

Pandas Dataframe 的新增與刪除



簡報閱讀



範例與作業



問題討論



學習心得(完成)

重要知識點

從 DataFrame 中插入或
刪除資料

= 可以用來增加行 (欄)

append() 可以用來新增列
(資料)



重要知識點



- 正確的從 DataFrame 中插入或刪除資料
- 正確的對 DataFrame 進行合併與重組
- 了解 DataFrame 中合併的方法差異

對於一個 DataFrame 可以進行新增或刪除的操作，又可以分為行或是列的方向：

- = 可以用來增加行（欄）
- append() 可以用來新增列（資料）
- del 或 pop() 可以用來刪除行（欄）
- drop() 可以用來刪除列（資料）

= 可以用來增加行（欄）

可以利用指派運算（=）值些產生新的欄位：

```
1 import pandas as pd
2
3 df = pd.DataFrame([[1], [2]], columns = ['a'])
4 print(df)
```

| | a |
|---|---|
| 0 | 1 |
| 1 | 2 |

```
1 df['b'] = pd.Series([3, 4])
2 print(df)
```

| | a | b |
|---|---|---|
| 0 | 1 | 3 |
| 1 | 2 | 4 |

append() 可以用來新增列 (資料)

利用 append(...) 增加新的資料：

```
1 import pandas as pd
2
3 df = pd.DataFrame([[1, 2]], columns = ['a', 'b'])
4 print(df)
```

| | a | b |
|---|---|---|
| 0 | 1 | 2 |

```
1 df = df.append(pd.DataFrame([[3, 4]], columns = ['a', 'b']))
2 print(df)
```

| | a | b |
|---|---|---|
| 0 | 1 | 2 |
| 0 | 3 | 4 |

但仔細看一下，會發現索引重複了，這邊利用 reset_index() 修正：

```
2 df = df.reset_index(drop=True)
3 print(df)
```

| | a | b |
|---|---|---|
| 0 | 1 | 2 |
| 1 | 3 | 4 |

del 或 pop() 可以用來刪除行 (欄)

利用 del 或是 pop() 的方法增加新的欄位：

```
1 import pandas as pd
2
3 df = pd.DataFrame([[1, 2, 3]], columns = ['a',
4 print(df)
```

| | a | b | c |
|---|---|---|---|
| 0 | 1 | 2 | 3 |



```
1 del df['a']
2 df.pop('c')
```

| | |
|---|---|
| | b |
| 0 | 2 |

drop() 可以用來刪除列 (資料)

可以利用 drop(...) 刪除列的資料：

```
1 import pandas as pd
2
3 df = pd.DataFrame([[1], [2]], columns = ['a'])
4 print(df)
```

| | |
|---|---|
| | a |
| 0 | 1 |
| 1 | 2 |

```
1 df = df.drop(1)
2 print(df)
```

| | |
|---|---|
| | a |
| 0 | 1 |

DataFrame 的合併與重組

見可以分為以下幾種：

- 連集 (Concat)
- 合併 (merge)
- 連接 (Join)
- 分組 (Group)

連集 (Concat)

利用連集 (Concat) 上下相拼：

```
1 one = pd.DataFrame({
2     'id':[1, 2],
3     'Name': ['Alex', 'Amy'],
4 })
5 two = pd.DataFrame({
6     'id':[1, 2],
7     'Name': ['Bob', 'Tom']
8 })
9
10 pd.concat([one, two])
```

| | id | name |
|---|----|------|
| 0 | 1 | Alex |
| 1 | 2 | Amy |

| | id | name |
|---|----|------|
| 0 | 1 | Bob |
| 1 | 2 | Tom |

| | id | Name |
|---|----|------|
| 0 | 1 | Alex |
| 1 | 2 | Amy |
| 0 | 1 | Bob |
| 1 | 2 | Tom |

有時候會有索引重複的現象，請務必要修正：

```

1 one = pd.DataFrame({
2     'id':[1, 2],
3     'Name': ['Alex', 'Amy'],
4 })
5 two = pd.DataFrame({
6     'id':[1, 2],
7     'Name': ['Bob', 'Tom']
8 })
9
10 pd.concat([one, two]).reset_index(drop=True)
```

| | id | Name |
|---|----|------|
| 0 | 1 | Alex |
| 1 | 2 | Amy |

| | id | Name |
|---|----|------|
| 0 | 1 | Bob |
| 1 | 2 | Tom |

| | id | Name |
|---|----|------|
| 0 | 1 | Alex |
| 1 | 2 | Amy |
| 0 | 1 | Bob |
| 1 | 2 | Tom |

合併 (merge)

利用合併 (merge) 實現欄位左右相拼：

```

1  one = pd.DataFrame({
2      'id':[1, 2],
3      'Name': ['Alex', 'Amy'],
4  })
5  two = pd.DataFrame({
6      'id':[1, 2],
7      'Score': [98, 60]
8  })
9
10 pd.merge(one, two, on='id')
```


| | id | name |
|---|----|------|
| 0 | 1 | Alex |
| 1 | 2 | Amy |

| | id | Score |
|---|----|-------|
| 0 | 1 | 98 |
| 1 | 2 | 60 |

| | id | Name | Score |
|---|----|------|-------|
| 0 | 1 | Alex | 98 |
| 1 | 2 | Amy | 60 |



不同的 merge 規則

合併除了可以指定欄位之外，也可以設定「拼」的方法：

`pandas.merge(left, right, how='inner', on=None ...)`

- left、right：必填，任何 *dataFrame* 物件
- how：提供四種不同的合併方法（參考下圖）
- on：用來合併的相依欄位

連接 (Join)

利用連接 (Join) 實現索引左右相拼：

```
1 one = pd.DataFrame({
2     'Name': ['Alex', 'Amy'],
3 })
4 two = pd.DataFrame({
5     'Score': [98, 60]
6 })
7
8 one.join(two)
```

| | Name |
|---|------|
| 0 | Alex |
| 1 | Amy |

| | Score |
|---|-------|
| 0 | 98 |
| 1 | 60 |

| | Name | Score |
|---|------|-------|
| 0 | Alex | 98 |
| 1 | Amy | 60 |

另外有一種比較特別的用法稱為是分組 (Group) ，會依照資料內容重新組裝：

```
1 df = pd.DataFrame({
2     'A' : ['foo', 'bar', 'foo', 'bar'],
3     'B' : ['one', 'one', 'two', 'three'],
4     'C' : [1,2,3,4],
5     'D' : [10, 20, 30, 40]
6 })
```

| | A | B | C |
|---|-----|-------|---|
| 0 | foo | one | 1 |
| 1 | bar | one | 2 |
| 2 | foo | two | 3 |
| 3 | bar | three | 4 |



我們直接來看這個例子，可以對分組後的結果進行運算：

```
1 df.groupby('A').sum()
2 df.groupby('A').agg(sum)
3 df.groupby(['A', 'B']).sum()
```

| A | C | D |
|-----|---|----|
| bar | 6 | 60 |
| foo | 4 | 40 |

| A | C | D |
|-----|---|----|
| bar | 6 | 60 |
| foo | 4 | 40 |

| A | B | C | D |
|-----|-------|---|----|
| bar | one | 2 | 20 |
| | three | 4 | 40 |
| foo | one | 1 | 10 |
| | two | 3 | 30 |



知識點回顧

- 正確的從 DataFrame 中插入或刪除資料
- 正確的對 DataFrame 進行合併與重組
- 了解 DataFrame 中合併的方法差異

參考資料

Merge, join, concatenate and compare

網站：[pandas](#)



pandas

Getting started **User Guide** API reference Development Release notes

🔍 Search the docs ...

- 10 minutes to pandas
- Intro to data structures
- Essential basic functionality
- IO tools (text, CSV, HDF5, ...)
- Indexing and selecting data
- Multindex / advanced indexing
- Merge, join, concatenate and compare**
- Reshaping and pivot tables
- Working with text data
- Working with missing data
- Duplicate Labels
- Categorical data
- Nullable integer data type
- Nullable Boolean data type
- Visualization
- Computational tools
- Group by: split-apply-combine
- Windowing Operations
- Time series / date functionality
- Time deltas

Merge, join, concatenate and compare

pandas provides various facilities for easily combining together Series or DataFrame with various kinds of set logic for the indexes and relational algebra functionality in the case of join / merge-type operations.

In addition, pandas also provides utilities to compare two Series or DataFrame and summarize their differences.

Concatenating objects

The `concat()` function (in the main pandas namespace) does all of the heavy lifting of performing concatenation operations along an axis while performing optional set logic (union or intersection) of the indexes (if any) on the other axes. Note that I say "if any" because there is only a single possible axis of concatenation for Series.

Before diving into all of the details of `concat` and what it can do, here is a simple example:

```

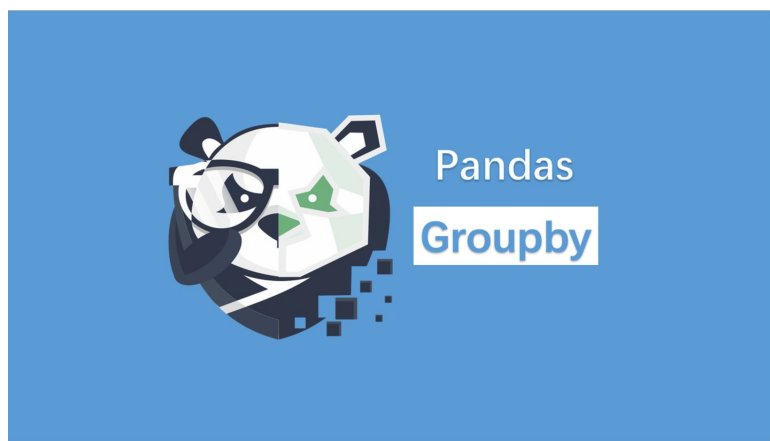
In [1]: df1 = pd.DataFrame(
...:     {
...:         "A": ["A0", "A1", "A2", "A3"],
...:         "B": ["B0", "B1", "B2", "B3"],
...:         "C": ["C0", "C1", "C2", "C3"],
...:         "D": ["D0", "D1", "D2", "D3"],
...:     },
...:     index=[0, 1, 2, 3],
...: )
...:
In [2]: df2 = pd.DataFrame(
...:     {
...:         "A": ["A4", "A5", "A6", "A7"],
...:         "B": ["B4", "B5", "B6", "B7"],
...:         "C": ["C4", "C5", "C6", "C7"],
...:     },
...: )

```

Pandas 之超好用的 Groupby 用法詳解

網站：zhuanlan

本文提供完整的 Groupby 用法，適合對於分組有興趣的同學深入理解。



[下一步：閱讀範例與完成作業](#)

