12. Lập chỉ mục nâng cao (Advanced or Multi-Level Indexing, labeling)

Chúng ta đã học cách tạo và tương tác với các đối tượng Series và DataFrame, Panel. Trong phần này, chúng ta sẽ học thêm một số khía cạnh của các cấu trúc. Chúng tôi sẽ bắt đầu với khả năng lập chỉ mục nâng cao trong Pandas.

Trước khi bắt đầu, chúng ta cùng ôn lại về pandas Data Structure. Các khối thành phần chính của pandas data structure là:

1. **Indexes**: Có hai tính chất cơ bản là bất biến (Immutable) và các chỉ mục phải là cùng kiểu dữ liệu (Homogenous in data type).

2. **Series**: kiểu 1D array và có index

3. **Dataframe**: 2D array và mỗi cột của dataframe là kiểu Series.

Chỉ mục nâng cao hoặc đa cấp có sẵn cho cả Series và DataFrames. Đây là một cách thú vị để làm việc với dữ liệu nhiều chiều, sử dụng cấu trúc dữ liệu Pandas. Đây là một cách hiệu quả để lưu trữ và thao tác với dữ liệu nhiều chiều tùy ý trong cấu trúc dạng bảng 1 chiều (series) và 2 chiều (dataframe). Nói cách khác, chúng ta có thể làm việc với dữ liệu nhiều chiều hơn trong một dữ liệu có chiều thấp hơn. Đã đến lúc trình bày một ví dụ:

```
>>> import pandas as pd
>>> cities = ["Vienna", "Vienna", "Vienna","Hamburg", "Hamburg", "Hamburg","Berlin", "Berlin", "Berlin","Zürich", "Zürich", "Zürich"]
>>> index = [cities, ["country", "area", "population","country", "area", "population","country", "area", "population","country", "area", "population"]]
>>> print(index)
[['Vienna', 'Vienna', 'Vienna', 'Hamburg', 'Hamburg', 'Hamburg', 'Berlin', 'Berlin', 'Berlin', 'Z\x81rich', 'Z\x81rich', 'Z\x81rich'], ['country', 'area', 'population', 'country', 'area', 'population', 'country', 'ar
>>> data = ["Austria", 414.60,     1805681,"Germany",   755.00,   1760433,"Germany",   891.85,     3562166,"Switzerland", 87.88, 378884]
>>> city_series = pd.Series(data, index=index)
>>> print(city_series)
Vienna   country          Austria
         area               414.6
         population       1805681
Hamburg  country          Germany
         area                755
         population       1760433
Berlin   country          Germany
         area              891.85
         population       3562166
Z?rich   country      Switzerland
         area               87.88
         population        378884
dtype: object
>>>
```

Truy cập dữ liệu

Ta có thể truy cập dữ liệu của một thành phố qua cách sau:**df[index][label]** hoặc **df[index, label]**

```
>>> print(city_series["Vienna"])
country       Austria
area            414.6
population    1805681
dtype: object
>>>
```

Ta cũng có thể truy cập thông tin về country, area hoặc population của một city. Chúng ta có thể làm điều này theo hai cách:

```
>>> print(city_series["Vienna"]["area"])
414.6
>>> print(city_series["Vienna", "area"])
414.6
```

Nếu chỉ mục được sắp xếp, ta cũng có thể áp dụng kĩ thuật slicing để truy cập:

```
>>> city_series = city_series.sort_index()
>>> print("city_series with sorted index:")
city_series with sorted index:
>>> print(city_series)
Berlin   area             891.85
         country         Germany
```

```
        population       3562166
Hamburg  area                755
        country          Germany
        population       1760433
Vienna   area               414.6
        country          Austria
        population       1805681
Z?rich   area               87.88
        country       Switzerland
        population        378884
dtype: object
>>> print("\n\nSlicing the city_series:")
Slicing the city_series:
>>> print(city_series["Berlin":"Vienna"])
Berlin   area           891.85
        country       Germany
        population    3562166
Hamburg  area             755
        country       Germany
        population    1760433
Vienna   area           414.6
        country       Austria
        population    1805681
dtype: object
>>>
```

Nó là hoàn toàn có thể trong việc truy cập các key sâu hơn, như area.

```
>>> print(city_series[:, "area"])
Berlin     891.85
Hamburg       755
Vienna      414.6
Z?rich      87.88
dtype: object
>>>
```

Sử dụng multicolumn dataframe

Ví dụ ta tạo ra một multicolumn dataframe như sau:

```
>>> columns = [['bar', 'bar', 'baz', 'baz', 'foo', 'foo', 'qux', 'qux'],['one', 'two', 'one', 'two', 'one', 'two', 'one', 'two']]
>>> df = pd.DataFrame(np.random.randn(3, 8), index=['A', 'B', 'C'], columns=columns)
>>> df
       bar                 baz                 foo                 qux  \
       one       two       one       two       one       two       one
A -1.454281  0.512414 -1.455039  0.110496  1.939493 -1.038009 -0.754450
B  2.325193 -1.616415 -0.948290  0.009425 -0.594611  0.519266  0.022691
C -2.006198 -0.220591  1.377502  0.491257  0.557574 -0.816445 -0.303094



       two
A -0.627585
B -0.209629
C -0.342264
>>>
```

Truy cập vào từng cột như sau:

```
>>> df.bar.one
```

```
A   -1.454281
B    2.325193
C   -2.006198
Name: one, dtype: float64
>>> df["bar","one"]
A   -1.454281
B    2.325193
C   -2.006198
Name: (bar, one), dtype: float64
>>> df['bar']['one']
A   -1.454281
B    2.325193
C   -2.006198
Name: one, dtype: float64
>>>
```

Thực hiện chuyển vị bảng để có kết quả multiindex dataframe.

```
>>> df = df.T
>>> df
                A         B         C
bar one -1.454281  2.325193 -2.006198
    two  0.512414 -1.616415 -0.220591
baz one -1.455039 -0.948290  1.377502
    two  0.110496  0.009425  0.491257
foo one  1.939493 -0.594611  0.557574
    two -1.038009  0.519266 -0.816445
qux one -0.754450  0.022691 -0.303094
    two -0.627585 -0.209629 -0.342264
>>>
```

Truy cập dựa trên chỉ mục hàng giống với bài trước, có thể dùng ix hoặc loc như sau:

```
>>> df.loc['bar','one']
A   -1.454281
B    2.325193
C   -2.006198
Name: (bar, one), dtype: float64
>>> df.ix['bar','one']
A   -1.454281
B    2.325193
C   -2.006198
Name: (bar, one), dtype: float64
>>>
```

Kết hợp với slicing ta có kết quả sau:

```
>>> df.ix['baz':'foo']
                A         B         C
baz one -1.455039 -0.948290  1.377502
    two  0.110496  0.009425  0.491257
foo one  1.939493 -0.594611  0.557574
    two -1.038009  0.519266 -0.816445
>>> df.loc['baz':'foo']
                A         B         C
baz one -1.455039 -0.948290  1.377502
    two  0.110496  0.009425  0.491257
foo one  1.939493 -0.594611  0.557574
```

```
   two -1.038009  0.519266 -0.816445

>>>

>>> df['baz':'foo']

             A         B         C

baz one -1.455039 -0.948290  1.377502

   two  0.110496  0.009425  0.491257

foo one  1.939493 -0.594611  0.557574

   two -1.038009  0.519266 -0.816445
```

Như ta đã biết một tuple, ví dụ ('bar', 'two') là một key giúp ta truy xuất trong trường hợp multiindex/multicolumn dataframe, ta có thể truyền vào một list các tuples để có thể nhận được nhiều hàng hay cột tùy ý như ví dụ dưới đây:

```
>>> df.loc[[('bar', 'two'), ('qux', 'one')]]

             A         B         C

bar two  0.512414 -1.616415 -0.220591

qux one -0.754450  0.022691 -0.303094

>>>
```

Các bạn sẽ nhận thấy khó trong việc chỉ lấy ra các giá trị chỉ mục "one" ở layer thứ 2 của tất cả các chỉ mục của layer 1. Kĩ thuật sclicing và cụ thể lớp slicers sinh ra để giải quyết việc này.

Cú pháp tạo một slice là : df.loc[(slice('A1','A3'),.....), :]

Nếu slice(None) nghĩa là bạn lựa chọn hết các chỉ mục. Các ví dụ sau sẽ giúp bạn hiểu kĩ hơn.

Trường hợp này slice(None) tương ứng với ':'

```
>>> df.loc[(slice(None)),:]
             A         B         C
bar one -1.454281  2.325193 -2.006198
   two  0.512414 -1.616415 -0.220591
baz one -1.455039 -0.948290  1.377502
   two  0.110496  0.009425  0.491257
foo one  1.939493 -0.594611  0.557574
   two -1.038009  0.519266 -0.816445
qux one -0.754450  0.022691 -0.303094
   two -0.627585 -0.209629 -0.342264
```

Slicing từ chỉ mục 'bar' đến 'foo'.

```
>>> df.loc[(slice("bar","foo")),:]
             A         B         C
bar one -1.454281  2.325193 -2.006198
   two  0.512414 -1.616415 -0.220591
baz one -1.455039 -0.948290  1.377502
   two  0.110496  0.009425  0.491257
foo one  1.939493 -0.594611  0.557574
   two -1.038009  0.519266 -0.816445
```

Có 2 lớp indexes, ta muốn chọn tất cả các index có chứa "one".

```
>>> df.loc[(slice(None),slice("one")),:]
             A         B         C
bar one -1.454281  2.325193 -2.006198
baz one -1.455039 -0.948290  1.377502
foo one  1.939493 -0.594611  0.557574
qux one -0.754450  0.022691 -0.303094
```

Sau ví dụ trên các bạn có thể hình dung cách slicing đối với dataframe. Còn cách khác tự nhiên hơn đó là dùng IndexSlice

```
>>> idx = pd.IndexSlice
>>> df.loc[idx[:,:],["B","C"]]
             B         C
bar one  2.325193 -2.006198
   two -1.616415 -0.220591
baz one -0.948290  1.377502
   two  0.009425  0.491257
foo one -0.594611  0.557574
   two  0.519266 -0.816445
qux one  0.022691 -0.303094
   two -0.209629 -0.342264
>>> print "Select index 'one' of all index of 1st layer"
Select index 'one' of all index of 1st layer
>>> df.loc[idx[:,"one"],["B","C"]]
             B         C
bar one  2.325193 -2.006198
baz one -0.948290  1.377502
foo one -0.594611  0.557574
qux one  0.022691 -0.303094
>>>
```

Kết hợp slicing với Filtering, hãy chọn ra cột 'B' và 'C' sao cho cột A không âm. Chỉ nhận giá trị của chỉ mục "one"

```
>>> df.loc[idx[:,"one"],["B","C"]][df.A>0]

             B         C

foo one -0.594611  0.557574
```

Hoặc

```
>>> mask = df['A'] > 0
>>> mask
bar  one    False
     two     True
baz  one    False
     two     True
foo  one     True
     two    False
qux  one    False
     two    False
Name: A, dtype: bool
>>> df.loc[idx[mask,['one']],["B","C"]]
                B         C
foo one -0.594611  0.557574
>>>
```

Kết luận

Chúng ta thực sự đã tìm hiểu khá nhiều phương pháp để truy cập và thao tác dựa trên chỉ mục của các đối tượng series và dataframe trong Pandas. Bài học này đem đến một cái nhìn sâu rộng hơn và mang nhiều ý nghĩa hơn với việc sử dụng chỉ mục nâng cao để làm việc với dữ liệu nhiều chiều. Các bạn hoàn toàn có thể áp dụng kĩ thuật slicing, ix, loc đã biết từ những bài học trước hoặc slice, pd.IndexSlice để làm việc với các lớp chỉ mục hàng/cột linh hoạt hơn.

Như vậy sau hơn 10 bài chúng ta đã có thể tạo và tương tác với pandas data structure tương đối thoải mái.

1. Tạo Series, dataframe, panel

2. Selection, indexing, labeling

3. Filtering