**Xử lý dữ liệu với Pandas trong Python**

**Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và mang hàm ý. Tên thư viện được bắt nguồn từ panel data (bảng dữ liệu). Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, có tiềm năng không đồng nhất) và dữ liệu chuỗi thời gian.**

Mục tiêu của pandas là trở thành **khối căn bản** (building block) cấp cao cơ bản cho công việc thực tế, phân tích dữ liệu thế giới thực trong Python, và rộng hơn là trở thành công cụ thao tác / phân tích mã nguồn mở mạnh mẽ và linh hoạt nhất có sẵn trong bất kỳ loại ngôn ngữ lập trình nào.

**Vì sao bạn nên chọn pandas?**

Pandas rất phù hợp với nhiều loại dữ liệu khác nhau:

* Dữ liệu dạng bảng với các cột được nhập không đồng nhất, như trong bảng SQL hoặc bảng tính Excel.
* Dữ liệu chuỗi thời gian theo thứ tự và không có thứ tự (không nhất thiết phải có tần số cố định).
* Dữ liệu ma trận tùy ý (được nhập đồng nhất hoặc không đồng nhất) với nhãn hàng và cột.
* Bất kỳ hình thức khác của các bộ dữ liệu quan sát / thống kê. Dữ liệu thực sự không cần phải được dán nhãn vào cấu trúc dữ liệu pandas.
* Pandas được xây dựng dựa trên [NumPy](https://codelearn.io/sharing/tim-hieu-thu-vien-numpy-trong-python). Hai cấu trúc dữ liệu chính của pandas là **Series** (1 chiều) và **DataFrame** (2 chiều) xử lý được phần lớn các trường hợp điển hình trong tài chính, thống kê, khoa học xã hội và nhiều lĩnh vực kỹ thuật.

Ưu điểm của pandas:

* Dễ dàng xử lý dữ liệu mất mát, được biểu thị dưới dạng NaN, trong dữ liệu dấu phẩy động cũng như dấu phẩy tĩnh theo ý người dùng mong muốn: bỏ qua hoặc chuyển sang 0
* Khả năng thay đổi kích thước: các cột có thể được chèn và xóa khỏi DataFrame và các đối tượng chiều cao hơn
* Căn chỉnh dữ liệu tự động và rõ ràng: các đối tượng có thể được căn chỉnh rõ ràng với một bộ nhãn hoặc người dùng chỉ cần bỏ qua các nhãn và để Series, DataFrame, v.v. tự động căn chỉnh dữ liệu cho bạn trong các tính toán
* Chức năng group by mạnh mẽ, linh hoạt để thực hiện các hoạt động kết hợp phân tách áp dụng trên các tập dữ liệu, cho cả dữ liệu tổng hợp và chuyển đổi
* Dễ dàng chuyển đổi dữ liệu rời rạc (ragged), chỉ mục khác nhau (differently-indexed) trong các cấu trúc dữ liệu khác của Python và NumPy thành các đối tượng DataFrame
* Cắt lát (slicing) thông minh dựa trên nhãn, lập chỉ mục ưa thích (fancy indexing) và tập hợp lại (subsetting) các tập dữ liệu lớn
* Gộp (merging) và nối (joining) các tập dữ liệu trực quan
* Linh hoạt trong định hình lại (reshaping) và xoay (pivoting) các tập dữ liệu
* Dán nhãn phân cấp (hierarchical) của các trục (có thể có nhiều nhãn trên mỗi đánh dấu)
* Các công cụ IO mạnh mẽ để tải dữ liệu từ các tệp phẳng (flat file) như CSV và delimited, tệp Excel, cơ sở dữ liệu và lưu / tải dữ liệu từ định dạng HDF5 cực nhanh
* Chức năng theo chuỗi thời gian (time series) cụ thể: tạo phạm vi ngày và chuyển đổi tần số, thống kê cửa sổ di chuyển, dịch chuyển ngày và độ trễ.
* Tích hợp tốt với các thư viện khác của python như SciPy, Matplotlib, [Plotly](https://codelearn.io/sharing/thu-vien-plotly-trong-python-la-gi), v.v.
* Hiệu suất tốt

**Cài đặt thư viện Pandas**

* Sử dụng pip và gõ lệnh: pip install pandas
* Hoặc bằng [Anaconda](https://www.anaconda.com/), dùng lệnh: conda install pandas

Để tìm các cách cài đặt pandas khác bạn hãy xem thêm tại [ĐÂY](https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html).  
*Lưu ý*: Bạn cần cài đặt thư viện [NumPy](https://codelearn.io/sharing/tim-hieu-thu-vien-numpy-trong-python) trước (nếu bạn cài bằng Anaconda thì NumPy đã có sẵn).

**Khai báo thư viện Pandas**

import pandas as pd   
Bạn không nên thay đổi từ pd bằng từ khác vì các tài liệu hướng dẫn đều ngầm quy ước như vậy.

Trong phạm vi bài viết này mình sẽ giới thiệu về các xử lý dữ liệu cho 2 cấu trúc dữ liệu cơ bản trong pandas.

**Thao tác với cấu trúc dữ liệu cơ bản**

Pandas có 2 cấu trúc dữ liệu cơ bản là:

* Series (1 chiều)
* DataFrame (2 chiều).

Panel (3 chiều) từng là một cấu trúc dữ liệu trong pandas trước khi bị gỡ bỏ từ phiên bản 0.25. Bạn có thể tham khảo về [panel ở phiên bản 0.24](https://pandas.pydata.org/pandas-docs/version/0.24.2/reference/panel.html).

**1. Series**

Series([data, index, dtype, name, copy, . . . ])

Series là mảng một chiều giống như mảng Numpy, hay như một cột của một bảng, nhưng nó bao gồm thêm một bảng đánh label. Series có thể được khởi tạo thông qua NumPy, kiểu Dict hoặc các dữ liệu vô hướng bình thường. Series có nhiều thuộc tính như *index, array, values, dtype,*v.v. Bạn có thể thực hiện chuyển đổi Series sang dạng dtype xác định, tạo bảng copy, trả về dạng bool của một thành phần, chuyển Series từ DatetimeIndex sang PeriodIndex, v.v.

Một số ví dụ về thao tác với Series:

**Tạo Series**

*Ví dụ 1*: Không truyền index

import pandas as pd

s = pd.Series([0,1,2,3])

print(s)

Output:

0 0

1 1

2 2

3 3

dtype: int64

pandas sẽ mặc định truyền indextừ 0 đến len(data)-1.*Ví dụ 2*: Có truyền index

import pandas as pd

s = pd.Series([0,1,2,3], index=["a","b","c","d"])

print(s)

Output:

a 0

b 1

c 2

d 3

dtype: int64

*Ví dụ 3*: Tạo Series từ dict

import pandas as pd

data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}

ser = pd.Series(data,index=['a','c','b','d','e','f'])

print(ser)

Output:

a -1.3

c NaN

b 11.7

d 2.0

e NaN

f 10.0

dtype: float64

Chúng ta tạo dict có index a, b, d, f, g. Sau đó tạo Series từ dict data này nhưng các index c và e không có trong dict nên dữ liệu tại các index này bị thiếu (missing data). pandas hiển thị NaN để báo các dữ liệu này bị trống.

*Ví dụ* *4*: Tạo Series từ Scalar   
Nếu dữ liệu là một giá trị scalar, index phải được cung cấp. Giá trị sẽ được lặp lại để phù hợp với độ dài của index.

import pandas as pd

ser = pd.Series(5, index=[1, 2, 3, 4, 5])

print(ser)

Output:

1 5

2 5

3 5

4 5

5 5

dtype: int64

**Truy cập dữ liệu từ Series với index và vị trí**   
Truy cập dữ liệu của Series tương tự với ndarray trong NumPy.  
*Ví dụ 5*: Lấy dữ liệu tại index cụ thể

import pandas as pd

data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}

ser = pd.Series(data,index=['a','c','b','d','e','f'])

print(ser['d'])

print(ser['c'])

Output:

2.0

nan

*Ví dụ 6*: Lấy dữ liệu từ đầu đến vị trí index cụ thể

import pandas as pd

data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}

ser = pd.Series(data,index=['a','c','b','d','e','f'])

print(ser[:'d'])

Output:

a -1.3

c NaN

b 11.7

d 2.0

dtype: float64

*Ví dụ 7*: Lấy dữ liệu theo vị trí: 2 dữ liệu đầu

import pandas as pd

data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}

ser = pd.Series(data,index=['a','c','b','d','e','f'])

print(ser[:2])

Output:

a -1.3

c NaN

dtype: float64

*Ví dụ 8*: Lấy 3 dữ liệu cuối

import pandas as pd

data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}

ser = pd.Series(data,index=['a','c','b','d','e','f'])

print(ser[-3:])

Output:

d 2.0

e NaN

f 10.0

dtype: float64

**Chuyển đổi sang dạng khác**

*Ví dụ 9*: Lấy dạng array của Series bằng numpy.asarray

import pandas as pd

import numpy as np

data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}

ser = pd.Series(data,index=['a','c','b','d','e','f'])

a = np.asarray(ser)

print(a)

Output:

[-1.3 nan 11.7 2. nan 10. ]

Còn rất nhiều thao tác khác mà bạn thực hiện được với Series của pandas tại [ĐÂY](https://pandas.pydata.org/docs/reference/series.html).

**2. DataFrame**

DataFrame([data, index, columns, dtype, copy])

Dataframe là cấu trúc dữ liệu được gắn nhãn hai chiều với các cột và hàng như bảng tính (spreadsheet) hoặc bảng (table). Giống như Series, DataFrame có thể chứa bất kỳ loại dữ liệu nào. Một điều quan trọng cần làm nổi bật là tất cả các cột trong khung dữ liệu là series Pandas. Vì vậy, một DataFrame là sự kết hợp của nhiều Series đóng vai trò như các cột! DataFrame được sử dụng rộng rãi và là một trong những cấu trúc dữ liệu quan trọng nhất.

Chúng ta hãy bắt đầu với **tạo DataFrame**  
*Ví dụ 1*: Tạo DataFrame từ dict các Series 1

import pandas as pd

# tạo dict từ các series

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}

# tại DataFrame từ dict

df = pd.DataFrame(s)

print(df)

Output:

một hai

a 1.0 1.0

b 2.0 2.0

c 3.0 3.0

d NaN 4.0

e 5.0 NaN

*Ví dụ 2*: Tạo DataFrame từ dict các Series 2

import pandas as pd

# tạo các series

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}

# tạo DataFrame từ dict theo các index được chọn

df = pd.DataFrame(s, index=['a','c','d'])

print(df)

Output:

một hai

a 1.0 1.0

c 3.0 3.0

d NaN 4.0

Ngoài ra còn nhiều cách tạo DataFrame khác như từ dict các ndarray/list, từ list của dict, từ một Series, v.v.

**Các thao tác chọn, thêm, xóa cột**

*Ví dụ 3:* Chọn cột (column selection)

import pandas as pd

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),

'ba': pd.Series([9., -1.3, 3.5, 41.1], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(s)

# chọn cột hai

df\_hai = df['hai']

print(df\_hai)

Output:

a 1.0

b 2.0

c 3.0

d 4.0

e NaN

Name: hai, dtype: float64

*Ví dụ 4*: Một số cách thêm cột (column addition)

import pandas as pd

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),

'ba': pd.Series([9., -1.3, 3.5, 41.1], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(s)

# thêm cột bốn với giá trị mỗi ô theo công thức

df['bốn'] = df['hai'] - df['ba']

# thêm cột với giá trị vô hướng (scalar value)

df['Chuẩn'] = 'OK'

# thêm cột không cùng số lượng index với DataFrame

df['Khác'] = df['hai'][:3]

# thêm cột True/False theo điều kiện

df['KT'] = df['một'] == 3.0

# dùng hàm insert. Cột "chèn" bên dưới sẽ ở vị trí 2 (tính từ 0), có giá trị bằng cột một

df.insert(2, 'chèn', df['một'])

print(df)

Output:

một hai chèn ba bốn Chuẩn Khác KT

a 1.0 1.0 1.0 9.0 -8.0 OK 1.0 False

b 2.0 2.0 2.0 -1.3 3.3 OK 2.0 False

c 3.0 3.0 3.0 3.5 -0.5 OK 3.0 True

d NaN 4.0 NaN 41.1 -37.1 OK NaN False

e 5.0 NaN 5.0 NaN NaN OK NaN False

*Ví dụ 5*: Xóa cột (column deletion)  
Có thể xóa cột bằng lệnh def hoặc hàm pop

import pandas as pd

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),

'ba': pd.Series([9., -1.3, 3.5, 41.1], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(s)

# xóa cột hai

del df['hai']

# pop cột ba với dict tv\_ba

tv\_ba = df.pop('ba')

print( df)

Output:

một

a 1.0

b 2.0

c 3.0

d NaN

e 5.0

**Lập chỉ mục/ lựa chọn**

Những điều cơ bản của việc lập chỉ mục như sau

|  |  |  |
| --- | --- | --- |
| **Thi hành** | **Cú pháp** | **Kết quả** |
| Chọn cột | df[col] | Series |
| Chọn dòng theo label | df.loc[label] | Series |
| Chọn dòng theo vị trí nguyên | df.iloc[loc] | Series |
| Cắt các dòng | df[3:7] | DataFrame |
| Chọn các dòng theo vector boolean | df[bool\_vector] | DataFrame |

*Ví dụ 1*: Chọn dòng theo label

import pandas as pd

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),

'ba': pd.Series([9., -1.3, 3.5, 41.1], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(s)

# chọn dòng theo label

d = df.loc['a']

print(d)

Output:

một 1.0

hai 1.0

ba 9.0

Name: a, dtype: float64

*Ví dụ 2*: Chọn dòng theo vị trí nguyên

import pandas as pd

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),

'ba': pd.Series([9., -1.3, 3.5, 41.1], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(s)

# chọn dòng theo vị trí nguyên

d = df.iloc[4]

print(d)

Output:

một 5.0

hai NaN

ba NaN

Name: e, dtype: float64

*Ví dụ 3*: Cắt (slice) các dòng

import pandas as pd

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),

'ba': pd.Series([9., -1.3, 3.5, 41.1], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(s)

# cắt lấy ra từ dòng 3 đến dòng 4

d = df[2:4]

print(d)

Output:

một hai ba

c 3.0 3.0 3.5

d NaN 4.0 41.1

Bên trên là một số thao tác cơ bản với DataFrame. Bạn có thể tham khảo thêm chi tiết về DataFrame tại [ĐÂY](https://pandas.pydata.org/docs/reference/frame.html). Ngoài các thao tác với Series và DataFrame, pandas còn rất nhiều các chức năng tuyệt vời [khác](https://pandas.pydata.org/pandas-docs/stable/reference/index.html).

**Tạm kết**

pandas cung cấp một nền tảng vững chắc để thiết lập một hệ sinh thái phân tích dữ liệu rất mạnh. Sự xuất hiện của thư viện này đã giúp cho Python trở thành một lựa chọn hấp dẫn cho các ứng dụng phân tích dữ liệu. Nếu bạn có bất kỳ góp ý xin hãy để lại lời nhắn ở phần bình luận và đánh giá 5 sao nếu bạn thấy hay. Cảm ơn các bạn đã đọc. Hẹn gặp lại các bạn trong các bài viết tiếp theo.