Giới thiệu về DataFrame

1 Giới thiệu

`pandas.DataFrame` (gọi tắt là `DataFrame`) là một mảng hai chiều có gắn nhãn.

`DataFrame` có một số đặc điểm sau:

- `DataFrame` là một mảng hai chiều.
- `DataFrame` có thể xem như là nhiều `Series` có chung label (index) được ghép kế tiếp nhau.

[1]:
```
# ví dụ DataFrame
import pandas as pd

pd.DataFrame({
    'Product' : ['Apple', 'Banana', 'Cherry'],
    'Quantity' : [12, 34, 56],
    'Price' : [10, 5, 8]
})
```

[1]:

	Product	Quantity	Price
0	Apple	12	10
1	Banana	34	5
2	Cherry	56	8

2 Cách khởi tạo `DataFrame`

Để khởi tạo một `DataFrame`, bạn có thể dùng cú pháp sau :

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

Trong đó:

- `data`: dữ liệu truyền vào (có thể bỏ trống).
- `index`: label của từng dòng (có thể bỏ trống).
- `columns`: label của từng cột (có thể bỏ trống).

Ngoài ra, còn 2 tham số khác nữa là `dtype` và `copy`.

Cùng xem một số ví dụ để hiểu rõ hơn về cách tạo `DataFrame`.

[2]:
```
# tạo DataFrame từ list
l = [0, 1, 2, 3, 5]
pd.DataFrame(data = l)
```

[2]:

	0
0	0
1	1
2	2
3	3
4	5

[3]:
```
# Tạo DataFrame từ list mà các phần tử là list
ll = [
  ['Apple', 100],
  ['Banana', 25],
  ['Cherry', 36]
]
pd.DataFrame(ll)
```

[3]:

	0	1
0	Apple	100
1	Banana	25
2	Cherry	36

[4]:
```
# tạo DataFrame, có chỉ ra index và columns
pd.DataFrame(
    ll,
    index = ['a', 'b', 'c'],
    columns = ['Product', 'Quantity']
)
```

[4]:

	Product	Quantity
a	Apple	100
b	Banana	25
c	Cherry	36

Câu hỏi : Nếu các `list` con trong `ll` có độ dài khác nhau thì có thể tạo được `DataFrame` từ `ll` hay không?

[5]:
```
# tạo DataFrame từ dictionary
d1 = {
    'col1' : ['Apple', 'Banana'],
    'col2' : [1, 2],
    'col3' : ['2019-10-02', '2019-11-01']
}

pd.DataFrame(d1, index = [1, 2])
```

[5]:
```
     col1  col2        col3
1   Apple     1  2019-10-02
2  Banana     2  2019-11-01
```

Câu hỏi: Nếu độ dài các `list` khác nhau thì có thể tạo được `DataFrame` từ `d1` hay không?

Tạo `DataFrame` từ `list` các `dictionary`

[6]:
```
# tạo list mà mỗi phần tử là một dictionary
ld = [
      {'Date' : '2019-10-01', 'Ticker' : 'AAA', 'Open' : 100},
      {'Date' : '2019-10-01', 'Ticker' : 'BBB', 'Close' : 200}
]
# tạo DataFrame
pd.DataFrame(ld)
```

[6]:
```
         Date Ticker   Open  Close
0  2019-10-01    AAA  100.0    NaN
1  2019-10-01    BBB    NaN  200.0
```

Câu hỏi : Trong trường hợp `ld` ở trên, nếu `dictionary` thứ 2 là

`{'Date' : '2019-10-01', 'Ticker' : 'BBB', 'Open': 205, 'Close' : 200}`

thì có tạo được `DataFrame` từ `d2` hay không?

Bài tập : Tạo một `DataFrame` rỗng?

3 Các thao tác với `DataFrame`

Trong phần này, chúng ta sẽ sử dụng `DataFrame` mẫu sau

[7]:
```
import numpy as np

d2 = {
    'col1' : pd.date_range(start = '2019-11-20', periods = 10, freq = 'D'),
    'col2' : np.random.choice(['Apple', 'Banana', 'Cherry'], size = 10),
    'col3' : np.random.randint(100, size = 10)
}
df = pd.DataFrame(d2, index = list('abcdefghij'))
df
```

[7]:
```
        col1    col2  col3
a 2019-11-20  Banana    73
b 2019-11-21  Cherry    24
c 2019-11-22   Apple    12
d 2019-11-23   Apple    65
e 2019-11-24   Apple    95
f 2019-11-25   Apple    98
g 2019-11-26  Banana    36
```

```
h 2019-11-27  Banana    73
i 2019-11-28   Apple     9
j 2019-11-29  Cherry    55
```

3.1 Thông tin cơ bản của một `DataFrame`

Bạn có thể xem một số thông tin cơ bản của `DataFrame` thông qua những thuộc tính sau:

- `.size`: trả về số lượng phần tử của `DataFrame`.
- `.shape`: trả về (dòng, cột) của `DataFrame`.
- `.empty`: trả về `True` nếu `DataFrame` là rỗng.
- `.dtypes`: trả về kiểu dữ liệu của từng cột.
- `.columns`: trả về danh sách các cột của `DataFrame`.
- `.index`: trả về danh sách các dòng của `DataFrame`.

Câu hỏi : Giả sử một `DataFrame` được tạo như sau

```
df1 = pd.DataFrame(columns = ['x', 'y', 'z'])
```

Vậy, `df1` có rỗng hay không?

3.2 Đổi tên dòng và cột

Để đổi tên dòng và cột, ta có thể dùng phương thức `.rename()` như sau :

```
<tên_DataFrame>.rename(index = <tên_dòng_mới>, columns = <tên_cột_mới>)
```

Trong đó, `tên_dòng_mới` và `tên_cột_mới` là `dictionary` với cấu trúc

```
{<tên_cũ_1> : <tên_mới_1>, <tên_cũ_2> : <tên_mới_2>, ...}
```

[8]:
```
# đổi tên cột `col1` thành `Date`
df.rename(columns = {'col1' : 'Date'})
name = {'col1': 'Date', 'col2': 'Product'}
df2 = df.rename(columns = name)
```

Lưu ý:

- `.rename()` sẽ trả về một `DataFrame` mới, không làm thay đổi `DataFrame` gốc. Nếu muốn thay đổi trực tiếp trên `DataFrame` gốc, thêm tham số `inplace = True`.
- Có thể bỏ qua `index` nếu chỉ đổi tên cột và bỏ qua `columns` nếu chỉ đổi tên dòng.

Ngoài ra, bạn có thể đổi tên *nhanh* toàn bộ cột theo cách sau :

```
<tên_DataFrame>.columns = <danh_sách_tên_cột_mới>
```

Cách này cũng đổi trực tiếp tên cột trong `DataFrame`.

Tương tự bạn có thể đổi tên *nhanh* toàn bộ dòng theo cách sau :

```
<tên_DataFrame>.index = <danh_sách_tên_dòng_mới>
```

[9]:
```
# tạo bản sao của df
df2 = df.copy()
# đổi tên toàn bộ cột của df2
df2.columns = ['Date', 'aaa', 'bbb']
df2
```

[9]:
```
        Date     aaa  bbb
a 2019-11-20  Banana   73
b 2019-11-21  Cherry   24
c 2019-11-22   Apple   12
d 2019-11-23   Apple   65
e 2019-11-24   Apple   95
f 2019-11-25   Apple   98
g 2019-11-26  Banana   36
h 2019-11-27  Banana   73
i 2019-11-28   Apple    9
j 2019-11-29  Cherry   55
```

3.3 Trích xuất dữ liệu theo cột

Để lấy dữ liệu từ 1 cột, ta có thể thực hiện theo 2 cách : 1. `<tên_DataFrame>.<tên_cột>`, hoặc 2. `<tên_DataFrame>[<tên_cột>]`

Lưu ý là bạn chỉ có thể dùng cách 1 khi mà tên cột **không** chứa khoảng trắng.

[10]:
```
# lấy dữ liệu từ cột có tên 'col1'
df['col1']
```

[10]:
```
a   2019-11-20
b   2019-11-21
c   2019-11-22
d   2019-11-23
e   2019-11-24
f   2019-11-25
g   2019-11-26
h   2019-11-27
i   2019-11-28
j   2019-11-29
Name: col1, dtype: datetime64[ns]
```

Câu hỏi : Kiểu dữ liệu của `df['col1']` trong ví dụ trên là gì?

Để lấy dữ liệu từ nhiều cột, ta thực hiện như sau

`<tên_DataFrame>[<danh_sách_tên_cột>]`

[11]:
```
# lấy dữ liệu từ hai cột 'col1' và 'col2'
df[['col1', 'col2']]
```

[11]:
```
        col1    col2
a 2019-11-20  Banana
b 2019-11-21  Cherry
c 2019-11-22   Apple
d 2019-11-23   Apple
e 2019-11-24   Apple
f 2019-11-25   Apple
g 2019-11-26  Banana
h 2019-11-27  Banana
i 2019-11-28   Apple
j 2019-11-29  Cherry
```

Câu hỏi : Phân biệt `df['col1']` và `df[['col1']]`.

3.4 Trích xuất dữ liệu theo dòng

Giống như `Series`, bạn có thể trích xuất dữ liệu từ một (hoặc nhiều) dòng thông qua label hoặc số thứ tự của dòng.

Tuy nhiên, bạn lại không thể dùng

`<tên_DataFrame>[<label_dòng_cần lấy>]`

vì đây là cách dùng để trích xuất cột.

[12]:
```
# Thử gọi dòng có label là 'a'
df['a'] #báo lỗi
```

Để lấy dữ liệu theo dòng, có thể dùng:

- `.loc`: lấy dữ liệu dòng theo tên.
- `.iloc`: lấy dữ liệu dòng theo số thứ tự.

[13]:
```
# Gọi dòng có label là 'a' bằng .loc
df.loc['a']
```

[13]:
```
col1    2019-11-20 00:00:00
col2                 Banana
col3                     73
Name: a, dtype: object
```

[14]:
```
# Gọi dòng có số thứ tự 2 bằng .iloc
df.iloc[2]
```

[14]:
```
col1    2019-11-22 00:00:00
col2                  Apple
col3                     12
Name: c, dtype: object
```

Bài tập : Lấy hai dòng có label là `'a'` và `'c'` từ `df`.

3.5 Trích xuất theo cả dòng và cột

Ngoài việc giúp bạn có thể lấy dòng theo label, `.loc` còn thể giúp bạn lấy thêm những cột mong muốn (bằng label) theo cú pháp:

`<tên_DataFrame>.loc[<dòng_cần_lấy>, <cột_cần_lấy>]`

Tương tự, `.iloc` sẽ giúp bạn lấy dòng và cột theo số thứ tự.

[15]:
```
# Lấy dòng có label 'a' và 'c', cột có label là 'col2' và 'col3'
df.loc[['a', 'c'], ['col2', 'col3']]
```

[15]:

	col2	col3
a	Banana	73
c	Apple	12

3.6 Trích xuất theo điều kiện (Lọc)

Ngoài ra, `DataFrame` còn cho phép bạn lấy dữ liệu theo một điều kiện nào đó theo cú pháp:

`<tên_DataFrame>[điều_kiện]`

`điều_kiện` ở đây có thể là điều kiện đơn hoặc điều kiện ghép (tức là được ghép từ các điều kiện đơn).

[16]:
```
# Lọc từ df những dòng có cột 'col2' là 'Apple'
df[(df.col2 == 'Apple')]
```

[16]:

	col1	col2	col3
c	2019-11-22	Apple	12
d	2019-11-23	Apple	65
e	2019-11-24	Apple	95
f	2019-11-25	Apple	98
i	2019-11-28	Apple	9

Để ghép nhiều điều kiện lại với nhau bạn có thể dùng toán tử `&` cho phép toán `and`, `|` cho phép toán `or` và `~` cho phép toán `not`.

Một điểm cần lưu ý khi ghép, các điều kiện đơn phải nằm trong dấu `()`. Ví dụ:

`(điều_kiện_1) & (điều_kiện_2) | (điều_kiện_3)`

Bài tập : Lấy những dòng có `'col2'` là `'Banana'` và `'col3'` > 50