

Pandas

Data Analysis - Pandas

Pandas



Python Data Analysis

Pandas

- ◆ Pandas 是Python的協力廠商函式庫，提供資料框架等獨特資料結構以及各式各樣的資料處理功能，是解析資料時不可或缺的工具。
- ◆ Pandas也能跟NumPy一樣，高速地進行矩陣運算，在處理龐大的資料時相當實用。

Pandas



Pandas 資料類型 - Series & DataFrame

- ◆ Pandas 取名自 pan(el)-da(ta)-s，也與套件主要提供的三個資料結構：**Panel**、**DataFrame** 與 **Series** 相呼應
- ◆ Pandas 主要的資料結構：Series與DataFrame。

Series

	apples
0	3
1	2
2	0
3	1

Series

	oranges
0	0
1	3
2	7
3	2

+

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Pandas 資料類型 - Series

- ◆ Series，是一個類似陣列的物件，裡面可包含陣列的資料，最簡單的Series格式就是一個一維陣列的資料：
- ◆ 下麵我們宣告一個Pandas套件的Series類型資料，使用同名函數.Series()將串列進行轉換。
- ◆ 若要讀取Series的value我們可以使用values 屬性，例如ss.values。
- ◆ 若要讀取Series的index我們可以使用index 屬性，例如ss.index。
- ◆ Series的index若沒有自己設定的話它會自動分配。

```
import pandas as pd  
  
ss = pd.Series([1, -3, 5, -7])  
  
print(ss)  
  
print('----') # 分隔線  
  
print(ss.values)  
  
print('----') # 分隔線  
  
print(ss.index)
```

```
0    1  
1   -3  
2    5  
3   -7  
dtype: int64  
----  
[ 1 -3  5 -7]  
----  
RangeIndex(start=0, stop=4, step=1)
```

Index	Data
0	4
1	7
2	-5
3	3

Index Data

1	'A'
2	'B'
3	'C'
4	'D'
5	'E'

Pandas 資料類型 - Series

- ◆ index不一定要從0開始，也可以是其他的內容：

```
import pandas as pd
ss1 = pd.Series([4, 7, -5, 3], index=['a', 'b', 'c', 'd'])

print(ss1.index)          # ss1的索引資料
print(ss1['a'])           # 索引「a」的內容
print('a' in ss1)         # 索引「a」有沒有在ss1中
print(7 in ss1)           # 索引「7」有沒有在ss1中
print(7 in ss1.values)    # 內容「7」有沒有在ss1中
```

```
Index(['a', 'b', 'c', 'd'], dtype='object')
4
True
False
True
```

Pandas 資料類型 - list/dict to Series

- ◆ 我們可以從 list 或 dict 轉成 Series 型態，進行後續處理。

```
import pandas as pd

#list 轉成 Series
list_ex=["A",123,"B",456,"C",789]
list_to_Series=pd.Series(list_ex)
print(list_to_Series)

print('---'*10)

#dict 轉成 Series
dic_ex={"A":123,"B":456,"C":789}
dic_to_Series=pd.Series(dic_ex)
print(dic_to_Series)
```

Index	0
A	123
B	456
C	789

Index	0
0	A
1	123
2	B
3	456
4	C
5	789

```
0      A
1    123
2      B
3    456
4      C
5    789
dtype: object
-----
A    123
B    456
C    789
dtype: int64
```

Pandas 資料類型 - DataFrame

- ◆ DataFrame就像是我們在使用的excel表格一樣，是一個二維的數據，有row和column，我們可以透過index和column來找到某一筆資料。

Regd. No	Name	Marks%
1000	Steve	86.29
1001	Mathew	91.63
1002	Jose	72.90
1003	Patty	69.23
1004	Vin	88.30

Pandas 資料類型 - DataFrame

- ◆ 接下來我們來定義了一組 dict 資料型態 data，裡面包含name、year、month、day，接著用DataFrame()函式將它轉成DataFrame資料型態將它轉成dataframe的格式並顯示。
- ◆ DataFrame和Series一樣，若沒有特別定義index的話它會自動的分配。
- ◆ name、year、month、day就是剛剛提到的columns，左邊的0到1就是index。

```
import pandas as pd
data={"name":['Bob','Nancy'], "year":[1996, 1997], "month":[8,1], "day":[11,8]}
df=pd.DataFrame(data)
print(df)
```

	name	year	month	day
0	Bob	1996	8	11
1	Nancy	1997	1	8

Pandas 資料類型 - DataFrame

- ◆ 假如需要調整欄位置的順序，只要在函式設定columns參數，就可以進行調整。也可以針對index做設定。
- ◆ 若columns沒有對應欄位，就會產生新欄位，但內容會是NaN。

```
import pandas as pd
data={"name":['Bob','Nancy'], "year":[1996, 1997], "month":[8,1], "day":[11,8]}
df=pd.DataFrame(data)
print(df)
print('-----')
df1=pd.DataFrame(data, columns=["name","day","month","year","ABC"], index=['a','b'])
print(df1)
```

	name	year	month	day	
0	Bob	1996		8	11
1	Nancy	1997		1	8

	name	day	month	year	ABC
a	Bob	11		8	1996
b	Nancy	8		1	1997

Dataframe 概觀 - 屬性

- ◆ 接下來介紹dataframe一些基本屬性與函式，首先 import pandas：

```
import pandas as pd
```

- ◆ 接著使用read_csv()函式讀取一個CSV的檔案

```
df = pd.read_csv('csv檔案位置(包含副檔名)')
```

- ◆ 使用以下三個屬性來瞭解df的概觀

- ndim 屬性：幾維
- shape 屬性：欄列個數；顯示列與欄(rows, columns)的數量。
- dtypes 屬性：資料類型

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

```
print(df.ndim)
```

```
print("---") # 分隔線
```

```
print(df.shape)
```

```
print("---") # 分隔線
```

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

```
print("---") # 分隔線
```

```
2  
---  
(480, 6)  
---  
sno          int64  
sna          object  
tot          int64  
sbi          int64  
sarea        object  
bemp         int64  
dtype: object
```

Dataframe 概觀 - 函式

- ◆ 使用.head()函式預設最後5筆資料，也可以自訂，只需要在括弧內加上需要顯示的數量即可。

df.head(6)

- ◆ 使用.tail()函式預設顯示最後5筆資料，和head()一樣，只需要在括弧內加上需要顯示的數量即可。

df.tail(3)

- ◆ 使用.info()函式可以看到該檔案的資訊，包含欄位名稱、筆數、大小和資料類型等。

df.info()

```
import pandas as pd  
  
df=pd.read_csv('csvsample.csv')  
  
print(df.head(6))  
  
print("---") # 分隔線  
  
print(df.tail(3))  
  
print("---") # 分隔線  
  
print(df.info())
```

```
sno          sna  tot  sbi  sarea  bemp  
0  1001      大鵬華城   38   24  新店區    14  
1  1002      汐止火車站   56    7  汐止區    49  
2  1003      汐止區公所   46    5  汐止區    41  
3  1004      國泰綜合醫院   56   31  汐止區    25  
4  1005      裕隆公園    40   11  新店區    29  
5  1006  捷運大坪林站(3號出口)   32   17  新店區    12  
---  
           sno          sna  tot  sbi  sarea  bemp  
477  1491      山佳火車站   32    8  樹林區    24  
478  1492  新北市勞工活動中心   34    3  五股區    31  
479  1493      重慶公園.   40   31  板橋區     9  
---  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 480 entries, 0 to 479  
Data columns (total 6 columns):  
sno      480 non-null int64  
sna      480 non-null object  
tot      480 non-null int64  
sbi      480 non-null int64  
sarea    480 non-null object  
bemp     480 non-null int64  
dtypes: int64(4), object(2)  
memory usage: 22.6+ KB
```

資料選擇查詢 - 自行建立dataframe查詢資料

- ◆ 接下來進行資料的選擇查詢，以下我們自行建立一個dataframe物件df1，並顯示相關的資料。

```
import pandas as pd  
  
X=[['Amy','F',80],['Bob','M',65],['Cindy','F',73],['Dave','M',46],['Eva','F',46]]  
  
df1 =pd.DataFrame(X, columns=['name','gender','mathgrade'])
```

```
print("-----顯示資料---索引方式-----")
```

```
print(df1['name'])          #用欄名稱，顯示這一欄的index名稱，內容及資料型態
```

```
print(df1['name'].values)   #用欄名稱，僅顯示內容
```

```
print(df1['name'][1])       #用欄列名稱，先欄再列，且僅顯示內容
```

```
print(df1['name'][1].value) #無法執行
```

```
-----顯示資料---索引方式-----  
0      Amy  
1      Bob  
2     Cindy  
3      Dave  
4      Eva  
Name: name, dtype: object  
['Amy' 'Bob' 'Cindy' 'Dave' 'Eva']  
Bob  
Traceback (most recent call last):  
  
  File "<ipython-input-1-0875e1c0f11b>", line 11, in <module>  
    print(df1['name'][1].value) #無法執行  
  
AttributeError: 'str' object has no attribute 'value'
```

資料選擇查詢 - 讀取資料進行查詢

- ◆ 讀取檔案

```
df = pd.read_csv("nba.csv")
```

- ◆ 若要在右圖中選擇一項資料(例如Name)，可以使用以下方式：

```
print(df['Name'])
```

0	Avery Bradley
1	Jae Crowder
2	John Holland
3	R.J. Hunter
4	Jonas Jerebko
5	Amir Johnson
6	Jordan Mickey
7	Kelly Olynyk
8	Terry Rozier
9	Marcus Smart
10	Jared Sullinger

- ◆ 若只要選擇某幾筆資料，可以使用以下方式，程式碼內會先挑出Name這個column內所有資料，[0:6]則是會從索引值0到索引值5的資料。*(先column，再index)*

```
print(df['Name'][0:6])
```

0	Avery Bradley
1	Jae Crowder
2	John Holland
3	R.J. Hunter
4	Jonas Jerebko
5	Amir Johnson

Name: Name, dtype: object

資料選擇查詢 - 讀取資料進行查詢

- ◆ 若要選擇多項資料，只需要把要選擇的資料用[](List)的資料格式既可。

```
print(df[['Name','Team']].head(10))
```

	Name	Team
0	Avery Bradley	Boston Celtics
1	Jae Crowder	Boston Celtics
2	John Holland	Boston Celtics
3	R.J. Hunter	Boston Celtics
4	Jonas Jerebko	Boston Celtics
5	Amir Johnson	Boston Celtics
6	Jordan Mickey	Boston Celtics
7	Kelly Olynyk	Boston Celtics
8	Terry Rozier	Boston Celtics
9	Marcus Smart	Boston Celtics

欄位新增

- ◆ 我們可以使用insert()函式新增資料。參數裡面依序為索引位置、column名稱以及預設值(value)設定。

```
df.insert(1, column="Sport", value="checked")
print(df.head())
```

	Name	Team	Number	Position	Age	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0

	Name	Sport	Team	...	Position	Age	Salary
0	Avery Bradley	checked	Boston Celtics	...	PG	25	7730337.0
1	Jae Crowder	checked	Boston Celtics	...	SF	25	6796117.0
2	John Holland	checked	Boston Celtics	...	SG	27	NaN
3	R.J. Hunter	checked	Boston Celtics	...	SG	22	1148640.0
4	Jonas Jerebko	checked	Boston Celtics	...	PF	29	5000000.0

[5 rows x 7 columns]

欄位/資料刪除

- ◆ 我們可以透過 drop() 函式來刪除資料筆數或欄位，指定參數 axis = 0 表示要刪除筆數 (row) ，
指定參數 axis = 1 表示要刪除欄位 (column) 。

```
# 刪除欄位  
df = df.drop("Sport", axis = 1)  
print(df.head())  
print("---") # 分隔線  
  
# 刪除資料  
df = df.drop(0, axis = 0)  
print(df.head())
```

	Name	Team	Number	Position	Age	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0

	Name	Team	Number	Position	Age	Salary
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0
5	Amir Johnson	Boston Celtics	90	PF	29	12000000.0

空資料刪除

- ◆ 資料不一定都是有內容的，有些可能是空的。而在某些情況下可能需要進行空資料的刪除。而dropna()的函式就可以幫我們刪除空(NaN)資料。

df = df.dropna()

- ◆ 如右圖，可以看到在有NaN的資料內容，執行dropna()後，可以看到它不見了。

	Name	Team	Number	Position	Age	Salary
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0
5	Amir Johnson	Boston Celtics	90	PF	29	12000000.0

	Name	Team	Number	Position	Age	Salary
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0
5	Amir Johnson	Boston Celtics	90	PF	29	12000000.0
6	Jordan Mickey	Boston Celtics	55	PF	21	1170960.0

空資料填充

- ◆ 若我們不打算刪除空的資料，我們可以使用 `fillna()` 將 `NaN` 的資料代換掉。在括弧中填入自己想要填入的值即可。

```
import pandas as pd  
df = pd.read_csv("nba.csv")  
print(df.head())
```

	Name	Team	Number	Position	Age	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0

```
# 填入空資料  
df = df.fillna(10000)  
print(df.head())
```

	Name	Team	Number	Position	Age	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	10000.0
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0

資料排序

- ◆ 接下來我們來進行資料的排序。若我們想排序Name的資料，只需要在資料後面加上`.sort_values()`並在括弧內加上要排序的column即可。

```
print(df.sort_values("Name"))
```

- ◆ 若想將排序反過來，就設定ascending為False。

```
print(df.sort_values("Name", ascending= False))
```

```
import pandas as pd  
  
df = pd.read_csv("nba.csv")
```

```
print(df.head(6))
```

```
print(df.sort_values('Age').head(6))
```

```
print(df.sort_values("Name", ascending= False).head(6))
```

	Name	Team	Number	Position	Age	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0
5	Amir Johnson	Boston Celtics	90	PF	29	12000000.0
	Name	Team	...	Age	Salary	
226	Rashad Vaughn	Milwaukee Bucks	...	19	1733040.0	
122	Devin Booker	Phoenix Suns	...	19	2127840.0	
40	Kristaps Porzingis	New York Knicks	...	20	4131720.0	
401	Tyus Jones	Minnesota Timberwolves	...	20	1282080.0	
427	Cliff Alexander	Portland Trail Blazers	...	20	525093.0	
116	D'Angelo Russell	Los Angeles Lakers	...	20	5103120.0	

[6 rows x 6 columns]

	Name	Team	...	Age	Salary
237	Zaza Pachulia	Dallas Mavericks	...	32	5200000.0
271	Zach Randolph	Memphis Grizzlies	...	34	9638555.0
402	Zach LaVine	Minnesota Timberwolves	...	21	2148360.0
270	Xavier Munford	Memphis Grizzlies	...	24	NaN
386	Wilson Chandler	Denver Nuggets	...	29	10449438.0
25	Willie Reed	Brooklyn Nets	...	26	947276.0

[6 rows x 6 columns]

進階資料選擇 (索引應用) - loc() 函式

- ◆ 隨機生DataFrame 型別資料

```
import pandas as pd
import numpy as np
frame = pd.DataFrame(np.random.rand(3,3),index=list('xyz'),columns=list('XYZ'))
print(frame)
```

	X	Y	Z
x	0.705781	0.407168	0.219671
y	0.144400	0.607952	0.761035
z	0.876232	0.251522	0.466767

進階資料選擇 (索引應用) - loc() 函式

- ◆ loc() 函式會以行標籤和列標籤 (x_label 、 y_label) 進行索引
- ◆ 先行後列，中間用逗號 (,) 分割，例如取 x 和 X 對應的資料

```
print(frame.loc['x','X'])
```

```
0.705780857583845
```

- ◆ 取前兩行對應資料

```
print(frame.loc['x':'y',:])
```

	X	Y	Z
x	0.705781	0.407168	0.219671
y	0.144400	0.607952	0.761035

- ◆ 取前兩列對應資料

```
print(frame.loc[:, 'X':'Y'])
```

	X	Y
x	0.705781	0.407168
y	0.144400	0.607952
z	0.876232	0.251522

進階資料選擇 (索引應用) - loc() 函式

- ◆ 取前兩行和前兩列對應資料

```
print(frame.loc['x':'y','X':'Y'])
```

	X	Y
x	0.705781	0.407168
y	0.144400	0.607952
	X	Z

- ◆ 取第一行和第三行、第一列和第三列對應的資料

```
print(frame.loc[['x','z'],['X','Z']])
```

	X	Z
x	0.705781	0.219671
z	0.876232	0.466767

進階資料選擇 (索引應用) - iloc() 函式

- ◆ 隨機生DataFrame 型別資料

```
import pandas as pd
import numpy as np
frame = pd.DataFrame(np.random.rand(3,3),index=list('xyz'),columns=list('XYZ'))
print(frame)
```

	X	Y	Z
x	0.611111	0.434309	0.848773
y	0.833439	0.605168	0.997388
z	0.804863	0.241948	0.768817

進階資料選擇 (索引應用) - iloc() 函式

- ◆ iloc 是以行索引和列索引 (index , columns) 都是從 0 開始 。
- ◆ 如果資料的行標籤和列標籤名字太長或不容易記，則用 iloc 很方便，只需記標籤對應的索引即可 。
- ◆ 先行後列，中間用逗號 (,) 分割，例如取 x 和 X 對應的資料

```
print(frame.iloc[0,0])
```

```
0.6111110571386257
```

- ◆ 取前兩行對應資料

```
print(frame.iloc[0:2,:])
```

```
          X         Y         Z
x  0.611111  0.434309  0.848773
y  0.833439  0.605168  0.997388
```

- ◆ 取前兩列對應資料

```
print(frame.iloc[:,0:2])
```

```
          X         Y
x  0.611111  0.434309
y  0.833439  0.605168
z  0.804863  0.241948
```

進階資料選擇 (索引應用) - iloc() 函式

- ◆ 取前兩行和前兩列對應資料

```
print(frame.iloc[0:2,0:2])
```

	X	Y
x	0.611111	0.434309
y	0.833439	0.605168

- ◆ 取第一行和第三行、第一列和第三列對應的資料

```
print(frame.iloc[[0,2],[0,2]])
```

	X	Z
x	0.611111	0.848773
z	0.804863	0.768817

資料篩選(遮罩)

- ◆ 接下來要說明如何過濾資料只取得我們所要的數值，這會用到一些運算元的觀念。

```
import pandas as pd  
df = pd.read_csv("nba.csv")  
print(df.head(10))
```

	Name	Team	Number	Position	Age	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0
5	Amir Johnson	Boston Celtics	90	PF	29	12000000.0
6	Jordan Mickey	Boston Celtics	55	PF	21	1170960.0
7	Kelly Olynyk	Boston Celtics	41	C	25	2165160.0
8	Terry Rozier	Boston Celtics	12	PG	22	1824360.0
9	Marcus Smart	Boston Celtics	36	PG	22	3431040.0

資料篩選 - 運算元運用

◆ 假設我現在要找出Age是否大於25，我們可以用「>」比較運算元判斷。

◆ 執行以下的程式碼可以看到若Age>25則顯示True否則False。

```
print(df["Age"] > 25)
```

◆ 接著我們以這個當作過濾器，執行就可以看到顯示出來的資料Age大於25。

◆ 除了>當然也可以用!、==、<、>

```
mask = (print(df["Age"] > 25))
```

```
print(df[mask])
```

```
import pandas as pd

df = pd.read_csv("nba.csv")
print(df["Age"] >=25)

mask = (df["Age"] >=25)
print(df[mask].head(8))
```

```
0      True
1      True
2      True
3     False
4      True
5      True
6     False
7      True
8     False
9     False
10    False
11    True
12    True
```

	Name	Team	Number	Position	Age	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
4	Jonas Jerebko	Boston Celtics	8	PF	29	5000000.0
5	Amir Johnson	Boston Celtics	90	PF	29	12000000.0
7	Kelly Olynyk	Boston Celtics	41	C	25	2165160.0
11	Isaiah Thomas	Boston Celtics	4	PG	27	6912869.0
12	Evan Turner	Boston Celtics	11	SG	27	3425510.0

資料篩選 - 運算元運用

- ◆ 上面用比較運算元取出範圍，也可以用.between()函式取出中間的範圍：

使用.between(20,28)可以得到20到28間的數值。

`df["Age"] .between(20,28)`

```
import pandas as pd  
  
df = pd.read_csv("nba.csv")  
  
mask1 = (df["Age"] .between(20,28))  
  
print(df[mask1].head(8))
```

	Name	Team	Number	Position	Age	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6796117.0
2	John Holland	Boston Celtics	30	SG	27	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	1148640.0
6	Jordan Mickey	Boston Celtics	55	PF	21	1170960.0
7	Kelly Olynyk	Boston Celtics	41	C	25	2165160.0
8	Terry Rozier	Boston Celtics	12	PG	22	1824360.0
9	Marcus Smart	Boston Celtics	36	PG	22	3431040.0

資料分組 - groupby()

- ◆ groupby() 函式方法可以將資料依照自己要的欄位(column)進行分組，我們用classA的內容做分組的依據。

```
import pandas as pd
col = ['class', 'name', 'bd']
data = [['classA', '小明', '1995-08-01'],
        ['classB', '小美', '1995-10-02'],
        ['classC', '小黃', '1995-06-01'],
        ['classC', '小陳', '1993-11-03'],
        ['classA', '小花', '1996-01-02'],
        ['classB', '小雨', '1996-02-03']]
```

```
frame = pd.DataFrame(data, columns=col)
frame_class = frame.groupby('class')
```

依class進行分組，資料被分成classA, classB, classC，且顯示每個組有什麼資料與資料類型
print(frame_class.groups)

```
# 取出classA的資料
print(frame_class.get_group("classA"))
```

```
{'classA': Int64Index([0, 4], dtype='int64'),
 'classB': Int64Index([1, 5], dtype='int64'),
 'classC': Int64Index([2, 3], dtype='int64')}
```

	class	name	bd
0	classA	小明	1995-08-01
4	classA	小花	1996-01-02

資料分組 - groupby()

◆ 計算分組總合

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
{'A' : ['foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo', 'foo'],
 'B' : ['one', 'one', 'two', 'three', 'two', 'two', 'one', 'three'],
 'C' : np.random.randn(8),
 'D' : np.random.randn(8)})
print(df)
print(df.groupby('A').sum())          # 計算總和
print(df.groupby(['B', 'A']).sum())    # 計算總和
```

	A	B	C	D
0	foo	one	1.428961	-0.566043
1	bar	one	0.964058	-0.241626
2	foo	two	-0.657903	-0.372607
3	bar	three	2.885062	1.136388
4	foo	two	1.090373	-0.290389
5	bar	two	-0.004571	0.404098
6	foo	one	-0.928698	-0.065476
7	foo	three	1.228067	1.454848

	C	D
A		
bar	3.844549	1.298860
foo	2.160801	0.160333

	C	D
B	A	
one	bar	0.964058 -0.241626
	foo	0.500264 -0.631519
three	bar	2.885062 1.136388
	foo	1.228067 1.454848
two	bar	-0.004571 0.404098
	foo	0.432470 -0.662996

資料分組 - groupby()

◆ groupby() 函式方法可以將資料依照自己要的欄位(column)進行分組，我們用key1的內容做分組的依據。

```
import numpy as np
import pandas as pd
df = pd.DataFrame({ 'key1': [ 'A', 'B', 'A', 'C', 'B', 'D', 'C', 'D', 'A', 'C'],
'key2': [ 'one', 'two', 'one', 'two', 'one', 'two', 'one', 'two', 'one', 'one'],
'data1': np.random.randn(10),
'data2': np.random.randn(10)})
print(df)
print()
sector = df.groupby("key1")      # 以key1進行分組
print(sector)                   # 輸出sector
print()
print(type(sector))            # 輸出資料類型
print()
print(sector.size())           # 組別內的大小
print(sector.get_group("A"))    # 取出key1是A的資料
print(sector.get_group("B"))    # 取出key1是B的資料
```

	key1	key2	data1	data2
0	A	one	-0.597776	0.382917
1	B	two	0.169656	0.733708
2	A	one	-1.849105	-0.404591
3	C	two	0.522837	1.676578
4	B	one	-0.048603	-0.115005
5	D	two	0.689877	0.859946
6	C	one	-0.098791	-1.859839
7	D	two	-0.023356	-0.192720
8	A	one	-1.542370	-1.397175
9	C	one	0.071008	0.121564

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000000008F42710>
```

```
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>
```

```
key1
```

```
A 3
```

```
B 2
```

```
C 3
```

```
D 2
```

```
dtype: int64
```

	key1	key2	data1	data2
0	A	one	-0.597776	0.382917
2	A	one	-1.849105	-0.404591
8	A	one	-1.542370	-1.397175

	key1	key2	data1	data2
1	B	two	0.169656	0.733708
4	B	one	-0.048603	-0.115005

Q & A