

# Kungfu Pandas

Lê Huỳnh Đức

2021-06-03



# Kungfu Pandas

Lê Huỳnh Đức

2021-06-03



# Contents

<b>C u trúc và ki u d li u</b>	<b>ix</b>
0.1 Series . . . . .	ix
0.1.1 Các cách kh i tạo . . . . .	ix
0.1.2 M t s thao tác c b n . . . . .	x
0.2 DataFrame . . . . .	xii
0.2.1 Các cách kh i tạo . . . . .	xiii
0.2.2 Các hàm kh i tạo thay th . . . . .	xvi
0.3 Data type trong pandas . . . . .	xviii
<b>Nh p xu t trong pandas</b>	<b>xxi</b>
0.4 Đ o c và l u file . . . . .	xxi
0.5 C u hình pandas . . . . .	xxi
<b>M t s hàm c b n</b>	<b>xxiii</b>
<b>L p trong Pandas</b>	<b>xxv</b>
0.6 S dụng vectorizer . . . . .	xxv
0.7 S dụng apply . . . . .	xxv
0.8 S dụng iterator . . . . .	xxv
0.9 X lý song song trong pandas . . . . .	xxv
<b>Select và Filter</b>	<b>xxvii</b>
0.10 Index . . . . .	xxvii
0.11 loc và iloc . . . . .	xxvii
0.12 Lo c theo đi u ki n . . . . .	xxvii
<b>Các cách ph i h p nhi u b ng v i nhau</b>	<b>xxix</b>
0.13 Join . . . . .	xxix
0.14 Merge . . . . .	xxix
0.15 Concat . . . . .	xxix
<b>Groupby và Aggregate</b>	<b>xxxi</b>
<b>Làm vi c v i s ki u d li u</b>	<b>xxxiii</b>

0.16	X lý d li u dạng text . . . . .	xxxiii
0.17	X lý d li u dạng timestamp . . . . .	xxxiii
0.18	Category trong pandas . . . . .	xxxiii
0.19	X lý Missing data . . . . .	xxxiii
<b>M t s k i n t h c n â n g c a o</b>		<b>xxxv</b>
0.20	MultiIndex . . . . .	xxxv
0.21	Pivot và Merge . . . . .	xxxv
0.22	Resample . . . . .	xxxv
0.23	Window . . . . .	xxxv
<b>Anomaly Detection Project</b>		<b>xxxvii</b>
<b>Visualize v i Matplotlib</b>		<b>xxxix</b>

# **L i n ó i đ u**

**G i i t h i u c u n s á c h**

**C à i đ t J u p y t e r L a b**

**C à i đ t P a n d a s**





# Cấu trúc và kiểu dữ liệu

## 0.1 Series

Trong Pandas, `Series` là một chỉ số bao gồm một danh sách giá trị, và một mảng chứa index của các giá trị. Trong dữ liệu bảng, mỗi `Series` được xem như là một cột của bảng đó. Cách định nghĩa để tạo 1 series như sau

```
s = pd.Series(data, index=None, name=None)
```

Trong đó `data` có thể có dạng:

- `numpy.ndarray`, `List`
- Python `dict`
- `Scalar`

`index` có thể truy cập hoặc không, tùy vào dạng của `data` mà `index` sẽ có định nghĩa khác nhau. `name` là tên của `Series`, giá trị này cũng không nhất thiết phải truy cập vào.

### 0.1.1 Các cách khởi tạo

#### Khởi tạo Series bằng array

Khi không truy cập giá trị `index`, `Series` sẽ mặc định index của nó là 1 mảng số nguyên từ 0 đến `len(data) - 1`

```
In [1]: pd.Series(data=[0, 1, 2], index=["a", "b", "c"], name="meow")
Out[1]:
a      0
b      1
c      2
Name: meow, dtype: int64
```

### Kh i tạo Series b ng dict

```
In [1]: pd.Series({"b": 1, "a": 0, "c": 2})
Out[1]:
b    1
a    0
c    2
dtype: int64
```



**L u ý:** Trong tr ng h p bạn truy n b i n index vào, Series s đánh index d a vào th t trong index, và ch ch a các giá trị c a dict có key n m trong index. V i các giá trị trong b i n index không có trong keys c a dict, Series s tạo ra các giá trị b i thi u NaN.

```
In [1]: pd.Series({"a": 0, "b": 1, "c": 2, "e": 4}, index=["b", "c", "d", "a"])
Out[1]:
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```



**L u ý:** NaN là giá trị m c định cho d li u b i thi u trong pandas và giá trị này có ki u là float64 nên ki u d li u c a Series cũng là float64 khác v i int64 ví dụ tr c đó.

### Kh i tạo Series b ng m t giá trị (Scalar)

```
In [1]: pd.Series(data=1, index=["a", "b", "c"])
Out[1]:
a    1
b    1
c    1
dtype: int64
```

## 0.1.2 M t s thao tác c b n

Thao tác trên Series cũng gi ng v i thao tác trên `numpy.array`. Ngoài ra chúng ta còn có th tác v i Series d a vào index

Ví dụ;

```
In [1]: s = pd.Series(data=[0, 1, 2, 3, 4, 5], index=["a", "b", "c", "d", "e", "f"])
```

**Hình thức toàn bộ giá trị của Series** Ta gọi thu c tính `.values`

```
In [1]: s.values
Out[1]:
array([0, 1, 2, 3, 4, 5])
```

### Ly theo indice

```
In [2]: s[2]
Out[2]: 2
```

### Ly theo index

```
In [3]: s["c"]
Out[3]: 2
```

### Slice indice

```
In [4]: s[1:3]
Out[4]:
b    1
d    2
dtype: int64
```

### Slice index

```
In [5]: s["b":"c"]
Out[5]:
b    1
c    2
dtype: int64
```

### List indice

```
In [6]: s[[1, 2, 4]]
Out[6]:
b    1
c    2
e    4
dtype: int64
```

**List index**

```
In [7]: s[["b", "c", "e"]]
Out[7]:
b    1
c    2
e    4
dtype: int64
```

**Đi u ki n**

```
In [5]: s[s > s.mean()]
Out[5]:
d    3
e    4
f    5
dtype: int64
```

## 0.2 DataFrame

`DataFrame` là c u trúc d li u chính và cũng là đ c tr ng c a pandas. Cũng gi ng nh SQL Table, `DataFrame` là m t b ng g m m t hay nhi u c t d li u. H o c có th nói rõ h n là `DataFrame` là t p h p các `Series` lại v i nhau.

Cách kh i tạo `DataFrame` nh sau

```
df = pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
```

Cũng gi ng nh `Series`, data c a `DataFrame` có nhi u cách kh i tạo khác nhau nh :

- dict c a `Series`, dict C a `numpy.array/List`
- M ng 2 chỉ u `numpy.ndarray`, List C a List
- M ng có c u trúc<sup>1</sup>
- T l `Series`
- T `DataFrame` khác

Tùy vào c u trúc c a data mà chúng ta có th b qua bi n index. Bi n columns th hi n tên c a các `Series`. dtype s định nghĩa các ki u d li u c a d li u, chúng ta s th o lu n v nó ph n k ti p c a ch ng này. copy dùng đ tạo b n sao t d li u data, nó ch nh h ng khi data là `DataFrame` khác ho c `numpy.ndarray`, vì c copy này s tránh tr ng h p 2 bi n cùng tr v cùng l b nh .

<sup>1</sup><https://numpy.org/doc/stable/user/basics.rec.html>

### 0.2.1 Các cách kh i tạo

#### Kh i tạo DataFrame t dict c a Series

Khi không truy n b i n index vào, thì index c a DataFrame s là h p gi a 2 index c a Series và chúng s đ c s p x p theo th t t v ng. N u ta không truy n columns thì các c t c a DataFrame s đ c s p x p theo th t truy n vào các keys c a dict.

Khi truy n b i n index vào, t ng t nh Series, ch nh ng index n m trong index m i đ c chọn, còn nh ng index bị th i u s đ c đi n giá trị NaN

Khi truy n giá trị columns, DataFrame s chọn nh ng Series thu c dict có key thu c columns, giá trị trong columns không có trong key c a dict s đ c gán NaN

```
In [1]: d = {
        "one": pd.Series([1, 2, 3], index=["c", "b", "a"]),
        "two": pd.Series([1, 2, 3, 4], index=["c", "a", "b", "d"])
    }
In [2]: pd.DataFrame(d)
Out[2]:
   one  two
a  3.0    2
b  2.0    3
c  1.0    1
d  NaN    4

In [3]: pd.DataFrame(d, index=["d", "b", "a"])
Out[3]:
   one  two
d  NaN    4
b  2.0    3
a  3.0    2

In [4]: pd.DataFrame(d, index=["d", "b", "a"], columns=["two", "three"])
Out[4]:
   two  three
d     4    NaN
b     3    NaN
a     2    NaN
```

#### Kh i tạo DataFrame t dict c a numpy.ndarray/List

Đ i v i v i c kh i tạo này, b t bu c các m ng ph i có cùng đ dài. Khi không truy n index vào thì index c a DataFrame s đ c tạo t 0 đ n len(n) - 1 trong đó n là đ dài c a m ng. Khi truy n giá trị columns, DataFrame s chọn nh ng key thu c dict và cũng thu c columns, giá trị trong columns không có trong key c a dict s đ c gán NaN

```
In [1]: d = {
        "one": [1, 2, 3, 4],
        "two": [1, 2, 3, 4],
        "three": [1, 2, 3, 4]
    }
In [2]: pd.DataFrame(data=d,
                    index=["a", "b", "c", "d"],
                    columns=["one", "two", "four"])
Out[2]:
   one  two  four
a    1    1   NaN
b    2    2   NaN
c    3    3   NaN
d    4    4   NaN
```

### Kh i tạo DataFrame t M ng 2 chỉ u/ 2-d numpy.ndarray

Khi không truy n index vào thì index c a DataFrame s đ c tạo t 0 đ n  $\text{len}(n) - 1$  trong đó n là s l ng List con ho c là s dòng hay `shape[0]` c a `numpy.ndarray`. Khi không truy n columns thì tên columns s đ c tạo t 0 đ n  $\text{len}(n) - 1$  vì n là đ dài l n nh t c a List con ho c `shape[1]` c a `numpy.ndarray`

```
In [1]: pd.DataFrame(data=[[1, 2], [3, 4, 5]],
                    index=["a", "b"],
                    columns=['one', 'two', 'three'])
Out[1]:
   one  two  three
a    1    2   NaN
b    3    4   5.0

In [2]: pd.DataFrame(data=np.random.rand(2,3),
                    index=["a", "b"],
                    columns=['one', 'two', 'three']))
Out[2]:
   one      two      three
a  0.662008  0.085735  0.331281
b  0.115360  0.358092  0.862477
```

### Kh i tạo DataFrame t danh sách các dict

cách kh i tạo này, bạn hãy t ng t ng r ng m i dict là m t dòng c a DataFrame vì các key là tên c t và value là giá trị tại c t đó. Vì c truy n thêm ho c không truy n index cũng gì ng nh các tr ng h p kh i tạo trên.



**Lưu ý:** Trong trường hợp này, nếu bạn truyền columns vào thì columns bắt buộc phải chứa tất cả các keys của dict

Trong ví dụ đi đây, columns phải chứa toàn bộ keys ["one", "two", "three"], nếu thiếu 1 trong 3 sẽ phát sinh lỗi.

```
In [1]: d = [{"one": 1, "two": 2}, {"one": 4, "two": 5, "three": 6}]
In [2]: pd.DataFrame(d, index=["a", "b"], columns=["one", "two", "three", "four"])
Out[2]:
```

	one	two	three	four
a	1	2	NaN	NaN
b	4	5	6.0	NaN

### Khi tạo DataFrame từ Mảng có cấu trúc

Mảng có cấu trúc là mảng mà các phần tử của nó là một cấu trúc, bao gồm các thành phần như tên, các thành phần này được đặt tên và khai báo kiểu dữ liệu. Đây là một ví dụ về mảng có cấu trúc trong numpy

```
In [1]: data = np.array([('pikachu', 9, 27.0), ('mewtwo', 3, 81.0)],
                        dtype=[('name', 'U10'), ('age', 'i4'), ('weight', 'f4')])
In [2]: pd.DataFrame(data)
Out[2]:
```

	name	age	weight
0	pikachu	9	27.0
1	mewtwo	3	81.0

### Khi tạo DataFrame từ namedtuple

Các trường trong namedtuple sẽ được gán thành tên các columns trong DataFrame. Nếu giá trị của namedtuple sẽ được xem là 1 dòng trong DataFrame. Số lượng cột của DataFrame sẽ phụ thuộc vào số lượng giá trị của phần tử namedtuple đầu tiên. Nếu các phần tử phía sau có số lượng giá trị ít hơn thì sẽ được điền NaN và ngược lại sẽ trả lại số lượng giá trị của namedtuple lớn nhất số lượng giá trị của phần tử namedtuple đầu tiên.

Ví dụ về cách tạo namedtuple

```
from collections import namedtuple
Point2D = namedtuple("Point2D", "x y")
Point3D = namedtuple("Point3D", "x y z")
```

Tạo DataFrame từ namedtuple Point2D

```
In [1]: pd.DataFrame([Point2D(0, 0), Point2D(0, 1), Point2D(0, 2)])
Out[1]:
   x  y
0  0  0
1  0  1
2  0  2
```

Tạo DataFrame từ namedtuple của Point2D và Point3D

```
In [1]: pd.DataFrame([Point3D(0, 0, 0), Point2D(0, 1), Point3D(0, 2, 3)])
Out[1]:
   x  y  z
0  0  0  0.0
1  0  1  NaN
2  0  2  3.0
```

Như ta thấy, tại phần tử thứ 2 chỉ có 2 giá trị, trong khi phần tử thứ nhất có 3 giá trị, vì vậy nên phần tử bị thiếu tại cột z sẽ được gán NaN

### Khởi tạo DataFrame từ Series

```
In [1]: s = pd.Series(data=[0, 1, 2], index=["a", "b", "c"], name="meow")
In [2]: pd.DataFrame(s)
Out[2]:
   meow
a      0
b      1
c      2
```

name của Series s là tên cột của DataFrame và index của Series s là index của DataFrame. Nếu ta không truyền các biến index, columns khi khởi tạo pd.DataFrame

## 0.2.2 Các hàm khởi tạo thay thế

### DataFrame.from\_dict

Cách khởi tạo

```
pd.DataFrame.from_dict(data, orient='columns', dtype=None, columns=None)
```

data truyền vào là 1 dict, orient có 2 giá trị có thể đưa vào là {"columns", "index"}, columns là danh sách tên các cột của DataFrame.





**Lưu ý:** Chẩn tra nghiệm columns khi orient="index". Khi orient="columns" s báo li.

Ví dụ, tạo DataFrame khi orient="columns". Vì cách khi tạo này tên các cột của DataFrame s là key của dict

```
In [1]: data = {"col_1": [3, 2, 1, 0], "col_2": ["a", "b", "c", "d"]}
In [2]: pd.DataFrame.from_dict(data)
Out[2]:
   col_1 col_2
0      3     a
1      2     b
2      1     c
3      0     d
```

Ví dụ, tạo DataFrame khi orient="index". Vì cách khi tạo này index của DataFrame s là key của dict.

```
In [1]: data = {"col_1": [3, 2, 1, 0], "col_2": ["a", "b", "c", "d"]}
In [2]: pd.DataFrame.from_dict(data, orient="index",
                                columns=["one", "two", "three", "four"])
Out[2]:
   col_1 col_2
0      3     a
1      2     b
2      1     c
3      0     d
```

### DataFrame.from\_records

Cách khi tạo

```
pd.DataFrame.from_records(data)
```

data tra nghiệm vào có thể là m t m ng có c u trúc

```
In [1]: data = np.array([('Rex', 9, 81.0), ('Fido', 3, 27.0)],
                          dtype=[('name', 'U10'), ('age', 'i4'), ('weight', 'f4')])
In [2]: pd.DataFrame.from_records(data, index=["a", "b"])
Out[2]:
   name age weight
a  Rex   9   81.0
b  Fido   3   27.0
```

D li u có thể m t danh sách các namedtuple

```

from collections import namedtuple
Point2D = namedtuple("Point2D", "x y")
Point3D = namedtuple("Point3D", "x y z")
pd.DataFrame.from_records([Point3D(0, 0, 0), Point2D(0, 1), Point3D(0, 2, 3)],
                           columns=["x", "y", "z"], index=["a", "b", "c"])

```

```

      x  y  z
a  0  0  0.0
b  0  1  NaN
c  0  2  3.0

```

Ho c 1 danh sách các dict

```

In [1]: d = [{"one": 1, "two": 2}, {"one": 4, "two": 5, "three": 6}]
In [2]: pd.DataFrame.from_records(d, index=["a", "b"], columns=["one", "two", "three", "four"])
Out[2]:
      one  two  three  four
a      1    2    NaN   NaN
b      4    5    6.0   NaN

```

### 0.3 Data type trong pandas

Đ kiểm tra kiểu dữ liệu của Series hay DataFrame bạn có thể gọi thuộc tính dtypes. Các kiểu dữ liệu thường gặp của Pandas được mô tả theo bảng dưới đây:

Các kiểu dữ liệu phổ biến	Numpy/Pandas object	Hình thức
Boolean	np.bool	<i>bool</i>
Integer	np.int, np.uint	<i>int uint</i>
Float	np.float	<i>float</i>
Object	np.object	<i>O, object</i>
Datetime	np.datetime64, pd.Timestamp	<i>datetime64</i>
Timedelta	np.timedelta64, pd.Timedelta	<i>timedelta64</i>
Category	pd.Categorical	<i>category</i>
Complex	np.complex	<i>complex</i>

Ví dụ,

```
In [1]: df = pd.DataFrame({
        'col_1': [1, 0, 1, 0],
        'col_2': [1.0, 2.0, 3.0, 4.0],
        'col_3': ['1', '2', '3', '4'],
        'col_4': ['1', 2, '3', 4],
        'col_5': [True, False, True, False]})

In [2]: df
Out[2]:
```

	col_1	col_2	col_3	col_4	col_5
0	1	1.0	1	1	True
1	0	2.0	2	2	False
2	1	3.0	3	3	True
3	0	4.0	4	4	False

```
In [3]: df.dtypes
Out[3]:
col_1      int64
col_2     float64
col_3      object
col_4      object
col_5       bool
dtype: object
```

Nếu không khai báo kiểu dữ liệu khi khởi tạo, pandas sẽ mặc định kiểu dữ liệu là `int64`, `float64` và `object` và `bool`.



# Nhập xuất trong pandas

0.4 Đọc và lưu file

0.5 Cấu hình pandas



## M t s hàm c b n





# **L p trong Pandas**

**0.6 S dụng vectorizer**

**0.7 S dụng apply**

**0.8 S dụng iterator**

**0.9 X lý song song trong pandas**



# Select và Filter

0.10 Index

0.11 loc và iloc

0.12 Loç theo đi u ki n



## **Các cách ph i h p nhi u b ng v i nhau**

**0.13 Join**

**0.14 Merge**

**0.15 Concat**



## **Groupby và Aggregate**





# **Làm việc với các kiểu dữ liệu**

**0.16 Xử lý dữ liệu dạng text**

**0.17 Xử lý dữ liệu dạng timestamp**

**0.18 Category trong pandas**

**0.19 Xử lý Missing data**



## M t s k i n t h c n â n g c a o

0.20 MultiIndex

0.21 Pivot và Merge

0.22 Resample

0.23 Window



# **Anomaly Detection Project**



# Visualize v i Matplotlib