台塑企業
FORMOSA PLASTICS GROUP

台 塑 企 業

(AI基礎訓練中級班)

資 料 前 置 處 理

(工程類)

制定部門：總管理處總經理室技訓中心
編定日期：2020年07月21日編印
版次: R1版

目 錄

目　錄

目 錄

台塑企業
FORMOSA PLASTICS GROUP

課 程 目 的

在本課程中，將簡介常見的資料前置處理方法與其相對應的Python指令。

台塑企業
FORMOSA PLASTICS GROUP

課 程 大 綱

壹、資料前置處理的動機

貳、資料前置處理的主要項目

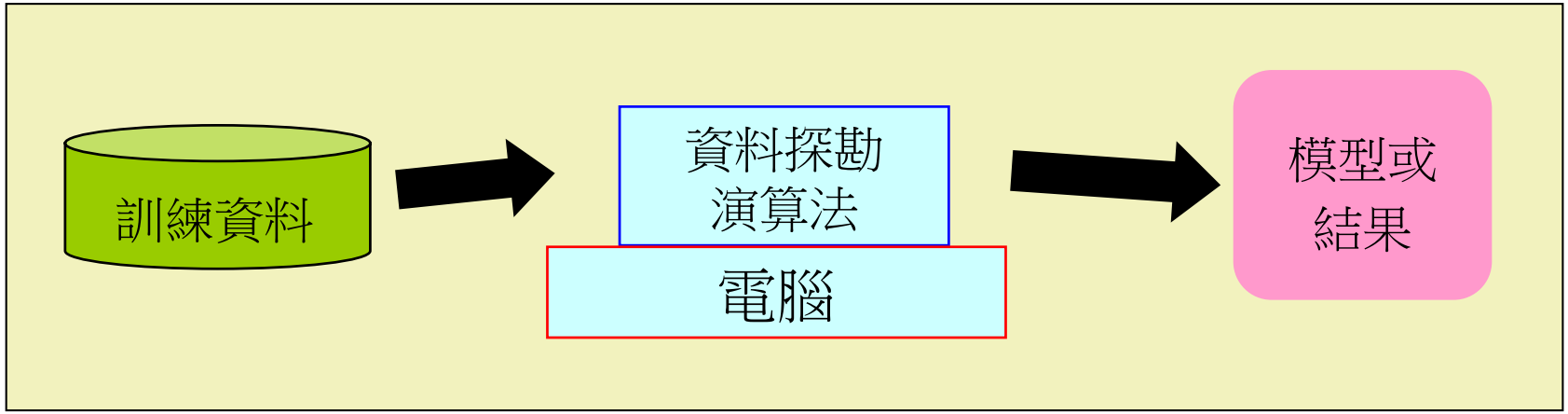
- 資料檢查
- 資料清理
- 資料合併

台塑企業
FORMOSA PLASTICS GROUP

工 作 目 的

1. 初步瞭解資料前置處理的主要項目與其相對應的Python指令。

資料前置處理的動機



垃圾進，垃圾出

真實的資料可能雜亂，需要清理與彙整

- 資料有遺漏值
- 資料有不一致值(度量衡值、命名不一致)
- 資料不見得全可用

df=pd.read_csv('centertraining.csv')

資料檢查

1. 以csv格式載入資料檔案，以 Dataframe 來作處理

```
import pandas as pd                          → 將pandas 匯入，並簡稱pd
                                             # 需先upload資料案centertraining.csv
df=pd.read_csv('centertraining.csv')         →匯入資料檔案，並簡稱df
print(df.head( ))                            →顯示df的前5筆資料
df2 = df                                     → 將df的資料複製給 df2
```

2. 如有 空欄位(Unnamed)可用以下指令刪除

```
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
```

2. 資料檢查的常用方法

1) 選取特定範圍的數據檢查 (若知道有異常事件發生時)
2) 資料隨機排列及抽樣後檢查 (若數據量太多時)
3) 數據分群後檢查 (須視數據群體特性來判斷時)
4) 數據視覺化後檢查(於資料視覺化部分作講解)

選取特定範圍的數據做檢查

1) 選取特定的人、事、時、地、物數據做觀察

例如選取特定的時間範圍並觀察數據

df['Time']　　→ 將df的Time的欄位資料顯示出來

```
df[df['Time'].between('1/10/2020','1/13/2020')]
```

→ 此處篩選 Time 欄位中 1/10/2020 至1/13/2020 之數據

→ 注意Time 欄位的格式為 西元 月/日/年

情境及使用目的：

操作員回報說 2020/1/10 至 2020/1/13 時段廠內有故障，因此想知道這時段各參數數值的狀況。

資料隨機排列及抽樣後檢查

(1) 進行隨機排列

```
import numpy as np
sampling = np.random.permutation(31)
```

np.random.permutation(隨機排列之數據範圍)

```
df.take(sampling)
```

(2) 進行隨機抽樣

```
df.sample(n=5)
```

此處欲隨機抽取5個數據

情境及使用目的：

在對數據了解程度不足或數據太多的情況下，想隨機找出數值來「快速了解」大概數值範圍或了解數據狀況。

數據分群檢查– 手動分群

(1) 定義分群標準

把數據分成 0-80 , 80-150 , 150-200 三群

```
bins = [0, 80, 150 ,200]
```

(2) 選擇需分群之欄位並進行分群

根據欲分群之欄位，此處是名稱為Value的欄位

```
data_inbins = pd.cut(df['Value'],bins)
```

(3) 評估各群之數據數量

```
pd.value_counts(data_inbins)
```

情境及使用目的：

在明確知道數據數值意義或想知道數值分佈的狀況下可用。

如操作員知道 0-80為原料不足所造成之數值，80-150為正常數值，150-200為突發狀況所造成之數值，則可用此方法去觀察狀況及次數

數據分群檢查– 電腦自動分群

(1) 定義(選擇)分群數量

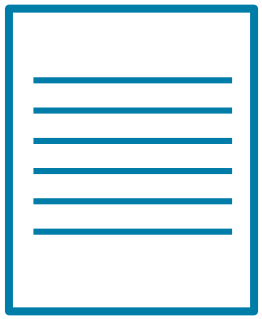
(2) 選擇需分群之欄位並進行分群

此處欲分群數量為 3，分群之欄位是名稱為Value的欄位

```
data_autobins = pd.qcut(df['Value'], 3)
pd.value_counts(data_autobins)
```

情境及使用目的：
在不清楚數據數值意義並想知道數值分佈的狀況下可用。

資料清理 - 觀察與分析資料缺陷的種類



數據

- 數據格式統一嗎?
- 數據值缺空?
- 數據是否重覆了?
- 數據那些是不合理的?

- 格式問題
- 缺失數據問題
- 數據重覆問題
- 離群值問題

資料清理 -資料缺陷處理的種類

常見的資料缺陷處理種類

1.異常數據處理

2.格式編輯處理

3.缺失資料處理

4.重覆數據處理

5.離群值處理

異常數值編輯替換

改寫特定異常數值

df.replace(需修改之數值, 目標數值)

```
df.replace(147.66,1111)
```

情境及使用目的：

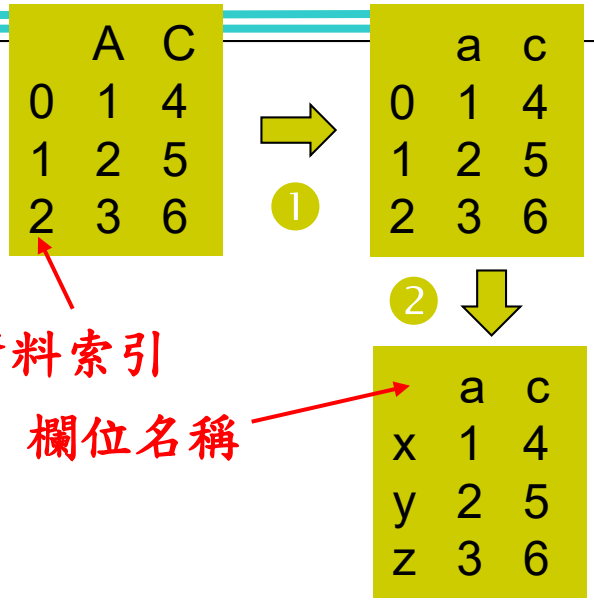
在了解數據異常數值意義之後，統一更改。

例如：在上述例子中，操作員知道147.66是一過低之異常值，需統一修正替換為1111。

格式編輯

更改欄位名稱或索引格式

- upper 大寫的更動
- lower 小寫的更動
- title 第一字母大寫



df.rename(index={'index name': 'new name'}, columns={'column name ': 'new name'})

```
df1=pd.read_csv('df1.csv')
print(df1)
1 df1=df1.rename(columns={'A': 'a', 'C': 'c'})
2 df1=df1.rename(index={0: 'x', 1: 'y', 2: 'z'})
print(df1)
```

情境及使用目的：

數據的欄位名稱或索引需更動下可使用。

缺失資料處理- 搜尋NaNs

(1) 找出 NaN：`df.isnull()`

NaN : Not a Number

(2) 找出有NaN 的 row：`df[df.isnull().any(axis=1)]`

(3) 檢查某一column是否含有NaN：

df [df [‘column name’] .isnull ()]

df [df [‘Length’] .isnull ()]

```
df[df['Flow Rate'].isnull()]
```

情境及使用目的：
在處理數據時偶而會遇到遺失的資料並造成空缺值，故需找出、刪除或用其他方式處理。

缺失資料處理 - 刪除及替換NaNs

(1) 刪除含NaN之數據：`df.dropna()`

(2) 以其他數值取代NaN

- 數字 0： `df.fillna(0)`
- 特定數字： `df.fillna({'Value': 1000, 'Length': 2000})`

  df.fillna({' column name ': number, ' column name ': number})
- 出現在NaN之前的數值：`df.fillna(method='ffill')`
- 平均值： `df.fillna(df.mean())`

處理重覆的數據

(1) 搜尋重覆數據 `df.duplicated()`

(2) 刪除重覆數據 `df.drop_duplicates()`

情境及使用目的：
因人為的重覆貼錯或數據本身的因素而造成重覆的資料，需進行刪除以免造成計算錯誤。

離群值的處理 - 1

- 顯示數據的統計資訊

```
df.describe()
```

會顯示以下訊息

情境及使用目的：
離群值常為系統異常值，故找出離群值以便對異常值進行處理。

		Value	Length	Width	Flow Rate	Weight	Height
數據數量	count	30.000000	30.000000	31.000000	31.000000	30.000000	31.000000
平均值	mean	86.799667	2.827333	2.841613	29.880645	0.832667	0.818065
標準差	std	40.026123	1.216306	2.793994	24.546239	0.379064	0.814708
最小值	min	7.510000	0.810000	0.080000	2.140000	0.240000	0.020000
25百分位數	25%	65.140000	1.962500	0.930000	10.685000	0.542500	0.210000
50百分位數 (即中位數)	50%	75.920000	2.620000	1.570000	22.380000	0.790000	0.470000
75百分位數	75%	104.015000	3.787500	4.475000	46.315000	1.132500	1.345000
最大值	max	196.430000	5.410000	10.310000	95.740000	1.620000	3.090000

離群值處理 - 2

- 檢視篩選的數據

  例如：下列為篩選 **Width** 欄位大於 **2.841613** 之數據

  ```
  df [ df ['Width'] > 2.841613 ]
  ```

- 檢視篩選離群值的數據

  建議可用平均值與標準差作離群值標準來篩選

  ```
  df[df['Width'] > 2.841613 + 2.793994 * 2 ]
  ```

  平均值　　標準差2倍

- 計算離群值個數

  ```
  len( df[df['Width'] > 2.841613 + 2.793994 * 2 ] )
  ```

離群值處理 - 3

	Time	Value	Length	Width	Flow Rate	Weight	Height
0	2020/1/1	147.66	3.91	1.05	50.00	1.17	0.09
1	2020/1/2	83.03	2.76	0.96	24.91	0.83	0.29
2	2020/1/3	69.80	1.35	3.07	5.94	0.40	0.92
3	2020/1/4	56.52	1.83	1.23	10.96	0.55	0.37
4	2020/1/5	76.90	NaN	0.49	2.37	0.26	0.15
5	2020/1/6	78.53	2.69	1.57	23.56	0.81	0.47
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
29	2020/1/30	105.27	4.03	4.69	53.18	1.21	1.41
30	2020/1/30	105.27	4.03	4.69	53.18	1.21	1.41
31	2020/1/31	NaN	NaN	NaN	NaN	NaN	NaN

df [df ['Width'] > 2.841613]

Width值大於某標的值的數據

	Time	Value	Length	Width	Flow Rate	Weight	Height
2	2020/1/3	69.80	1.35	3.07	5.94	0.40	0.92
6	2020/1/7	67.15	2.48	5.03	20.15	0.74	1.51
8	2020/1/9	64.47	2.43	3.22	19.34	0.73	0.97
9	2020/1/10	99.12	5.41	10.31	95.74	1.62	3.09
10	2020/1/11	100.25	3.70	5.15	44.78	1.11	1.55
13	2020/1/14	68.81	1.31	3.59	5.64	0.39	1.08
15	2020/1/16	7.51	0.83	2.87	2.25	0.25	0.47
16	2020/1/17	74.61	2.62	9.18	22.38	0.79	2.75
17	2020/1/18	63.80	2.42	9.18	19.14	0.73	2.29
19	2020/1/20	61.25	2.37	4.90	18.37	NaN	1.47
27	2020/1/28	70.88	2.55	3.28	21.26	0.77	0.98
28	2020/1/29	60.61	2.36	4.26	18.18	0.71	1.28
29	2020/1/30	105.27	4.03	4.69	53.18	1.21	1.41
30	2020/1/30	105.27	4.03	4.69	53.18	1.21	1.41

len(df [df ['Width'] > 2.841613])

14個大於標的值的數據

台塑企業
FORMOSA PLASTICS GROUP

離群值處理 - 4

	Time	Value	Length	Width	Flow Rate	Weight	Height
0	2020/1/1	147.66	3.91	1.05	50.00	1.17	0.09
1	2020/1/2	83.03	2.76	0.96	24.91	0.83	0.29
2	2020/1/3	69.80	1.35	3.07	5.94	0.40	0.92
3	2020/1/4	56.52	1.83	1.23	10.96	0.55	0.37
4	2020/1/5	76.90	NaN	0.49	2.37	0.26	0.15
5	2020/1/6	78.53	2.69	1.57	23.56	0.81	0.47
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
29	2020/1/30	105.27	4.03	4.69	53.18	1.21	1.41
30	2020/1/30	105.27	4.03	4.69	53.18	1.21	1.41
31	2020/1/31	NaN	NaN	NaN	NaN	NaN	NaN

df [df ['Width'] <= 2.841613]

Width值小於某標的值之資料列

	Time	Value	Length	Width	Flow Rate	Weight	Height
0	2020/1/1	147.66	3.91	1.05	50.00	1.17	0.09
1	2020/1/2	83.03	2.76	0.96	24.91	0.83	0.29
3	2020/1/4	56.52	1.83	1.23	10.96	0.55	0.37
4	2020/1/5	76.90	NaN	0.49	2.37	0.26	0.15
5	2020/1/6	78.53	2.69	1.57	23.56	0.81	0.47
7	2020/1/8	196.43	4.25	1.01	58.93	1.27	0.30
11	2020/1/12	158.33	3.81	2.58	47.50	1.14	0.77
12	2020/1/13	78.88	2.69	1.12	23.66	0.81	0.34
14	2020/1/15	34.70	1.79	0.90	10.41	0.54	0.27
18	2020/1/19	136.60	4.14	1.53	55.98	1.24	0.46
20	2020/1/21	130.17	5.28	1.22	90.95	1.58	0.37
21	2020/1/22	141.79	3.61	0.31	42.54	1.08	0.09
22	2020/1/23	NaN	3.72	0.08	45.13	1.11	0.02
23	2020/1/24	74.94	2.62	0.10	22.48	0.79	0.03
24	2020/1/25	86.87	1.28	0.13	5.36	0.38	0.04
25	2020/1/26	32.96	1.74	0.17	9.89	0.52	0.05
26	2020/1/27	70.88	0.81	0.22	2.14	0.24	0.07

資料表格的合併

指令：pd. concat

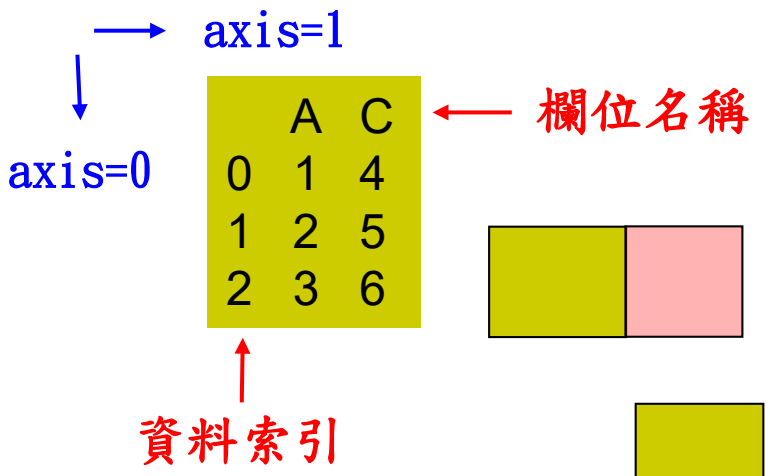
用於合併表格的指令，其參數可設定下列功能：

(1) 合併方向 (axis)

(2) 索引重建(ignore_index)

(3) 合併方式(join)

(4) 合併的主索引 (join_axes)



情境及使用目的：
因記錄數據的方法不同或各種原因，常需進行不同表格之數據合併。

資料表格合併 - 設定合併的方向(axis)

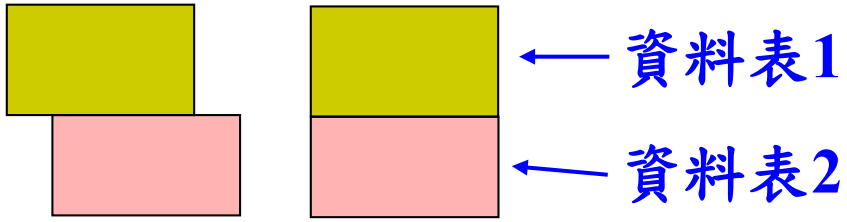
執行 pd.concat 時，若參數為

(1) axis=0 (為預設值: 即執行 pd.concat 時無參數)

表示兩資料表垂直合併(首尾相接)

```
pd.concat([df1,df2])
```

上 下

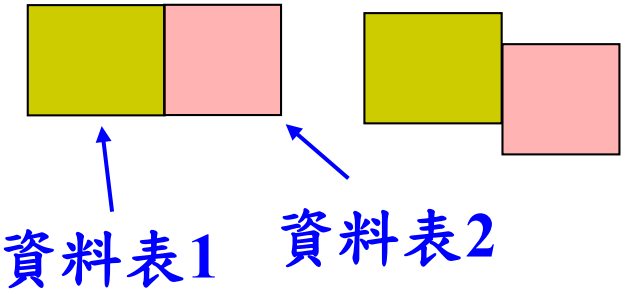


(2) axis=1：表示資料兩表水平合併

即橫向相接

```
pd.concat([df1, df2], axis=1)
```

左 右 參數



資料表格合併-設定索引的重建

- 資料表格合併時，可設定忽略原有的索引以重新編排新索引(ignore_index)

- 若忽略原有索引的參數未寫或設定為False，表示設為‘不’重新編排表格索引。

```
pd.concat([df1,df2], ignore_index=False )
```

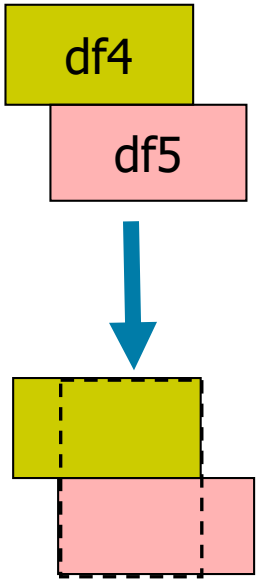
表格合併 - 設定合併方式：join='inner'

Dataframe 4 (df4)

	a	b	c	d
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0

Dataframe 5 (df5)

	b	c	d	e
2	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0



索引未重新編排

```
pd.concat([df4, df5], axis=0, join='inner')
```

兩資料表垂直合併、欄位交集合併

	b	c	d
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.0	0.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0

表格合併 - 設定合併的方式：join='outer'

Dataframe 4 (df4)

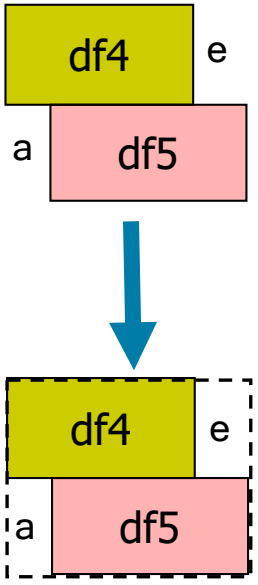
欄位名稱 →

	a	b	c	d
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0

索引 →

Dataframe 5 (df5)

	b	c	d	e
2	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0



索引未重新編排

```
pd.concat([df4, df5], axis=0, join='outer')
```

指兩資料表垂直合併、欄位外接合併

	a	b	c	d	e
1	0.0	0.0	0.0	0.0	NaN
2	0.0	0.0	0.0	0.0	NaN
3	0.0	0.0	0.0	0.0	NaN
2	NaN	1.0	1.0	1.0	1.0
3	NaN	1.0	1.0	1.0	1.0
4	NaN	1.0	1.0	1.0	1.0

NaN : Not a Number

按照 axes合併(指定合併主索引的水平合併) : join_axes

df6 的索引

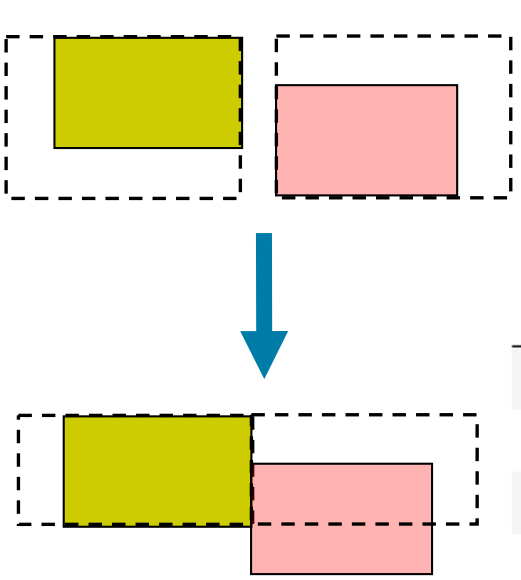
Dataframe 6 (df6)

	a	b	c	d
1	2.0	2.0	2.0	2.0
2	2.0	2.0	2.0	2.0
3	2.0	2.0	2.0	2.0

Dataframe 7 (df7)

	b	c	d	e
2	3.0	3.0	3.0	3.0
3	3.0	3.0	3.0	3.0
4	3.0	3.0	3.0	3.0

df7 的索引



pandas version < 1.0.0

設定join_axes參數，依照df6的索引水平合併

```
pd.concat([df6, df7], axis=1, join_axes=[df6.index])
```

或

```
pd.merge(df6, df7, how='left', left_index=True, right_index=True)
```

	a	b	c	d	b	c	d	e
1	2.0	2.0	2.0	2.0	NaN	NaN	NaN	NaN
2	2.0	2.0	2.0	2.0	3.0	3.0	3.0	3.0
3	2.0	2.0	2.0	2.0	3.0	3.0	3.0	3.0

pandas version >= 1.0.0

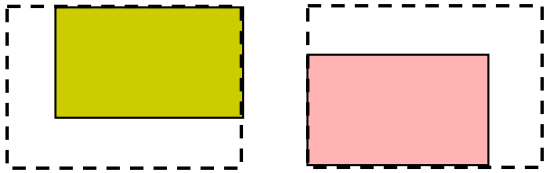
按照 axes合併(未指定合併主索引的水平合併) : join_axes

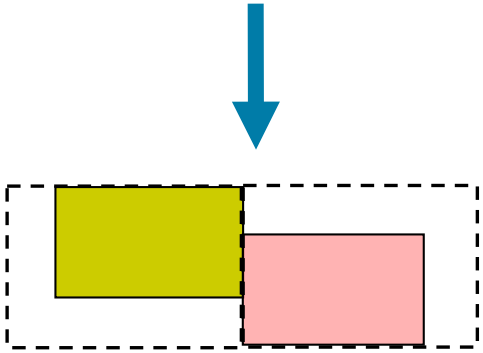
Dataframe 6 (df6)

	a	b	c	d
1	2.0	2.0	2.0	2.0
2	2.0	2.0	2.0	2.0
3	2.0	2.0	2.0	2.0

Dataframe 7 (df7)

	b	c	d	e
2	3.0	3.0	3.0	3.0
3	3.0	3.0	3.0	3.0
4	3.0	3.0	3.0	3.0





無join axes參數

```
pd.concat([df6, df7], axis=1)
```

指兩資料表水平合併

	a	b	c	d	b	c	d	e
1	2.0	2.0	2.0	2.0	NaN	NaN	NaN	NaN
2	2.0	2.0	2.0	2.0	3.0	3.0	3.0	3.0
3	2.0	2.0	2.0	2.0	3.0	3.0	3.0	3.0
4	NaN	NaN	NaN	NaN	3.0	3.0	3.0	3.0

感謝聆聽！