



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

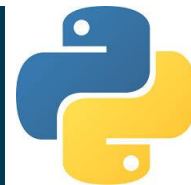
PYTHON FOR MACHINE LEARNING, DATA SCIENCE & DATA VISUALIZATION

Bài 4: Pandas



Phòng LT & Mạng

2019



Nội dung

1. Giới thiệu
2. Series
3. DataFrame
4. Panel

Giới thiệu

□ Pandas

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

- Là thư viện Python mã nguồn mở theo BSD-licensed, hiệu suất cao
- Cung cấp các công cụ phân tích dữ liệu, có cấu trúc dữ liệu dễ dàng sử dụng cho NNLT Python
- Python với Pandas được sử dụng trong nhiều lĩnh vực bao gồm khoa học, kinh tế, phân tích thống kê...
- **Cài đặt: `pip install pandas`**

Giới thiệu

- ❑ Pandas có 3 kiểu cấu trúc dữ liệu:
 - Series
 - DataFrame
 - Panel
- ❑ Các cấu trúc dữ liệu này được xây dựng trên Numpy array, có tốc độ nhanh.

Giới thiệu

□ Ưu điểm

- Hỗ trợ đa dạng dữ liệu
- Tích hợp dữ liệu
- Chuyển đổi dữ liệu
- Hỗ trợ dữ liệu time-series
- Thống kê mô tả

Giới thiệu

❑ Kích thước (Dimension) & mô tả

- Có thể xem Data Frame là một container của series, Panel là một container của Data Frame

Data Structure	Dimension	Mô tả
Series	1	1D array được gán nhãn đồng nhất, có kích thước không thay đổi
Data Frame	2	2D array được gán nhãn, cấu trúc bảng có kích thước có thể thay đổi, các cột có khả năng không đồng nhất
Panel	3	3D array được gán nhãn, kích thước có thể thay đổi

Ghi chú: DataFrame được sử dụng rộng rãi và là một trong những cấu trúc dữ liệu quan trọng nhất. Panel rất ít được sử dụng

Nội dung

1. Giới thiệu
2. Series
3. DataFrame

Series

- ❑ Series là mảng một chiều có gán nhãn, có khả năng chứa các phần tử với kiểu dữ liệu khác nhau (integer, string, float, python objects...).
- ❑ Các axis label của Series được gọi là index.

Series

❑ Phương thức tạo series

`pandas.Series(data, index, dtype, copy)`

● Trong đó:

- Data: dữ liệu từ nhiều dạng khác nhau như ndarray, list, constants
- Index: giá trị của index là duy nhất và số lượng index = số lượng phần tử. Mặc định sẽ có giá trị trong `np.arange(n)` nếu người dùng không tự tạo index.
- Dtype: kiểu dữ liệu (tùy chọn)
- Copy: tạo bản copy (tùy chọn)

Series

□ Tạo series

- Khởi tạo series

```
s = pd.Series()  
print(s)
```

```
Series([], dtype: float64)
```

- Tạo series từ ndarray, với index tự động

```
import numpy as np  
nd_array_1 = np.array(['a', 'b', 'c', 'd'])  
s1 = pd.Series(nd_array_1)  
print(s1)
```

```
0    a  
1    b  
2    c  
3    d  
dtype: object
```

Series

- Tạo series từ ndarray, với index do coder tạo

```
# Create a Series from ndarray
print(nd_array_1)
s2 = pd.Series(nd_array_1, index=[100,101,102,103])
print(s2)
```

```
['a' 'b' 'c' 'd']
100    a
101    b
102    c
103    d
dtype: object
```

- Tạo series từ dictionary

```
# Create a Series from dict
dic_1 = {'a' : 0., 'b' : 1., 'c' : 2.}
s3 = pd.Series(dic_1)
print(s3)
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

Series

- Tạo series từ một giá trị

```
# Create a Series from Scalar: index phải được thiết lập  
s4 = pd.Series(10, index=[0, 1, 2, 3])  
print(s4)
```

```
0    10  
1    10  
2    10  
3    10  
dtype: int64
```

Series

❑ Thao tác trên series

- Truy xuất phần tử trên series: theo index (lable)

```
# get element value from Series
s5 = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print(s5)
print('Cách 1:',s5['a'])
print('Cách 2:',s5[0])
print('Từ đều đến cận 2:')
print(s5[:2])
print('Từ 1 đến cận 3:')
print(s5[1:3])
print('Từ -3:')
print(s5[-3:])
print('Truy xuất nhiều phần tử theo index:')
print(s5[['a','c','d']])
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
Cách 1: 1
Cách 2: 1
Từ đều đến cận 2:
a    1
b    2
dtype: int64
Từ 1 đến cận 3:
b    2
c    3
dtype: int64
Từ -3:
c    3
d    4
e    5
dtype: int64
Truy xuất nhiều phần tử theo index:
a    1
c    3
d    4
dtype: int64
```

Series

- Cập nhật phần tử

```
# cập nhật phần tử  
s3 = pd.Series([12,21,11],index = ['idx1','idx2','idx3'])  
s3['idx3'] = 100  
print(s3)
```

```
idx1    12  
idx2    21  
idx3   100  
dtype: int64
```

Series

• Thêm phần tử

```
#Thêm phần tử vào Series - tương tự dictionary
print('Trước khi thêm:\n',s)
s['f'] = 6
print('Sau khi thêm:\n', s)
```

Trước khi thêm:

```
a    5
b    2
c    3
d    4
e    5
dtype: int32
Sau khi thêm:
a    5
b    2
c    3
d    4
e    5
f    6
dtype: int64
```

• Xóa phần tử

```
#Xóa phần tử
print(s)
t= s.drop(['a','e'])
print(t)
tt= s.drop('a')
print(tt)
```

```
a    5
b    2
c   10
d    4
e    5
f    6
dtype: int64
b    2
c   10
d    4
f    6
dtype: int64
b    2
c   10
d    4
e    5
f    6
dtype: int64
```

Series

- Lấy list của row **axis label**: dùng `axes`

```
s1 = pd.Series(np.random.randn(5))
print(s1)
```

```
0    -0.011797
1     1.430572
2     0.729355
3    -0.060106
4    -0.229390
dtype: float64
```

```
print("The axes are:")
print(s1.axes)
```

```
The axes are:
[RangeIndex(start=0, stop=5, step=1)]
```

```
s2 = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print(s2)
print("The axes are:")
print(s2.axes)
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
The axes are:
[Index(['a', 'b', 'c', 'd', 'e'], dtype='object')]
```


Series

- Lấy dtype của object: dùng dtype

```
print(s1)
print(s1.dtype)
```

```
0    -0.290882
1     2.262629
2    -0.042616
3     0.640973
4    -0.722389
dtype: float64
float64
```

```
print(s2)
print(s2.dtype)
```

```
a     1
b     2
c     3
d     4
e     5
dtype: int64
int64
```

- Kiểm tra series rỗng: dùng empty

```
print(s1.empty)
```

```
False
```

```
print(s2.empty)
```

```
False
```

Series

- Lấy số dimension của object: dùng ndim

```
print(s1.ndim)
```

1

```
print(s2.ndim)
```

1

- Lấy chiều dài (số phần tử) của series: dùng size

```
print(s1.size)
```

5

```
print(s2.size)
```

5

- Lấy giá trị của series dưới dạng array: dùng values

```
# get values as array
print(s1.values)
```

```
[-0.29088241  2.26262866 -0.04261633  0.64097325 -0.72238851]
```

```
print(s2.values)
```

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



Series

- Lấy n dòng đầu(head)/dòng cuối(tail); dùng head(n)/tail(n)

```
print(s1.head(2))
```

```
0    -0.290882
1     2.262629
dtype: float64
```

```
print(s2.tail(3))
```

```
c     3
d     4
e     5
dtype: int64
```

- Xóa phần tử theo index/label

```
a     1
b     2
c     3
d     4
e     5
dtype: int64
```

```
print(s2)
```

```
s3 = s2.drop(s2.index[[0, 2, 4]])
```

```
print(s3)
```

```
s3 = s2.drop(['c', 'd'])
```

```
print(s3)
```

```
b     2
d     4
dtype: int64
```

```
a     1
b     2
e     5
dtype: int64
```

Series

❑ Tạo series mới bằng cách ánh xạ từ series đang có: dùng `map (lambda)`

● Ví dụ

```
s2
```

```
a    1  
b    2  
c    3  
d    4  
e    5  
dtype: int64
```

```
s3 = s2.map(lambda x: x * x)  
s3
```

```
a    1  
b    4  
c    9  
d   16  
e   25  
dtype: int64
```

Series

□ Thống kê

S.No.	Function	Description
1	count()	Trả về số lượng các phần tử khác null
2	sum()	Trả về tổng các giá trị
3	mean()	Trả về giá trị số trung bình của các giá trị
4	median()	Trả về giá trị trung bình giữa của các giá trị
5	mode()	Trả về giá trị có tần suất xuất hiện nhiều nhất
6	std()	Trả về độ lệch chuẩn của các giá trị
7	min()	Trả về giá trị nhỏ nhất
8	max()	Trả về giá trị lớn nhất
9	abs()	Trả về giá trị tuyệt đối
10	prod()	Product of Values
11	cumsum()	Trả về tổng tích lũy (Cumulative Sum)
12	cumprod()	Cumulative Product

Series

```
s1 = pd.Series([25,26, np.NaN])
print(s1)
print('Số phần tử khác null:',s1.count())
```

```
0    25.0
1    26.0
2     NaN
dtype: float64
Số phần tử khác null: 2
```

```
print(s2.values)
print("Min:", s2.min())
print("Max:", s2.max())
print("Abs:", s2.abs().values)
print("Prod:", s2.prod())
print("Cumsum:", s2.cumsum().values)
print("Cumprod:", s2.cumprod().values)
```

```
[-3.24  2.98  3.98  2.56  3.2   4.6   3.8   3.78  2.98 -4.8   4.1   3.65
  2.98]
Min: -4.8
Max: 4.6
Abs: [3.24 2.98 3.98 2.56 3.2  4.6  3.8  3.78 2.98 4.8  4.1  3.65 2.98]
Prod: 13268383.171812728
Cumsum: [-3.24 -0.26  3.72  6.28  9.48 14.08 17.88 21.66 24.64 19.84 23.94 27.59
 30.57]
Cumprod: [-3.24000000e+00 -9.65520000e+00 -3.84276960e+01 -9.83749018e+01
-3.14799686e+02 -1.44807855e+03 -5.50269850e+03 -2.08002003e+04
-6.19845970e+04  2.97526066e+05  1.21985687e+06  4.45247757e+06
 1.32683832e+07]
```

```
s2 = pd.Series([-3.24,2.98,3.98,2.56,3.20,
               4.6,3.8,3.78,2.98,-4.80,4.10,3.65, 2.98])
print("Sum:", s2.sum())
print("Mean:", s2.mean())
print("Median:", s2.median())
print("Mode:", s2.mode())
print("Std:", s2.std())
```

```
Sum: 30.569999999999997
Mean: 2.3515384615384614
Median: 3.2
Mode: 0    2.98
dtype: float64
Std: 2.900390336241676
```

Series

- Thông tin thống kê chung: dùng describe()

```
print(s2.values)
print(s2.describe())
```

```
[-3.24  2.98  3.98  2.56  3.2   4.6   3.8   3.78  2.98 -4.8   4.1   3.65
 2.98]
```

```
count      13.000000
mean        2.351538
std         2.900390
min        -4.800000
25%         2.980000
50%         3.200000
75%         3.800000
max         4.600000
dtype: float64
```

Nội dung

1. Giới thiệu
2. Series
3. DataFrame
4. Panel

DataFrame

- ❑ DataFrame là một cấu trúc dữ liệu 2D (two-dimensional), dữ liệu được tổ chức theo dòng và cột.
- ❑ Đặc điểm
 - Các cột có nhiều kiểu dữ liệu khác nhau
 - Kích thước có thể thay đổi
 - Trục được gán nhãn (dòng và cột)
 - Có thể thực hiện các phép tính số học theo dòng và cột

DataFrame

❑ Phương thức tạo Data frame:

```
pandas.DataFrame( data, index,  
columns, dtype, copy)
```

● Trong đó:

- data: dữ liệu (list, dict, series, ndarray, dataframe)
- index: danh sách các dòng; index = ['tên dòng 1', 'tên dòng 2']
- columns: danh sách các cột; columns = ['tên cột 1', 'tên cột 2']
- dtype: kiểu dữ liệu của các cột
- copy: copy dữ liệu hay không, mặc định là False

DataFrame

□ Tạo DataFrame

● Khởi tạo

```
df1 = pd.DataFrame()
print(df1)
```

Empty DataFrame
Columns: []
Index: []

● Tạo data frame từ list

```
lst = [1,2,3,4,5]
df1 = pd.DataFrame(lst)
print(df1)
```

	0
0	1
1	2
2	3
3	4
4	5

```
lst2 = [['Bibi',7],['Xuxu',5],['Kent',10]]
df2 = pd.DataFrame(lst2,columns=['Name','Age'])
print(df2)
```

	Name	Age
0	Bibi	7
1	Xuxu	5
2	Kent	10

DataFrame

• Tạo data frame từ dictionary ndarray/list

```
dic1= {'Name':['Bibi', 'Xuxu', 'Kent', 'Mickey'], 'Age':[7,5,10,3]}
df3 = pd.DataFrame(dic1, index=['sv1','sv2','sv3','sv4'], columns=['Name','Age'])
print(df3)
```

	Name	Age
sv1	Bibi	7
sv2	Xuxu	5
sv3	Kent	10
sv4	Mickey	3

• Tạo data frame từ list dictionary

```
# Create a DataFrame from List of Dicts
data = [{'Ten': 'An', 'Tuoi': 12, 'Diem': 8.5}, {'Ten': 'Hoa', 'Tuoi': 12, 'Diem': 7.5},
        {'Ten': 'An', 'Tuoi': 12} ]
df = pd.DataFrame(data, index = ['st1', 'st2', 'st3'])
print(df)
```

	Diem	Ten	Tuoi
st1	8.5	An	12
st2	7.5	Hoa	12
st3	NaN	An	12

Chú ý: NaN sẽ được
thêm vào nơi bị
thiếu dữ liệu

DataFrame

• Tạo data frame từ list dictionary (tt)

```
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]
# 2 cột, tên cột là tên key của dictionary
df1 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])

# 2 cột, tên một cột là tên key của dictionary, tên một cột khác
df2 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])
print(df1)
print(df2)
```

	a	b
first	1	2
second	5	10

	a	b1
first	1	NaN
second	5	NaN

DataFrame

• Tạo data frame từ dictionary series

```
# Create a DataFrame from Dict of Series
d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd']),
     'three' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     }

df5 = pd.DataFrame(d, columns = ['one', 'two', 'three'])
print(df5)
```

	one	two	three
a	1.0	1	1.0
b	2.0	2	2.0
c	3.0	3	3.0
d	NaN	4	NaN

DataFrame

❑ Thao tác trên Data frame

● Lấy dữ liệu theo cột

```
# Lấy dữ liệu theo cột
print(df5)
print("Lấy cột 'one':")
print(df5['one'])
```

	one	two	three
a	1.0	1	1.0
b	2.0	2	2.0
c	3.0	3	3.0
d	NaN	4	NaN

Lấy cột 'one':

a	1.0
b	2.0
c	3.0
d	NaN

Name: one, dtype: float64

Thêm cột

```
# Thêm cột mới
df5['four'] = pd.Series([10, 20, 30, 40],
                        index=['a', 'b', 'c', 'd'])
print(df5)
```

	one	two	three	four
a	1.0	1	1.0	10
b	2.0	2	2.0	20
c	3.0	3	3.0	30
d	NaN	4	NaN	40

DataFrame

• Xóa cột dùng: dùng del/pop()

```
# using del function
print ("Dùng del:")
del df5['three']
print(df5)
```

Dùng del:

	one	two	four
a	1.0	1	10
b	2.0	2	20
c	3.0	3	30
d	NaN	4	40

```
# using pop function
print ("Dùng pop")
df5.pop('one')
print(df5)
```

Dùng pop

	two	four
a	1	10
b	2	20
c	3	30
d	4	40

• Xem danh sách các cột: dùng columns

	Name	Age
2	Alex	23.0
0	Jack	27.0
1	Clarke	32.0

```
print(df.columns)
```

```
Index(['Name', 'Age'], dtype='object')
```


DataFrame

• Lấy dữ liệu theo dòng

```
# lấy dữ liệu trên dòng 'b'
print(df.loc['b'])
```

```
one    2.0
two    2.0
Name: b, dtype: float64
```

```
# dùng index
df = pd.DataFrame(d)
print(df.iloc[1])
```

```
one    2.0
two    2.0
Name: b, dtype: float64
```

```
one two
a  1.0  1
b  2.0  2
c  3.0  3
d  NaN  4
```

```
# chọn nhiều dòng
print(df[2:4])
```

```
one two
c  3.0  3
d  NaN  4
```

• Cập nhật dữ liệu của dòng

df

	Name	Age
2	Alex	23.0
0	Jack	27.0
1	Clarke	32.0

```
# Cập nhật tuổi của dòng 1 thành 29
df.loc[1, 'Age'] = 29
df
```

	Name	Age
2	Alex	23.0
0	Jack	27.0
1	Clarke	29.0

Lấy dữ liệu theo index - iloc

	0	1	2	3
	country	continent	GDP	population
0	USA	North America	19,390,604	322,179,605
1	China	Asia	12,237,700	1,403,500,365
2	Japan	Asia	4,872,137	127,748,513
3	Germany	Europe	3,677,439	81,914,672
4	UK	Europe	2,622,434	65,788,574
5	India	Asia	2,597,491	1,324,171,354

```
your_dataframe.iloc[row-index, column-index]
```

Lấy dữ liệu theo nhãn - loc

column labels

	continent	GDP	population
USA	North America	19,390,604	322,179,605
China	Asia	12,237,700	1,403,500,365
Japan	Asia	4,872,137	127,748,513
Germany	Europe	3,677,439	81,914,672
UK	Europe	2,622,434	65,788,574
India	Asia	2,597,491	1,324,171,354

row
labels

```
your_dataframe.loc[row-label, column-label]
```

Lấy dữ liệu theo điều kiện

```
d = {'name' : pd.Series(['An', 'Hoa', 'Trung','Huy','Thanh','Dung'], index=[49,48,47, 1, 2, 3]),
      'age' : pd.Series([15, 22, 36,27,55],index=[49,48,47, 1, 3])
    }

df=pd.DataFrame(d,columns=['name','age'])
print('Dataframe\n',df)

print('Giá trị dòng 3- cột name là:',df.loc[3,'name'])

print('Lọc những nhân viên có tuổi >30: \n', df.loc[df['age']>30])

print('Lọc tên những nhân viên có tuổi >30: \n', df.loc[df['age']>30,'name'])
```

```
Dataframe
   name  age
1   Huy  27.0
2  Thanh  NaN
3   Dung  55.0
47  Trung  36.0
48   Hoa  22.0
49   An   15.0
Giá trị dòng 3- cột name là: Dung
Lọc những nhân viên có tuổi >30:
   name  age
3   Dung  55.0
47  Trung  36.0
Lọc tên những nhân viên có tuổi >30:
3      Dung
47     Trung
Name: name, dtype: object
```

Xác định lại nhãn dòng – set_index

country	continent	GDP	population
USA	North America	19,390,604	322,179,605
China	Asia	12,237,700	1,403,500,365
Japan	Asia	4,872,137	127,748,513
Germany	Europe	3,677,439	81,914,672
UK	Europe	2,622,434	65,788,574
India	Asia	2,597,491	1,324,171,354

```
country_data_df = country_data_df.set_index('country')
```

column labels

	continent	GDP	population
USA	North America	19,390,604	322,179,605
China	Asia	12,237,700	1,403,500,365
Japan	Asia	4,872,137	127,748,513
Germany	Europe	3,677,439	81,914,672
UK	Europe	2,622,434	65,788,574
India	Asia	2,597,491	1,324,171,354

row
labels


DataFrame

• Thêm dòng: dùng append()

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

```
# thêm dòng dùng append()
df2 = pd.DataFrame([{'one': 1, 'two': 2}, {'one': 5, 'two': 10}], index = ['e', 'f'])

df = df.append(df2)
print(df)
```



	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4
e	1.0	2
f	5.0	10

• Xóa dòng: dùng drop()

```
# xóa dòng theo label
df = df.drop(['a', 'b'])
print(df)
```

	one	two
c	3.0	3
d	NaN	4
e	1.0	2
f	5.0	10

DataFrame

☐ Sắp xếp

- Theo index: dùng `sort_index()`

```
print(df.sort_index())
```

	Name	Age
0	Jack	27.0
1	Clarke	32.0
2	Alex	23.0

- Theo giá trị: dùng `sort_values()`

```
print(df.sort_values(by='Name'))
```

	Name	Age
2	Alex	23.0
1	Clarke	32.0
0	Jack	27.0

	Name	Age
2	Alex	23.0
0	Jack	27.0
1	Clarke	32.0

DataFrame

❑ Xếp hạng: Dùng rank()

```
print(df.rank())
```

	Name	Age
2	1.0	1.0
0	3.0	2.0
1	2.0	3.0

	Name	Age
2	Alex	23.0
0	Jack	27.0
1	Clarke	32.0

❑ Xem thông tin Data Frame: dùng info()

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 2 to 1
Data columns (total 2 columns):
Name      3 non-null object
Age       3 non-null float64
dtypes: float64(1), object(1)
memory usage: 60.0+ bytes
None
```


DataFrame

□ any/all

- Kiểm tra bất kỳ phần tử nào đều là True: dùng any()

```
>>> df = pd.DataFrame({"A": [1, 2], "B": [0, 2], "C": [0, 0]})
>>> df
```

	A	B	C
0	1	0	0
1	2	2	0

```
>>> df.any()
A      True
B      True
C     False
dtype: bool
```

DataFrame

□ any/all

- Kiểm tra mọi phần tử đều là True: dùng all()

```
>>> df = pd.DataFrame({'col1': [True, True], 'col2': [True, False]})  
>>> df  
   col1  col2  
0  True  True  
1  True False
```

kiểm tra theo từng cột

```
>>> df.all()  
col1      True  
col2     False  
dtype: bool
```

kiểm tra theo từng dòng

```
>>> df.all(axis=1)  
0      True  
1     False  
dtype: bool
```

DataFrame

□ Thống kê

S.No.	Function	Description
1	count(axis={0 hoặc 1})	Trả về số lượng các phần tử khác null
2	sum(axis={0 hoặc 1})	Trả về tổng các giá trị
3	.mean(axis={0 hoặc 1})	Trả về giá trị số trung bình của các giá trị
4	median(axis={0 hoặc 1})	Trả về giá trị trung bình giữa của các giá trị
5	mode(axis={0 hoặc 1})	Trả về giá trị có tần suất xuất hiện nhiều nhất
6	std(axis={0 hoặc 1})	Trả về độ lệch chuẩn của các giá trị
7	min(axis={0 hoặc 1})	Trả về giá trị nhỏ nhất
8	max(axis={0 hoặc 1})	Trả về giá trị lớn nhất
9	abs()	Trả về giá trị tuyệt đối
10	prod()	Product of Values
11	cumsum()	Trả về tổng tích lũy (Cumulative Sum)
12	cumprod()	Cumulative Product

DataFrame

	Age	Name	Rating
0	25	Minh	4.23
1	26	Xuân	3.24
2	25	Thanh	3.98
3	23	Hoàng	2.56
4	30	Thắng	3.20
5	29	Hai	4.60
6	23	Thanh	3.80
7	34	Bình	3.78
8	40	Hậu	2.98
9	30	Huy	4.80
10	51	Thùy	4.10
11	46	Tuyền	3.65

```
print(df.sum()) # tổng theo cột
```

```
Age                                     382
Name      MinhXuânThanhHoàngThắngHaiThanhBìnhHậuHuyThùyT...
Rating                                     44.92
dtype: object
```

```
print(df.sum(1)) # tổng theo dòng
```

```
0      29.23
1      29.24
2      28.98
3      25.56
4      33.20
5      33.60
6      26.80
7      37.78
8      42.98
9      34.80
10     55.10
11     49.65
dtype: float64
```

```
print(df['Age'].mode())
```

```
0      23
1      25
2      30
dtype: int64
```

DataFrame

- Thông tin thống kê chung: dùng describe()

```
print(df.describe())
```

	Age	Rating
count	12.000000	12.000000
mean	31.833333	3.743333
std	9.232682	0.661628
min	23.000000	2.560000
25%	25.000000	3.230000
50%	29.500000	3.790000
75%	35.500000	4.132500
max	51.000000	4.800000

```
print(df. describe(include='all'))
```

	Age	Name	Rating
count	12.000000	12	12.000000
unique	NaN	11	NaN
top	NaN	Thanh	NaN
freq	NaN	2	NaN
mean	31.833333	NaN	3.743333
std	9.232682	NaN	0.661628
min	23.000000	NaN	2.560000
25%	25.000000	NaN	3.230000
50%	29.500000	NaN	3.790000
75%	35.500000	NaN	4.132500
max	51.000000	NaN	4.800000

Nội dung

1. Giới thiệu
2. Series
3. DataFrame
4. Panel

Panel

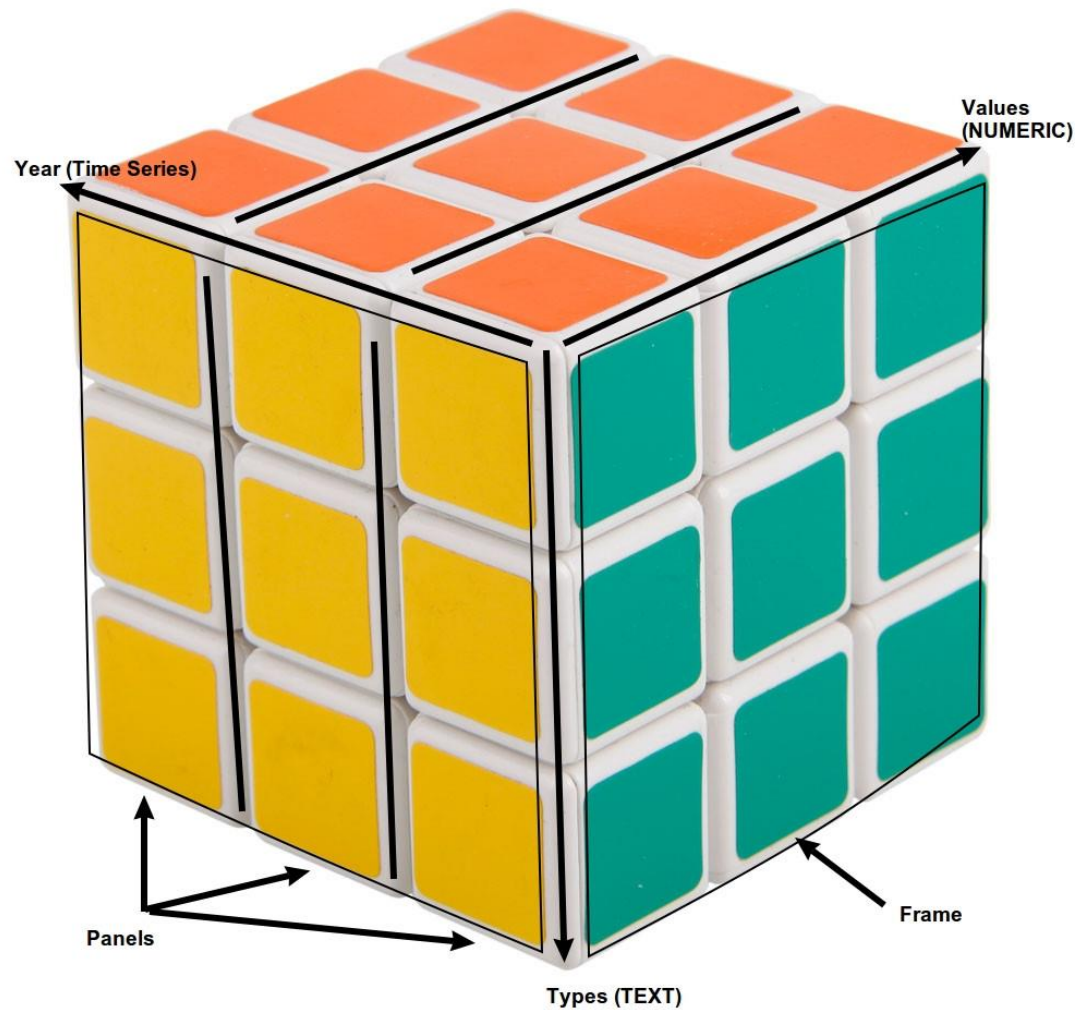
❑ Là một 3D container chứa dữ liệu

● Tên của 3 trục như sau:

- items – axis 0, mỗi item tương ứng một DataFrame chứa bên trong.
- major_axis – axis 1, index (rows) của mỗi DataFrame.
- minor_axis – axis 2, các cột của mỗi DataFrame.

Ghi chú: cấu trúc dữ liệu này ít sử dụng

Panel



Panel

❑ Phương thức tạo Panel

`pandas.Panel(data, items, major_axis, minor_axis, dtype, copy)`

● Trong đó:

- data: có thể là ndarray, series, map, lists, dict, constants hoặc DataFrame
- items: axis 0
- major_axis: axis 1
- minor_axis: axis 2
- dtype: kiểu dữ liệu của mỗi cột
- copy: copy dữ liệu nếu có

Panel

□ Tạo panel

- Khởi tạo

```
# khởi tạo  
p = pd.Panel()  
print(p)
```

```
<class 'pandas.core.panel.Panel'>  
Dimensions: 0 (items) x 0 (major_axis) x 0 (minor_axis)  
Items axis: None  
Major_axis axis: None  
Minor_axis axis: None
```

Panel

• Tạo panel từ ndarray

```
data = np.random.rand(2,4,5)
print(type(data))
p = pd.Panel(data)
print(p)
print("Data frame 1:")
print(p[0])
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 4 (major_axis) x 5 (minor_axis)
Items axis: 0 to 1
Major_axis axis: 0 to 3
Minor_axis axis: 0 to 4
Data frame 1:
```

	0	1	2	3	4
0	0.403008	0.606423	0.212630	0.379008	0.584206
1	0.505842	0.920907	0.911677	0.203077	0.083612
2	0.309360	0.695284	0.615508	0.339175	0.452895
3	0.231657	0.953858	0.416134	0.883392	0.713399

Panel

- Tù Data frame

```
print(df1)
```

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

```
print(df2)
```

	four	three
a	5	5
b	6	6
c	7	7
d	9	8

```
data = {'Item1' : df1,
        'Item2' : df2}
p = pd.Panel(data)
print(p)
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 4 (major_axis) x 4 (minor_axis)
Items axis: Item1 to Item2
Major_axis axis: a to d
Minor_axis axis: four to two
```

Panel

❑ Truy xuất dữ liệu từ panel

```
print(p)
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 4 (major_axis) x 4 (minor_axis)
Items axis: Item1 to Item2
Major_axis axis: a to d
Minor_axis axis: four to two
```

```
print(p['Item1'])
```

	four	one	three	two
a	NaN	1.0	NaN	1.0
b	NaN	2.0	NaN	2.0
c	NaN	3.0	NaN	3.0
d	NaN	NaN	NaN	4.0

```
print(p['Item2'])
```

	four	one	three	two
a	5.0	NaN	5.0	NaN
b	6.0	NaN	6.0	NaN
c	7.0	NaN	7.0	NaN
d	9.0	NaN	8.0	NaN

• Theo Major_axis (theo dòng)

```
print(p.major_xs('a'))
```

	Item1	Item2
four	NaN	5.0
one	1.0	NaN
three	NaN	5.0
two	1.0	NaN

• Theo Minor axis (theo cột)

```
print(p.minor_xs('three'))
```

	Item1	Item2
a	NaN	5.0
b	NaN	6.0
c	NaN	7.0
d	NaN	8.0

Panel

❑ Thống kê: trên từng thành phần của panel (data frame)

```
data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),
        'Item2' : pd.DataFrame(np.random.randn(4, 2))}
p = pd.Panel(data)
print(p['Item1'])
```

	0	1	2
0	-1.400572	-0.093098	0.232955
1	-1.171686	1.030805	1.140213
2	-1.916337	-1.280010	-2.386235
3	-1.205018	-0.745174	-1.294989

```
print(p['Item1'].describe(include = 'all'))
```

	0	1	2
count	4.000000	4.000000	4.000000
mean	-1.423403	-0.271869	-0.577014
std	0.343782	0.994868	1.569892
min	-1.916337	-1.280010	-2.386235
25%	-1.529513	-0.878883	-1.567800
50%	-1.302795	-0.419136	-0.531017
75%	-1.196685	0.187878	0.459769
max	-1.171686	1.030805	1.140213

