



Bài 10: Pandas 2

TS. Trịnh Tuấn Đạt
Trường CNTT-TT, ĐHBK Hà Nội

1

1



Nội dung

1. Thao tác đọc/ghi dữ liệu với file
2. Các thao tác cơ bản với Series và DataFrame
3. Đặc tả thống kê (Descriptive Statistics)
4. Sắp xếp trên DataFrame (Sorting)
5. Cắt nhỏ DataFrame (Slicing)
6. Lọc DataFrame (Filtering)
7. Duyệt trên Series và DataFrame (Iteration)
8. Đánh lại nhãn cho DataFrame (Reindexing)
9. Giống nhãn với reindex_like

2

2



Nội dung

- 1. Thao tác đọc/ghi dữ liệu với file**
2. Các thao tác cơ bản với Series và DataFrame
3. Đặc tả thống kê (Descriptive Statistics)
4. Sắp xếp trên DataFrame (Sorting)
5. Cắt nhỏ DataFrame (Slicing)
6. Lọc DataFrame (Filtering)
7. Duyệt trên Series và DataFrame (Iteration)
8. Đánh lại nhãn cho DataFrame (Reindexing)
9. Giống nhãn với `reindex_like`

3

3



1. Thao tác đọc/ghi dữ liệu với file

- Các định dạng file được hỗ trợ trong Pandas
 - Excel
 - CSV - Comma Separated Values
 - JSON

4

4



Đọc dữ liệu từ file CSV

- CSV là dạng thức đơn giản nhất để lưu trữ dữ liệu cấu trúc bảng dưới dạng thuần text
- Là định dạng dùng rất phổ biến trong khoa học dữ liệu
- **Comma Separated Values:** Các giá trị ngăn cách bằng dấu , nhưng có thể ngăn cách bằng dấu ;
- Cấu trúc file CSV

```
Salary_Data - Notepad
File Edit Format View Help
YearsExperience,Salary
1.1,39343.00
1.3,46205.00
1.5,37731.00
2.0,43525.00
2.2,39891.00
2.9,56642.00
3.0,60150.00
3.2,54445.00
3.2,64445.00
```

header

observations / records

5

5



Đọc dữ liệu từ file CSV

- Nếu dữ liệu chứa dấu , cần đặt trong cặp dấu "

```
emp_data_2_with_quotes.txt
1 id,name,organization,address,contact
2 1,Ranjith,tcs,"gachibowli, hyderabad",1234567890
3 2,sudha,altimetrik,"electonic city, bangalore",78945632210
4 3,Micheal,wipro,"dlf info city,chennai",2031456879
5 4,kiran,infosys,"Rajiv Gandhi Infotech Park,pune",5520369741
6 5,Ramesh,accenture,"financial district, hyderabad",5522001144
7 6,mohammed,capegemini,"Rajiv Gandhi Infotech Park, pune",10023654989
```

6

6



Đọc dữ liệu từ file CSV

- Đọc dữ liệu file CSV vào DataFrame

Duration,Pulse,Maxpulse,Calories

60,110,130,409.1

60,117,145,479.0

60,103,135,340.0

45,109,175,282.4

45,117,148,406.0

60,102,127,300.5

60,110,136,374.0

45,104,134,253.3

30,109,133,195.1

60,98,124,269.0

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df)
```

Duration Pulse Maxpulse Calories

0 60 110 130 409.1

1 60 117 145 479.0

2 60 103 135 340.0

3 45 109 175 282.4

4 45 117 148 406.0

...

164 60 105 140 290.8

165 60 110 145 300.4

166 60 115 145 310.2

167 75 120 150 320.4

168 75 125 150 330.4

[169 rows x 4 columns]

7

7



Đọc dữ liệu từ file JSON

- JSON: **J**ava**S**cript **O**bject **N**otation
- Là định dạng text, dùng để lưu trữ và **truyền tải dữ liệu**
- JSON có tính tự mô tả (self-describing), dễ hiểu
- Dùng rộng rãi trong hầu hết các ngôn ngữ lập trình
- Ví dụ một chuỗi dữ liệu lưu dưới dạng JSON:

```
{"name": "John", "age": 30, "car": null}
```

- Các ngôn ngữ lập trình (Python, Java Script, Java, PHP, ...) có thể phân tích một chuỗi JSON thành các đối tượng JSON tương ứng

8

8



Ví dụ dữ liệu định dạng JSON

```
["Sunday", "Monday", "Tuesday",  
 "Wednesday", "Thursday", "Friday",  
 "Saturday"]
```

```
{  
  "name": "John",  
  "age": 30,  
  "cars": ["Ford", "BMW", "Fiat"]  
}
```

```
{  
  "Duration": {  
    "0": 60,  
    "1": 60,  
    "2": 60,  
    "3": 45,  
    "4": 45,  
    "5": 60  
  },  
  "Pulse": {  
    "0": 110,  
    "1": 117,  
    "2": 103,  
    "3": 109,  
    "4": 117,  
    "5": 102  
  },  
  "Maxpulse": {  
    "0": 130,  
    "1": 145,  
    "2": 135,  
    "3": 175,  
    "4": 148,  
    "5": 127  
  }  
}
```

9

9



Các kiểu dữ liệu trong JSON

1	Number Số nguyên, số thực
2	String Đặt trong cặp nháy đơn/nháy kép
3	Boolean <i>true</i> hoặc <i>false</i>
4	Array (Mảng) Giống như kiểu List trong Python
5	Object (Đối tượng) an unordered collection of key:value pairs
6	null giá trị rỗng (không có giá trị)

JSON là sự kết hợp của **Dict** và **List** trong Python

10

10



Đọc dữ liệu từ file JSON

```
import pandas as pd
```

```
df = pd.read_json('data.json')
```

```
print(df)
```

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
...
164	60	105	140	290.8
165	60	110	145	300.4
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

[169 rows x 4 columns]

```
{  "Duration":{
    "0":60,
    "1":60,
    "2":60,
    "3":45,
    "4":45,
    "5":60
  },
  "Pulse":{
    "0":110,
    "1":117,
    "2":103,
    "3":109,
    "4":117,
    "5":102
  },
  "Maxpulse":{
    "0":130,
    "1":145,
    "2":135,
    "3":175,
    "4":148,
    "5":127
  },
  "Calories":{
    "0":409.1,
    "1":479.0,
    "2":340.0,
    "3":282.4,
    "4":406.0,
    "5":300.5
  }
}
```

11

11



Đọc dữ liệu từ file JSON

So sánh sự dễ dàng khi đọc dữ liệu từ file JSON, CSV, và file Excel?

12

12



Ghi dữ liệu ra file CSV, JSON

- Khi ghi dữ liệu ra file CSV, có thể bỏ qua nhãn hàng (index=false), bỏ qua nhãn cột (header=false), chọn cột muốn ghi (tham số columns)

```
import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])
print(df)
```

```
df.to_csv('data1.csv');
df.to_csv('data2.csv', index = False, columns=["Name"]);
df.to_csv('data3.csv', header = False, columns=["Age"]);
```

```
df.to_json('data.json');
```

	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

data1		
	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

data2	
	Name
	Tom
	Jack
	Steve
	Ricky

data3	
rank1	28
rank2	34
rank3	29
rank4	42

13

13



Nội dung

- Thao tác đọc/ghi dữ liệu với file
- Các thao tác cơ bản với Series và DataFrame**
- Đặc tả thống kê (Descriptive Statistics)
- Sắp xếp trên DataFrame (Sorting)
- Cắt nhỏ DataFrame (Slicing)
- Lọc DataFrame (Filtering)
- Duyệt trên Series và DataFrame (Iteration)
- Đánh lại nhãn cho DataFrame (Reindexing)
- Giống nhãn với reindex_like

14

14



Các thao tác cơ bản với Series

```
import pandas as pd
```

```
calories = {
    "day1": 420,
    "day2": 380,
    "day3": 390,
    "day4": 350,
    "day5": 410,
    "day6": 310}
```

```
myvar = pd.Series(calories)
```

```
print(myvar.axes) # Lấy các nhãn (hoặc myvar.index)
print(myvar.empty) # Series có rỗng?
print(myvar.ndim) # Số chiều dữ liệu (là 1)
print(myvar.size) # Kích thước Series
print(myvar.values) # Series dưới dạng List
print(myvar.head()) # Tạo series con: 5 phần tử đầu
print(myvar.head(3)) # Tạo series con: 3 phần tử đầu
print(myvar.tail()) # Tạo series con: 5 phần tử cuối
print(myvar.tail(2)) # Tạo series con: 2 phần tử cuối
```

```
[Index(['day1', 'day2',
'day3', 'day4', 'day5',
'day6'], dtype='object')]
False
1
6
[420 380 390 350 410 310]
day1    420
day2    380
day3    390
day4    350
day5    410
dtype: int64
day1    420
day2    380
day3    390
dtype: int64
day2    380
day3    390
day4    350
day5    410
day6    310
dtype: int64
day5    410
day6    310
dtype: int64
```

15

15



Các thao tác cơ bản với DataFrame

- Chuyển vị (Transpose: chuyển hàng → cột, cột → hàng)

```
import pandas as pd
```

```
data = {
    'Name': ['Tom', 'Jack', 'Steve', 'Ricky'],
    'Age': [28, 34, 29, 42]}
```

```
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])
```

```
print(df)
print(df.T)
```

	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

	rank1	rank2	rank3	rank4
Name	Tom	Jack	Steve	Ricky
Age	28	34	29	42

16

16



Các thao tác cơ bản với DataFrame

■ Lấy ra nhãn hàng, cột

```
import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])

print(type(df.axes))
print(df.axes)

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

	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

```
<class 'list'>
[Index(['rank1', 'rank2', 'rank3', 'rank4'], dtype='object'),
Index(['Name', 'Age'], dtype='object')]

Index(['rank1', 'rank2', 'rank3', 'rank4'], dtype='object')
Index(['Name', 'Age'], dtype='object')
```

17

17



Các thao tác cơ bản với DataFrame

■ Lấy ra kiểu dữ liệu cho các cột

```
import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])

print(df.dtypes)
print()
print(type(df.dtypes))
```

	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

```
Name      object
Age      int64
dtype: object

<class 'pandas.core.series.Series'>
```

18

18



Các thao tác cơ bản với DataFrame

■ Các thao tác khác

```
import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])

print(df.empty) # Có rỗng không
print(df.ndim) # Số chiều - luôn là 2
print(df.shape) # Số hàng, số cột
print(df.size) # Tổng số phần tử (hàng, cột)
print(df.values) # Trả về dưới dạng list
```

False
2
(4, 2)
8
[['Tom' 28]
 ['Jack' 34]
 ['Steve' 29]
 ['Ricky' 42]]

	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

19

19



Các thao tác cơ bản với DataFrame

■ Phương thức head, tail

```
import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])

print(df.head())
print(df.head(1))
print(df.tail())
print(df.tail(2))
```

rank1 Tom 28
rank2 Jack 34
rank3 Steve 29
rank4 Ricky 42

Name Age
rank1 Tom 28

Name Age
rank1 Tom 28
rank2 Jack 34
rank3 Steve 29
rank4 Ricky 42

Name Age
rank3 Steve 29
rank4 Ricky 42

20

20

Nội dung

1. Thao tác đọc/ghi dữ liệu với file
2. Các thao tác cơ bản với Series và DataFrame
- 3. Đặc tả thống kê (Descriptive Statistics)**
4. Sắp xếp trên DataFrame (Sorting)
5. Cắt nhỏ DataFrame (Slicing)
6. Lọc DataFrame (Filtering)
7. Duyệt trên Series và DataFrame (Iteration)
8. Đánh lại nhãn cho DataFrame (Reindexing)
9. Giống nhãn với reindex_like

21

21

Đặc tả thống kê

- Xét DataFrame sau:

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Ricky	30	3.20
5	Smith	29	4.60
6	Jack	23	3.80
7	Lee	34	3.78
8	David	40	2.98
9	Gasper	30	4.80
10	Betina	51	4.10
11	Andres	46	3.65

22

22



Đặc tả thống kê – hàm sum

```
import pandas as pd
import numpy as np

d = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin', 'Steve', 'Smith', 'Jack',
            'Lee', 'David', 'Gasper', 'Betina', 'Andres'],
    'Age': [25, 26, 25, 23, 30, 29, 23, 34, 40, 30, 51, 46],
    'Rating': [4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8, 3.78, 2.98, 4.80, 4.10, 3.65]
}

df = pd.DataFrame(d)
sum = df.sum() # Tương đương với df.sum(axis = 0): cộng trên cột
print(type(sum))
print(sum)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Smith	29	4.60
6	Jack	34	3.78
7	Lee	40	2.98
8	David	30	4.80
9	Gasper	51	4.10
10	Betina	46	3.65
11	Andres		

```
<class 'pandas.core.series.Series'>
Name      TomJamesRickyVinSteveSmithJackLeeDavidGasperBe...
Age                      382
Rating                     44.92
dtype: object
```

23



Đặc tả thống kê – hàm sum

```
import pandas as pd
import numpy as np

d = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin', 'Steve', 'Smith', 'Jack',
            'Lee', 'David', 'Gasper', 'Betina', 'Andres'],
    'Age': [25, 26, 25, 23, 30, 29, 23, 34, 40, 30, 51, 46],
    'Rating': [4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8, 3.78, 2.98, 4.80, 4.10, 3.65]
}

df = pd.DataFrame(d)
sum = df.sum(axis = 1) # Cộng trên hàng
print(type(sum))
print(sum)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Smith	29	4.60
6	Jack	34	3.78
7	Lee	40	2.98
8	David	30	4.80
9	Gasper	51	4.10
10	Betina	46	3.65
11	Andres		

```
<class 'pandas.core.series.Series'>
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
```

24



Đặc tả thống kê – hàm mean

```
import pandas as pd
import numpy as np

d = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin', 'Steve', 'Smith', 'Jack',
             'Lee', 'David', 'Gasper', 'Betina', 'Andres'],
    'Age': [25, 26, 25, 23, 30, 29, 23, 34, 40, 30, 51, 46],
    'Rating': [4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8, 3.78, 2.98, 4.80, 4.10, 3.65]
}

df = pd.DataFrame(d)
mean = df.mean(numeric_only=True)
print(type(mean))
print(mean)
```

<class 'pandas.core.series.Series'>
Age 31.833333
Rating 3.743333
dtype: float64

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Smith	29	4.60
6	Jack	34	3.78
7	Lee	40	2.98
8	David	30	4.80
9	Gasper	51	4.10
10	Betina	46	3.65

25



Đặc tả thống kê – hàm std

```
import pandas as pd
import numpy as np

d = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin', 'Steve', 'Smith', 'Jack',
             'Lee', 'David', 'Gasper', 'Betina', 'Andres'],
    'Age': [25, 26, 25, 23, 30, 29, 23, 34, 40, 30, 51, 46],
    'Rating': [4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8, 3.78, 2.98, 4.80, 4.10, 3.65]
}

df = pd.DataFrame(d)
std = df.std(numeric_only=True)
print(type(std))
print(std)
```

<class 'pandas.core.series.Series'>
Age 9.232682
Rating 0.661628
dtype: float64

26

26



Đặc tả thống kê

Sr.No.	Function	Description
1	count()	Number of non-null observations
2	sum()	Sum of values
3	mean()	Mean of Values
4	median()	Median of Values
5	mode()	Mode of values
6	std()	Standard Deviation of the Values
7	min()	Minimum Value
8	max()	Maximum Value
9	abs()	Absolute Value
10	prod()	Product of Values
11	cumsum()	Cumulative Sum
12	cumprod()	Cumulative Product

27

27



Đặc tả thống kê – phương thức describe

■ Phương thức đặc tả

```
df.describe(include)
```

■ include nhận 3 giá trị:

- 'object': Thống kê với cột dữ liệu khác số
- 'number': Thống kê với cột dữ liệu số
- 'all': Thống kê tất cả các cột dữ liệu

28

28



include = 'object'

```
import pandas as pd
import numpy as np

d = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin', 'Ricky', 'Smith', 'Jack',
            'Lee', 'David', 'Gasper', 'Betina', 'Andres'],
    'Age': [25, 26, 25, 23, 30, 29, 23, 34, 40, 30, 51, 46],
    'Rating': [4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8, 3.78, 2.98, 4.80, 4.10, 3.65]
}

df = pd.DataFrame(d)
desc = df.describe(include='object')
print(type(desc))
print(desc)
```

<class 'pandas.core.frame.DataFrame'>

	Name	Age	Rating
count	12		
unique	11		
top	Ricky		
freq	2		

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Smith	29	4.60

29



include = 'number'

```
import pandas as pd
import numpy as np

d = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin', 'Ricky', 'Smith', 'Jack',
            'Lee', 'David', 'Gasper', 'Betina', 'Andres'],
    'Age': [25, 26, 25, 23, 30, 29, 23, 34, 40, 30, 51, 46],
    'Rating': [4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8, 3.78, 2.98, 4.80, 4.10, 3.65]
}

df = pd.DataFrame(d)
desc = df.describe(include='number')
print(desc)
```

	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

30

30



include = 'all'

```
import pandas as pd
import numpy as np

d = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin', 'Ricky', 'Smith', 'Jack',
            'Lee', 'David', 'Gasper', 'Betina', 'Andres'],
    'Age': [25, 26, 25, 23, 30, 29, 23, 34, 40, 30, 51, 46],
    'Rating': [4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8, 3.78, 2.98, 4.80, 4.10, 3.65]
}

df = pd.DataFrame(d)
desc = df.describe(include='all')
print(desc)
```

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

31

31



Nội dung

1. Thao tác đọc/ghi dữ liệu với file
2. Các thao tác cơ bản với Series và DataFrame
3. Đặc tả thống kê (Descriptive Statistics)
- 4. Sắp xếp trên DataFrame (Sorting)**
5. Cắt nhỏ DataFrame (Slicing)
6. Lọc DataFrame (Filtering)
7. Duyệt trên Series và DataFrame (Iteration)
8. Đánh lại nhãn cho DataFrame (Reindexing)
9. Giống nhãn với reindex_like

32

32

Sắp xếp

- Có thể sắp xếp DataFrame theo nhãn các hàng/cột, hoặc theo giá trị
- Có thể sắp xếp DataFrame theo thứ tự tăng dần, giảm dần

33

33

Sắp xếp nhãn của hàng/cột

- **axis=1**: sắp xếp nhãn của cột. **axis=0**: sắp xếp nhãn của hàng. Mặc định **axis=0**
- Ví dụ sắp xếp nhãn của cột:

```
import pandas as pd

data = {
    'c': [5, 9, 1, 10],
    'd': [28, 34, 29, 42],
    'b': [15, 16, 11, 24],
    'a': [46, 27, 10, 30]
}
df = pd.DataFrame(data,
    index = [2, 0, 3, 1])
print(df)

df1 = df.sort_index(axis = 1)
print(df1)
```

	c	d	b	a
2	5	28	15	46
0	9	34	16	27
3	1	29	11	10
1	10	42	24	30

	a	b	c	d
2	46	15	5	28
0	27	16	9	34
3	10	11	1	29
1	30	24	10	42

34

34



Sắp xếp nhãn của hàng/cột

- **axis=1**: sắp xếp theo nhãn của cột. **axis=0**: sắp xếp theo nhãn của hàng. Mặc định **axis=0**
- Ví dụ sắp xếp nhãn của hàng

```
import pandas as pd

data = {
    'c': [5, 9, 1, 10],
    'd': [28, 34, 29, 42],
    'b': [15, 16, 11, 24],
    'a': [46, 27, 10, 30]
}
df = pd.DataFrame(data,
    index = [2, 0, 3, 1])
print(df)

df1 = df.sort_index()
print(df1)
```

	c	d	b	a
2	5	28	15	46
0	9	34	16	27
3	1	29	11	10
1	10	42	24	30

	c	d	b	a
0	9	34	16	27
1	10	42	24	30
2	5	28	15	46
3	1	29	11	10

35

35



Sắp xếp nhãn của hàng/cột

- Có thể sắp xếp nhãn theo thứ tự tăng dần (**ascending=True**) hoặc giảm dần (**ascending=False**). Mặc định là sắp xếp tăng dần

```
import pandas as pd

data = {
    'c': [5, 9, 1, 10],
    'd': [28, 34, 29, 42],
    'b': [15, 16, 11, 24],
    'a': [46, 27, 10, 30]
}
df = pd.DataFrame(data,
    index = [2, 0, 3, 1])
print(df)

df1 = df.sort_index(axis=1, ascending=False)
print(df1)
```

	c	d	b	a
2	5	28	15	46
0	9	34	16	27
3	1	29	11	10
1	10	42	24	30

	d	c	b	a
2	28	5	15	46
0	34	9	16	27
3	29	1	11	10
1	42	10	24	30

36

36



Sắp xếp theo giá trị

- Có thể sắp xếp theo giá trị. Tham số **axis** chỉ ra sắp xếp các hàng/cột. Tham số **by** chỉ ra nhãn của hàng/cột được lấy làm cơ sở sắp xếp. Có thể là giá trị đơn hoặc một list.
- **by** đi theo **axis**. VD: nếu **axis=1** là sắp xếp lại cột, **by** sẽ là nhãn của 1 hàng nào đó, và giá trị của hàng này được lấy làm cơ sở để sắp xếp các cột

```
import pandas as pd
data = {
    'a': [1,5,5,10],
    'b': [28,29,34,42],
    'c': [15,16,11,24],
    'd': [46,27,10,30]
}
df = pd.DataFrame(data)
print(df)

df1 = df.sort_values(
    axis=1, ascending=True,
    by=0)
print(df1)
```

	a	b	c	d
0	1	28	15	46
1	5	29	16	27
2	5	34	11	10
3	10	42	24	30

	a	c	b	d
0	1	15	28	46
1	5	16	29	27
2	5	11	34	10
3	10	24	42	30

Lưu ý: trong ví dụ này, hàng không có nhãn, nên hàng sẽ nhận nhãn mặc định theo STT

37

37



Sắp xếp theo giá trị

- **by** nhận giá trị List → có thể sắp xếp các giá trị trên **nhiều hàng, nhiều cột**
- VD: sắp xếp các hàng theo giá trị trên 2 cột: cột a trước, cột b sau

```
import pandas as pd
data = {
    'a': [1,5,5,10],
    'b': [28,29,34,42],
    'c': [15,16,11,24],
    'd': [46,27,10,30]
}
df = pd.DataFrame(data)
print(df)

df1 = df.sort_values(
    axis=0, ascending=False,
    by=['a', 'b'])
print(df1)
```

	a	b	c	d
0	1	28	15	46
1	5	29	16	27
2	5	34	11	10
3	10	42	24	30

	a	b	c	d
3	10	42	24	30
2	5	34	11	10
1	5	29	16	27
0	1	28	15	46

38

38

Nội dung

1. Thao tác đọc/ghi dữ liệu với file
2. Các thao tác cơ bản với Series và DataFrame
3. Đặc tả thống kê (Descriptive Statistics)
4. Sắp xếp trên DataFrame (Sorting)
5. **Cắt nhỏ DataFrame (Slicing)**
6. Lọc DataFrame (Filtering)
7. Duyệt trên Series và DataFrame (Iteration)
8. Đánh lại nhãn cho DataFrame (Reindexing)
9. Giống nhãn với `reindex_like`

39

39

Cắt nhỏ dữ liệu (slicing) trong DataFrame

- Lấy ra tất cả các hàng trong 1 cột (lấy ra 1 cột) với thuộc tính **loc**

```
import pandas as pd

d = {'one': [1, 2, 3], 'two': [5, 6, 7]}
df = pd.DataFrame(d, index=['a', 'b', 'c'])

print(df, end="\n\n")

# Lấy ra tất cả các hàng trong 1 cột
print(df.loc[:, 'two']) # Trả về 1 Series
```

	one	two
a	1	5
b	2	6
c	3	7

a	5
b	6
c	7

Name: two, dtype: int64

40

40



Cắt nhỏ dữ liệu (slicing) trong DataFrame

- Lấy ra tất cả các hàng trong nhiều cột với thuộc tính **loc**

```
import pandas as pd

d = {
    'one': [1, 2, 3],
    'two': [5, 6, 7],
    'three': [8, 9, 10]}
df = pd.DataFrame(d, index=['a', 'b', 'c'])

print(df, end="\n\n")

# Lấy ra tất cả các hàng trong 2 cột
print(df.loc[:, ['two', 'one']]) # Trả về 1 DataFrame
```

	one	two	three
a	1	5	8
b	2	6	9
c	3	7	10

	two	one
a	5	1
b	6	2
c	7	3

41

41



Cắt nhỏ dữ liệu (slicing) trong DataFrame

- Lấy ra một vài hàng trong nhiều cột với thuộc tính **loc**

```
import pandas as pd

d = {
    'one': [1, 2, 3],
    'two': [5, 6, 7],
    'three': [8, 9, 10]}
df = pd.DataFrame(d, index=['a', 'b', 'c'])

print(df, end="\n\n")

# Lấy ra tất cả dữ liệu trong 2 hàng, 2 cột
print(df.loc[['c', 'a'], ['two', 'one']]) # Trả về 1 DataFrame
```

	one	two	three
a	1	5	8
b	2	6	9
c	3	7	10

	two	one
c	7	3
a	5	1

42

42



Cắt nhỏ dữ liệu (slicing) trong DataFrame

- Lấy ra một dải các hàng trong tất cả các cột với thuộc tính **loc**

```
import pandas as pd

d = {
    'one': [1, 2, 3],
    'two': [5, 6, 7],
    'three': [8, 9, 10]}
df = pd.DataFrame(d, index=['a', 'b', 'c'])

print(df, end="\n\n")

# Lấy ra dải các hàng
print(df.loc['a':'b']) # Trả về 1 DataFrame
```

	one	two	three
a	1	5	8
b	2	6	9
c	3	7	10

	one	two	three
a	1	5	8
b	2	6	9

43

43



Cắt nhỏ dữ liệu (slicing) trong DataFrame

- Lấy ra một dải các hàng và cột với thuộc tính **loc**

```
import pandas as pd

d = {
    'one': [1, 2, 3],
    'two': [5, 6, 7],
    'three': [8, 9, 10]}
df = pd.DataFrame(d, index=['a', 'b', 'c'])

print(df, end="\n\n")

# Lấy ra dải các hàng và cột
print(df.loc['a':'b', 'two':'three']) # Trả về 1 DataFrame
```

	one	two	three
a	1	5	8
b	2	6	9
c	3	7	10

	two	three
a	5	8
b	6	9

44

44



Lấy ra DataFrame các giá trị Boolean

- Thuộc tính **loc** có thể dùng kết hợp với biểu thức

```
import pandas as pd
```

```
d = {
    'one': [1, 2, 3],
    'two': [5, 6, 7],
    'three': [8, 9, 10]}
df = pd.DataFrame(d, index=['a', 'b', 'c'])
```

```
print(df, end="\n\n")
```

```
# Lấy ra 1 dải các hàng và cột
```

```
print(df.loc['a':'b', 'two':'three']) # Trả về 1 DataFrame
```

```
# Lấy ra DataFrame các giá trị boolean theo điều kiện check
```

```
print(df.loc['a':'b', 'two':'three']>6) # Trả về 1 DataFrame
```

	one	two	three
a	1	5	8
b	2	6	9
c	3	7	10

	two	three
a	5	8
b	6	9

	two	three
a	False	True
b	False	True

45

45



Cắt nhỏ dữ liệu trong DataFrame với phương thức **iloc**

- iloc**: tương tự như **loc**, nhưng không dùng nhãn, mà dùng STT, đánh số từ 0

```
import pandas as pd
```

```
d = {'one': [1, 2, 3], 'two': [5, 6, 7], 'three': [8, 9, 10]}
df = pd.DataFrame(d, index=['a', 'b', 'c'])
```

```
print(df)
```

```
print(df.iloc[1:3, 0:1])
```

```
print(df.iloc[1:2, :])
```

```
print(df.iloc[:, 1])
```

```
print(df.iloc[1:])
```

	one	two	three
a	1	5	8
b	2	6	9
c	3	7	10

	one
b	2
c	3

	one	two	three
b	2	6	9

	one	two	three
a	1	5	8

	one	two	three
b	2	6	9
c	3	7	10

46

46

Nội dung

1. Thao tác đọc/ghi dữ liệu với file
2. Các thao tác cơ bản với Series và DataFrame
3. Đặc tả thống kê (Descriptive Statistics)
4. Sắp xếp trên DataFrame (Sorting)
5. Cắt nhỏ DataFrame (Slicing)
- 6. Lọc DataFrame (Filtering)**
7. Duyệt trên Series và DataFrame (Iteration)
8. Đánh lại nhãn cho DataFrame (Reindexing)
9. Giống nhãn với reindex_like

47

47

Lọc DataFrame

- Có thể dùng các toán tử logic trên các cột để lọc ra các hàng trong DataFrame

```
import pandas as pd
import numpy as np

data = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin'],
    'Age': [25, 26, 25, 23],
    'Rating': [4.23, 3.24, 3.98, 2.56]
}

df = pd.DataFrame(data)
print(df)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56

```
df1 = df[df.Age==25] # Hoặc dùng df["Age"]==25
print(df1)
```

	Name	Age	Rating
0	Tom	25	4.23
2	Ricky	25	3.98

48

48



Lọc DataFrame – kết hợp điều kiện

- Có thể kết hợp các điều kiện. Lưu ý: Mỗi điều kiện phải đặt trong cặp dấu ngoặc tròn (). Toán tử and là &. Toán tử or là |. Toán tử not là ~

```
import pandas as pd
import numpy as np

data = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin'],
    'Age': [25, 26, 25, 23],
    'Rating': [4.23, 3.24, 3.98, 2.56]
}

df = pd.DataFrame(data)
print(df)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56

```
df1 = df[(df.Name=="James") | (df["Age"]==25)]
print(df1)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98

49

49



Lọc DataFrame – phương thức isin

- Phương thức **isin**

```
import pandas as pd
import numpy as np

data = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin'],
    'Age': [25, 26, 25, 23],
    'Rating': [4.23, 3.24, 3.98, 2.56]
}

df = pd.DataFrame(data)
print(df)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56

```
names = ['Bob', 'David', 'Ricky', 'Vin']
df1 = df[df.Name.isin(names)]
print(df1)
```

	Name	Age	Rating
2	Ricky	25	3.98
3	Vin	23	2.56

50

50



Lọc DataFrame – Xử lý sâu

- Xử lý sâu

```
import pandas as pd
import numpy as np
data = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin'],
    'Age': [25, 26, 25, 23],
    'Rating': [4.23, 3.24, 3.98, 2.56]
}
df = pd.DataFrame(data)
print(df)

df1 = df[df.Name.str.startswith("J")]
print(df1)

df2 = df[df.Name.str.contains("i")]
print(df2)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56

	Name	Age	Rating
1	James	26	3.24

	Name	Age	Rating
2	Ricky	25	3.98
3	Vin	23	2.56

51

51



Lọc DataFrame – phương thức query

- Phương thức **query**: xử lý code ngắn và linh hoạt hơn. Nhưng dễ bị lỗi do câu truy vấn là dạng sâu

```
import pandas as pd
import numpy as np
data = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin'],
    'Age': [25, 26, 25, 23],
    'Rating': [4.23, 3.24, 3.98, 2.56]
}
df = pd.DataFrame(data)
print(df)


df1 = df.query('Name!="Tom" and Age>23')
print(df1)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56

	Name	Age	Rating
1	James	26	3.24
2	Ricky	25	3.98

52

52



Lọc DataFrame – phương thức nlargest và nsmallest

- Lấy ra **N** hàng có giá trị lớn nhất/nhỏ nhất. Chỉ áp dụng được với cột có kiểu Number. Sẽ sắp xếp lại các hàng luôn.

```
import pandas as pd
import numpy as np

data = {
    'Name': ['Tom', 'James', 'Ricky', 'Vin'],
    'Age': [25, 26, 25, 23],
    'Rating': [4.23, 3.24, 3.98, 2.56]
}

df = pd.DataFrame(data)
print(df)
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56

```
df1 = df.nlargest(3, 'Age')
print(df1)
```


	Name	Age	Rating
1	James	26	3.24
0	Tom	25	4.23
2	Ricky	25	3.98

```
df2 = df.nsmallest(2, 'Rating')
print(df2)
```

	Name	Age	Rating
3	Vin	23	2.56
1	James	26	3.24

53

53



Nội dung

1. Thao tác đọc/ghi dữ liệu với file
2. Các thao tác cơ bản với Series và DataFrame
3. Đặc tả thống kê (Descriptive Statistics)
4. Sắp xếp trên DataFrame (Sorting)
5. Cắt nhỏ DataFrame (Slicing)
6. Lọc DataFrame (Filtering)
- 7. Duyệt trên Series và DataFrame (Iteration)**
8. Đánh lại nhãn cho DataFrame (Reindexing)
9. Giống nhãn với reindex_like

54

54



Duyệt trên DataFrame và Series

- Vòng lặp **for i in Series**: i lần lượt là các giá trị trong Series
- Vòng lặp **for i in DataFrame**: i lần lượt là các nhãn cột

```
import pandas as pd
data = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'LA']
}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])
print(df)
```

	Name	Age	Place
rank1	Tom	28	Paris
rank2	Jack	34	London
rank3	Nam	29	Hanoi
rank4	Ricky	42	LA

```
for col in df:
    print(col)
```

```
for value in df['Name']:
    print(value, end=" ")
```

Lưu ý: df['Name'] là một Series

Tom Jack Nam Ricky

55

55



iteritems()

- **DataFrame.iteritems()**: trả về đối tượng dạng như dict, với key là nhãn của một cột và value là Series ứng với cột đó

```
import pandas as pd
data = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'LA']
}
df = pd.DataFrame(data,
    index=['rank1', 'rank2', 'rank3', 'rank4'])

for key, value in df.iteritems():
    print(key, type(value))
    print(value)
```

```
Name <class
'pandas.core.series.Series'>
rank1      Tom
rank2      Jack
rank3      Nam
rank4      Ricky
Name: Name, dtype: object
Age <class
'pandas.core.series.Series'>
rank1      28
rank2      34
rank3      29
rank4      42
Name: Age, dtype: int64
Place <class
'pandas.core.series.Series'>
rank1      Paris
rank2      London
rank3      Hanoi
rank4      LA
Name: Place, dtype: object
```

56

56

iterrows()

- **DataFrame.iterrows()**: trả về đối tượng dạng như dict, với key là nhãn của một hàng và value là Series ứng với hàng đó

```
import pandas as pd

data = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'LA']
}
df = pd.DataFrame(data,
    index=['rank1', 'rank2', 'rank3', 'rank4'])

for key, value in df.iterrows():
    print(key)
    print(value)
```

```
rank1
Name      Tom
Age       28
Place     Paris
Name: rank1, dtype: object
rank2
Name      Jack
Age       34
Place     London
Name: rank2, dtype: object
rank3
Name      Nam
Age       29
Place     Hanoi
Name: rank3, dtype: object
rank4
Name      Ricky
Age       42
Place     LA
Name: rank4, dtype: object
```

57

57

itertuples()

- **itertuples**: trả về các dòng, mỗi dòng như một namedtuples

```
import pandas as pd

data = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'LA']
}
df = pd.DataFrame(data,
    index=['rank1', 'rank2', 'rank3', 'rank4'])

print(df)

for row in df.itertuples():
    print(type(row), end="\n\n") <class 'pandas.core.frame.Pandas'>
    print(row, end="\n\n")
    print(list(row), end="\n\n") Pandas(Index='rank1', Name='Tom', Age=28, Place='Paris')
    print(row.Place)
    print(row[3]) ['rank1', 'Tom', 28, 'Paris']
    break;
    Paris
    Paris
```

58

58



Duyệt trên DataFrame

- **Lưu ý:** khi duyệt DataFrame, chỉ nên đọc dữ liệu. Không nên thay đổi DataFrame, vì đây là 1 việc vô ích.
- Khi duyệt, các đối tượng được duyệt là bản sao dữ liệu từ DataFrame. Thay đổi trên các bản sao này sẽ không cập nhật được DataFrame

59

59



Nội dung

1. Thao tác đọc/ghi dữ liệu với file
2. Các thao tác cơ bản với Series và DataFrame
3. Đặc tả thống kê (Descriptive Statistics)
4. Sắp xếp trên DataFrame (Sorting)
5. Cắt nhỏ DataFrame (Slicing)
6. Lọc DataFrame (Filtering)
7. Duyệt trên Series và DataFrame (Iteration)
8. **Đánh lại nhãn cho DataFrame (Reindexing)**
9. Giống nhãn với reindex_like

60

60

Đánh lại nhãn với **reindex**

■ Chọn và sắp xếp lại nhãn hàng, cột

```
import pandas as pd
data = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])
print(df)

df_reindexed = df.reindex(
    index=['rank1', 'rank4'],
    columns=['Age', 'Name', 'Language'])
print(df_reindexed)
```

	Name	Age	Place
rank1	Tom	28	Paris
rank2	Jack	34	London
rank3	Nam	29	Hanoi
rank4	Ricky	42	Newyork

Lưu ý: Có thể bỏ cột cũ, sắp lại thứ tự cột, hoặc thêm cột mới, như cột Language. Các ô trong cột mới sẽ có giá trị là NaN

	Age	Name	Language
rank1	28	Tom	NaN
rank4	42	Ricky	NaN

61

61

Đánh lại nhãn với **reindex**

■ Chọn và sắp xếp lại nhãn hàng, cột

```
import pandas as pd
data = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
df = pd.DataFrame(data)
print(df)

df_reindexed = df.reindex(
    index=[0, 2, 3, 4],
    columns=['Age', 'Name', 'Language'])
print(df_reindexed)
```

	Name	Age	Place
0	Tom	28	Paris
1	Jack	34	London
2	Nam	29	Hanoi
3	Ricky	42	Newyork

	Age	Name	Language
0	28.0	Tom	NaN
2	29.0	Nam	NaN
3	42.0	Ricky	NaN
4	NaN	NaN	NaN

62

62



Đổi tên nhãn cho các cột, hàng

```
import pandas as pd
```

```
data = {  
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],  
    'Age': [28, 34, 29, 42],  
    'Place': ['Paris', 'London', 'Hanoi', 'LA']  
}
```

```
df = pd.DataFrame(data,  
    index=['rank1', 'rank2', 'rank3', 'rank4'])
```

```
print(df)
```

```
df = df.rename(  
    columns={'Name': 'Tên', 'Age': 'Tuổi', 'New': 'Mới'},  
    index={'rank1': 'Hạng 1', 'rank3': 'Hạng 3'})  
)  
print("\nSau khi đổi nhãn: \n")  
print(df)
```

	Name	Age	Place
rank1	Tom	28	Paris
rank2	Jack	34	London
rank3	Nam	29	Hanoi
rank4	Ricky	42	LA

Sau khi đổi tên nhãn:

	Tên	Tuổi	Place
Hạng 1	Tom	28	Paris
rank2	Jack	34	London
Hạng 3	Nam	29	Hanoi
rank4	Ricky	42	LA

63

63



Nội dung

1. Thao tác đọc/ghi dữ liệu với file
2. Các thao tác cơ bản với Series và DataFrame
3. Đặc tả thống kê (Descriptive Statistics)
4. Sắp xếp trên DataFrame (Sorting)
5. Cắt nhỏ DataFrame (Slicing)
6. Lọc DataFrame (Filtering)
7. Duyệt trên Series và DataFrame (Iteration)
8. Đánh lại nhãn cho DataFrame (Reindexing)
9. **Giống nhãn với reindex_like**

64

64

Nâng cao: giống nhãn với `reindex_like`

65

65

Giống nhãn với `reindex_like`

Lưu ý: `df1.reindex_like(df2)` sẽ lấy tất cả các nhãn của `df2`, nhưng lấy dữ liệu tương ứng với nhãn đó trong `df1` để làm dữ liệu cho các cell

```
import pandas as pd
```

```
data1 = {  
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],  
    'Age': [28, 34, 29, 42],  
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']  
}
```

```
df1 = pd.DataFrame(data1,  
    index=['rank1', 'rank2', 'rank3', 'rank4'])
```

```
data2 = {  
    'Name': ['David', 'Alice', 'Bob'],  
    'Age': [29, 24, 20],  
    'Place': ['Rio', 'LA', 'Berlin']  
}
```

```
df2 = pd.DataFrame(data2,  
    index=['rank2', 'rank1', 'rank5'])
```

```
print(df1)  
print(df2)  
df1 = df1.reindex_like(df2)  
print(df1)
```

	Name	Age	Place
rank1	Tom	28	Paris
rank2	Jack	34	London
rank3	Nam	29	Hanoi
rank4	Ricky	42	Newyork
	Name	Age	Place
rank2	David	29	Rio
rank1	Alice	24	LA
rank5	Bob	20	Berlin
	Name	Age	Place
rank2	Jack	34.0	London
rank1	Tom	28.0	Paris
rank5	NaN	NaN	NaN

66

66



Điền bổ sung dữ liệu khi đánh lại nhãn

- Các phương thức **reindex**, **reindex_like** có thể nhận tham số **method** để bổ sung dữ liệu ở các trường **NaN**
 - **pad/ffill** – Điền giá trị tiến lên (giá trị với index phía trước dùng để điền cho giá trị NaN của index phía sau)
 - **bfill/backfill** – Điền giá trị lùi lại (giá trị sau dùng để điền cho giá trị NaN phía trước)
 - **nearest** – Tìm giá trị với index gần nhất để điền
 - Chỉ dùng được khi **nhãn của hàng** có kiểu Number
- Lưu ý: **df1.reindex_like(df2)** sẽ lấy tất cả các nhãn của **df2**, nhưng lấy giá trị của **df1** để điền bổ sung dữ liệu

67

67



Ví dụ: nhãn của hàng là kiểu Number

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
df1 = pd.DataFrame(data1, index=[1, 2, 3, 4])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
df2 = pd.DataFrame(data2, index=[4, 3, 10, -1])
```

```
print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill"))
print(df1.reindex_like(df2, method="bfill"))
print(df1.reindex_like(df2, method="nearest"))
```

	Name	Age	Place
1	Tom	28	Paris
2	Jack	34	London
3	Nam	29	Hanoi
4	Ricky	42	Newyork

	Name	Age	Place
4	David	29	Rio
3	Alice	24	LA
10	Bob	20	Berlin
-1	Kelvin	22	Rome

68

68



Ví dụ: nhãn của hàng là kiểu Number

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
```

```
df1 = pd.DataFrame(data1, index=[1, 2, 3, 4])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
```

```
df2 = pd.DataFrame(data2, index=[4, 3, 10, -1])
```

```
print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill"))
print(df1.reindex_like(df2, method="bfill"))
print(df1.reindex_like(df2, method="nearest"))
```

	Name	Age	Place
4	Ricky	42.0	Newyork
3	Nam	29.0	Hanoi
10	NaN	NaN	NaN
-1	NaN	NaN	NaN
	Name	Age	Place
4	Ricky	42.0	Newyork
3	Nam	29.0	Hanoi
10	Ricky	42.0	Newyork
-1	NaN	NaN	NaN
	Name	Age	Place
4	Ricky	42.0	Newyork
3	Nam	29.0	Hanoi
10	NaN	NaN	NaN
-1	Tom	28.0	Paris
	Name	Age	Place
4	Ricky	42	Newyork
3	Nam	29	Hanoi
10	Ricky	42	Newyork
-1	Tom	28	Paris

69

69



Lưu ý: dữ liệu được lấy từ df1 để điền

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky', "Alice"],
    'Age': [28, 34, 29, 42, 30],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork', "LA"]
}
```

```
df1 = pd.DataFrame(data1, index=[1, 2, 3, 4, 5])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
```

```
df2 = pd.DataFrame(data2, index=[4, 3, 10, -1])
```

```
print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill"))
print(df1.reindex_like(df2, method="bfill"))
print(df1.reindex_like(df2, method="nearest"))
```

	Name	Age	Place
1	Tom	28	Paris
2	Jack	34	London
3	Nam	29	Hanoi
4	Ricky	42	Newyork
5	Alice	30	LA

	Name	Age	Place
4	David	29	Rio
3	Alice	24	LA
10	Bob	20	Berlin
-1	Kelvin	22	Rome

70

70



Lưu ý: dữ liệu được lấy từ df1 để điền

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky', 'Alice'],
    'Age': [28, 34, 29, 42, 30],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork', 'LA']
}
df1 = pd.DataFrame(data1, index=[1, 2, 3, 4, 5])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', 'Kelvin'],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', 'Rome']
}
df2 = pd.DataFrame(data2, index=[4, 3, 10, -1])
```

```
print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill"))
print(df1.reindex_like(df2, method="bfill"))
print(df1.reindex_like(df2, method="nearest"))
```

	Name	Age	Place
4	Ricky	42.0	Newyork
3	Nam	29.0	Hanoi
10	NaN	NaN	NaN
-1	NaN	NaN	NaN
	Name	Age	Place
4	Ricky	42.0	Newyork
3	Nam	29.0	Hanoi
10	Alice	30.0	LA
-1	NaN	NaN	NaN
	Name	Age	Place
4	Ricky	42.0	Newyork
3	Nam	29.0	Hanoi
10	NaN	NaN	NaN
-1	Tom	28.0	Paris
	Name	Age	Place
4	Ricky	42	Newyork
3	Nam	29	Hanoi
10	Alice	30	LA
-1	Tom	28	Paris

71

71



Ví dụ: nhãn của hàng là kiểu String

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
df1 = pd.DataFrame(data1, index=['b', 'c', 'd', 'e'])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', 'Kelvin'],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', 'Rome']
}
df2 = pd.DataFrame(data2, index=['e', 'c', 'f', 'a'])
```

```
print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill"))
print(df1.reindex_like(df2, method="bfill"))
```

```
# print(df1.reindex_like(df2, method="nearest")) # Bị lỗi nếu dùng
```

	Name	Age	Place
b	Tom	28	Paris
c	Jack	34	London
d	Nam	29	Hanoi
e	Ricky	42	Newyork

	Name	Age	Place
e	David	29	Rio
c	Alice	24	LA
f	Bob	20	Berlin
a	Kelvin	22	Rome

72

72



Ví dụ: nhãn của hàng là kiểu String

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
df1 = pd.DataFrame(data1, index=['b', 'c', 'd', 'e'])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
df2 = pd.DataFrame(data2, index=['e', 'c', 'f', 'a'])
```

```
print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill"))
print(df1.reindex_like(df2, method="bfill"))

# print(df1.reindex_like(df2, method="nearest")) # Bị lỗi nếu dùng
```

	Name	Age	Place
e	Ricky	42.0	Newyork
c	Jack	34.0	London
f	NaN	NaN	NaN
a	NaN	NaN	NaN
	Name	Age	Place
e	Ricky	42.0	Newyork
c	Jack	34.0	London
f	Ricky	42.0	Newyork
a	NaN	NaN	NaN
	Name	Age	Place
e	Ricky	42.0	Newyork
c	Jack	34.0	London
f	NaN	NaN	NaN
a	Tom	28.0	Paris

73

73



Thiết lập giới hạn khi điền dữ liệu bổ sung

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
df1 = pd.DataFrame(data1, index=[1, 2, 3, 4])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
df2 = pd.DataFrame(data2, index=[-1, 4, 6, 7])
```

```
print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill", limit=1))
print(df1.reindex_like(df2, method="bfill", limit=1))
print(df1.reindex_like(df2, method="nearest", limit=1))
```

	Name	Age	Place
1	Tom	28	Paris
2	Jack	34	London
3	Nam	29	Hanoi
4	Ricky	42	Newyork
	Name	Age	Place
-1	David	29	Rio
4	Alice	24	LA
6	Bob	20	Berlin
7	Kelvin	22	Rome

Lưu ý: Để dùng tham số limit, index của df2 phải **đánh tăng dần**.
Nếu index của df2 là [4, -1, 6, 7] sẽ bị lỗi

74

74



Thiết lập giới hạn khi điền dữ liệu bổ sung

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
```

```
df1 = pd.DataFrame(data1, index=[1, 2, 3, 4])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
```

```
df2 = pd.DataFrame(data2, index=[-1, 4, 6, 7])
```

```
print(df1.reindex_like(df2))
```

```
print(df1.reindex_like(df2, method="ffill", limit=1))
```

```
print(df1.reindex_like(df2, method="bfill", limit=1))
```

```
print(df1.reindex_like(df2, method="nearest", limit=1))
```

	Name	Age	Place
-1	NaN	NaN	NaN
4	Ricky	42.0	Newyork
6	NaN	NaN	NaN
7	NaN	NaN	NaN
	Name	Age	Place
-1	NaN	NaN	NaN
4	Ricky	42.0	Newyork
6	Ricky	42.0	Newyork
7	NaN	NaN	NaN
	Name	Age	Place
-1	Tom	28.0	Paris
4	Ricky	42.0	Newyork
6	NaN	NaN	NaN
7	NaN	NaN	NaN
	Name	Age	Place
-1	Tom	28.0	Paris
4	Ricky	42.0	Newyork
6	Ricky	42.0	Newyork
7	NaN	NaN	NaN

Lưu ý: Để dùng tham số limit, index của df2 phải **đánh tăng dần**.

Nếu index của df2 là [4, -1, 6, 7] sẽ bị lỗi

75

75



Thiết lập giới hạn khi điền dữ liệu bổ sung

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
```

```
df1 = pd.DataFrame(data1, index=[1, 2, 3, 4])
```

```
data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
```

```
df2 = pd.DataFrame(data2, index=[-1, 4, 6, 7])
```

```
print(df1.reindex_like(df2))
```

```
print(df1.reindex_like(df2, method="ffill", limit=2))
```

```
print(df1.reindex_like(df2, method="bfill", limit=2))
```

```
print(df1.reindex_like(df2, method="nearest", limit=2))
```

	Name	Age	Place
-1	NaN	NaN	NaN
4	Ricky	42.0	Newyork
6	NaN	NaN	NaN
7	NaN	NaN	NaN
	Name	Age	Place
-1	NaN	NaN	NaN
4	Ricky	42.0	Newyork
6	Ricky	42.0	Newyork
7	Ricky	42.0	Newyork
	Name	Age	Place
-1	Tom	28.0	Paris
4	Ricky	42.0	Newyork
6	NaN	NaN	NaN
7	NaN	NaN	NaN
	Name	Age	Place
-1	Tom	28	Paris
4	Ricky	42	Newyork
6	Ricky	42	Newyork
7	Ricky	42	Newyork

76

76



Thiết lập giới hạn khi điền dữ liệu bổ sung

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
df1 = pd.DataFrame(data1, index=['b', 'c', 'd', 'e'])

data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
df2 = pd.DataFrame(data2, index=['a', 'c', 'y', 'z'])

print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill", limit=1))
print(df1.reindex_like(df2, method="bfill", limit=1))
```

	Name	Age	Place
b	Tom	28	Paris
c	Jack	34	London
d	Nam	29	Hanoi
e	Ricky	42	Newyork

	Name	Age	Place
a	David	29	Rio
c	Alice	24	LA
y	Bob	20	Berlin
z	Kelvin	22	Rome

77

77



Thiết lập giới hạn khi điền dữ liệu bổ sung

```
import pandas as pd
```

```
data1 = {
    'Name': ['Tom', 'Jack', 'Nam', 'Ricky'],
    'Age': [28, 34, 29, 42],
    'Place': ['Paris', 'London', 'Hanoi', 'Newyork']
}
df1 = pd.DataFrame(data1, index=['b', 'c', 'd', 'e'])

data2 = {
    'Name': ['David', 'Alice', 'Bob', "Kelvin"],
    'Age': [29, 24, 20, 22],
    'Place': ['Rio', 'LA', 'Berlin', "Rome"]
}
df2 = pd.DataFrame(data2, index=['a', 'c', 'y', 'z'])

print(df1.reindex_like(df2))
print(df1.reindex_like(df2, method="ffill", limit=1))
print(df1.reindex_like(df2, method="bfill", limit=1))
```

	Name	Age	Place
a	NaN	NaN	NaN
c	Jack	34.0	London
y	NaN	NaN	NaN
z	NaN	NaN	NaN

	Name	Age	Place
a	NaN	NaN	NaN
c	Jack	34.0	London
y	Ricky	42.0	Newyork
z	NaN	NaN	NaN

	Name	Age	Place
a	Tom	28.0	Paris
c	Jack	34.0	London
y	NaN	NaN	NaN
z	NaN	NaN	NaN

78

78