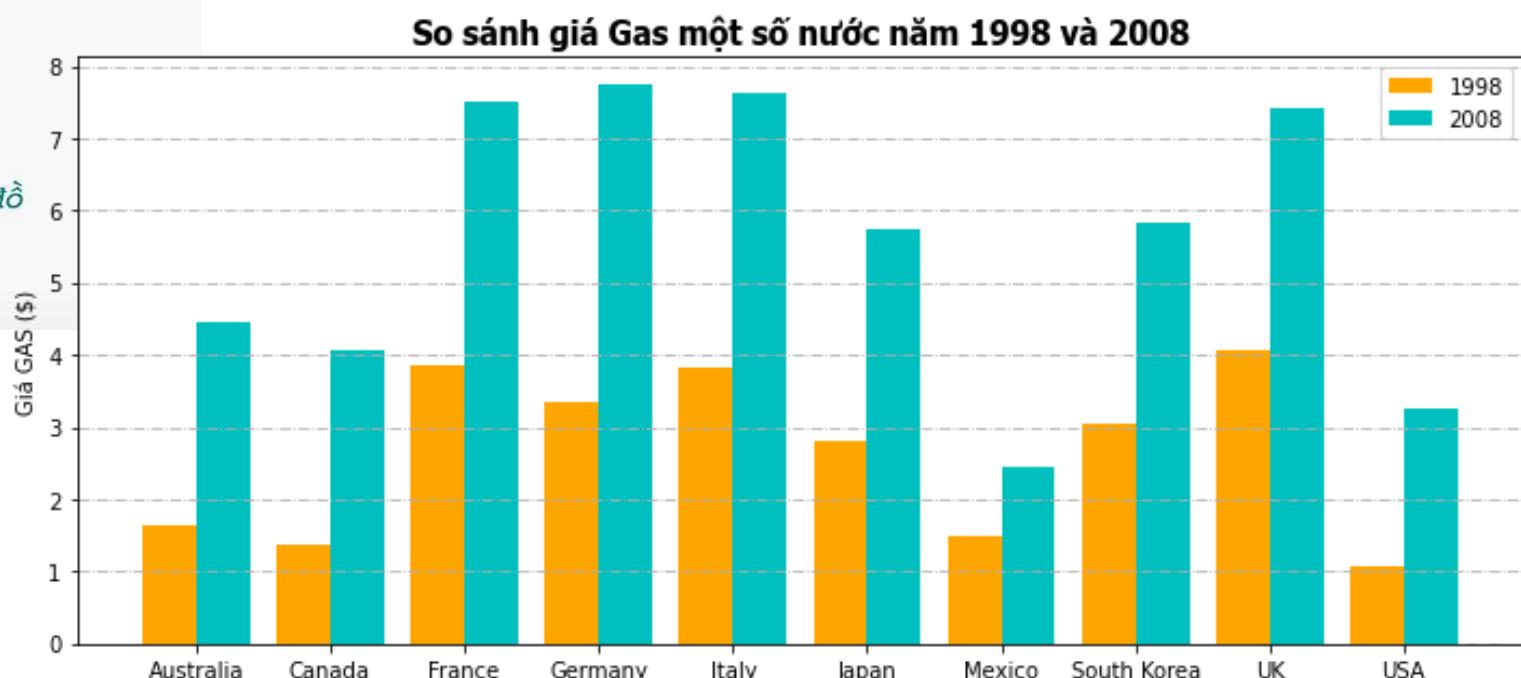




c. Grouped Bar chart

Nhiều cột trên biểu đồ

```
1 #Multiple bar:  
2 w=0.4 #Thiết lập độ rộng của thanh (Tổng < 1.0)  
3 bar1 = np.arange(len(labels))  
4 bar2 = [i+w for i in bar1]  
5  
6 plt.figure(figsize = (12,5))  
7 #Vẽ các biểu đồ cột cho từng bộ dữ liệu:  
8 plt.bar(bar1,y_1998,width=w,color='orange',label='1998')  
9 plt.bar(bar2,y_2008,width=w,color='c',label='2008')  
10  
11 plt.title('So sánh giá Gas một số nước năm 1998 và 2008',  
12     fontdict={'fontname':'Tahoma',  
13     'fontweight':'bold',  
14     'fontsize':15})  
15 plt.ylabel("Giá GAS ($)")  
16 plt.grid(axis='y',ls='-.')  
17 plt.legend()  
18  
19 #Hiển thị nhãn của trục x, căn vào giữa 2 biểu đồ  
20 plt.xticks(bar1+w/2,labels)  
21  
22 plt.show()
```



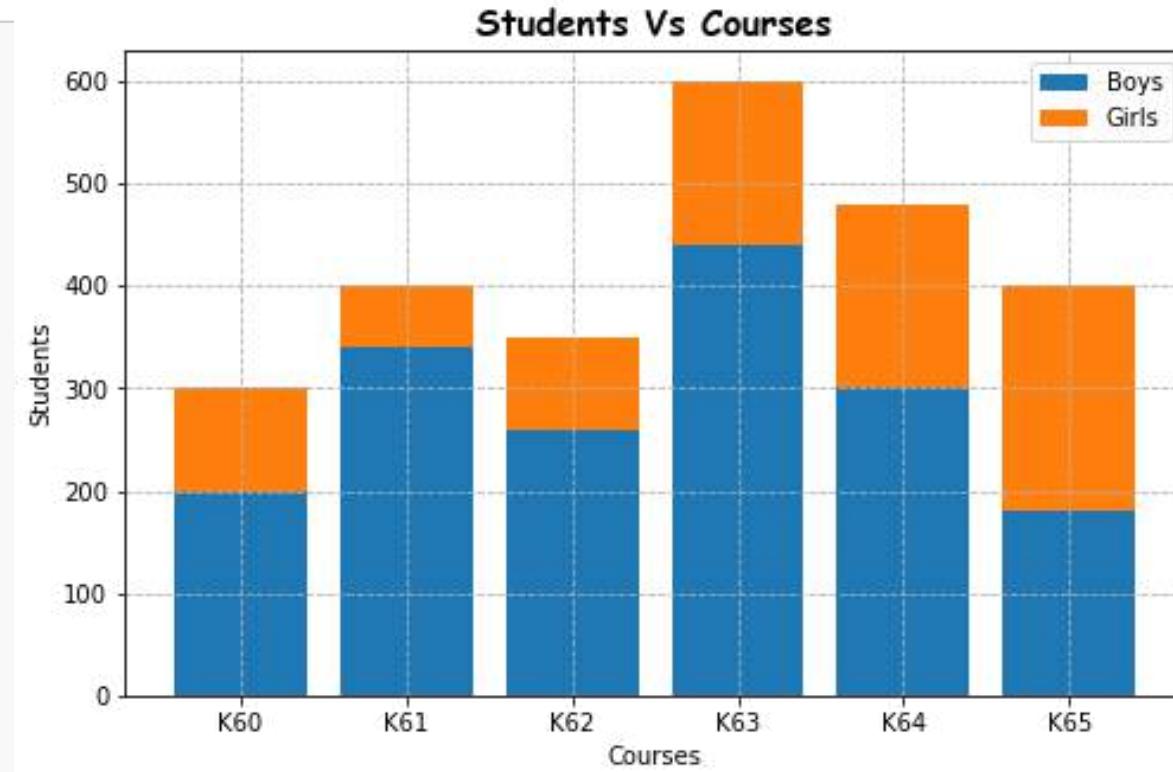


d. Stacked Bar chart

Biểu đồ cột xếp tầng

```
1 #Tạo dữ liệu: Số lượng SV Nam – Nữ theo từng khóa.  
2 labels = ['K60', 'K61', 'K62', 'K63', 'K64', 'K65']  
3 boys = [200, 340, 260, 440, 300, 180]  
4 girls = [100, 60, 90, 160, 180, 220]
```

```
1 plt.figure(figsize=(8,5))  
2  
3 #Biểu đồ cột 1 bên dưới cùng:  
4 plt.bar(labels,boys,label='Boys')  
5  
6 #Biểu đồ cột 2 xếp chồng lên biểu đồ 1:  
7 #Sử dụng: thuộc tính bottom chồng biểu đồ:  
8 plt.bar(labels,girls,bottom = boys, label='Girls')  
9  
10 plt.title('Students Vs Courses',  
11             fontdict={'fontname':'Comic Sans MS',  
12                     'fontweight':'bold',  
13                     'fontsize':15})  
14 plt.xlabel('Courses')  
15 plt.ylabel("Students")  
16 plt.grid(ls='--')  
17 plt.legend()  
18  
19 plt.show()
```



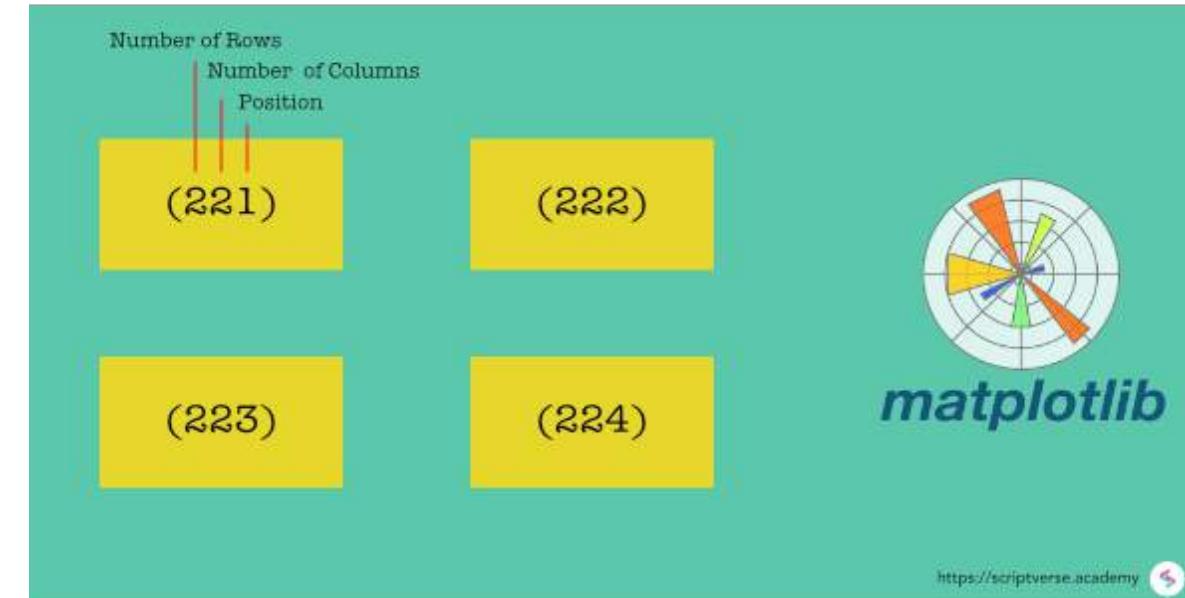
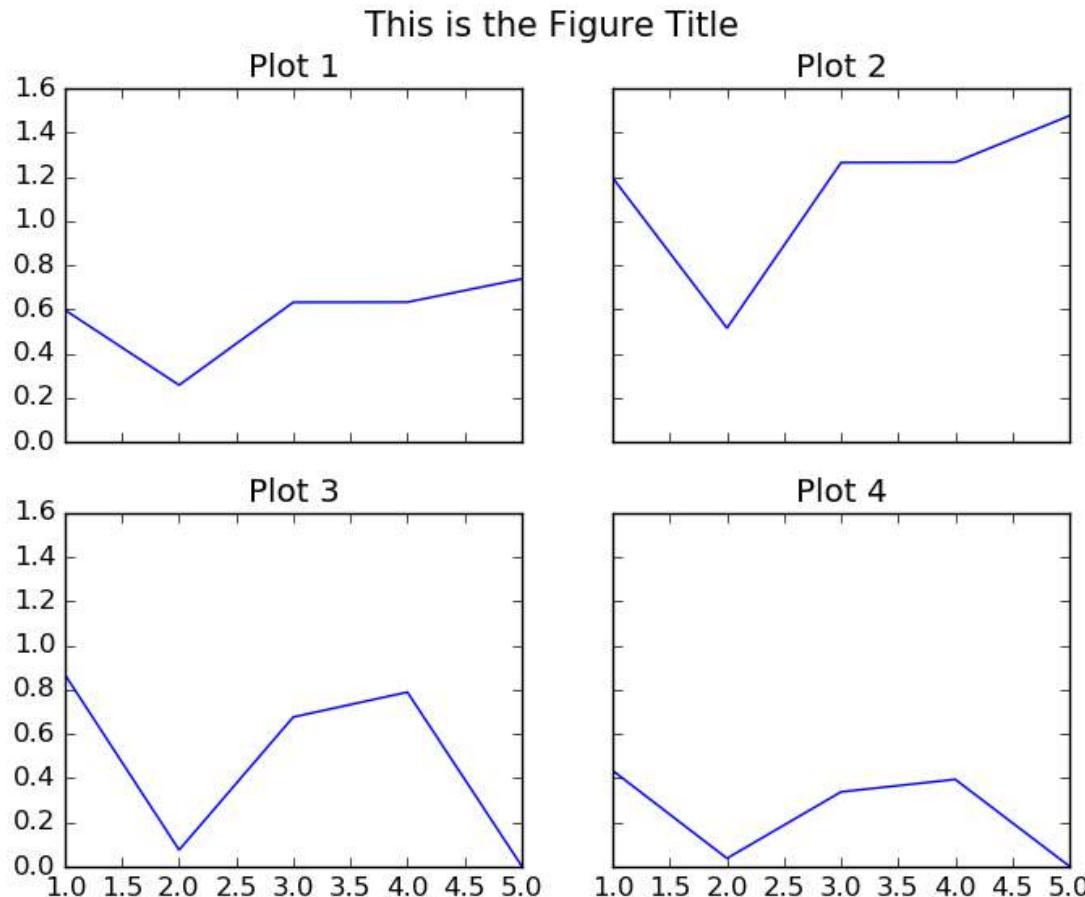


d. Hiển thị nhiều khung biểu đồ



Multiple plot

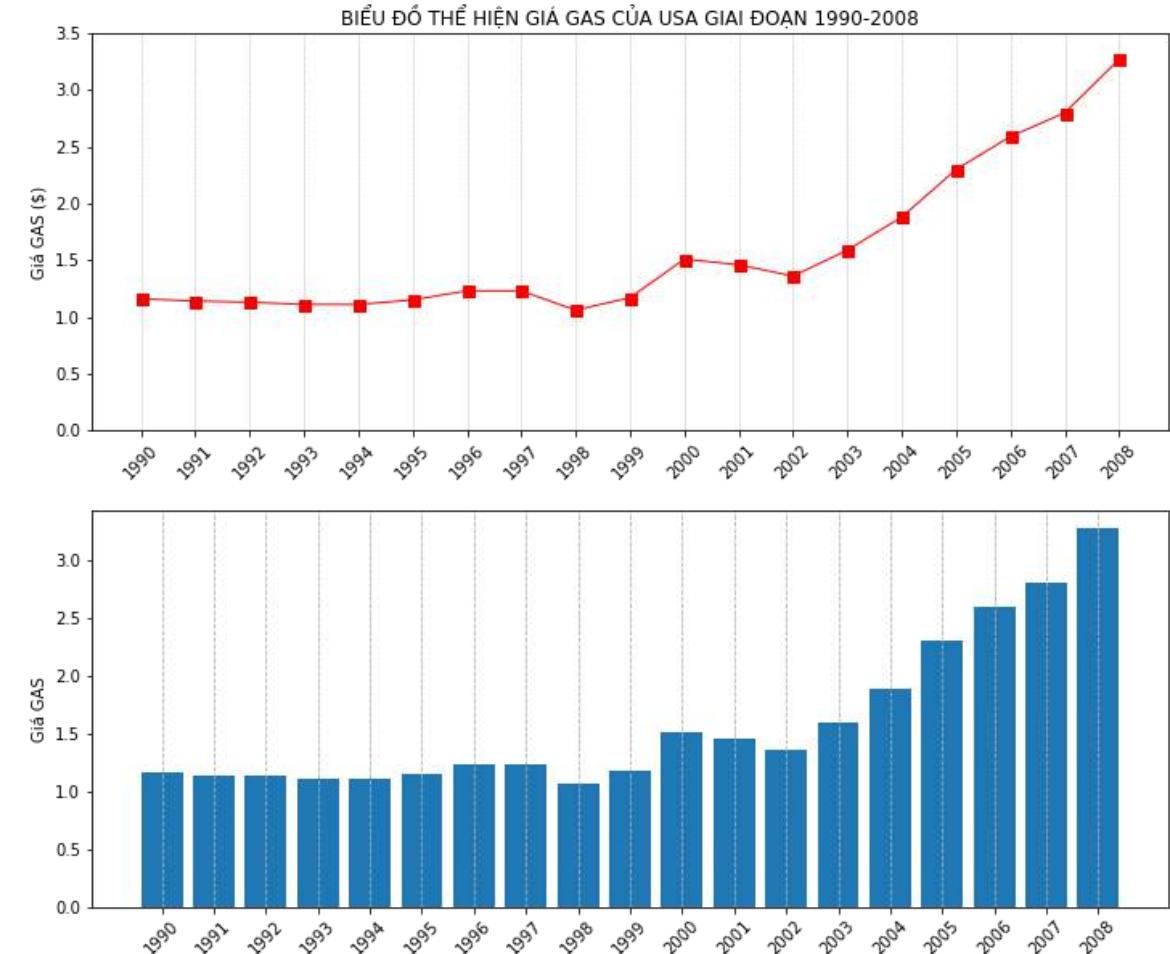
Cú pháp: plt.subplot (nrows, ncols,Position)



Multiple plot

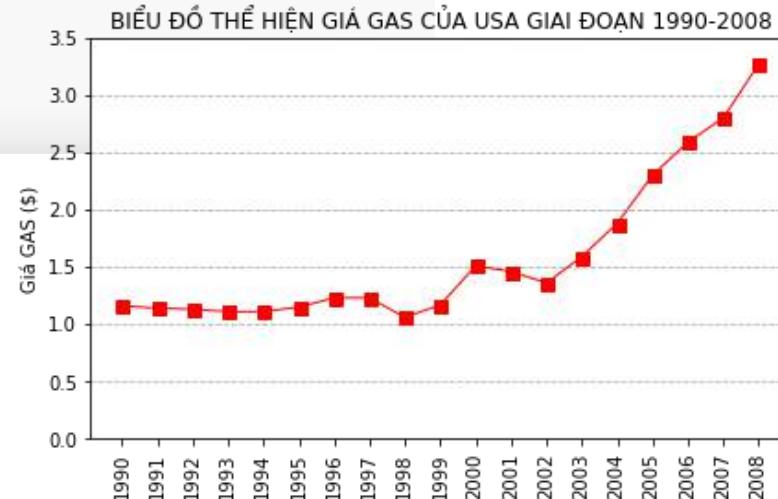
plt.subplot (2, 1, Position)

```
1 plt.figure(figsize = (12,10)) #Thiết lập kích thước biểu đồ
2 #Vẽ biểu đồ đường trên khung 1:
3 plt.subplot(2,1,1) #Thiết lập Khung biểu đồ gồm 2 hàng 1 cột
4
5 plt.plot(x, y, 'r-s', lw=1.0, ms=7) #Vẽ biểu đồ đường plot 1
6
7 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CỦA USA GIAI ĐOẠN 1990-2008')
8 plt.ylabel('Giá GAS ($)')
9 plt.ylim(0,3.5)
10 plt.xticks(x,rotation=45)
11 plt.grid(axis='x',ls='--')
12 #
13 #Vẽ biểu đồ Bar trên khung 2:
14 plt.subplot(2,1,2)
15
16 plt.bar(x,y) #Vẽ biểu đồ cột plot 2
17
18 plt.ylabel('Giá GAS')
19 plt.xticks(x,rotation=45)
20 plt.grid(axis='x', ls='--')
21
22 plt.show()
```

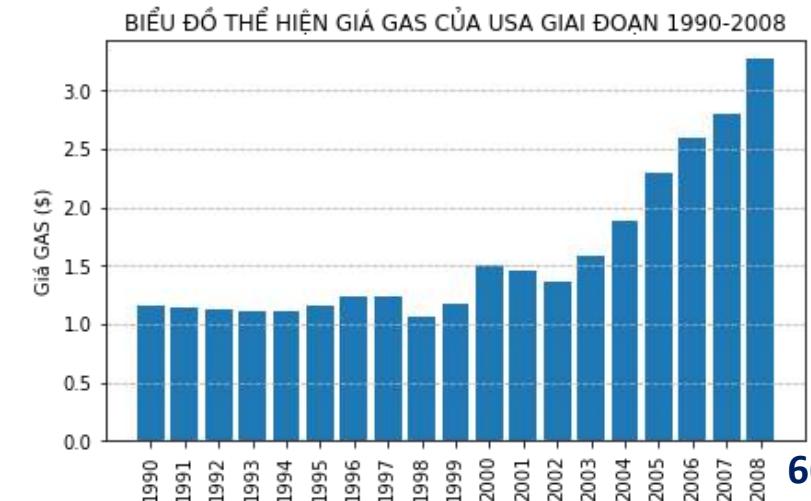


Multiple plot

```
1 plt.figure(figsize = (15,4))
2 #Vẽ biểu đồ đường trên khung 1:
3 plt.subplot(1,2,1) #Thiết lập Khung biểu đồ gồm 1 hàng 2 cột
4
5 plt.plot(x, y, 'r-s', lw=1.0, ms = 7) #Vẽ biểu đồ đường plot 1
6
7 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CỦA USA GIAI ĐOẠN 1990-2008')
8 plt.ylabel('Giá GAS ($)')
9 plt.ylim(0,3.5)
10 plt.xticks(x,rotation=90)
11 plt.grid(axis='y',ls='--')
12 #
13 #Vẽ biểu đồ Bar trên khung 2:
14 plt.subplot(1,2,2)
15
16 plt.bar(x,y) #Vẽ biểu đồ cột plot 2
17
18 plt.title('BIỂU ĐỒ THỂ HIỆN GIÁ GAS CỦA USA GIAI ĐOẠN 1990-2008')
19 plt.ylabel('Giá GAS ($)')
20 plt.xticks(x,rotation=90)
21 plt.grid(axis='y', ls='--')
22
23 plt.show()
```



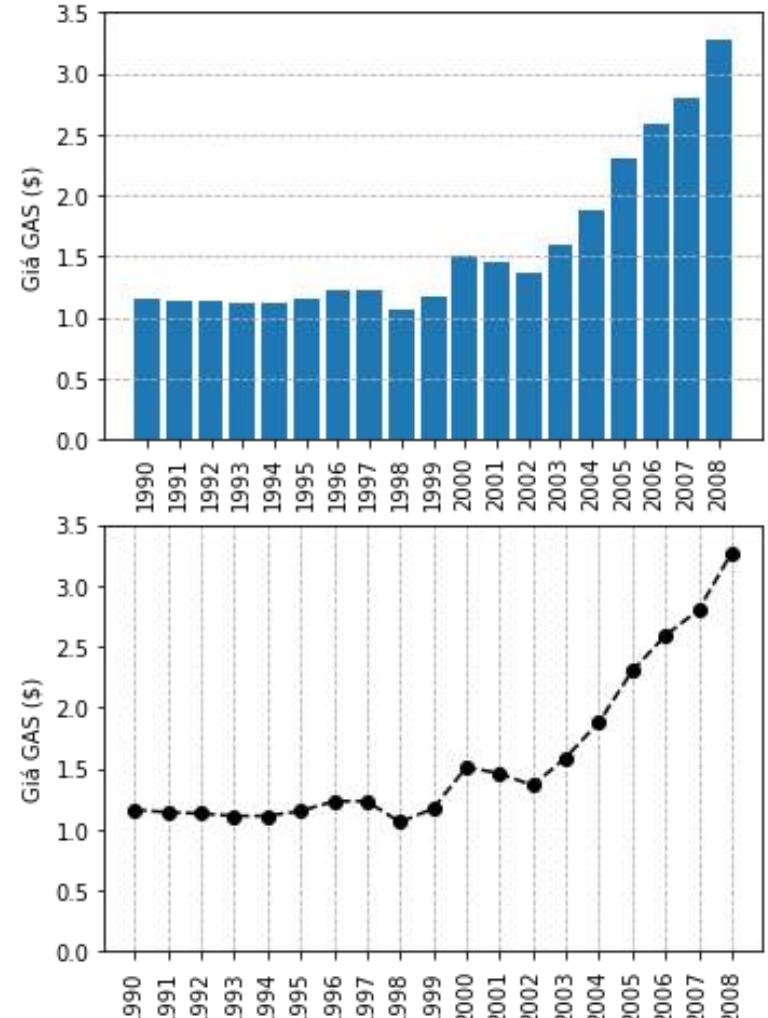
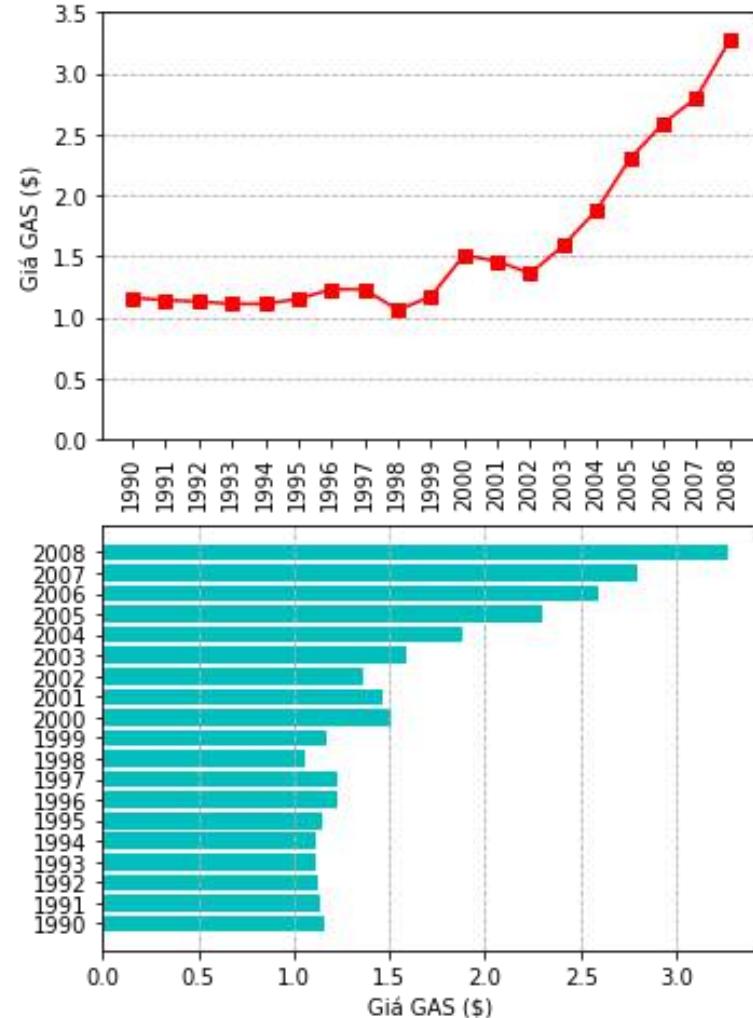
plt.subplot (1, 2, Position)



Multiple plot

```
1 plt.figure(figsize = (12,8))
2
3 #Thiết lập khung biểu đồ gồm 2 hàng 2 cột
4 #####Vẽ biểu đồ đường trên khung 1#####
5 plt.subplot(2,2,1)
6 #Vẽ biểu đồ đường trên plot 1:
7 plt.plot(x, y, 'r-s')
8
9 plt.ylabel('Giá GAS ($)')
10 plt.ylim(0,3.5)
11 plt.xticks(x,rotation=90)
12 plt.grid(axis='y',ls='--')
13 #####Vẽ biểu đồ đường trên khung 2#####
14 plt.subplot(2,2,2)
15 #Vẽ biểu đồ cột đứng trên plot 2:
16 plt.bar(x, y)
17 plt.ylabel('Giá GAS ($)')
18 plt.ylim(0,3.5)
19 plt.xticks(x,rotation=90)
20 plt.grid(axis='y',ls='--')
21 #####Vẽ biểu đồ đường trên khung 3#####
22 plt.subplot(2,2,3)
23 #Vẽ biểu đồ cột ngang trên plot 3:
24 plt.barch(x,y,color='c')
25 plt.xlabel('Giá GAS ($)')
26 plt.yticks(x)
27 plt.grid(axis='x',ls='--')
28 #####Vẽ biểu đồ đường trên khung 4#####
29 plt.subplot(2,2,4)
30 #Vẽ biểu đồ đường trên plot 4:
31 plt.plot(x, y, 'k--o')
32 plt.ylabel('Giá GAS ($)')
33 plt.ylim(0,3.5)
34 plt.xticks(x,rotation=90)
35 plt.grid(axis='x',ls='--')
36
37 plt.show()
```

plt.subplot (2, 2, Position)



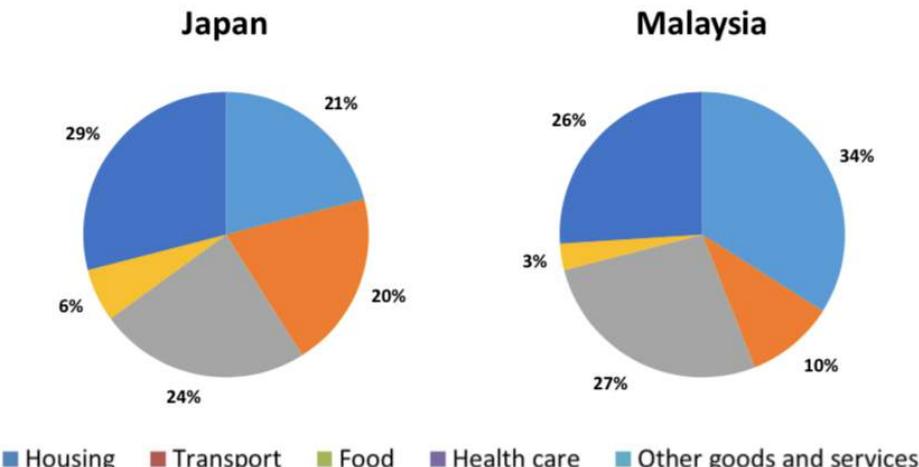
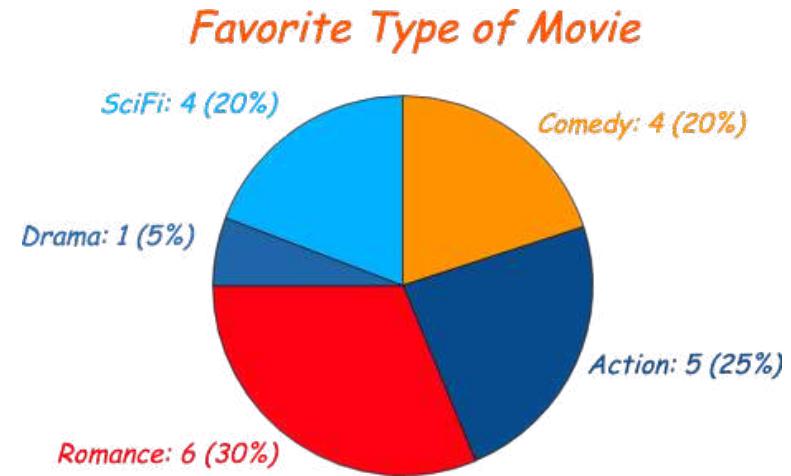


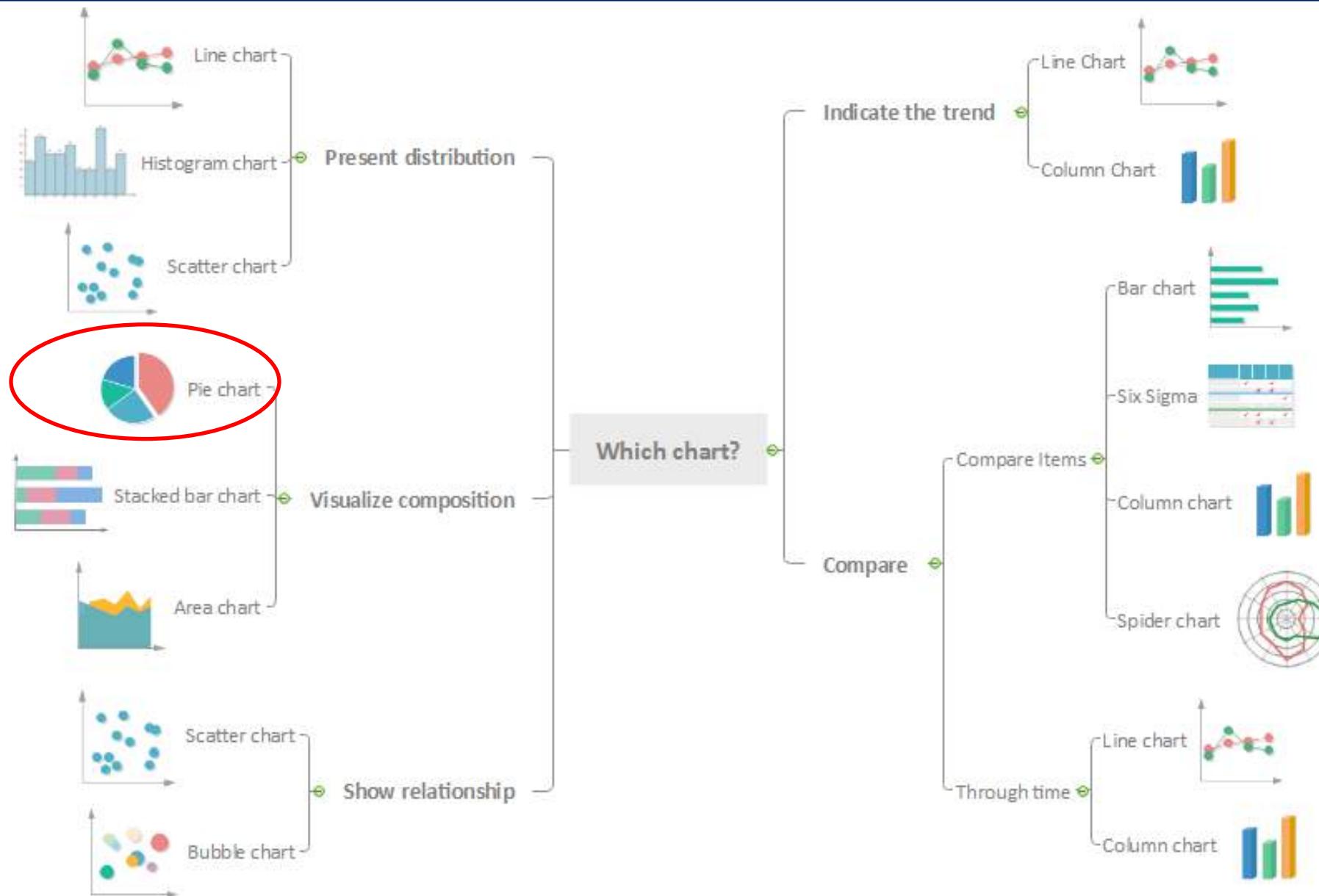
6. Biểu đồ tròn (hình bánh)



Biểu đồ tròn (Pie chart)

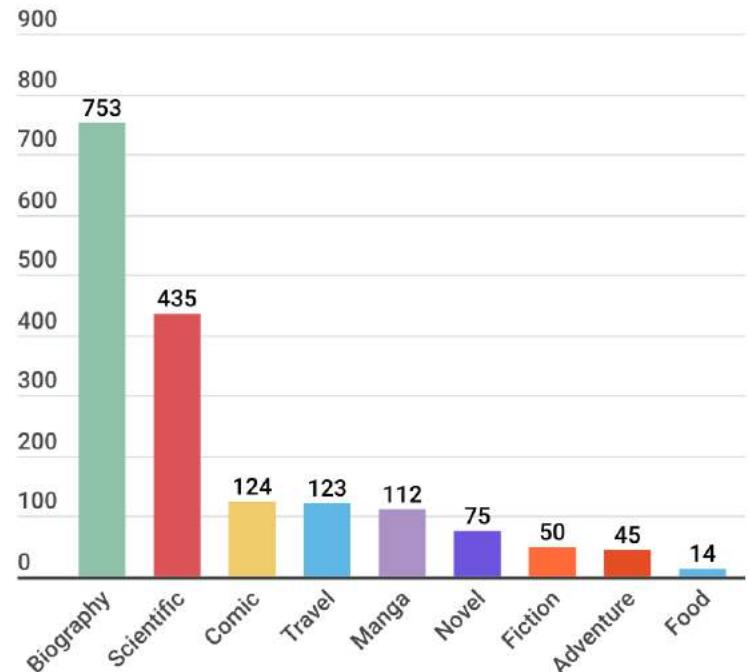
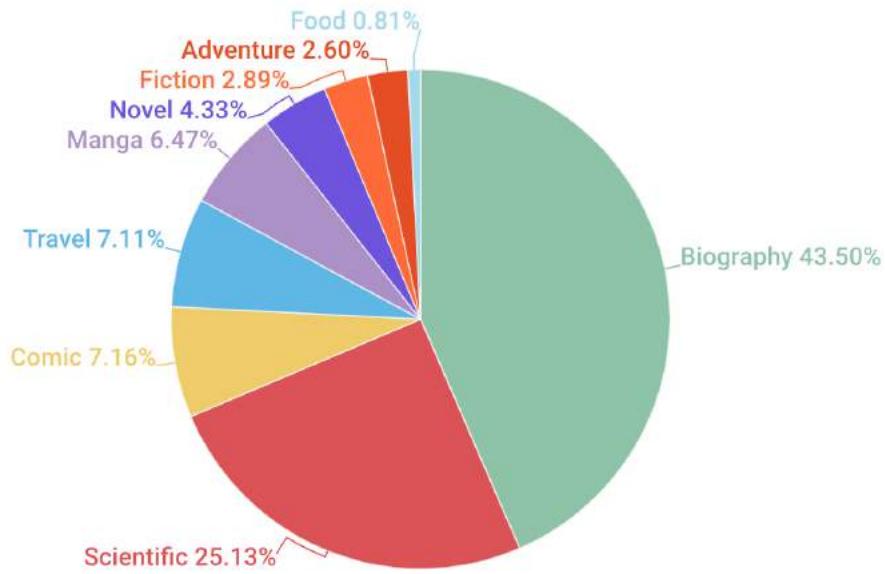
- **Biểu đồ tròn (Pie Chart)** là dạng biểu đồ hình tròn phẳng (cũng có tình huống được trình bày ở dạng 3D) dùng để so sánh giá trị phần trăm trong tổng thể.
- Các giá trị biểu diễn số liệu cho một đối tượng thông qua màu sắc riêng biệt. Đối tượng nào có màu sắc tương ứng đó và được liệt kê ở phần chú thích của biểu đồ. Phần màu càng lớn thì số liệu càng lớn và ngược lại.
- Pie Chart được sử dụng để biểu diễn tỉ lệ phần trăm của các thành phần so với tổng thể. Vì vậy, nó không được dùng để biểu diễn giá trị chính xác.





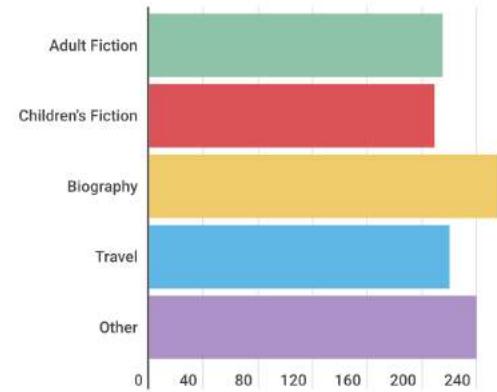
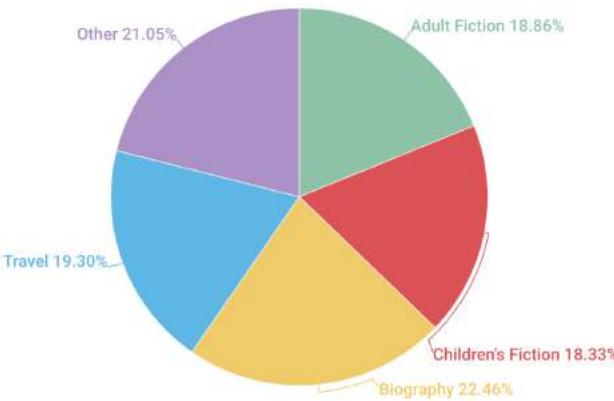
Lưu ý khi sử dụng Pie chart

- Đảm bảo tổng các thành phần là 100%:** Với các công cụ hỗ trợ thì không cần lo lắng về lỗi này vì các công cụ đã đảm bảo được sự chính xác của số liệu khi biểu diễn. Nếu vẽ Pie chart thủ công thì chúng ta cần kiểm tra lại tính đúng đắn một lần nữa.
- Chỉ dùng Pie chart khi số lượng thể loại ít hơn 6:** Việc sử dụng Pie Chart khi có quá nhiều thể loại sẽ khiến cho biểu đồ khá rối. Nếu có quá nhiều thể loại, nên xem xét một biểu đồ khác như Bar Chart.

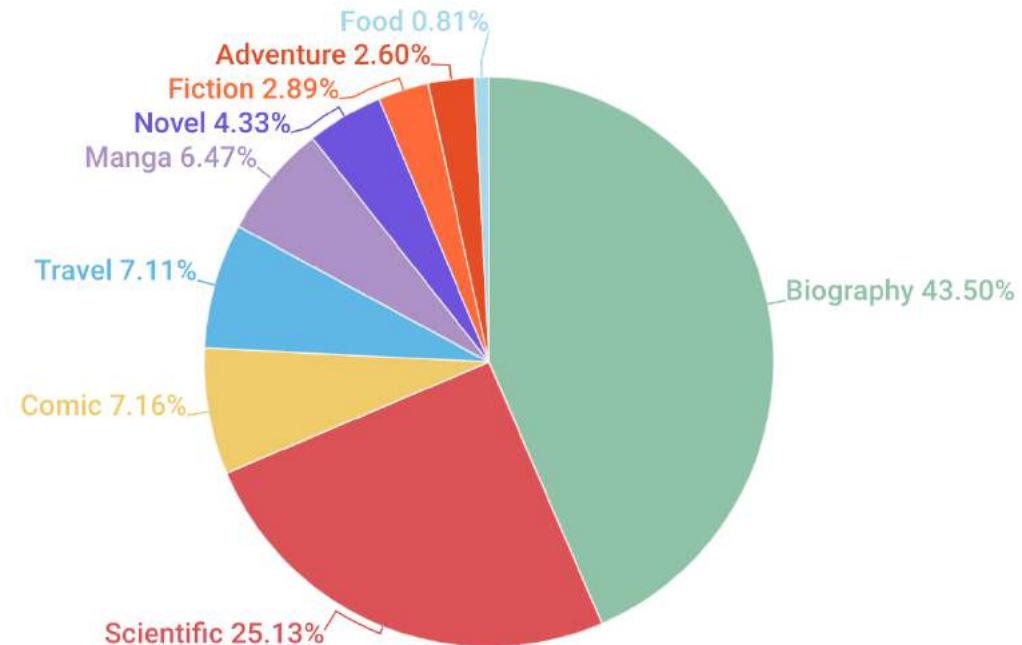


Lưu ý khi sử dụng Pie chart

3. Không dùng Pie Chart nếu tỉ lệ giữa các thể loại gần tương đương nhau: Nếu tỉ lệ giữa các thể loại là tương đương nhau thì dường như Pie Chart lúc này là vô dụng vì không thể hiện cụ thể một ý nghĩa gì. Giải pháp lúc này là xem xét một dạng biểu đồ khác như **Column Chart** hoặc **Bar Chart**.

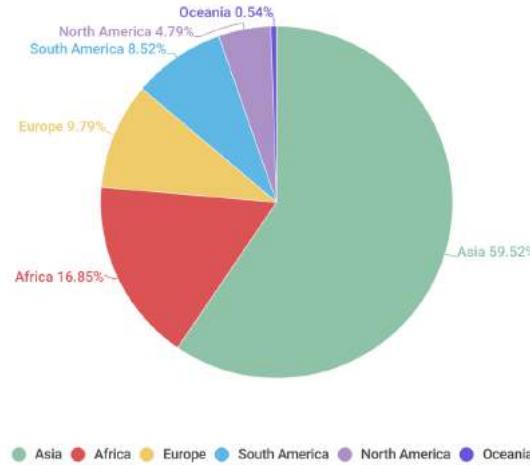


4. Nên sắp xếp giá trị các thể loại để dễ hiểu hơn: Sắp xếp lại dữ liệu giúp cho người xem nhận ra ngay thể loại có tỉ lệ cao nhất. Đồng thời với 2 thể loại gần nhau tương đương thì biết được thể loại nào có giá trị lớn hơn. Thông thường, giá trị trong Pie Chart được sắp xếp từ lớn đến nhỏ theo chiều kim đồng hồ như ví dụ bên dưới.

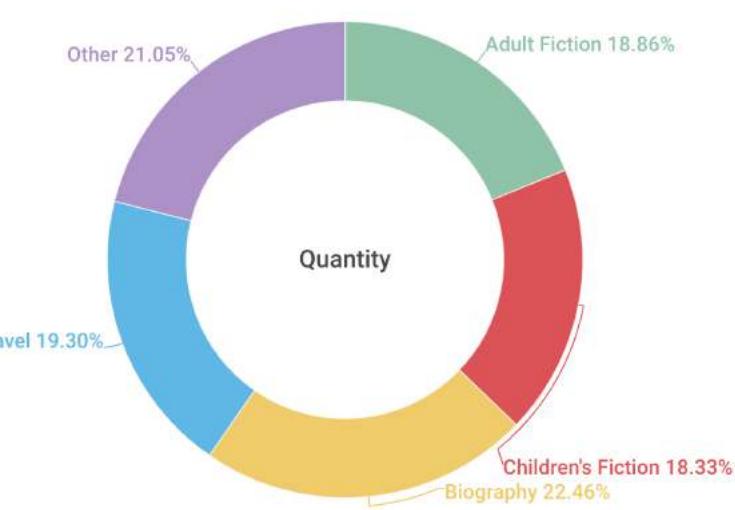


Một số dạng Pie chart

Global population by continent as of mid-2018

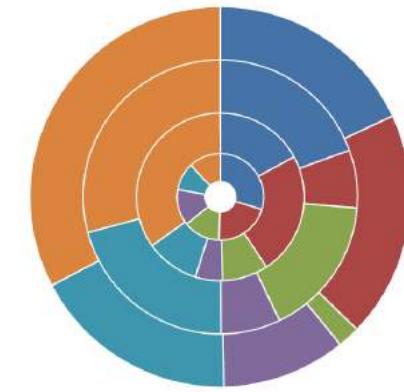


1. Pie chart



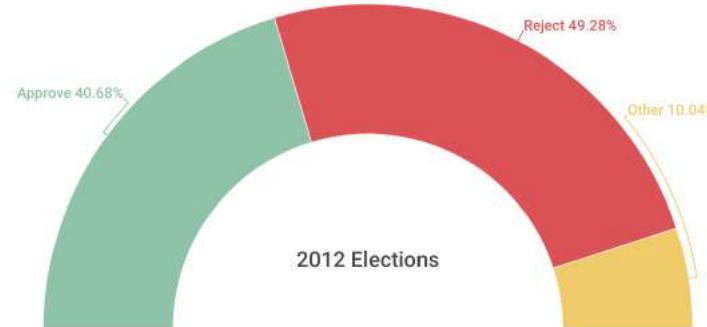
2. Donut chart

Total Revenue
Q1 1997 Q2 1997 Q3 1997 Q4 1997

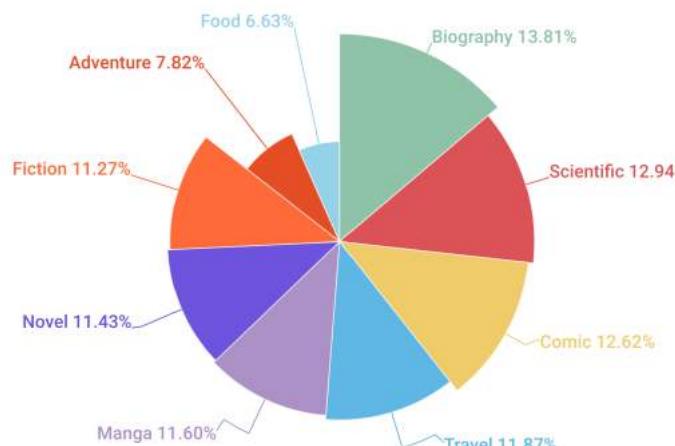


Margaret Peacock Janet Leverling Robert King Laura Callahan
Nancy Davolio All others

3. Stacked Donut chart



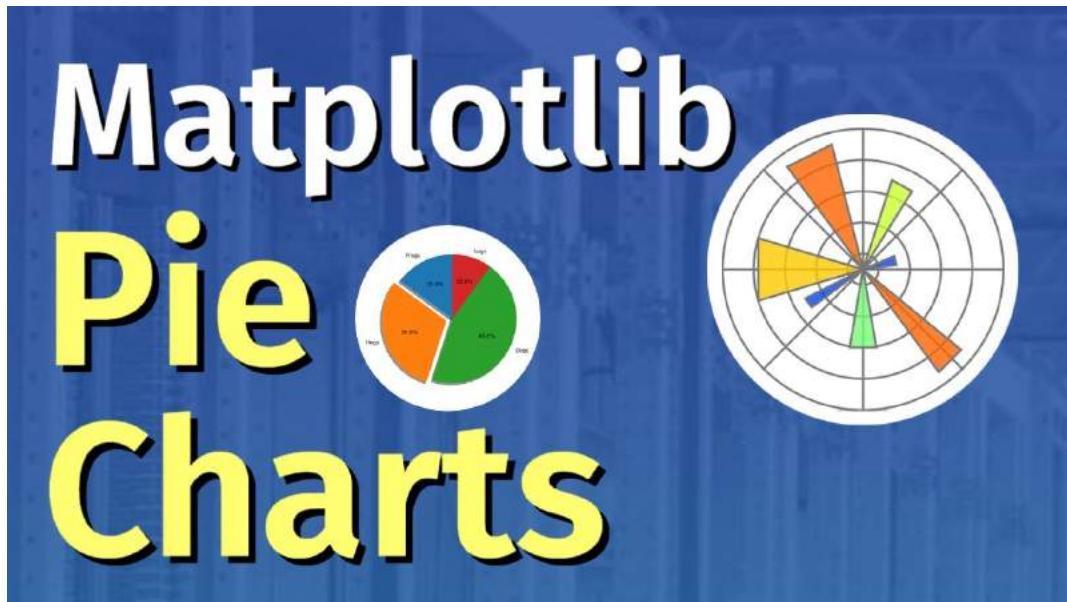
4. Semi-Circle Pie chart



5. Irregular Pie chart

Biểu đồ Bar chart với Matplotlib

Số liệu: Số lượng SV của các Khoa theo Giới tính.



Khoa	K60	K61	K62	K63	K64	K65
Nam	200	340	260	440	300	180
Nữ	30	60	90	160	180	220

```
1 #Tạo dữ liệu: Số lượng SV Nam - Nữ theo từng khoá.  
2 labels = ['K60','K61','K62','K63','K64','K65']  
3 boys = [200, 340, 260, 440, 300, 180]  
4 girls = [30, 60, 90, 160, 180, 220]  
5 total = list(np.array(boys) + np.array(girls))  
6  
7 sex =[ 'Nam','Nữ']  
8 sum_boy = sum(boys)  
9 sum_girl = sum(girls)
```

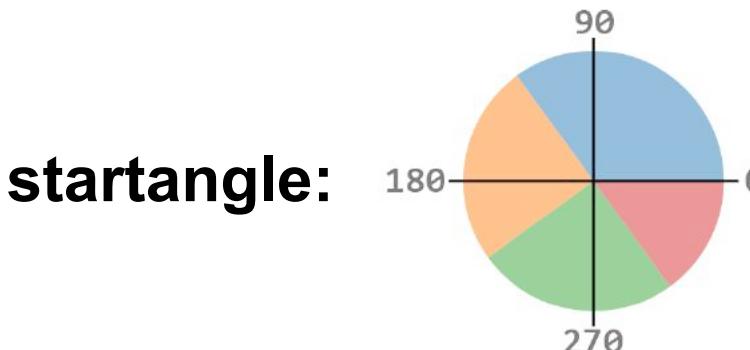


a. Pie chart

Biểu đồ tròn (Pie chart)

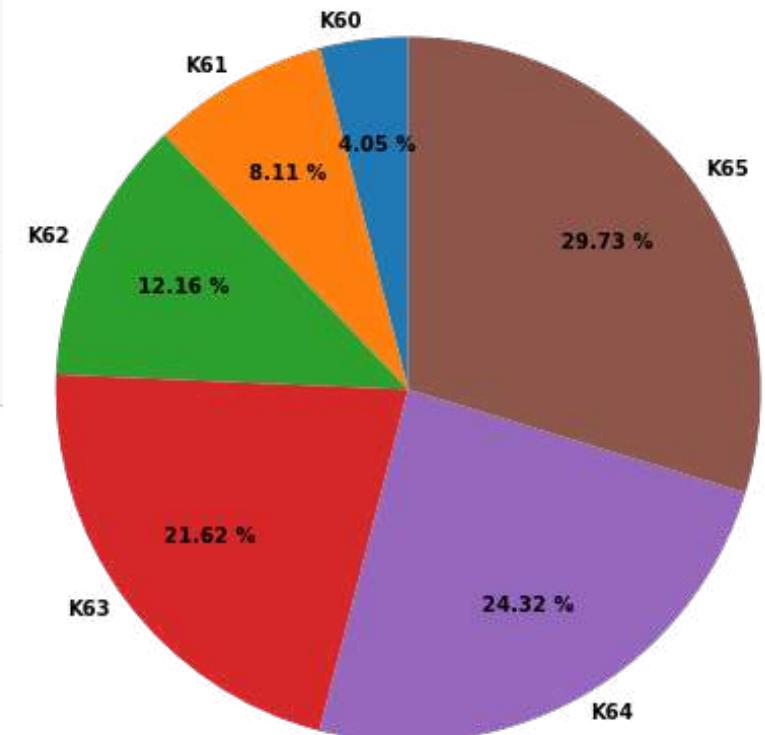
Cú pháp: plt.pie (values, labels)

```
1 plt.figure(figsize = (12,8))
2 #Vẽ biểu đồ tròn:
3 plt.pie(girls,           #Giá trị thể hiện
4          labels=labels,    #Nhãn tương ứng
5          autopct='%.2f %%', #Tính toán và hiển thị % tương ứng
6          pctdistance=0.7,   #Khoảng cách hiển thị giá trị % tới tâm.
7          startangle=90,     #Góc bắt đầu của biểu đồ
8          textprops={'color':'k','fontweight':'bold'},#thiết lập label
9          labeldistance=1.05, #Khoảng cách từ label tới biểu đồ
10         rotatelabels=False) #Label có xoay không?
11
12 plt.title('Tỷ lệ sinh viên Nữ theo từng Khoa', fontdict={'fontname':'Arial',
13                                         'fontweight':'bold',
14                                         'fontsize':18})
15 plt.show()
```



startangle:

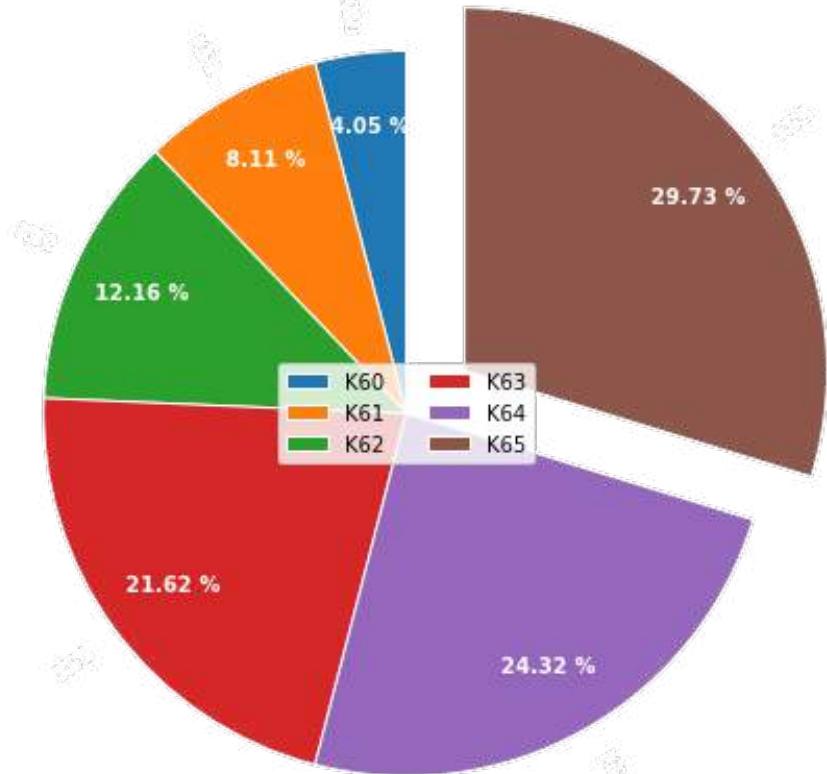
Tỷ lệ sinh viên Nữ theo từng Khoa



Biểu đồ tròn (Pie chart)

```
1 plt.figure(figsize = (12,8))
2 #Làm nổi bật một phần:
3 e = [0,0,0,0,0,0.2]
4 plt.pie(girls,
5         labels=labels,
6         autopct='%.2f %%',
7         pctdistance=0.8,
8         startangle=90,
9         labeldistance=1.05,
10        textprops={'color':'w','fontweight':'bold'},
11        rotatelabels=True,
12        wedgeprops=dict(edgecolor='w'),#Đường viền màu trắng
13        explode=e)#Làm nổi bật một phần
14
15 plt.title('Tỷ lệ sinh viên Nữ theo từng Khoa', fontdict={'fontname':'Arial',
16                                         'fontweight':'bold',
17                                         'fontsize':18})
18 plt.legend(ncol=2, loc='center')
19 plt.show()
```

Tỷ lệ sinh viên Nữ theo từng Khoa





b. Donus chart

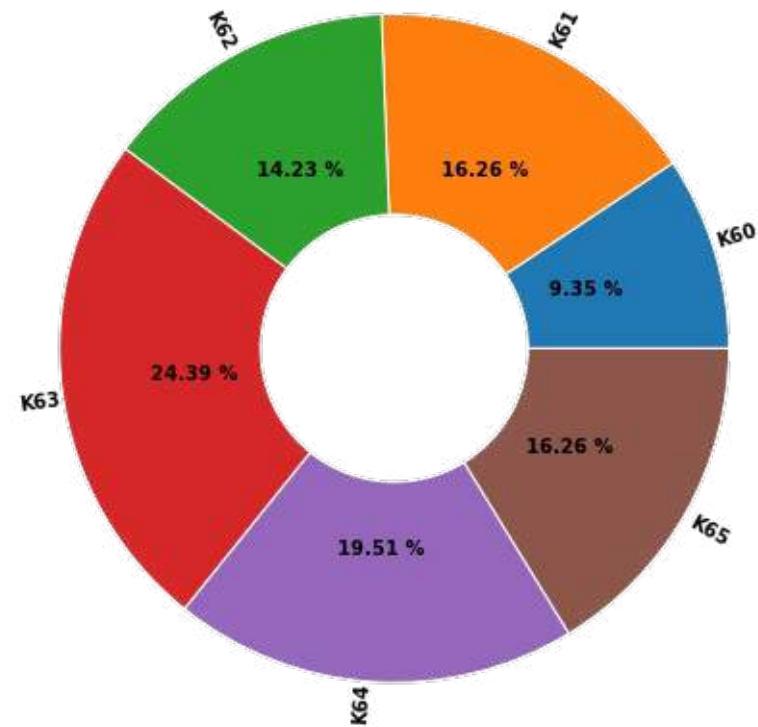


Donus chart

Cú pháp: plt.pie (values, labels)

```
1 plt.figure(figsize = (12,8))
2 #vẽ biểu đồ:
3 plt.pie(total,
4         labels=labels,
5         textprops={'color':'k','fontweight':'bold'},
6         rotatelabels=True,
7         labeldistance=1.0,
8         wedgeprops=dict(width=0.6,edgecolor='w'), #Xác định độ rộng của Pie
9         autopct='%.2f %%',
10        pctdistance=0.6)
11
12 plt.title('TỶ LỆ SINH VIÊN CỦA TỪNG KHOÁ', fontdict={'fontname':'Tahoma',
13                                         'fontweight':'bold',
14                                         'fontsize':18})
15 plt.show()
```

TỶ LỆ SINH VIÊN CỦA TỪNG KHOÁ



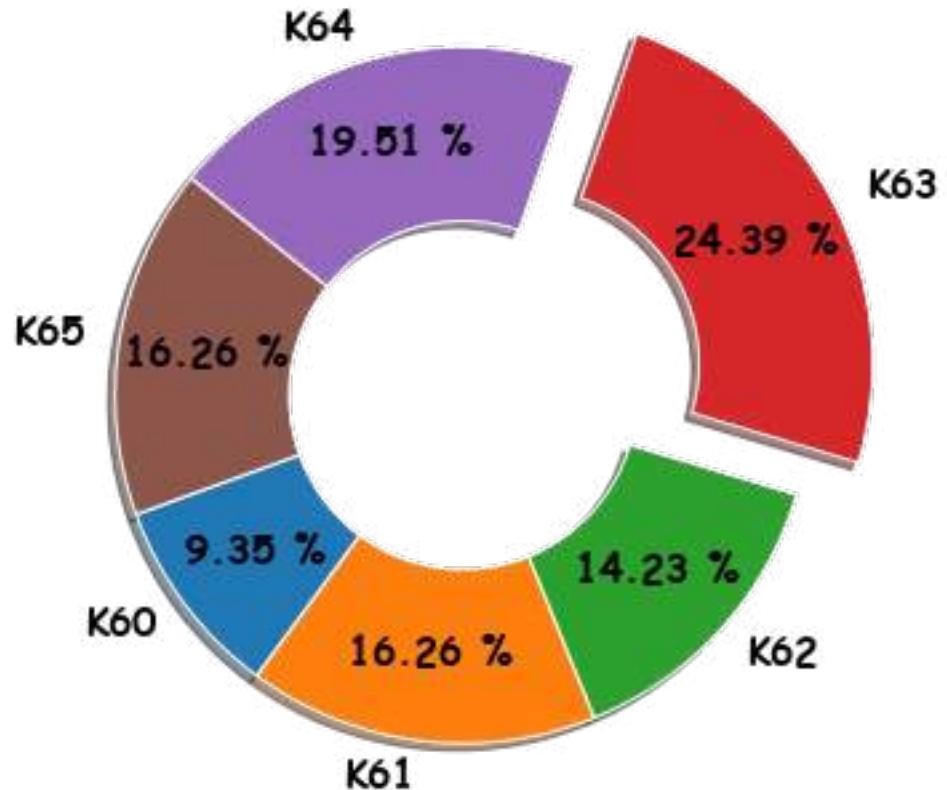
Donut chart

```

1 plt.figure(figsize = (12,6))
2
3 #Làm nổi bật một phần
4 e = [0,0,0,0.2,0,0]
5 plt.pie(total,
6         labels=labels,
7         explode=e, #Làm nổi bật biểu đồ
8         startangle=200,
9         textprops={'color':'k','fontweight':'bold',
10                 'fontname':'Comic Sans MS',
11                 'fontsize':15},
12         wedgeprops=dict(width=0.5,edgecolor='w'),
13         shadow=True, #Tạo bóng cho biểu đồ
14         autopct='%.2f %%',
15         pctdistance=0.75)
16
17 plt.title('TỶ LỆ SINH VIÊN CỦA TỪNG KHOÁ',
18             fontdict={'fontname':'Tahoma',
19                         'fontweight':'bold',
20                         'fontsize':18})
21 plt.show()

```

TỶ LỆ SINH VIÊN CỦA TỪNG KHOÁ

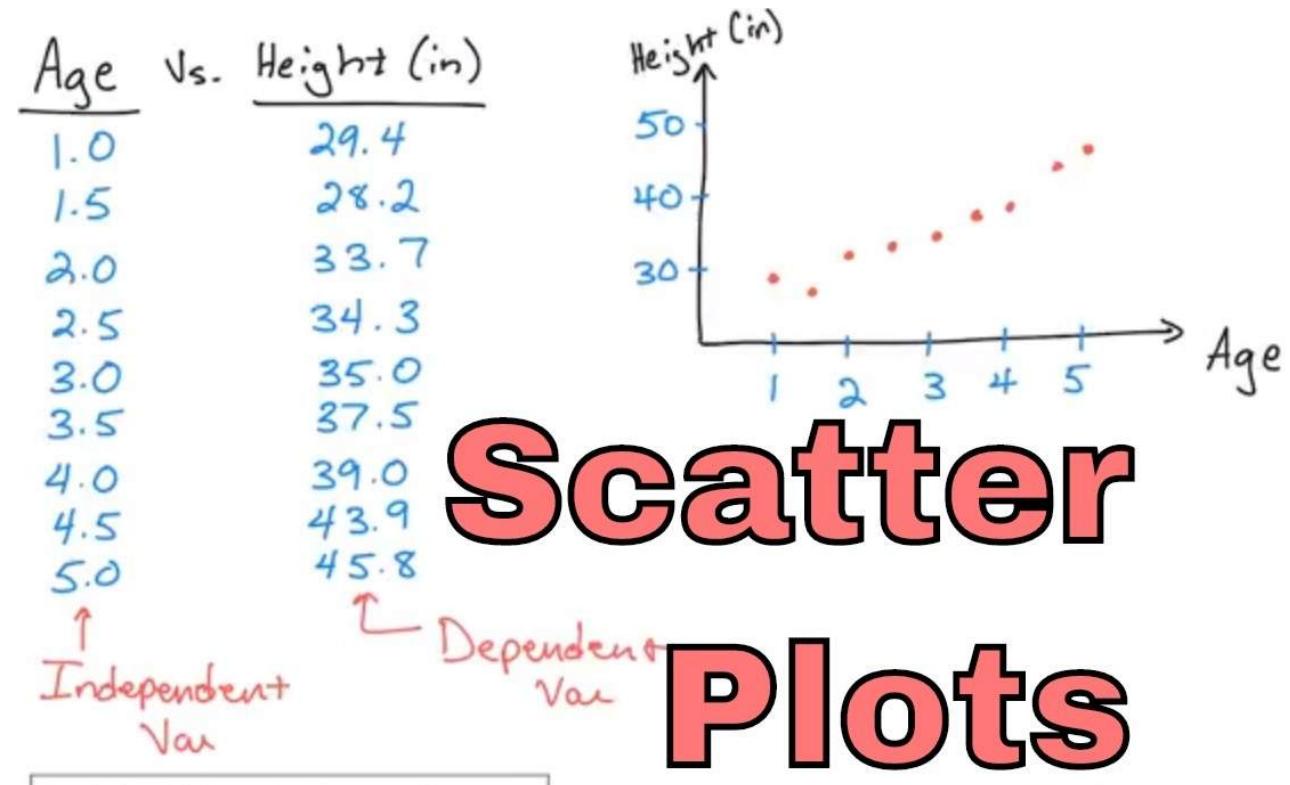


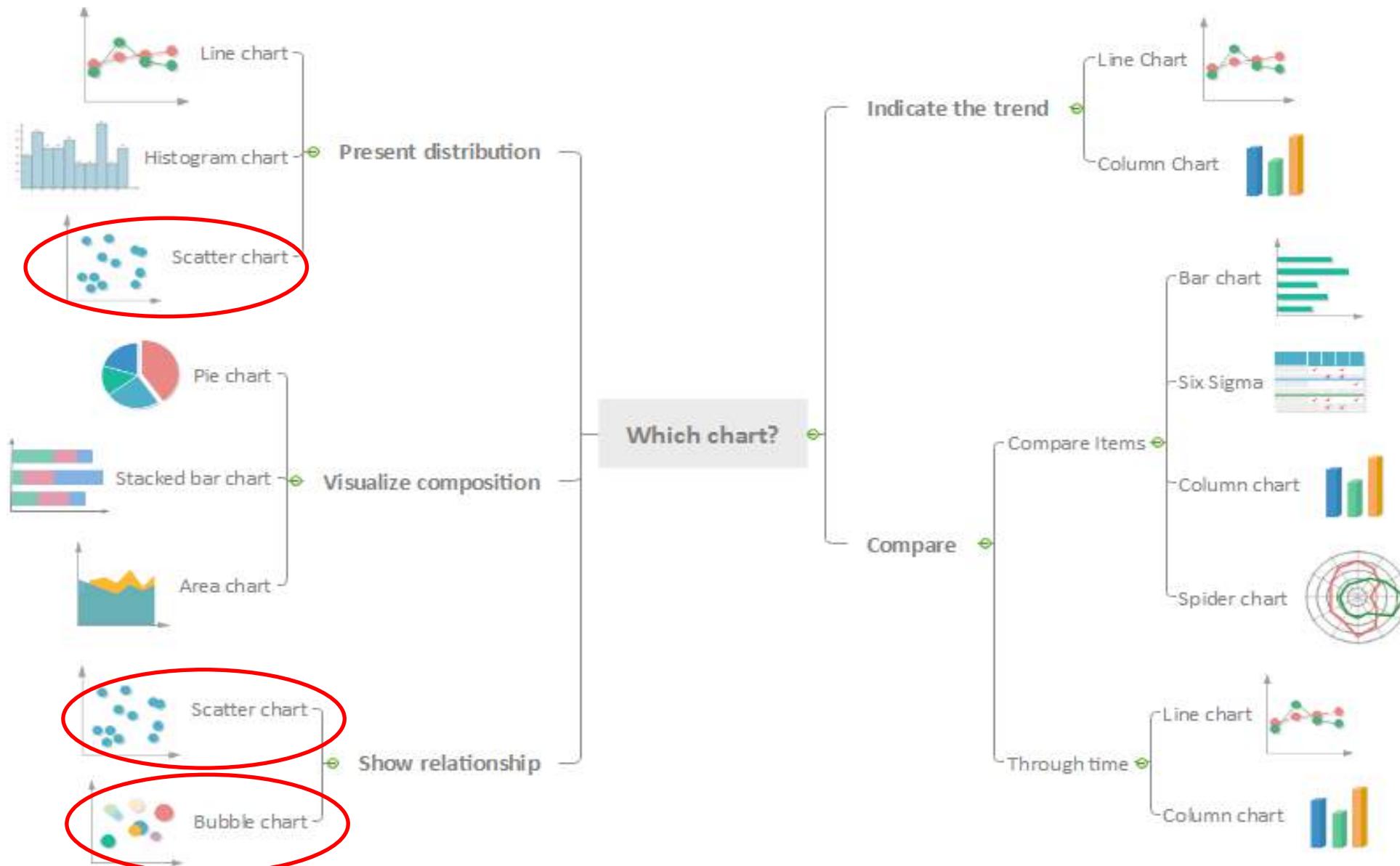


7. Biểu đồ phân tán (Scatter chart)

Biểu đồ phân tán (Scatter chart)

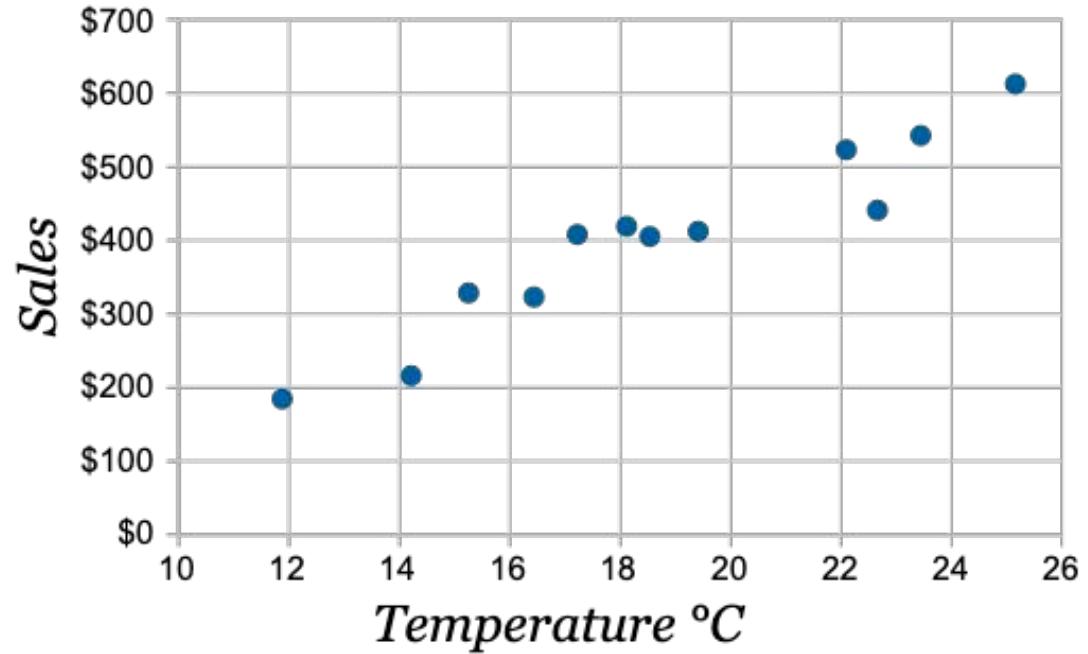
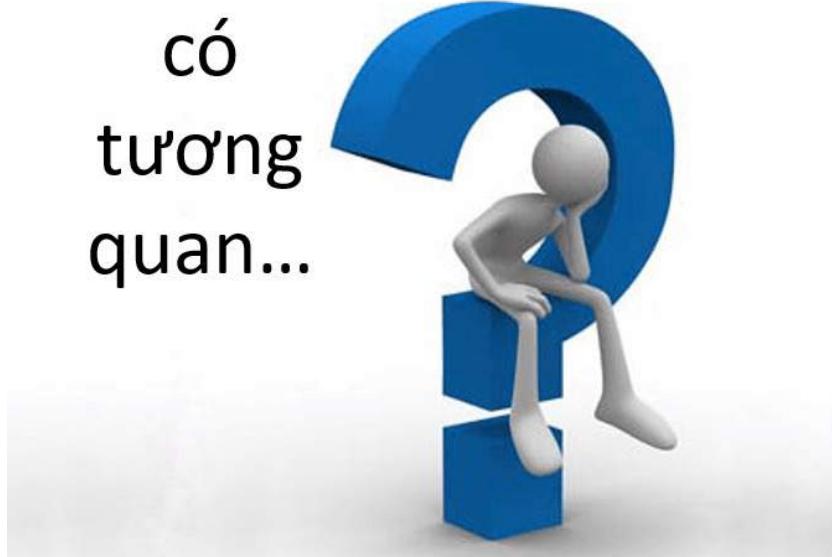
- **Biểu đồ phân tán (Scatter chart, Scatterplot, scatter graph)** là loại biểu đồ được dựng bởi các điểm theo tọa độ toán học để xác định mối tương quan giữa 2 biến.
- Đồ thị thể hiện 2 bộ dữ liệu, trục tung Y được sử dụng cho biến được dự đoán (biến phụ thuộc), trục hoành X được sử dụng cho biến dùng để dự đoán (biến độc lập).
- Scatter plot được sử dụng khi có 2 cặp dữ liệu (biến) và muốn xác định 2 biến có liên quan với nhau hay không? Liên quan nhiều hay ít và như thế nào?





Biểu đồ phân tán (Scatter chart)

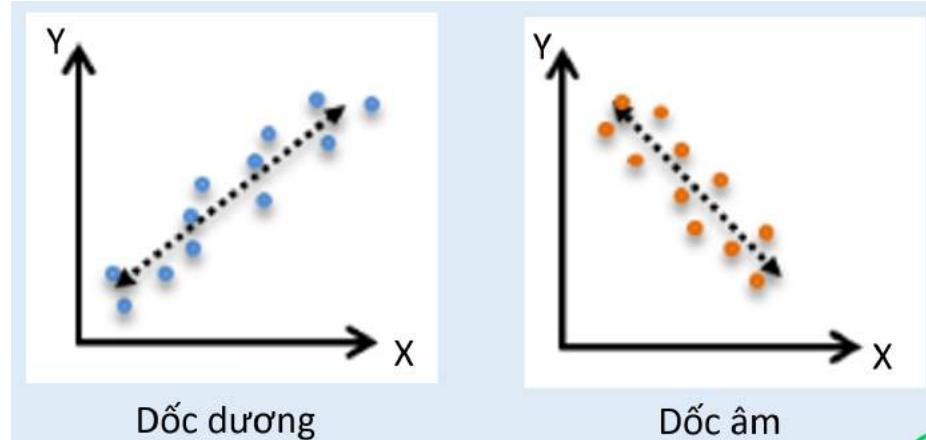
X và Y
có
tương
quan...



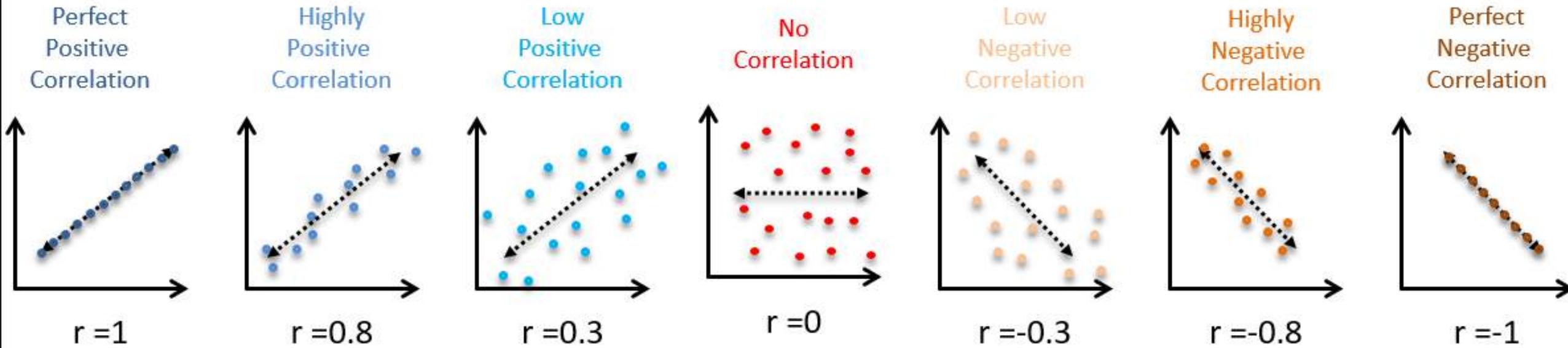
- **Biểu đồ phân tán** sẽ cho chúng ta thấy được mức độ tương quan giữa 2 biến

Biểu đồ phân tán (Scatter chart)

- Dựa vào hình dạng, bờ dốc và độ tập trung điểm của biểu đồ để xác định mối tương quan giữa 2 biến.

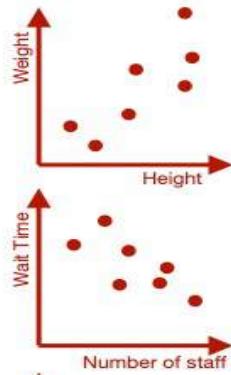


Scatter Plots & Correlation Examples



Biểu đồ phân tán (Scatter chart)

Scatter Graphs can show a relationship between two variables.



...such as people's height and weight.

...or the number of staff working in KFC and the wait time for food.

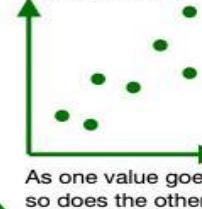


...or the distance people live from work and their best score in darts.

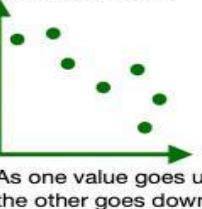
If the two variables have a relationship we call it correlation.

There are different types of correlation:

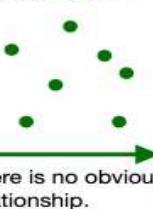
Positive correlation:



Negative correlation:

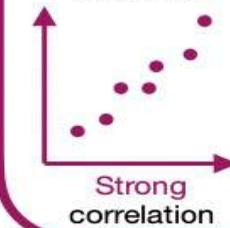


No correlation:



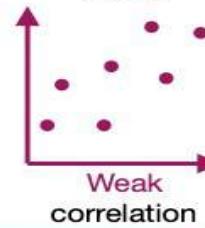
Correlation can be strong or weak.

If the correlation is strong, all the points will closely follow a straight line.



Strong correlation

If the correlation is weak, the points will follow the line more loosely.

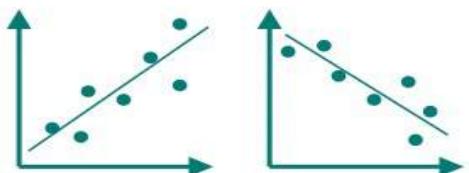


Weak correlation

Scatter Graphs

We can show the correlation more clearly by drawing a Line of Best Fit.

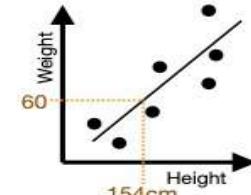
This should pass through the middle of all the points (but does not have to touch any of the points).



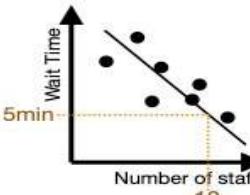
We can use the Line of Best Fit to make predictions of other results.

For example, we can estimate:

...someone's height if we know their weight is 60kg.



...or the wait time in KFC if we know they have 10 staff on today.



Sometimes you might be asked to explain the correlation in context.

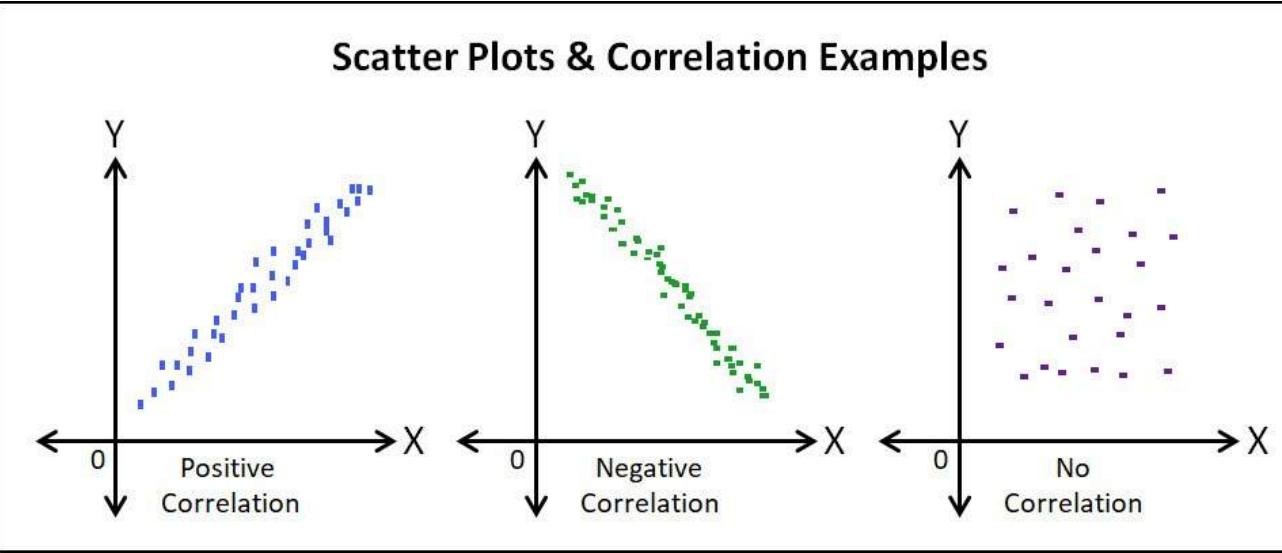
This means describing what is actually happening. eg:

"Taller people are usually heavier."

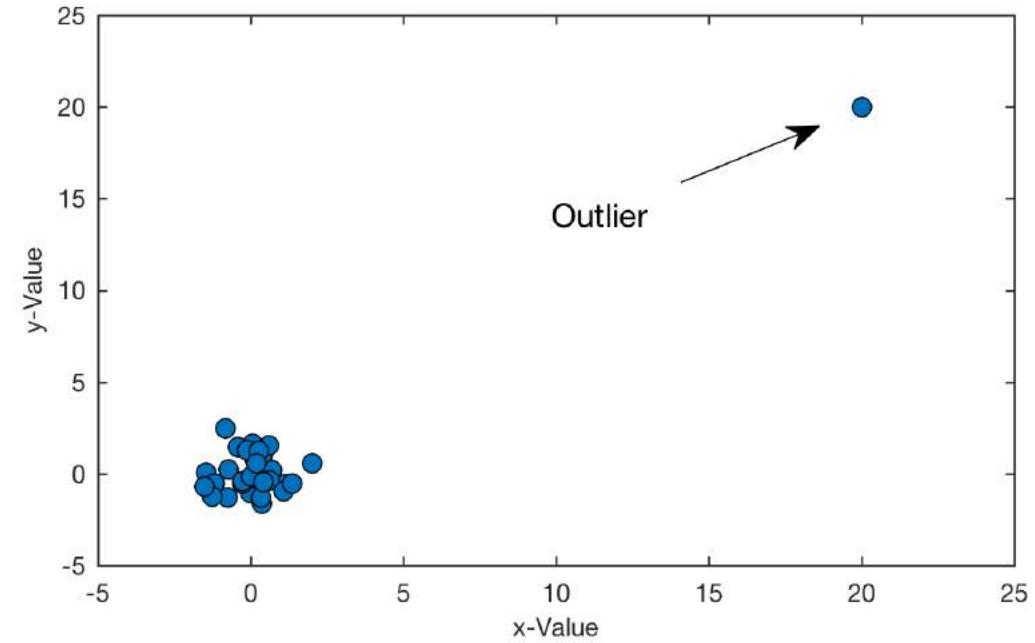
"When there are more staff working, you wait less."

"There is no relationship between how far people live from work and their darts ability."

Biểu đồ phân tán sử dụng để?



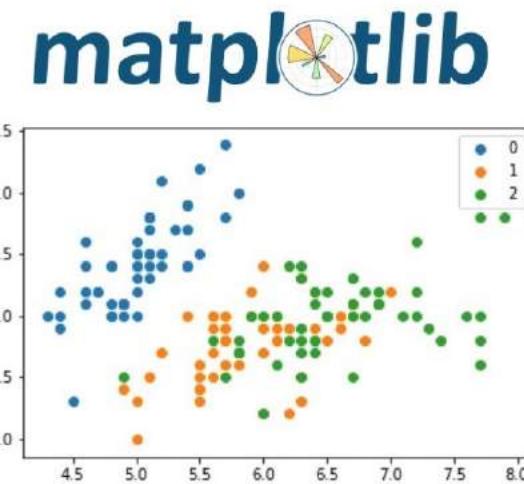
Xác định mối tương quan giữa các biến số (có hay không? Mạnh hay yếu?)



Phát hiện các điểm ngoại lai (outlier) trong tập dữ liệu

Scatter chart với Matplotlib

Tập dữ liệu **Diamonds.txt** lưu trữ trọng lượng (carat) và giá (\$) tương ứng của 50 viên kim cương.



	Dia...
0.23	484
0.31	942
0.2	345
1.02	4459
1.63	14022
1.14	4212
2.01	11925
1.28	9548
1.7	11605
1.01	4642
0.64	3541
0.97	4504
1.78	13691
3.4	15964
3.01	10453
1.51	11560
1.37	7979
1.5	9533
0.54	1723
0.72	3344
1.13	6133
2.24	13827
3.01	16538
4.5	18531
0.92	3625
1.05	7879
0.55	1319
0.74	2761
0.91	3620
1.23	6165

Biểu đồ phân tán (Scatter chart)

Cú pháp: plt.scatter (x, y)

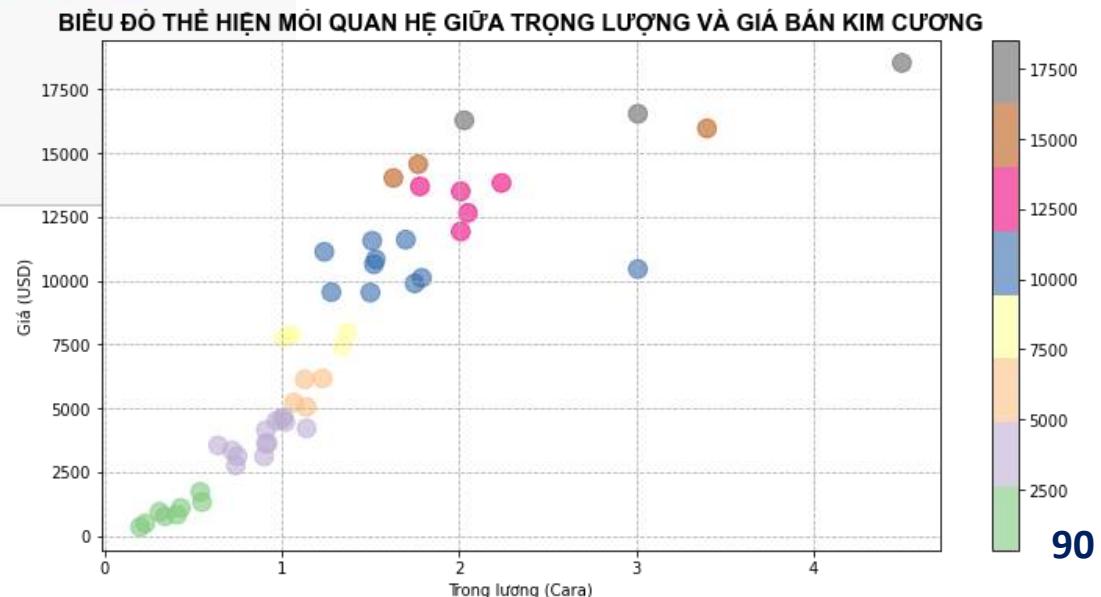
```
1 plt.figure(figsize = (10,5))
2
3 plt.scatter(weight,           #Dữ liệu trục X
4               price,          #Dữ liệu trục Y
5               c='g',            #Màu của Point
6               marker='d',       #Kiểu Point
7               s=120,             #Kích thước của Point
8               alpha=0.5)        #Độ trong suốt của Point
9
10 plt.title('BIỂU ĐỒ THỂ HIỆN MỐI QUAN HỆ GIỮA TRỌNG LƯỢNG VÀ GIÁ BÁN KIM CƯƠNG',
11           fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})
12 plt.grid(ls='--')
13 plt.xlabel('Trọng lượng (Cara)')
14 plt.ylabel('Giá (USD)')
15 plt.show()
```



Biểu đồ phân tán (Scatter chart)

Sử dụng colorbar:

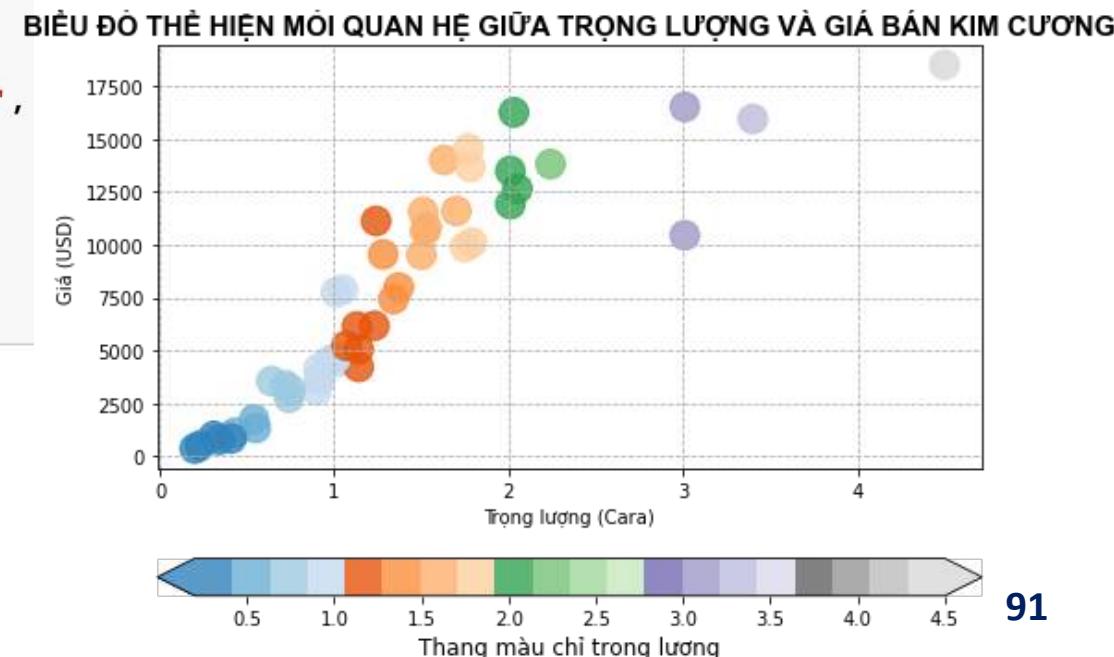
```
1 plt.figure(figsize = (12,6))
2 #Vẽ biểu đồ scatter:
3 plt.scatter(weight, price,
4             s=140,
5             alpha=0.6,
6             c=price,
7             cmap='Accent') #Sử dụng color map
8 plt.colorbar() #Hiển thị thanh color bar:
9
10
11 plt.title('BIỂU ĐỒ THỂ HIỆN MỐI QUAN HỆ GIỮA TRỌNG LƯỢNG VÀ GIÁ BÁN KIM CƯƠNG',
12           fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})
13 plt.grid(ls='--')
14 plt.xlabel('Trọng lượng (Cara)')
15 plt.ylabel('Giá (USD)')
16
17 plt.show()
```



Biểu đồ phân tán (Scatter chart)

Sử dụng colorbar:

```
1 plt.figure(figsize = (8,6))
2 #Vẽ biểu đồ scatter:
3 plt.scatter(weight, price,
4             s=250,
5             alpha=0.8,
6             c=weight,
7             cmap='tab20c') #Sử dụng color map
8
9 #Hiển thị và setup thanh color bar:
10 cbar = plt.colorbar(location ='bottom', #Vị trí của colorbar
11                      extend='both', #Đầu của colorbar
12                      pad=0.15) #Khoảng cách giữa thang màu và biểu đồ
13 cbar.set_label(label='Thang màu chỉ trọng lượng',size=12)
14
15 plt.title('BIỂU ĐỒ THỂ HIỆN MỐI QUAN HỆ GIỮA TRỌNG LƯỢNG VÀ GIÁ BÁN KIM CƯƠNG',
16            fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})
17 plt.grid(ls='--')
18 plt.xlabel('Trọng lượng (Cara)')
19 plt.ylabel('Giá (USD)')
20
21 plt.show()
```





a. Ứng dụng của Scatter chart

Đánh giá độ tương quan giữa 2 biến



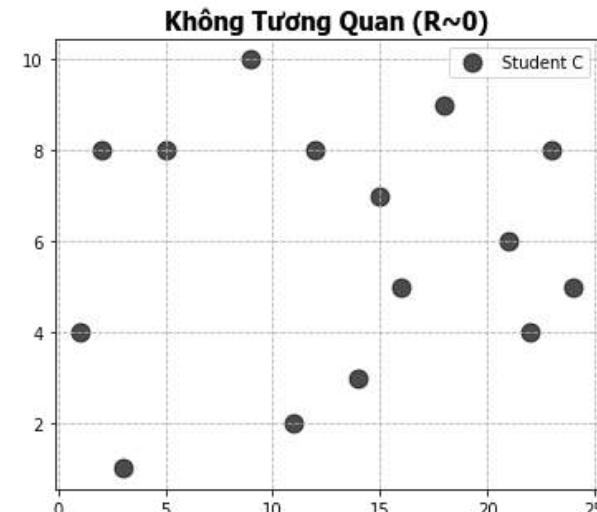
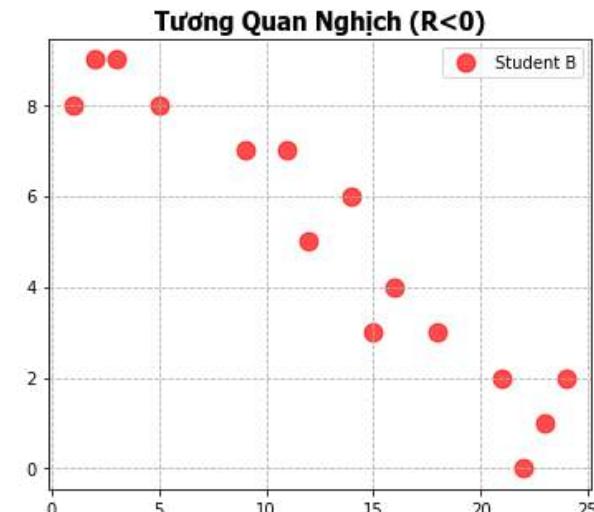
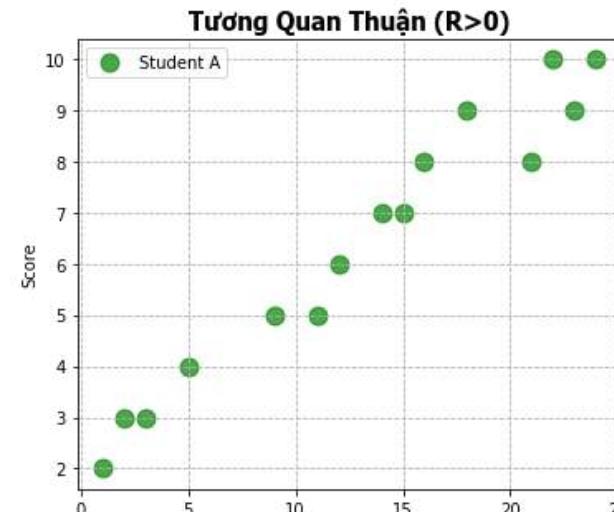
VINBIGDATA VINGROUP

Academy
Vietnam

Bảng dữ liệu thể hiện thời gian dành cho việc học và số điểm nhận được của 3 sinh viên A, B, C

Hour	Score_A	Score_B	Score_C
------	---------	---------	---------

0	1	2	8	4
1	2	3	9	8
2	3	3	9	1
3	5	4	8	8
4	9	5	7	10
5	11	5	7	2
6	12	6	5	8
7	14	7	6	3
8	15	7	3	7
9	16	8	4	5
10	18	9	3	9
11	21	8	2	6
12	22	10	0	4
13	23	9	1	8
14	24	10	2	5



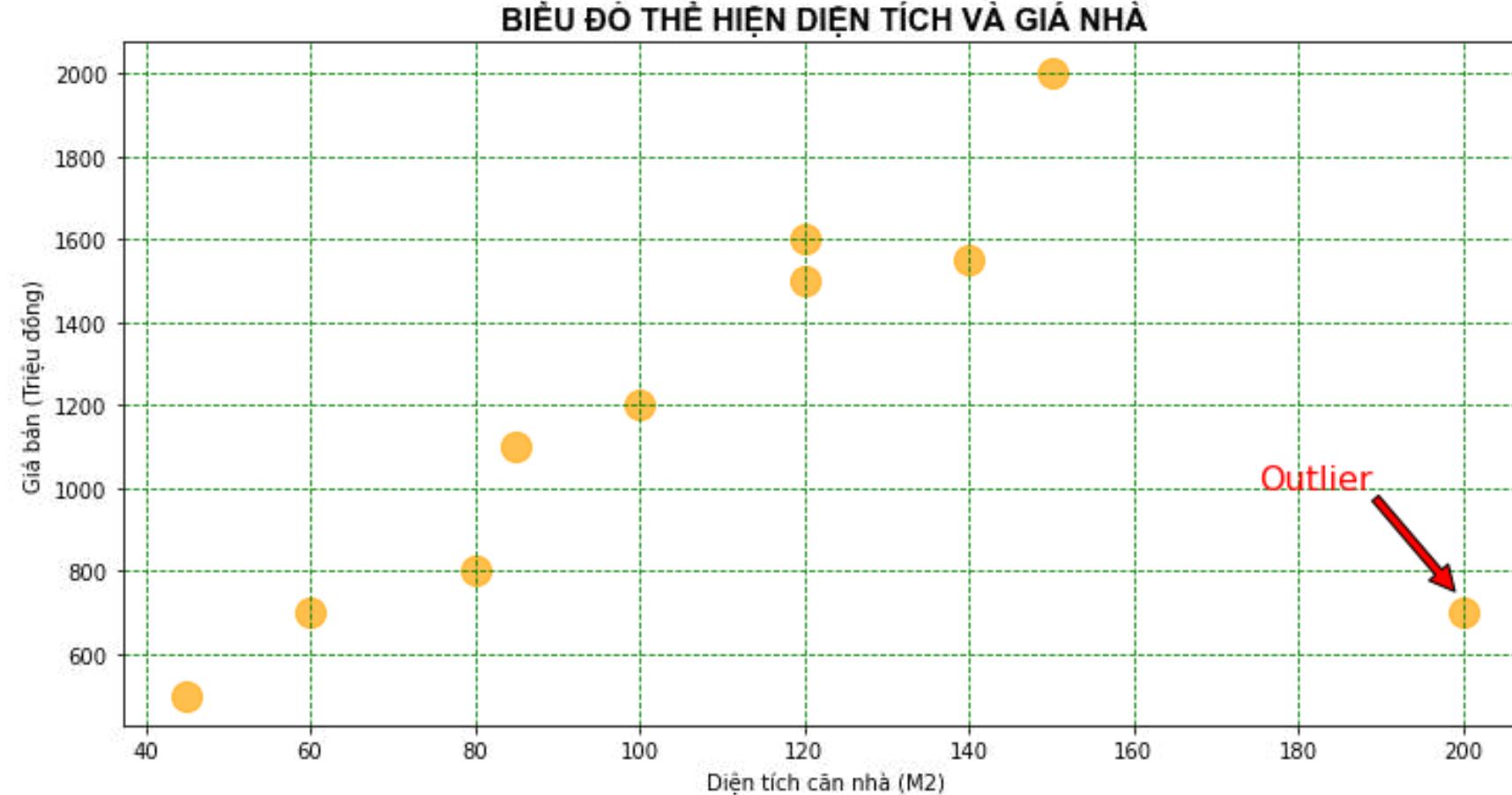
Phát hiện các giá trị ngoại lai



VINBIGDATA VINGROUP

Academy
Vietnam

```
1 #Dữ liệu bao gồm diện tích và giá của một số căn nhà:  
2 area_house = [45, 80, 120, 100, 150, 60, 200, 120, 85, 140]      #Diện tích (m2)  
3 price_house = [500, 800, 1600, 1200, 2000, 700, 700, 1500, 1100, 1550] #Giá nhà (Triệu đồng)
```



b. Mở rộng....

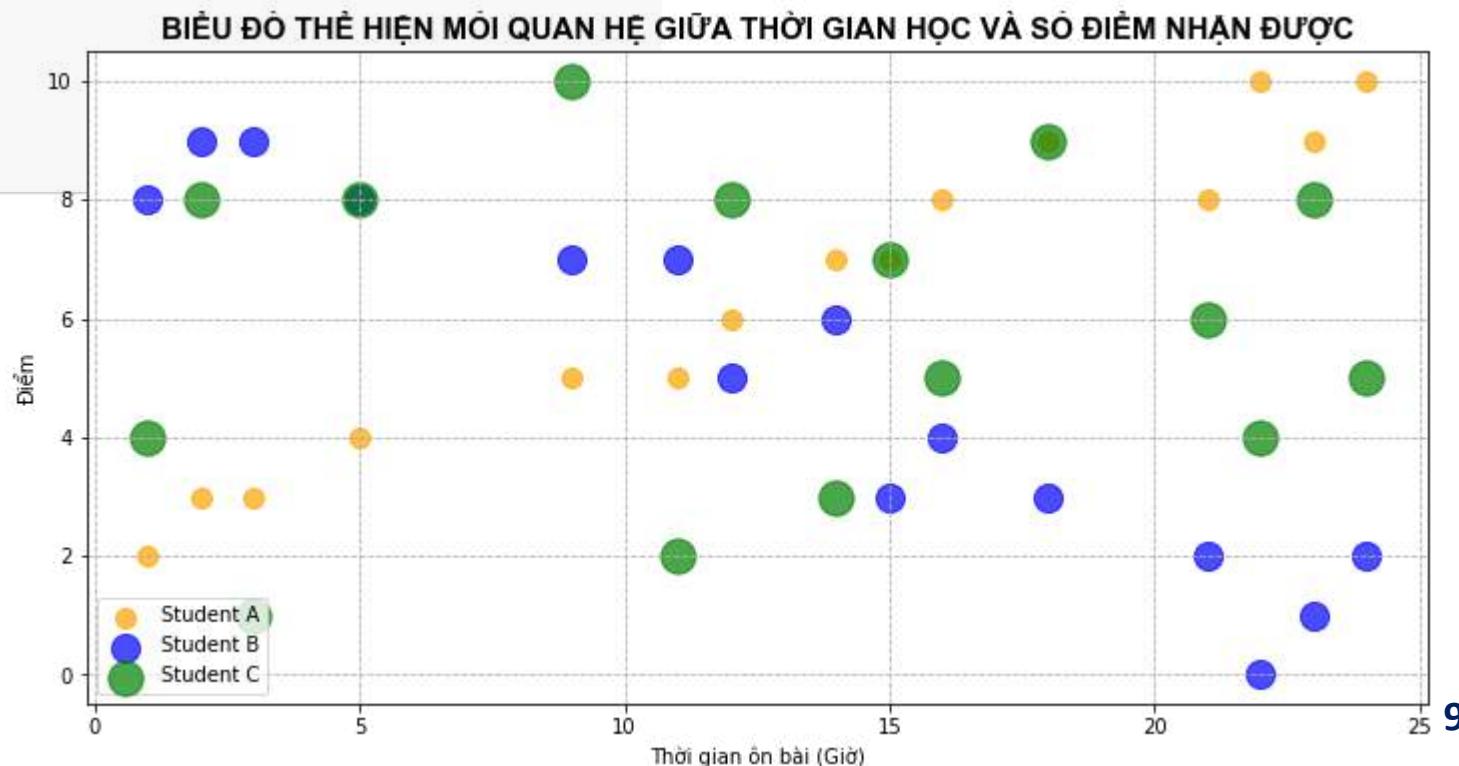
Nhiều scatter trên cùng một plot



VINBIGDATA
VINGROUP

Academy
Vietnam

```
1 plt.figure(figsize = (12,6))
2
3 #Vẽ nhiều biểu đồ scatter trên một plot:
4 plt.scatter(hour, scoreA,s=100, alpha=0.7, c='orange', label='Student A')
5 plt.scatter(hour, scoreB,s=200, alpha=0.7, c='blue', label='Student B')
6 plt.scatter(hour, scoreC,s=300, alpha=0.7, c='g', label='Student C')
7
8 plt.title('BIỂU ĐỒ THỂ HIỆN MỐI QUAN HỆ GIỮA THỜI GIAN HỌC VÀ SỐ ĐIỂM NHẬN ĐƯỢC',
9             fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})
10
11 plt.grid(ls='--')
12 plt.xlabel('Thời gian ôn bài (Giờ)')
13 plt.ylabel('Điểm')
14 plt.legend(loc='lower left')
15 plt.show()
```

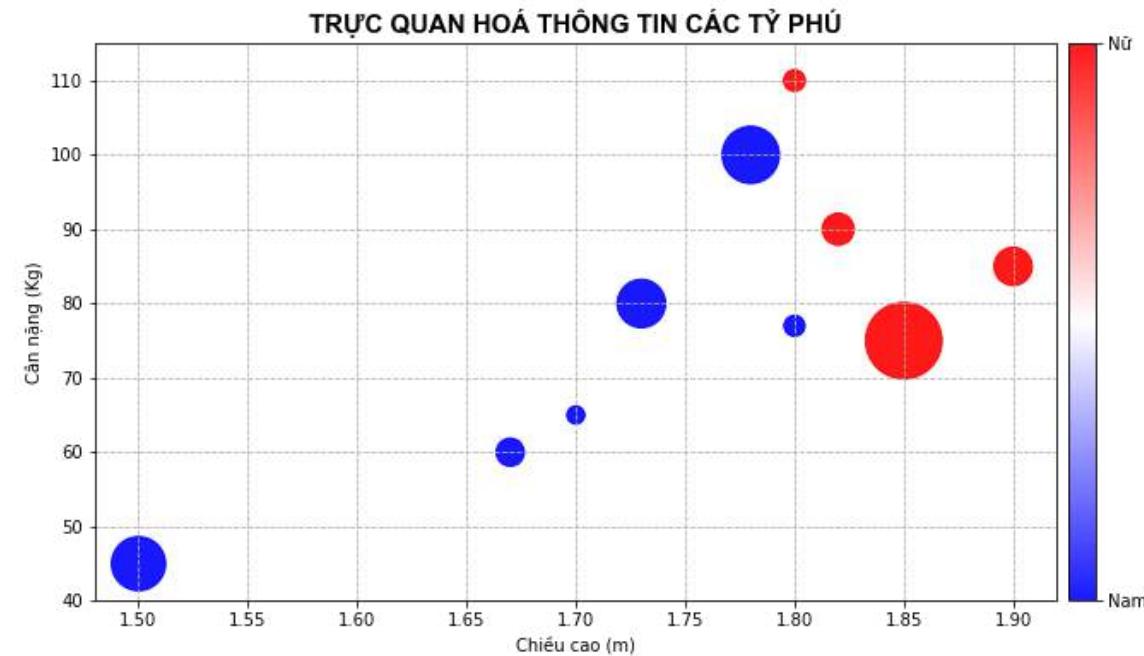


Nhiều biến trên một scatter

Mỗi một biểu đồ scatter có thể biểu diễn được tối đa 4 biến (x, y, size, color)

```
1 #Thông tin của 10 doanh nhân:  
2 weight = [ 45, 80, 110, 100, 75, 60, 85, 77, 65, 90] #Cân nặng (Kg)  
3 height = [1.50, 1.73, 1.80, 1.78, 1.85, 1.67, 1.90, 1.80, 1.70, 1.82] #Chiều cao (M)  
4 sex = [ 0, 0, 1, 0, 1, 0, 1, 0, 0, 1] #Giới tính (0:Nam - 1:Nữ)  
5 money = [1000, 800, 160, 1120, 2000, 270, 500, 150, 110, 350] # Tài sản (Triệu USD)
```

```
1 plt.figure(figsize = (12,6))  
2  
3 #Vẽ biểu đồ scatter biểu diễn thông tin của 10 doanh nhân:  
4 #Biến 1: Chiều cao - Trục X  
5 #Biến 2: Cân nặng - Trục Y  
6 #Biến 3: Giới tính - Màu sắc của Point  
7 #Biến 4: Tài sản - Kích thước của Point  
8 plt.scatter(height, weight, c=sex, s=money, alpha=0.9, cmap='bwr')  
9  
10 cbar = plt.colorbar(pad=0.01)  
11 cbar.set_ticks([0,1])  
12 cbar.set_ticklabels(["Nam", "Nữ"])  
13 plt.title('TRỰC QUAN HOÁ THÔNG TIN CÁC TỶ PHÚ',  
14 fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})  
15 plt.grid(ls='--')  
16 plt.xlabel('Chiều cao (m)')  
17 plt.ylabel('Cân nặng (Kg)')  
18 plt.ylim([40,115])  
19 plt.show()
```

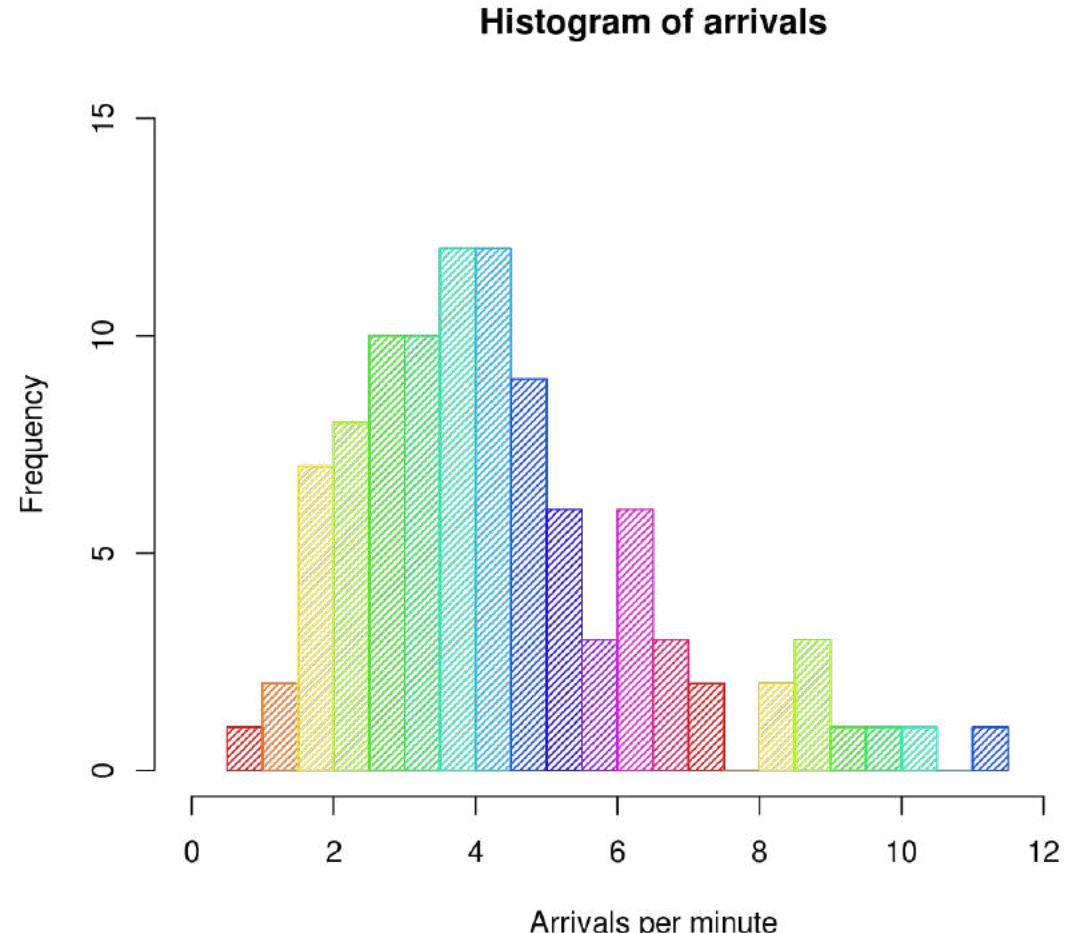




8. Biểu đồ tần suất (Histogram chart)

Biểu đồ tần suất (Histogram chart)

- **Biểu đồ Histogram chart** là một dạng biểu đồ thể hiện tần suất dạng cột. Nó mô tả dữ liệu một cách đơn giản mà không làm mất bất cứ thông tin thống kê nào của tập dữ liệu.
- Histogram cho thấy hình thái phân bố của dữ liệu. Sử dụng biểu đồ Histogram có thể trả lời cho các câu hỏi:
 - Kiểu phân bố của dữ liệu?
 - Độ rộng của dữ liệu thế nào?
 - Dữ liệu có đối xứng hay không?
 - Có dữ liệu nào nằm ngoài hay không?



Biểu đồ tần suất (Histogram chart)

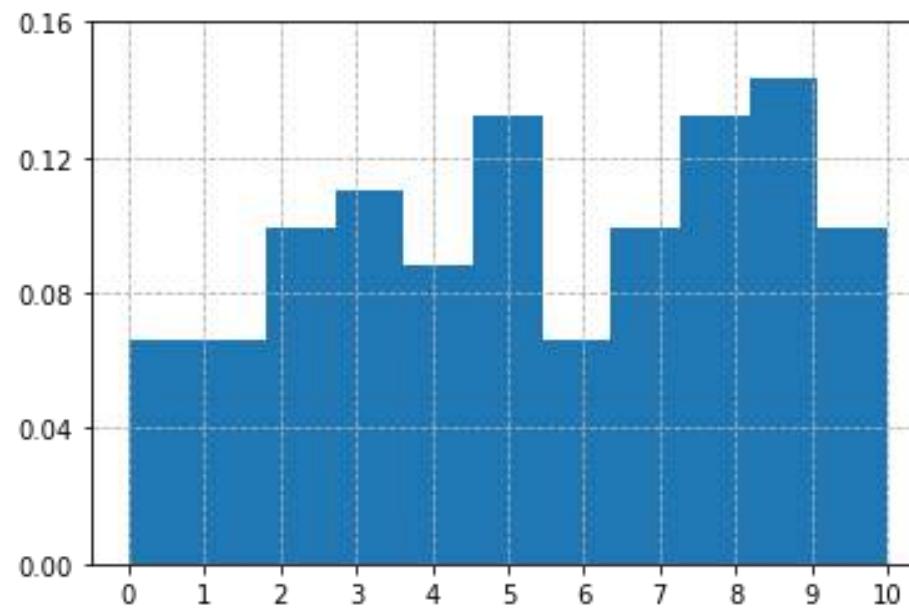
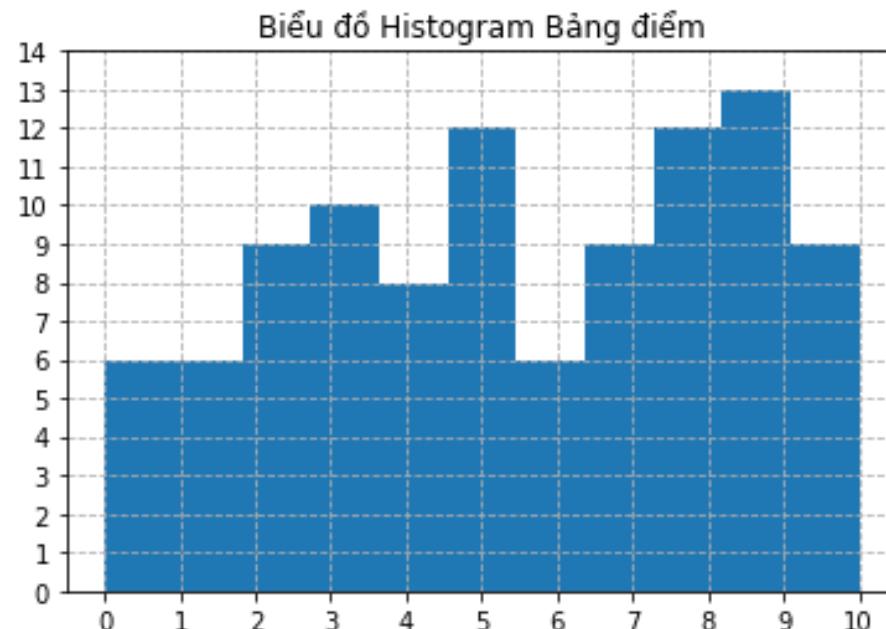


Cách xây dựng biểu đồ Histogram

- Bảng điểm của 100 học sinh khối lớp 5

```
[ 3,  0,  5,  6,  9,  5,  3,  2,  4,  5]
[ 8,  7,  8,  0,  1,  8,  8,  8,  3,  9]
[ 4,  8,  4,  7,  2,  9,  7,  4,  7,  10]
[ 0, 10,  7,  5,  0,  5,  5,  4,  1,  8]
[10,  9,  0,  2,  3,  8,  8,  7,  7,  6]
[ 3,  9,  6,  9,  5,  5, 10,  5, 10,  5]
[ 1,  9, 10,  2,  5, 10,  3,  1,  2,  7]
[ 4,  9,  9,  2,  3,  3,  2,  6,  9,  5]
[ 9,  0,  6,  9,  4,  9, 10,  8,  4,  3]
[ 1,  2, 10,  7,  8,  8,  1,  2,  3,  6]
```

Điểm	0	1	2	3	4	5	6	7	8	9	10	
n	6	6	9	10	8	12	6	9	12	13	9	100
P(n)	0.06	0.06	0.09	0.1	0.08	0.12	0.06	0.09	0.12	0.13	0.09	1



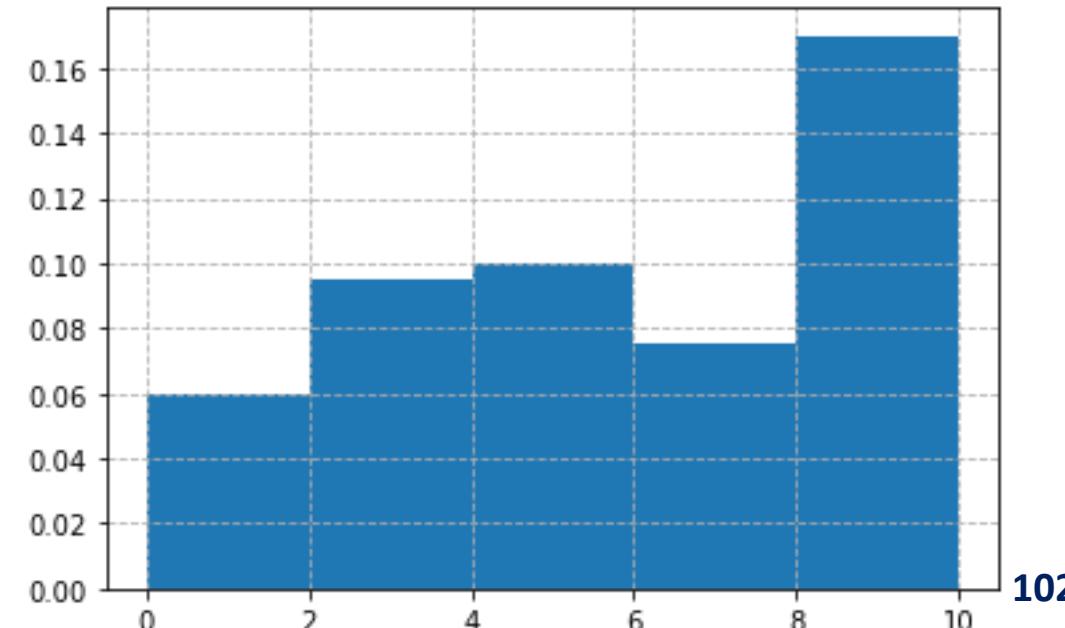
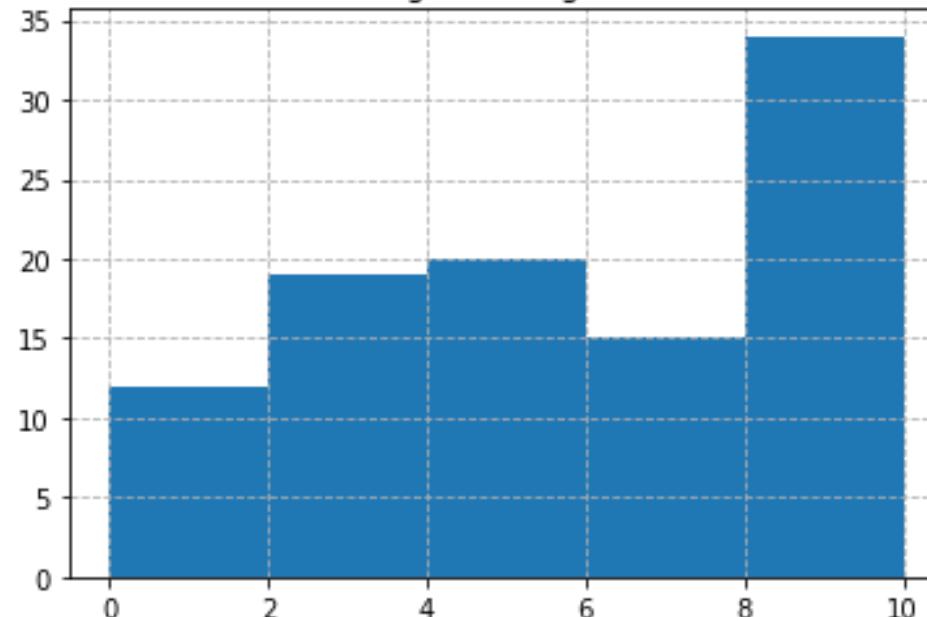
Cách xây dựng biểu đồ Histogram

- Bảng điểm của 100 học sinh khối lớp 5

```
[ 3,  0,  5,  6,  9,  5,  3,  2,  4,  5]
[ 8,  7,  8,  0,  1,  8,  8,  8,  3,  9]
[ 4,  8,  4,  7,  2,  9,  7,  4,  7,  10]
[ 0, 10,  7,  5,  0,  5,  5,  4,  1,  8]
[10,  9,  0,  2,  3,  8,  8,  7,  7,  6]
[ 3,  9,  6,  9,  5,  5, 10,  5, 10,  5]
[ 1,  9, 10,  2,  5, 10,  3,  1,  2,  7]
[ 4,  9,  9,  2,  3,  3,  2,  6,  9,  5]
[ 9,  0,  6,  9,  4,  9, 10,  8,  4,  3]
[ 1,  2, 10,  7,  8,  8,  1,  2,  3,  6]
```

Điểm	0-<2	2-<4	4-<6	6-<8	8-10	
n	12	19	20	15	34	100
P(n)	0.12	0.19	0.20	0.15	0.34	1

Biểu đồ Histogram Bảng điểm (bins=5)



Biểu đồ tần suất (Histogram chart)



VINBIGDATA VINGROUP

Academy Vietnam

Phân phối chuẩn: Biểu đồ có dạng hình chuông. Tần suất xuất hiện nhiều nhất ở trung tâm và giảm dần về hai phía (hai phía có dạng đối xứng).

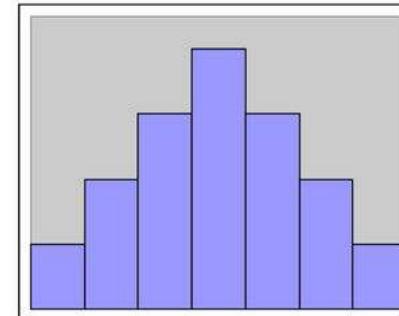
Phân phối đồng nhất là dạng phân bố mà ở đó tần suất xuất hiện của các giá trị là như nhau, không có đỉnh. Trông giống như một hình chữ nhật → phân phối hình chữ nhật.

Phân phối hai đỉnh: Biểu đồ thu được trông giống như lưng của một con lắc đà 2 biếu. Tần suất xuất hiện tại trung tâm thấp hơn các khoảng lân cận.

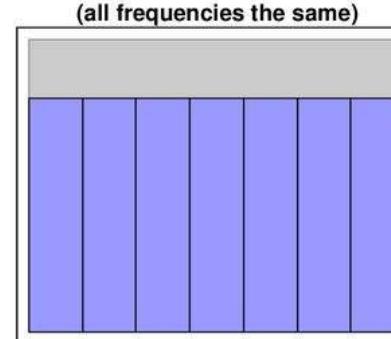
Phân phối lệch: là dạng phân bố không cân xứng. Giá trị trung bình của đồ thị bị lệch về bên trái hoặc bên phải tạo nên hình dáng không cân xứng.

Types of Histograms

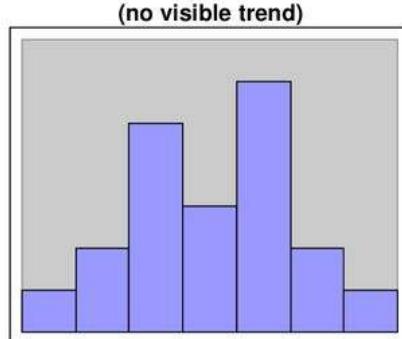
Symmetrical



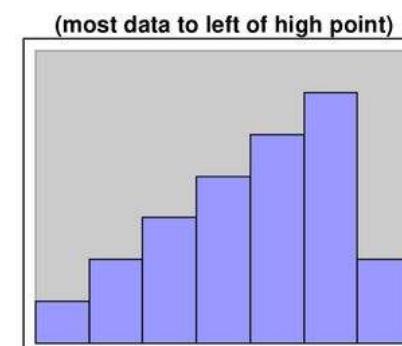
Uniform



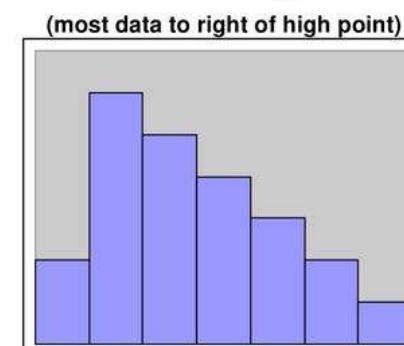
Bimodal



Skewed Left



Skewed Right



Histogram chart với Matplotlib

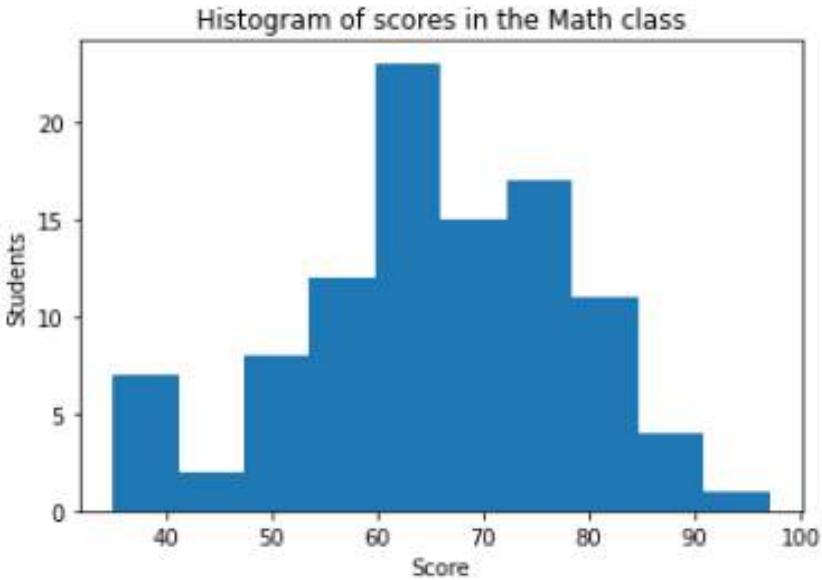


VINBIGDATA VINGROUP

Academy
Vietnam

Tập dữ liệu **Data_score.csv** lưu trữ dữ liệu 3 môn
Math, Science, History của 500 học sinh

matplotlib



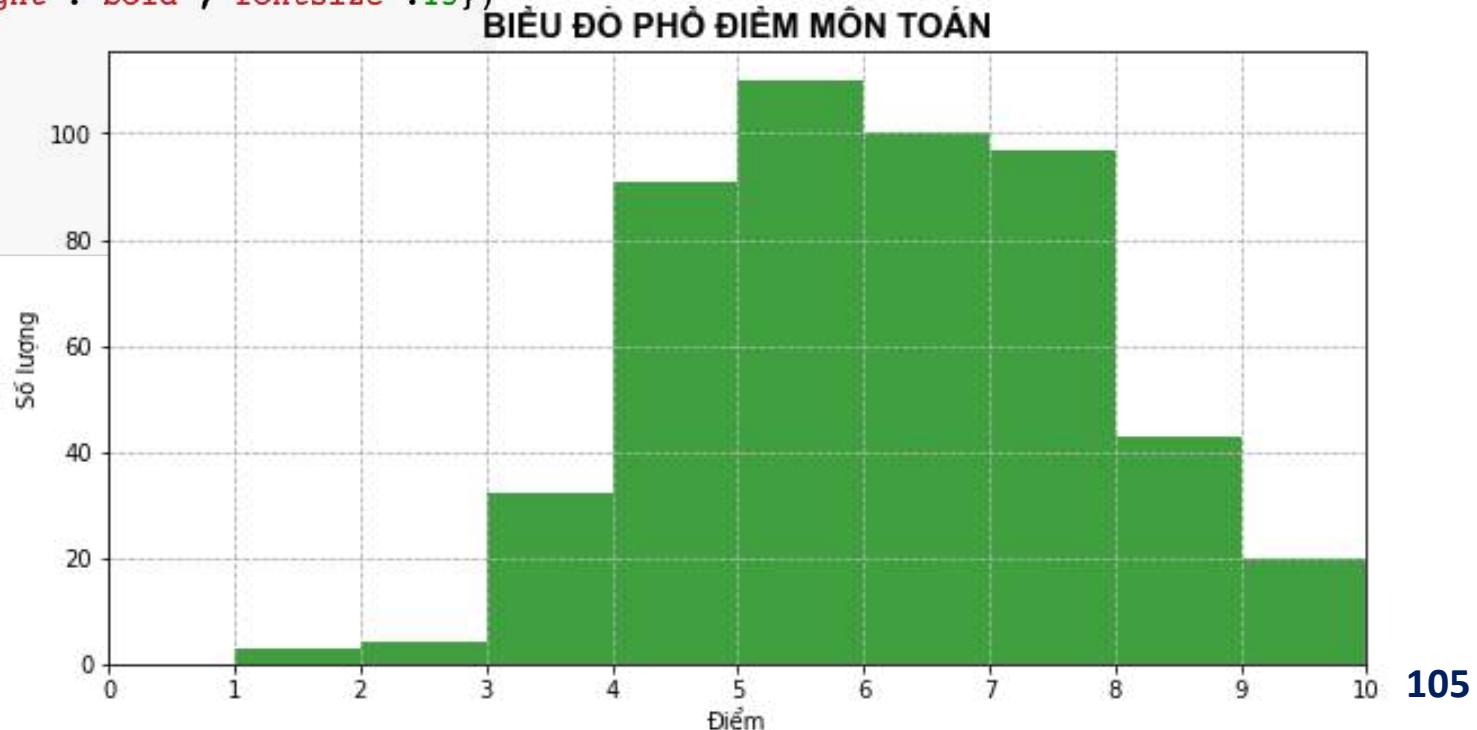
Plot a Histogram

	A	B	C
1	Math	Science	History
2	7	10	4
3	4	9	3
4	4	8	2
5	4	8	3
6	4	6	3
7	6	8	1
8	5	10	7
9	5	10	5
10	5	7	4
11	6	8	3
12	7	9	3
13	6	8	2
14	5	10	7
15	5	9	6
16	7	5	2
17	8	10	3
18	4	9	7
19	5	7	2
20	5	9	6

Biểu đồ tần suất (Histogram chart)

Cú pháp: plt.hist (x)

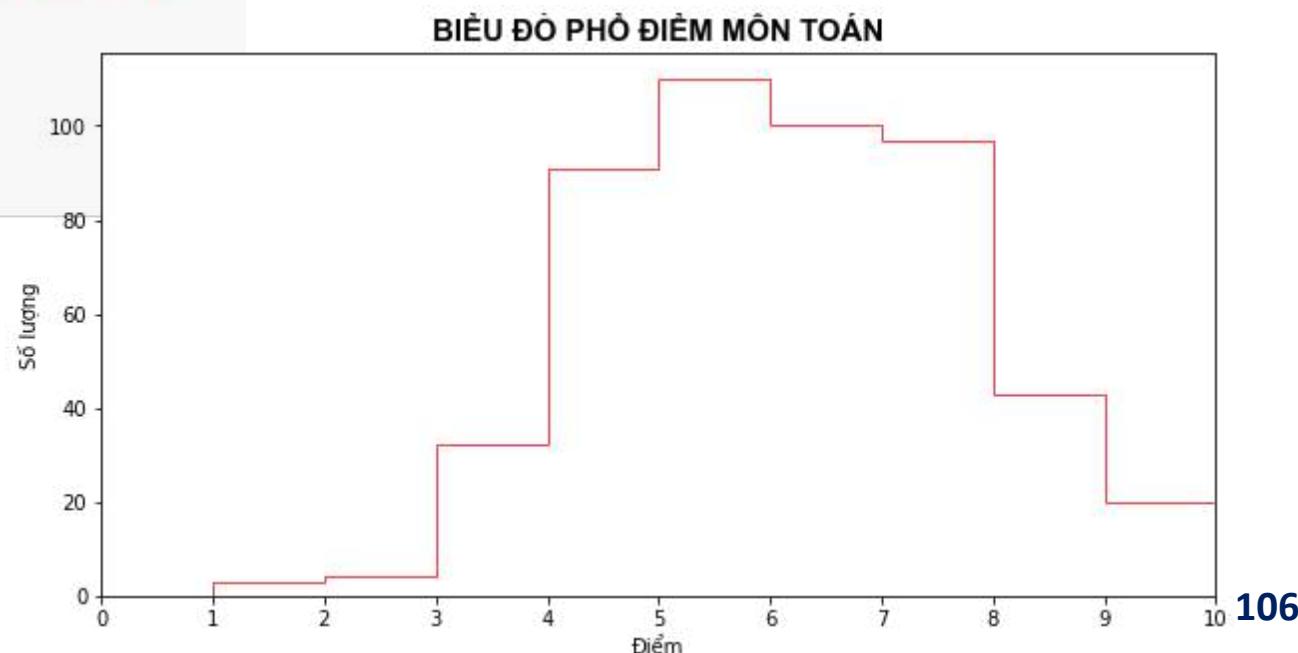
```
1 plt.figure(figsize = (10,5))
2 #Vẽ biểu đồ Histogram:
3 plt.hist(data['Math'],      #Dữ liệu thống kê tần suất
4           color='g',        #Màu của biểu đồ
5           alpha=0.75,       #Độ trong suốt của biểu đồ
6           bins=9)          #Số lượng class muốn thống kê
7
8 plt.title('BIỂU ĐỒ PHỒ ĐIỂM MÔN TOÁN',
9            fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})
10 plt.grid(ls='--')
11 plt.xlabel('Điểm')
12 plt.ylabel('Số lượng')
13 plt.xlim([0,10])
14 plt.xticks([0,1,2,3,4,5,6,7,8,9,10])
15 plt.show()
```



Biểu đồ tần suất (Histogram chart)

Cú pháp: plt.hist (x)

```
1 # Tuỳ chỉnh Histogram:  
2 plt.figure(figsize = (10,5))  
3 #Vẽ biểu đồ Histogram:  
4 plt.hist(data['Math'],  
5           alpha=0.75,  
6           bins=9,  
7           color='r',  
8           histtype='step') #Kiểu biểu diễn Histogram  
9  
10 plt.title('BIỂU ĐỒ PHỔ ĐIỀM MÔN TOÁN',  
11            fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})  
12 #plt.grid(ls='--')  
13 plt.xlabel('Điểm')  
14 plt.ylabel('Số lượng')  
15 plt.xlim([0,10])  
16 plt.xticks([0,1,2,3,4,5,6,7,8,9,10])  
17 plt.show()
```



Biểu đồ tần suất (Histogram chart)

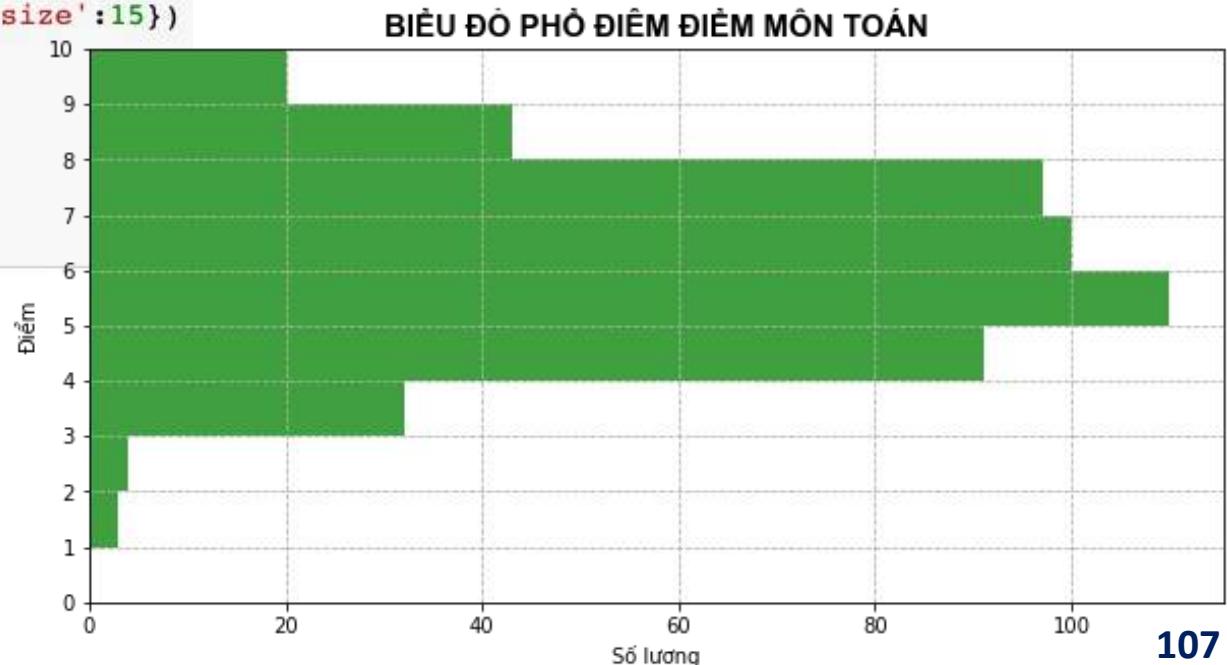


VINBIGDATA VINGROUP

Academy
Vietnam

Cú pháp: plt.hist (x)

```
1 plt.figure(figsize = (10,5))
2 #Vẽ biểu đồ Histogram:
3 plt.hist(data['Math'],
4           facecolor='g',
5           alpha=0.75,
6           bins=9,
7           orientation='horizontal') #Biểu diễn Histogram nằm ngang
8
9
10 plt.title('BIỂU ĐỒ PHỒ ĐIỂM MÔN TOÁN',
11            fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})
12 plt.grid(ls='--')
13 plt.ylabel('Điểm')
14 plt.xlabel('Số lượng')
15 plt.ylim([0,10])
16 plt.yticks([0,1,2,3,4,5,6,7,8,9,10])
17 plt.show()
```



Thiết lập bins

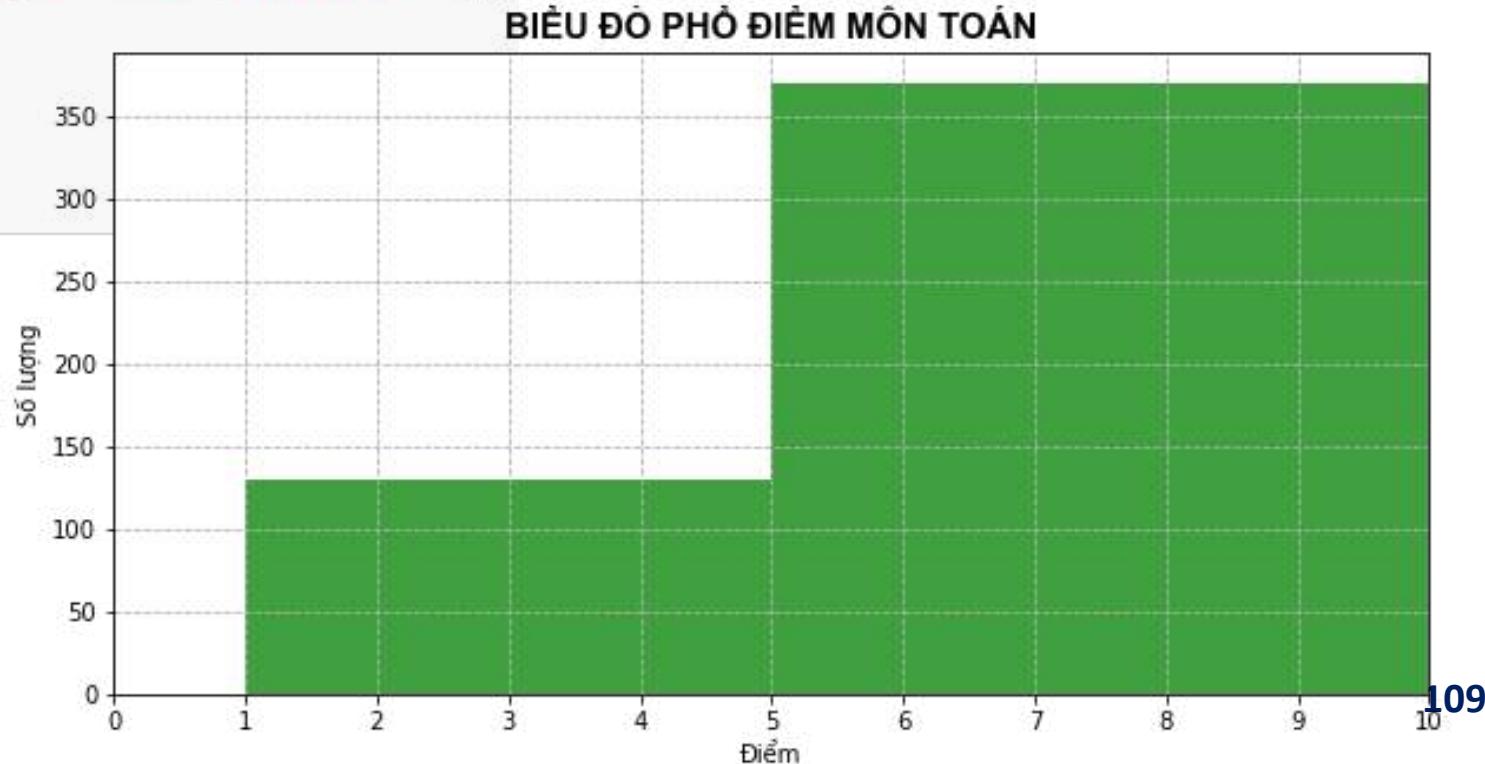
Biểu đồ tần suất (Histogram chart)



VINBIGDATA VINGROUP

Academy
Vietnam

```
1 plt.figure(figsize = (10,5))
2 #Vẽ biểu đồ Histogram:
3 t = plt.hist(data['Math'],
4               facecolor='g',
5               alpha=0.75,
6               bins=[1,5,10]) #Tách thành 2 nhóm theo ngưỡng thiết lập
7 #nhóm 1: 1--<5
8 #nhóm 2: 5--10
9
10 plt.title('BIỂU ĐỒ PHỒ ĐIỂM MÔN TOÁN',
11            fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})
12 plt.grid(ls='--')
13 plt.xlabel('Điểm')
14 plt.ylabel('Số lượng')
15 plt.xlim([0,10])
16 plt.xticks([0,1,2,3,4,5,6,7,8,9,10])
17 plt.show()
```



Biểu đồ tần suất (Histogram chart)



VINBIGDATA VINGROUP

Academy
Vietnam

```
1 plt.figure(figsize = (10,5))
2 #Vẽ biểu đồ Histogram:
3 t = plt.hist(data[ 'Math'],
4               facecolor='g',
5               alpha=0.75,
6               bins=[1,5,8,10]) #Tách thành 3 nhóm theo ngưỡng thiết lập
7 #nhóm 1: 1--<5
8 #nhóm 2: 5--<8
9 #Nhóm 3: 8--10
10
11 plt.title('BIỂU ĐỒ PHỒ ĐIỂM MÔN TOÁN',
12            fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})
13 plt.grid(ls='--')
14 plt.xlabel('Điểm')
15 plt.ylabel('Số lượng')
16 plt.xlim([0,10])
17 plt.xticks([0,1,2,3,4,5,6,7,8,9,10])
18 plt.show()
```





Bar chart & Histogram chart



Bar chart & Histogram chart

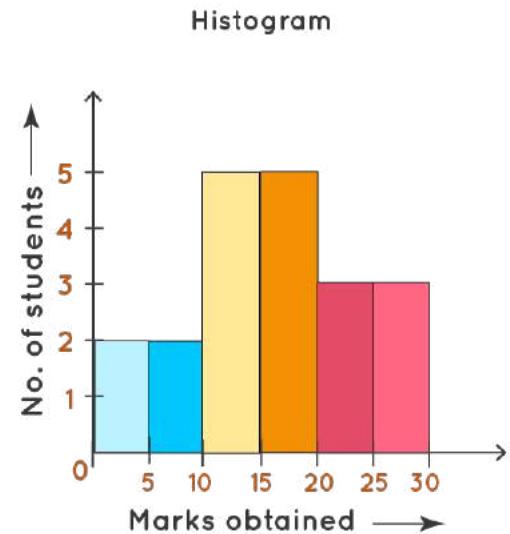
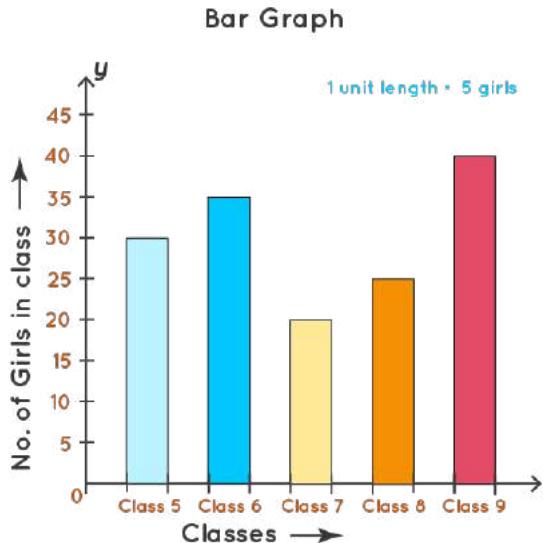


VINBIGDATA VINGROUP

Academy
Vietnam

Difference Between Bar Chart and Histogram

 cuemath
THE MATH EXPERT



Bar Graph

Equal space between every two consecutive bars.

X-axis can represent anything.

Histogram

No space between two consecutive bars. They should be attached to each other.

X-axis should represent only continuous data that is in terms of numbers.



9. Biểu đồ hộp (Boxplot)

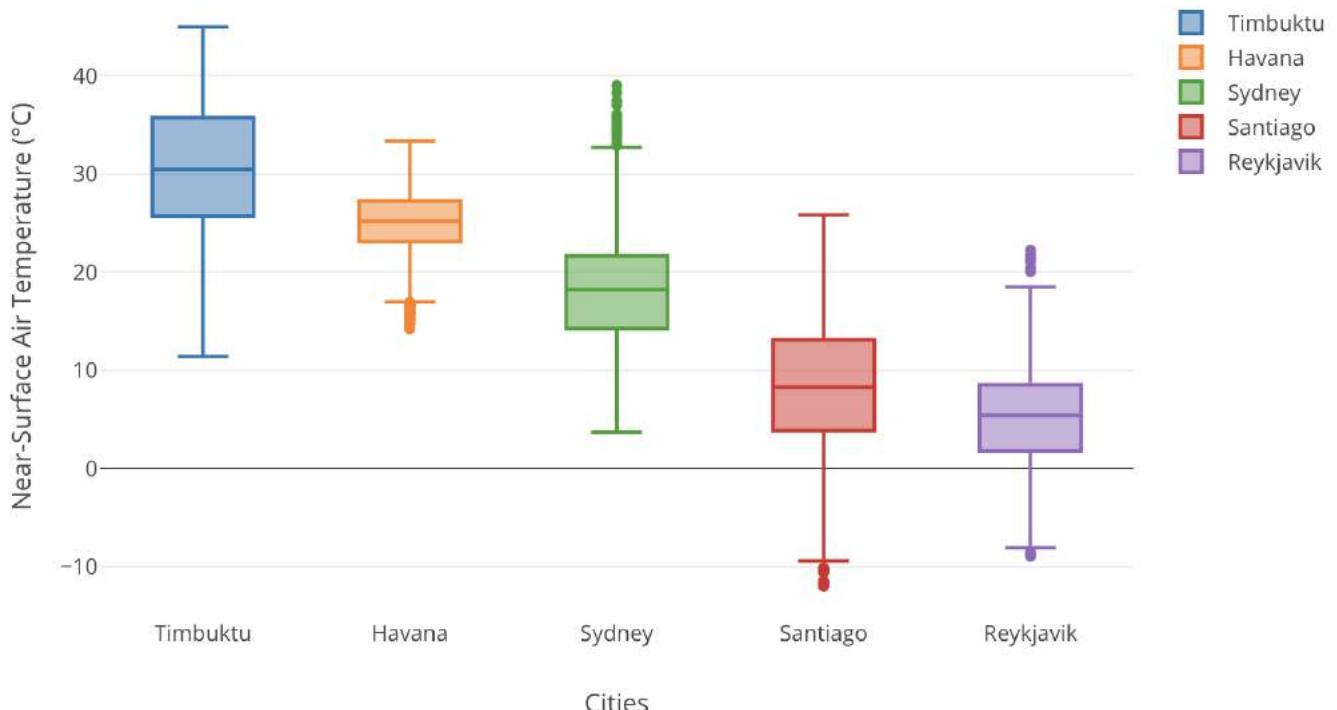


Boxplot

- **Boxplot** là một dạng biểu đồ thể hiện phân bố dữ liệu của các thuộc tính số thông qua các “*tứ phân vị*” và được giới thiệu lần đầu bởi John Tukey vào năm 1970.
- **Tứ phân vị** là một khái niệm trong thống kê dùng để mô tả sự phân bố và sự phân tán của tập dữ liệu, gồm 3 giá trị: Q1, Q2 và Q3 chia tập dữ liệu thành 4 phần bằng nhau.

Box plots

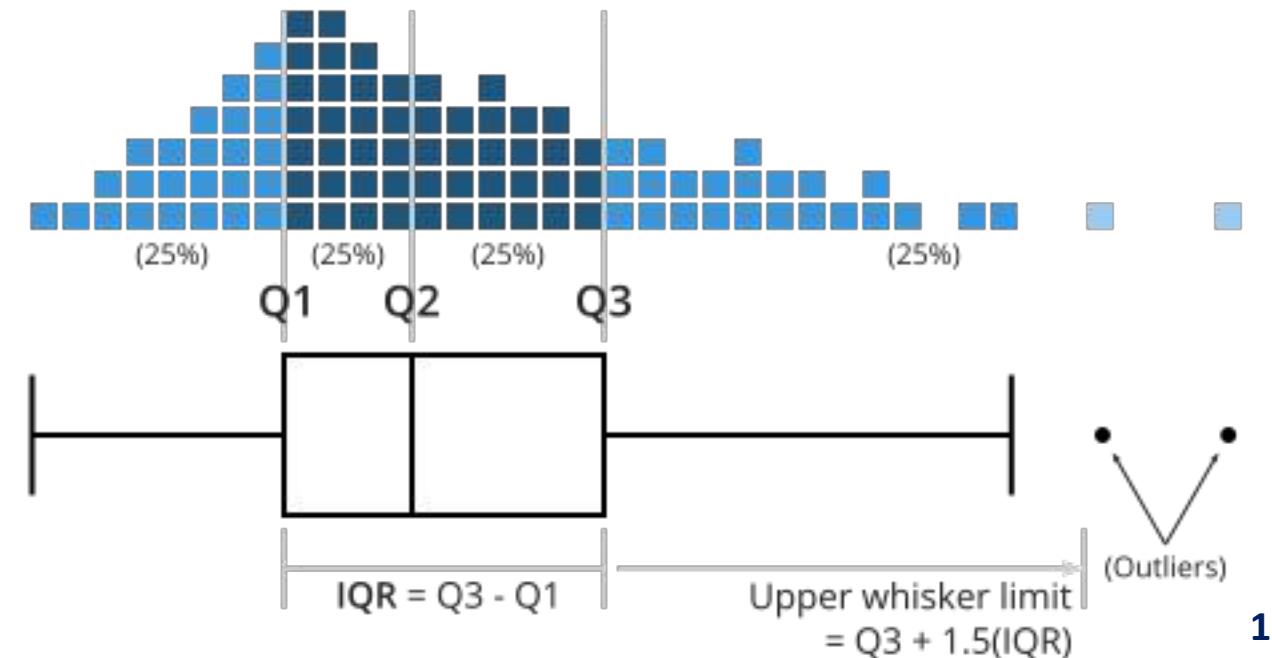
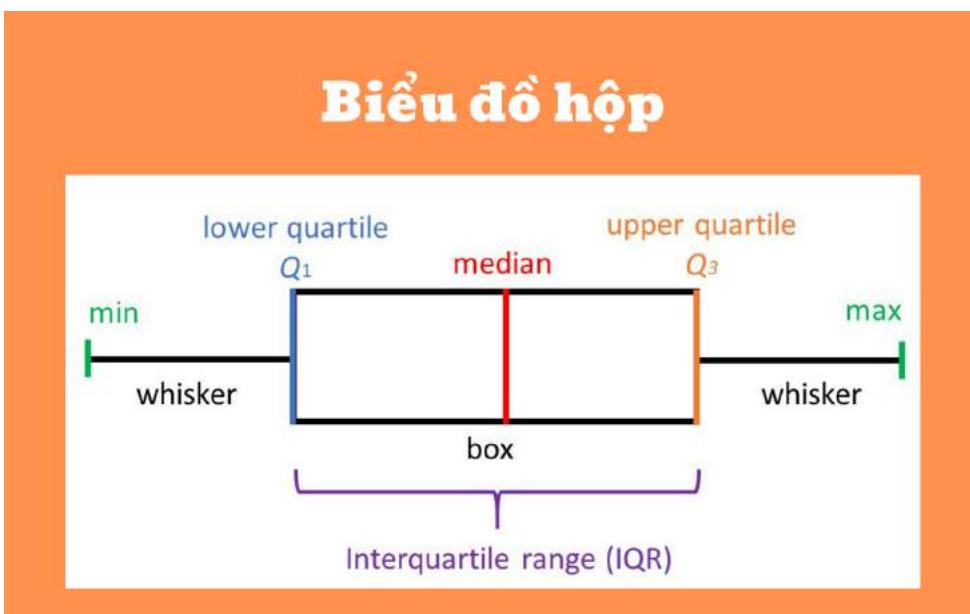
- Boxplot thể hiện các phân bố dữ liệu, nghĩa là giúp chúng ta biết được độ dàn trải của các điểm dữ liệu như thế nào, dữ liệu có đối xứng không, phân bố rộng hay hẹp, giá trị nhỏ nhất, lớn nhất và các điểm ngoại lệ.



Boxplot

Biểu đồ Boxplot thể hiện 5 thông số:

- **First quartile (Q1):** Trung vị giữa **Median** và **phần tử nhỏ nhất** trong tập dữ liệu. Còn gọi là **25th Percentile**.
- **Median (Q2):** Trung vị của tập dữ liệu, tức là giá trị ở phần tử giữa. (50%)
- **Third quartile (Q3):** Trung vị giữa **Median** và **phần tử lớn nhất** trong tập dữ liệu. Còn gọi là **75th Percentile**.
- **Minimum:** Phần tử nhỏ nhất không phải ngoại lệ.
- **Maximum:** Phần tử lớn nhất không phải là ngoại lệ.

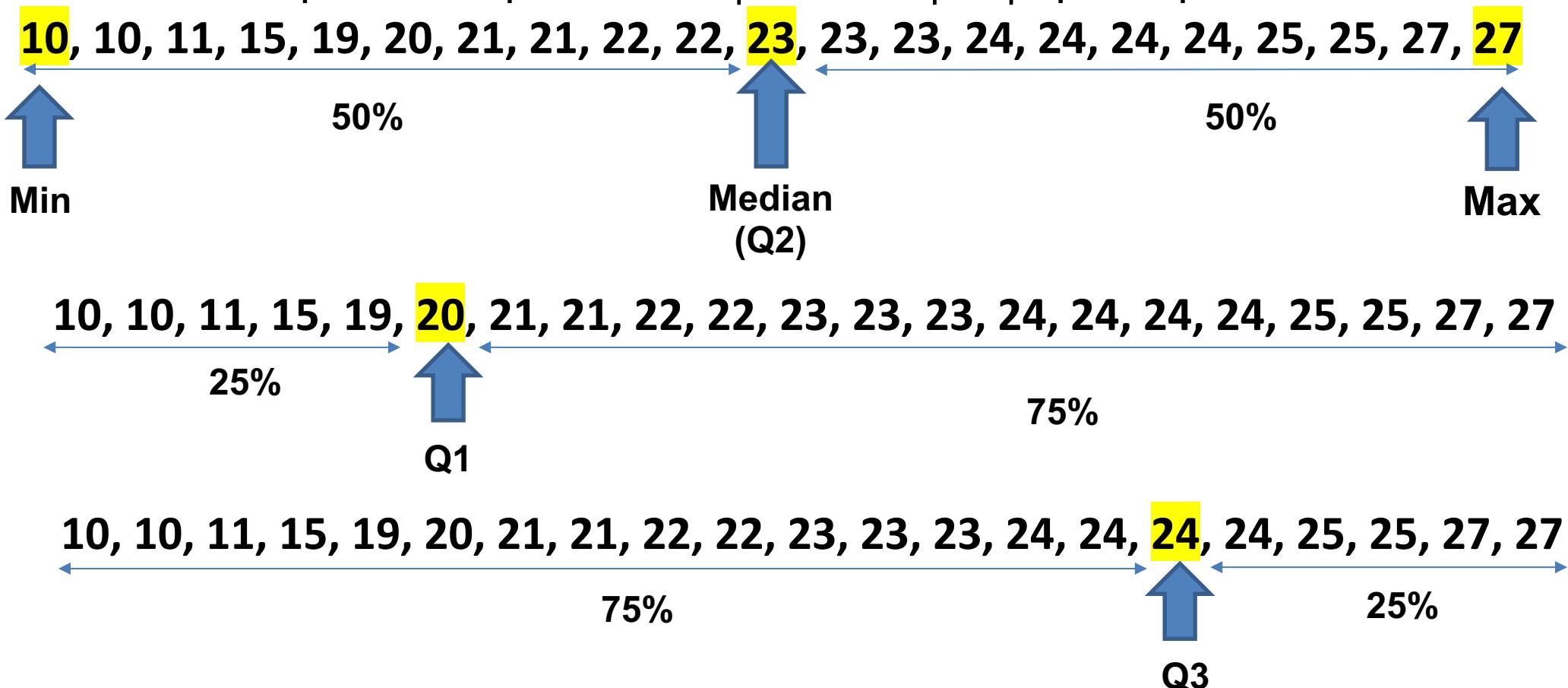


Cách xây dựng biểu đồ Boxplot

- Một nhà hàng ghi lại khoảng cách của 21 khách hàng đi từ nhà đến nhà hàng như sau:

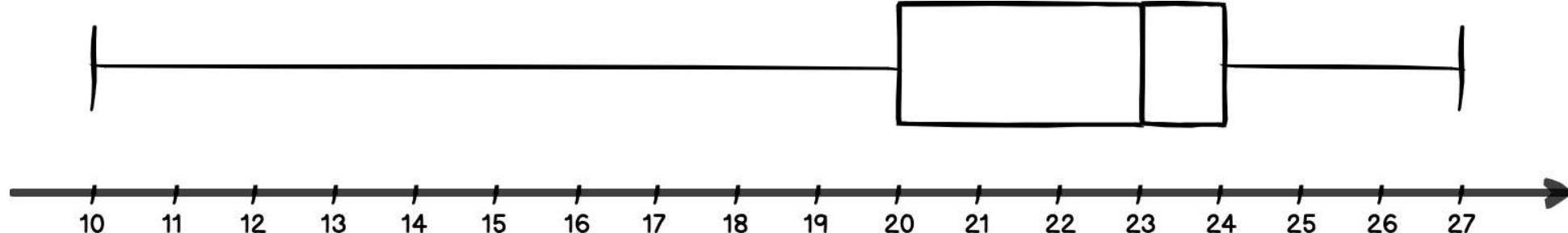
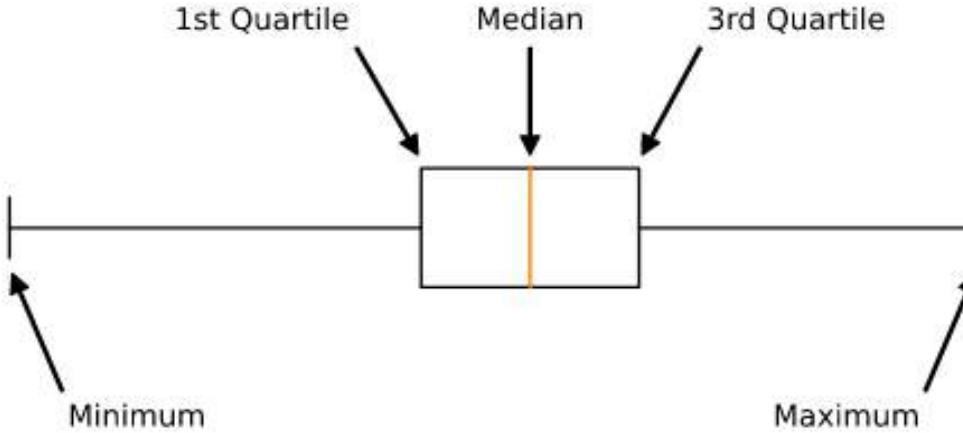
24, 10, 23, 11, 21, 22, 23, 15, 23, 21, 20, 25, 22, 24, 24, 10, 24, 25, 27, 27, 19

- Trước tiên để tìm được các số liệu để vẽ Boxplot cần sắp xếp lại dữ liệu:



Cách xây dựng biểu đồ Boxplot

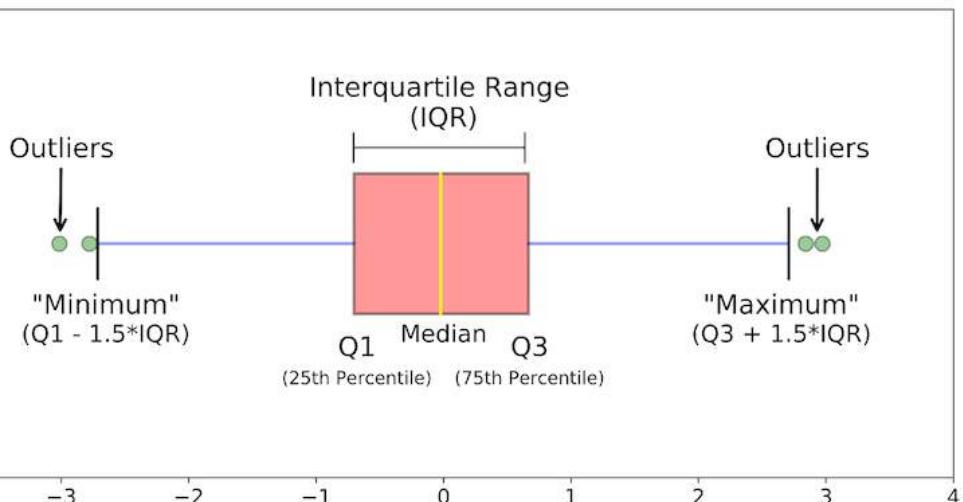
- Median = 23
- Q1 = 20
- Q3 = 24
- Min = 10
- Max = 27



Cách xây dựng biểu đồ Boxplot

- Trong thống kê, một ngoại lệ (outlier) là một điểm dữ liệu khác biệt đáng kể so với các quan sát khác. Một ngoại lệ có thể là do sự thay đổi trong phép đo hoặc là lỗi và thông thường được loại trừ khỏi tập dữ liệu bởi nó có thể gây ra vấn đề nghiêm trọng trong phân tích thống kê.
- Để tìm ngoại lệ, ta dùng thêm khái niệm **IQR**. **IQR (Interquartile Range)** là một khái niệm trong thống kê mô tả, dùng đo lường độ phân tán của dữ liệu và được tính toán bằng công thức:

$$\text{IQR} = Q_3 - Q_1$$



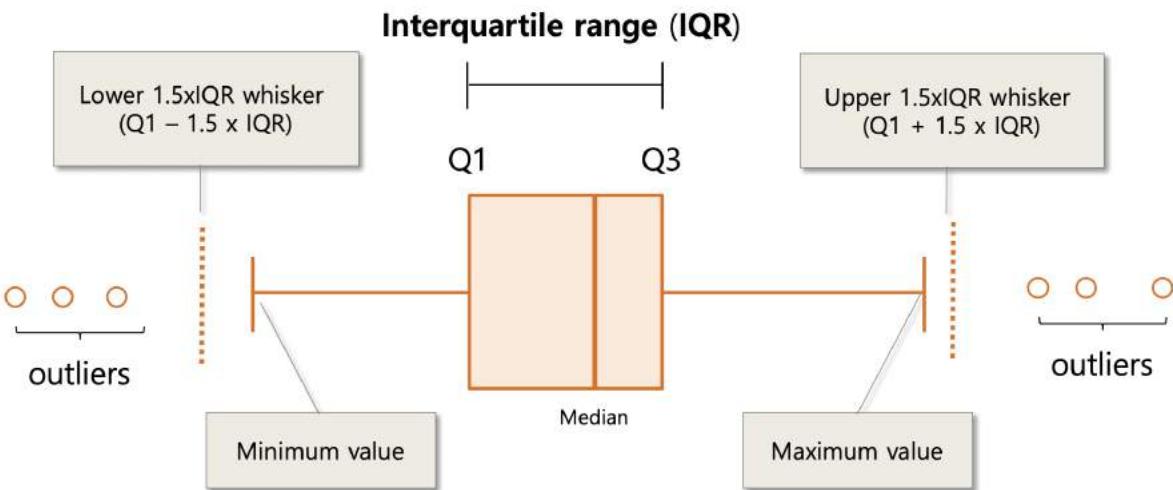
Điểm ngoại lệ sẽ là những điểm:
nhỏ hơn $< Q_1 - 1.5 * \text{IQR}$ và lớn hơn $> Q_3 + 1.5 * \text{IQR}$.

Cách xây dựng biểu đồ Boxplot

- $Q1 = 20$
- $Q3 = 24$ $\rightarrow IQR = Q3 - Q1 = 24 - 20 = 4$

Như vậy ta xác định được **Minimum** mới và **Maximum** mới như sau:

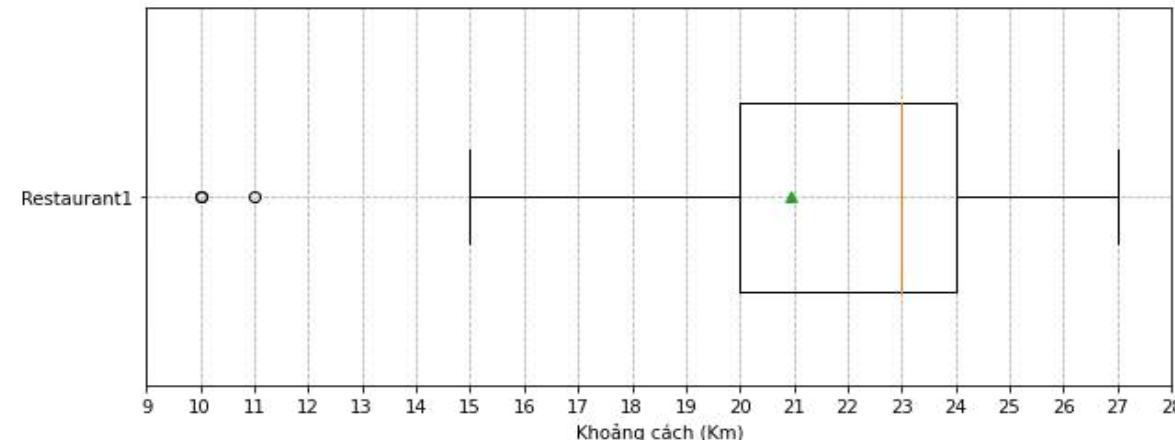
- $Q1 - 1.5 \times IQR = 20 - 1.5 \times 4 = 14$ (Trong dữ liệu giá trị nhỏ nhất ≥ 14 gần nhất là 15 $\rightarrow \text{Minimum} = 15$)
- $Q3 + 1.5 \times IQR = 24 + 1.5 \times 4 = 30$ (Trong dữ liệu giá trị lớn nhất ≤ 30 $\rightarrow \text{Maximum} = 27$)



10, 10, 11, **15**, 19, 20, 21, 21, 22, 22, 23, 23, 23, 24, 24, 24, 24, 24, 25, 25, 27, **27**

↑
Minimum

↑
Maximum



Cách xây dựng biểu đồ Boxplot

Một nhà hàng ghi lại độ tuổi của 37 khách hàng đến nhà hàng như sau:

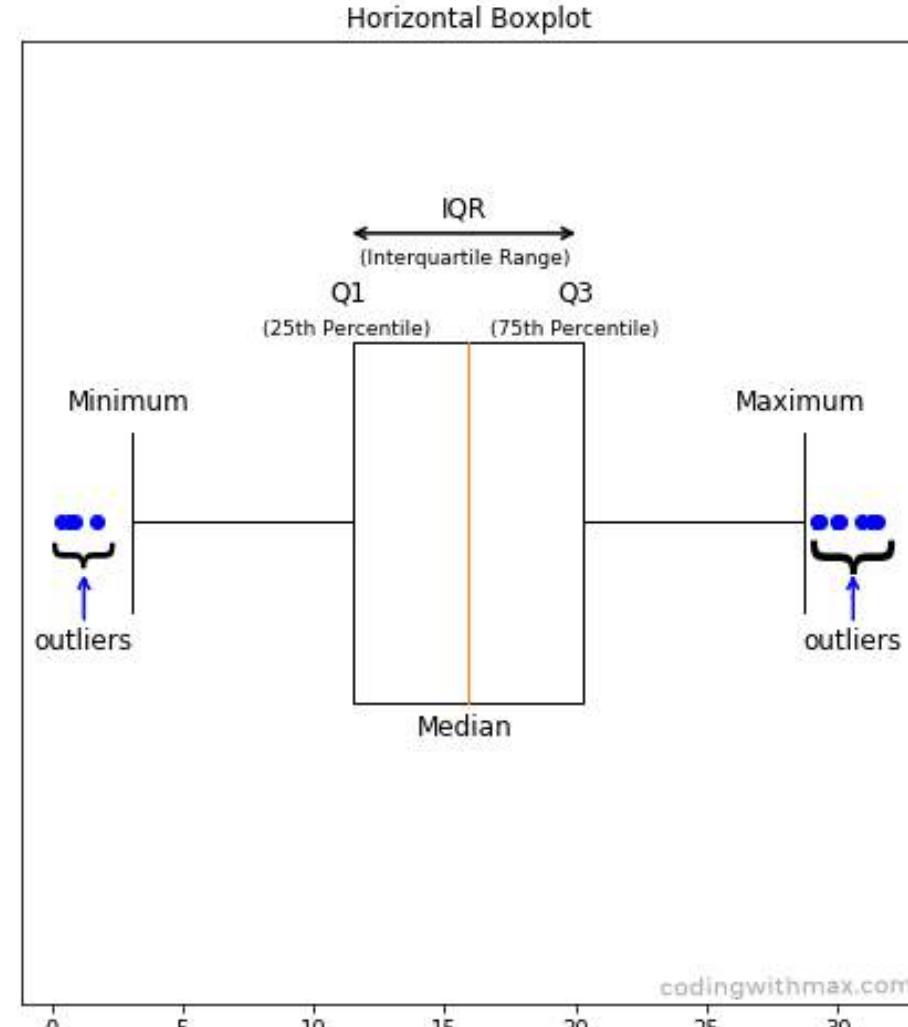
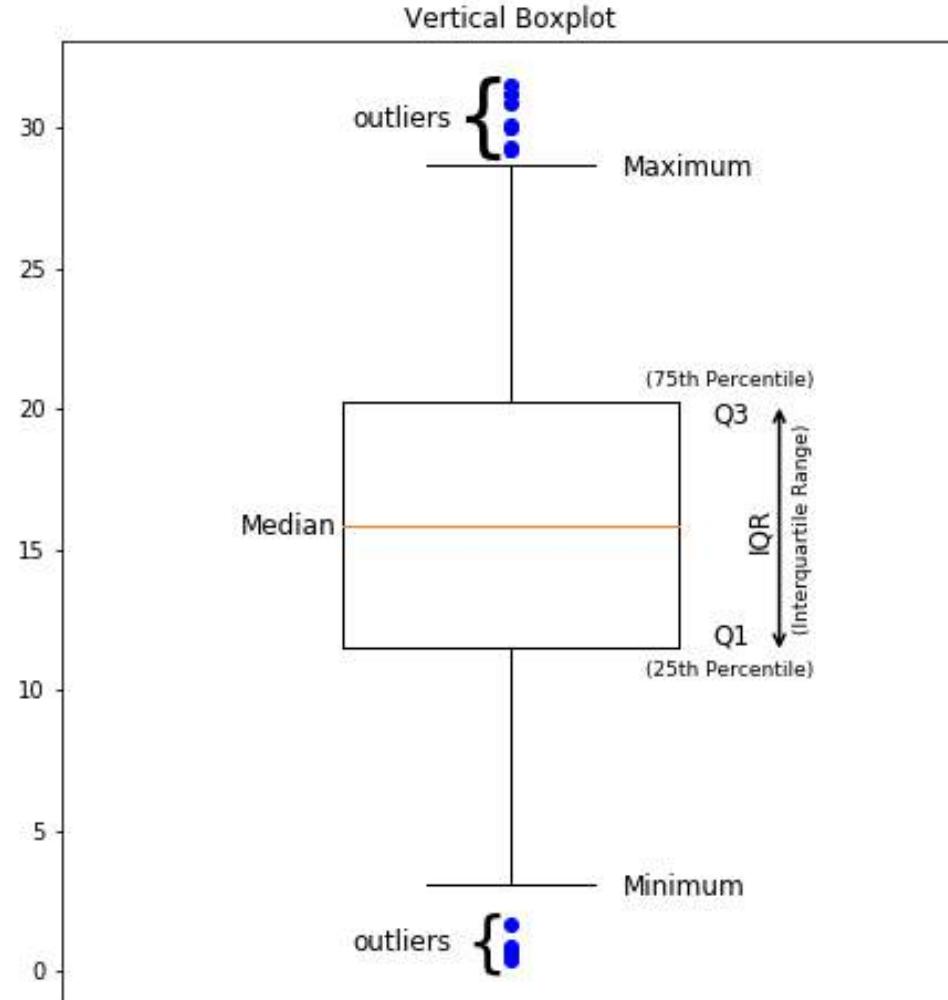
**24,32,40,26,23,31,39,21,22,23,35,23,21,20,8,22,24,24,40,24,25,27,37,25,24,28,33,29,
30,50,30,31,38,23,38,27,29**

Thực hiện Vẽ biểu đồ Boxplot cho dữ liệu trên?

BEST
PRACTICE



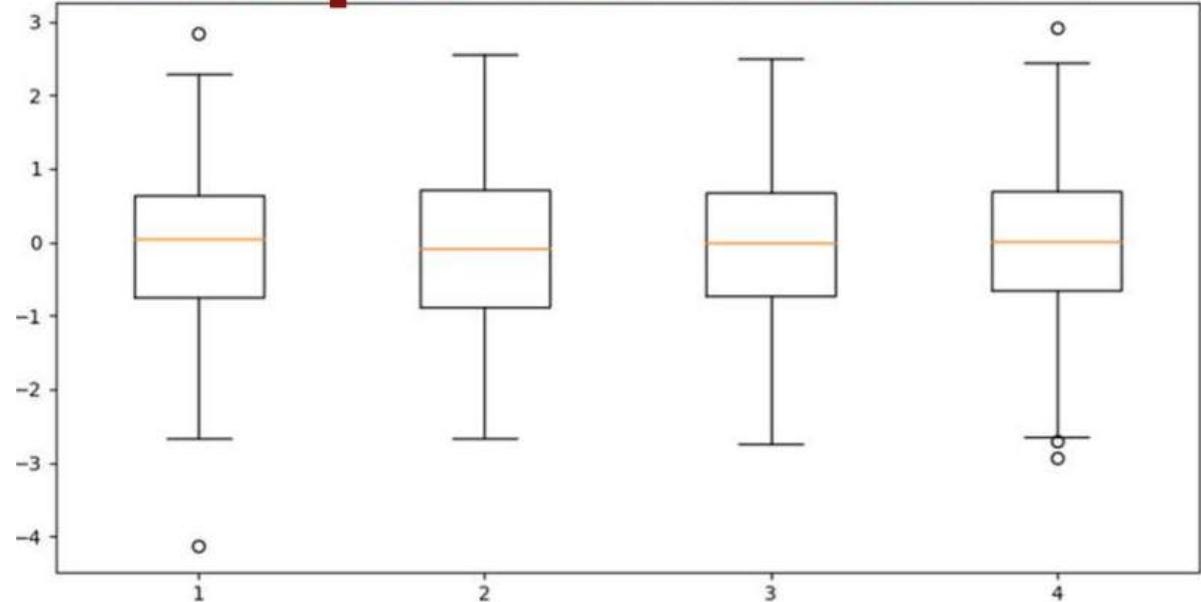
Các loại Biểu đồ Boxplot



Boxplot với Matplotlib

Tập dữ liệu **Data_score.csv** lưu trữ dữ liệu 3 môn Math, Science, History của 500 học sinh

Matplotlib BoxPlot

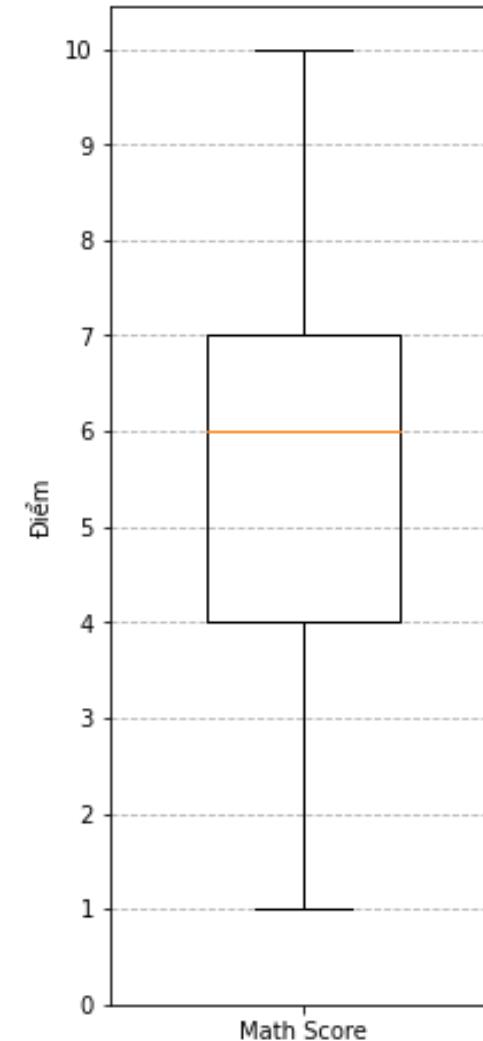


	A	B	C
1	Math	Science	History
2	7	10	4
3	4	9	3
4	4	8	2
5	4	8	3
6	4	6	3
7	6	8	1
8	5	10	7
9	5	10	5
10	5	7	4
11	6	8	3
12	7	9	3
13	6	8	2
14	5	10	7
15	5	9	6
16	7	5	2
17	8	10	3
18	4	9	7
19	5	7	2
20	5	9	6

Boxplot với Matplotlib

Cú pháp: plt.boxplot (x)

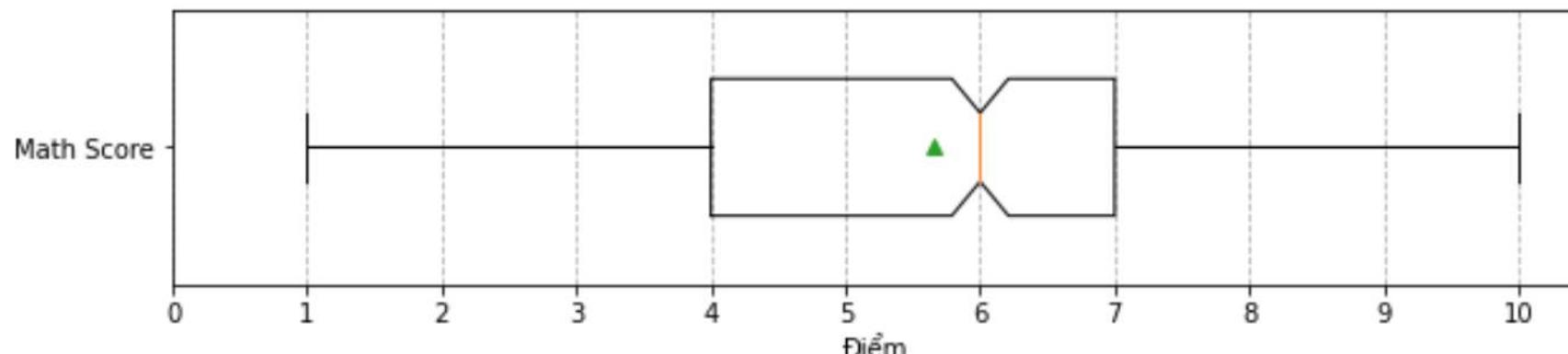
```
1 plt.figure(figsize = (3,8))
2 #Vẽ biểu đồ hộp Boxplot:
3 plt.boxplot(data[ 'Math' ],
4             widths=[0.5],
5             labels=[ 'Math Score' ]) #Dữ liệu đầu vào
6                                     #Độ rộng của Box
7                                     #Nhãn của Boxplot
8
9 plt.grid( axis = 'y', ls='--' )
10 plt.ylabel('Điểm')
11 plt.yticks([0,1,2,3,4,5,6,7,8,9,10])
12 plt.show()
```



Boxplot với Matplotlib

Cú pháp: plt.boxplot (x)

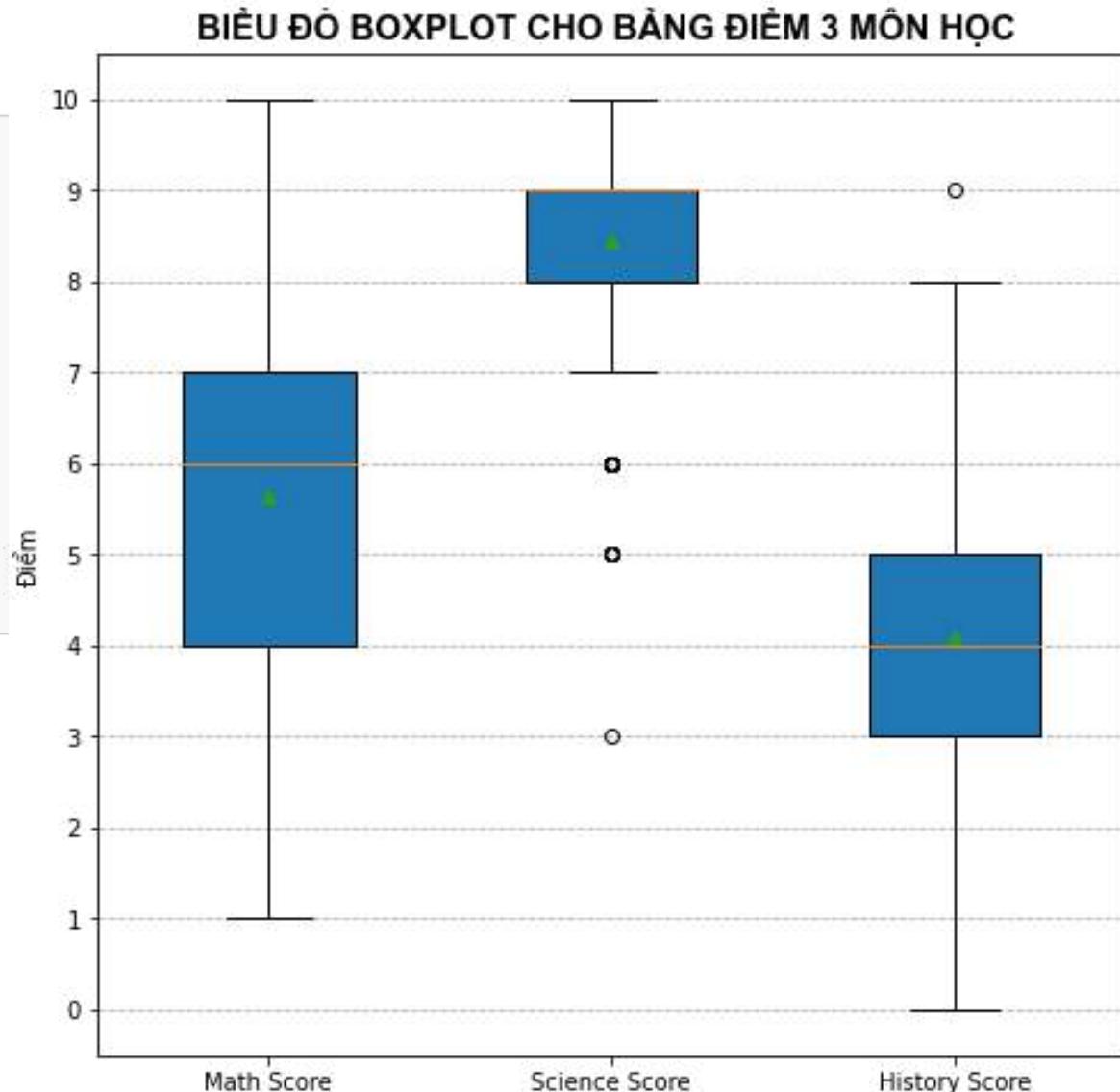
```
1 plt.figure(figsize = (10,2))
2 #Vẽ biểu đồ hộp Boxplot:
3 plt.boxplot(data['Math'],
4             labels=['Math Score'],
5             widths=[0.5],
6             notch=True,           #Tạo thóp tại vị trí Median
7             vert=False,          #Hiển thị boxplot theo chiều ngang False / chiều dọc True
8             showmeans=True)      #Hiển thị vị trí có giá trị trung bình (mean)
9
10 plt.grid( axis = 'x', ls='--')
11 plt.xlabel('Điểm')
12 plt.xticks([0,1,2,3,4,5,6,7,8,9,10])
13 plt.show()
```



Boxplot với Matplotlib

Cú pháp: plt.boxplot (x)

```
1 #Multiple boxplot trên cùng một plot:  
2 plt.figure(figsize = (8,8))  
3 plt.boxplot([data['Math'],data['Science'],data['History']],  
4             labels=['Math Score','Science Score','History Score'],  
5             widths=[0.5,0.5,0.5],  
6             showmeans=True,  
7             patch_artist=True)#Đỗ màu cho hộp  
8  
9 plt.title('BIỂU ĐỒ BOXPLOT CHO BẢNG ĐIỂM 3 MÔN HỌC ',  
10           fontdict={'fontname':'Arial','fontweight':'bold','fontsize':15})  
11 plt.grid( axis = 'y', ls='--')  
12 plt.ylabel('Điểm')  
13 plt.yticks([0,1,2,3,4,5,6,7,8,9,10])  
14  
15 plt.show()
```





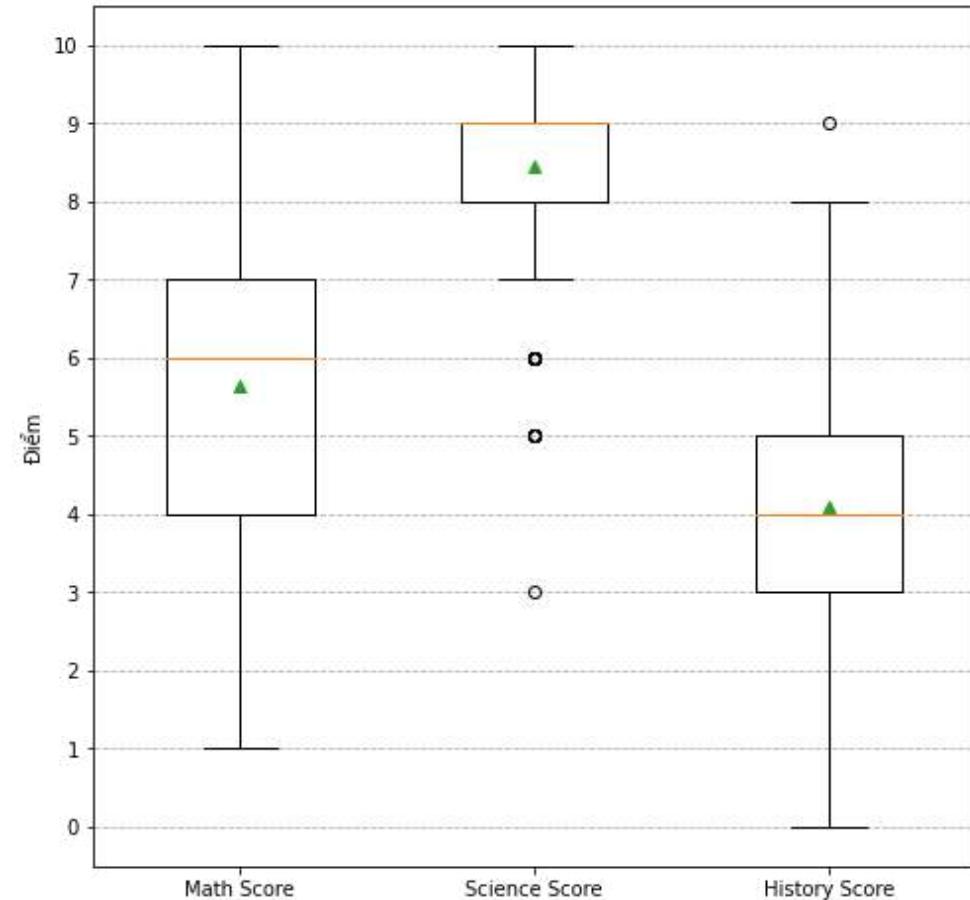
a. Boxplot sử dụng khi nào?



Boxplot sử dụng khi nào?

Khi muốn nhanh chóng có được cái nhìn tổng quan về dữ liệu. Boxplot cung cấp thông tin chung về một nhóm dữ liệu đối xứng? độ lệch, phương sai và giá trị ngoại lai? Có thể dễ dàng thấy phần lớn dữ liệu chính ở đâu.

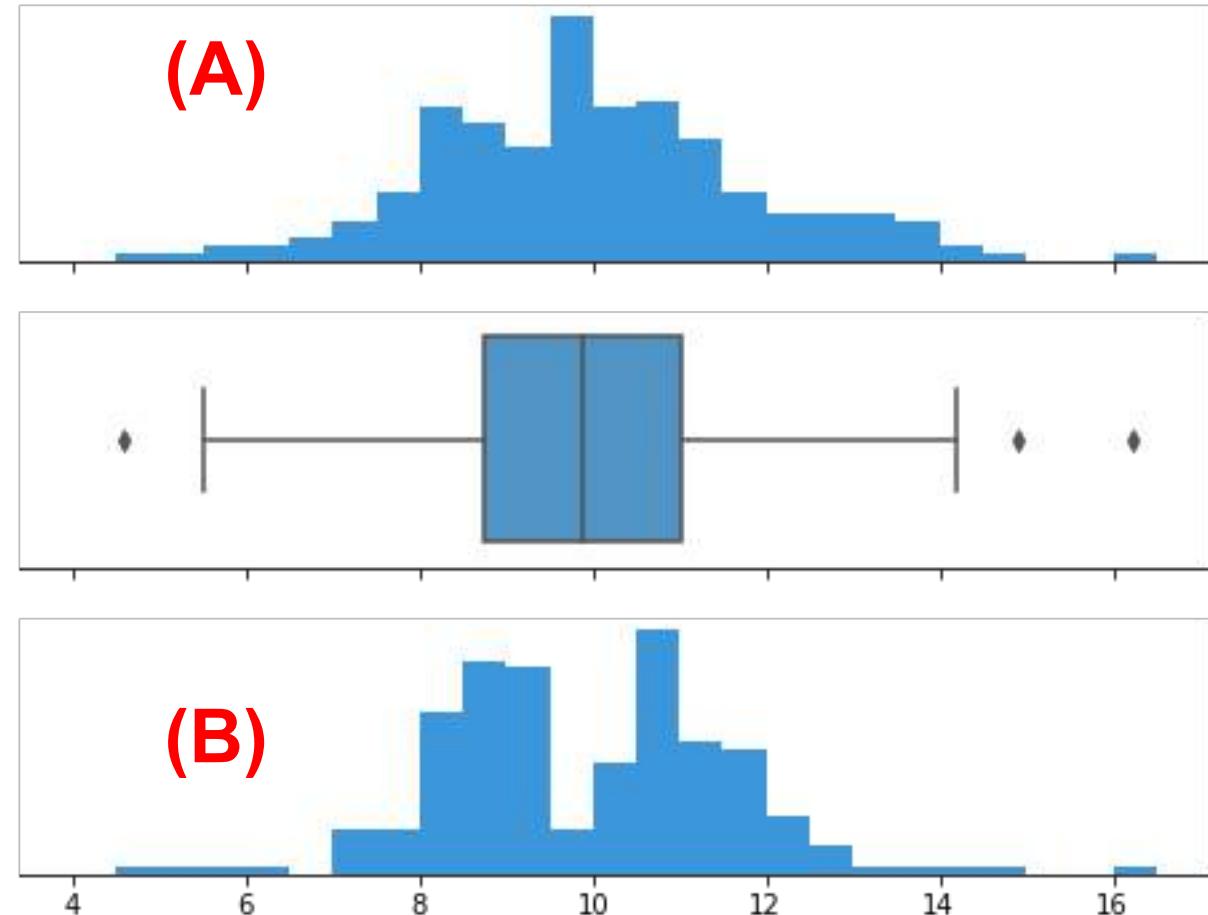
Boxplot được sử dụng tốt nhất khi cần phải thực hiện so sánh phân bố giữa các nhóm. Chúng nhỏ gọn trong việc tóm tắt dữ liệu và dễ dàng so sánh các nhóm thông qua vị trí của hộp và râu.



Boxplot sử dụng khi nào?

Hạn chế của Boxplot chính là sự đơn giản của biểu đồ, Boxplot không thể hiện được chi tiết về phân bố của dữ liệu.

Ví dụ như hình bên, Hai bộ dữ liệu (A) và (B) có phân bố dữ liệu khác nhau rất nhiều nhưng khi trực quan bằng Boxplot đều thu được một biểu đồ như nhau → Boxplot Không thể hiện được chi tiết của phân bố dữ liệu.

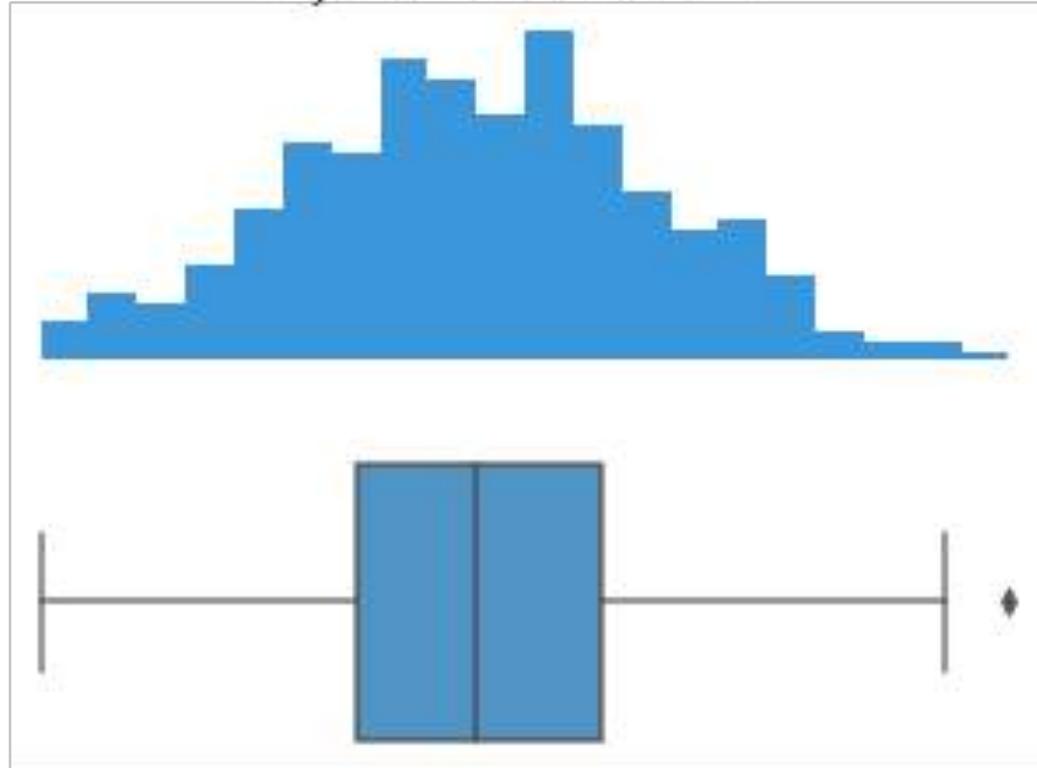




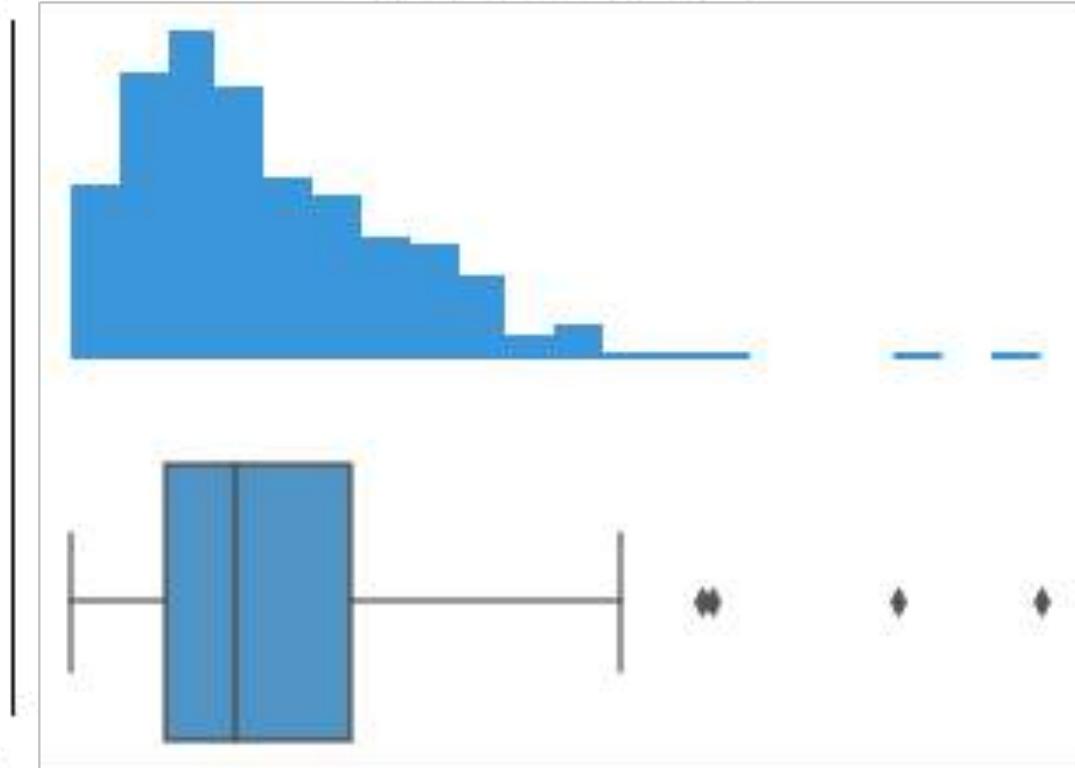
b. Boxplot & Histogram chart

Boxplot & Histogram chart

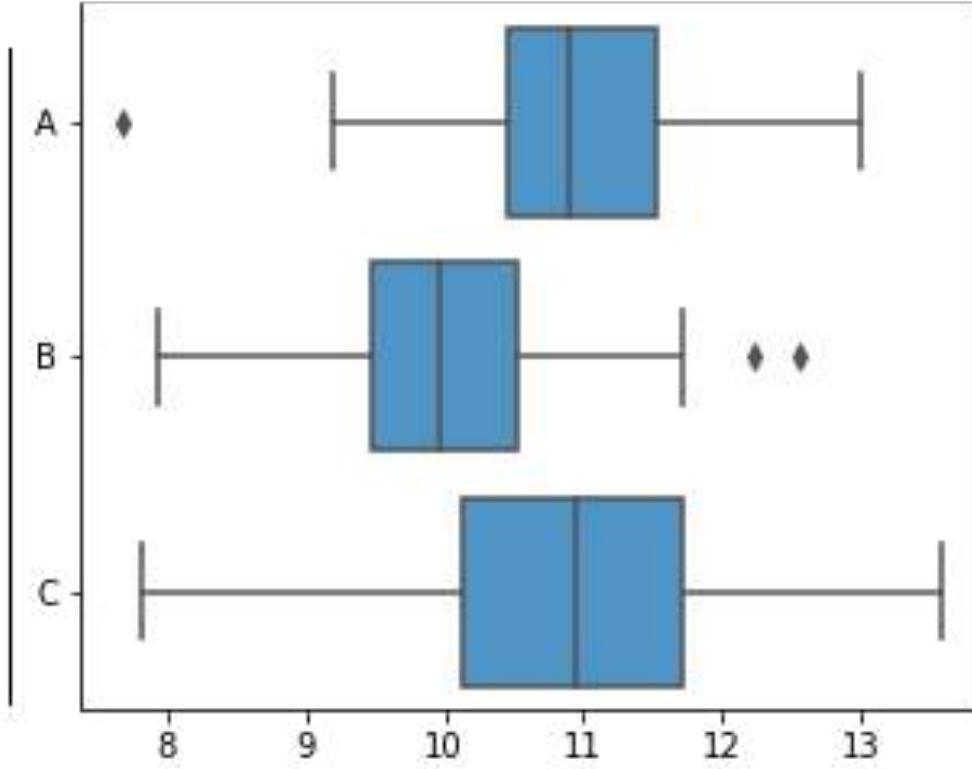
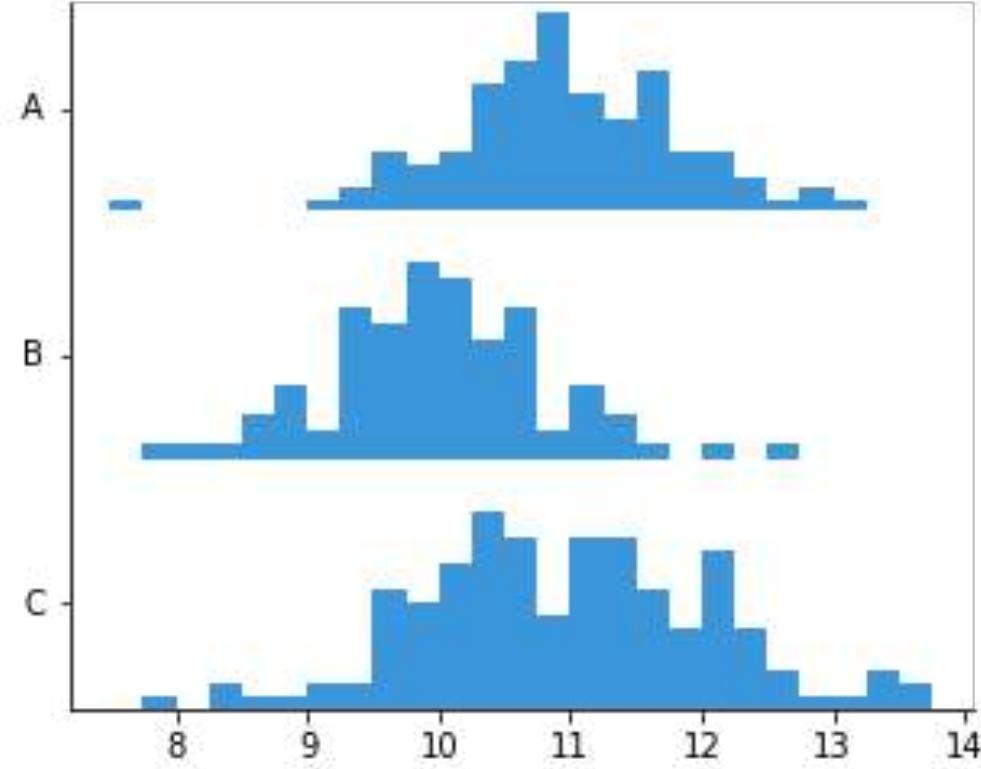
Symmetric distribution



Skewed distribution



Boxplot & Histogram chart





VINBIGDATA VINGROUP



AIS Academy Vietnam

Q & A
Thank you!