DataFrame筆記

📊 Creation of DataFrame

```
import numpy as np
import pandas as pd
```

載入 `numpy` 與 `pandas` 套件，準備後續建立與操作 DataFrame。

1. 透過 Series 建立 DataFrame

```
#create via Series
s1 = pd.Series([1, 2, 3, 4, 5])
s2 = pd.Series([6, 7, 8, 9, 10])
df = pd.DataFrame({"col1": s1, "col2": s2})
df
```

col1	col2	
0	1	6
1	2	7
2	3	8
3	4	9
4	5	10

- `pd.Series([...])`：建立一維資料序列（Series）。
- `pd.DataFrame({...})`：使用字典，key為 column 名稱，value為對應的 Series，建立二維的 DataFrame。
- 此方式可透過多個 Series 合併成多 column 的 DataFrame。

2. 透過 Dictionary 建立 DataFrame 並指定索引與欄位順序

```
#create via Dictionary
df = pd.DataFrame(
  {
    "name": ["tom", "jack", "alice", "bob", "allen", "mike"],
    "age": [15, 17, 20, 26, 30, 17],
    "score": [60.5, 80, 30.6, 80, 83.5, 70],
  }, index=[1, 2, 3, 4, 5, 6], columns=["name", "score", "age"]  # change arrangement
)
df
```

name	score	age	
1	tom	60.5	15
2	jack	80.0	17
3	alice	30.6	20
4	bob	80.0	26
5	allen	83.5	30
6	mike	70.0	17

- 使用字典建立 DataFrame，key 為欄位名稱，value 為資料列表。
- 透過 `index` 參數指定 DataFrame 的 row index。
- 透過 `columns` 參數可指定欄位顯示順序，覆蓋預設。
- 注意欄位順序與字典順序不必相同。

3. DataFrame 的屬性

```
# properties of DataFrame
print("row index")
print(df.index)    # 回傳 row index，資料結構為 Index 物件
print("column index")
print(df.columns)  # 回傳 column index，資料結構為 Index 物件
print("value: ")
print(df.values)   # 回傳所有元素的二維 numpy array
```

```
row index
Index([1, 2, 3, 4, 5, 6], dtype='int64')
```

```
column index
Index(['name', 'score', 'age'], dtype='object')
value:
[['tom' 60.5 15]
 ['jack' 80.0 17]
 ['alice' 30.6 20]
 ['bob' 80.0 26]
 ['allen' 83.5 30]
 ['mike' 70.0 17]]
```

- `.index` 屬性代表所有 row 的標籤（index）。
- `.columns` 屬性代表所有 column 的標籤。
- `.values` 屬性回傳資料內容為二維 numpy array。

```
print("dimension", df.ndim)  # DataFrame 維度，2 表示二維結構
print("data type")
print(df.dtypes)  # 各欄位的資料型態 (dtype)
```

輸出：

```
name     object
score   float64
age       int64
dtype: object
```

- `object` 表示字串或混合資料。
- `float64` 為浮點數。
- `int64` 為整數。

```
print("shape", df.shape)  # DataFrame 的尺寸 (row數, column數)
print("size", df.size)  # DataFrame 中所有元素數量 = row * column
```

```
shape (6, 3)
size 18
```

4. DataFrame 轉置

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

	1	2	3	4	5	6
name	tom	jack	alice	bob	allen	mike
score	60.5	80.0	30.6	80.0	83.5	70.0
age	15	17	20	26	30	17

- `df.T` 代表 DataFrame 的轉置（transpose），
- row 與 column 標籤對調。

```
print(df.T.index)  # 轉置後的 row index（原本的 column）
print(df.T.columns) # 轉置後的 column index（原本的 row）
```

```
'''
Index(['name', 'score', 'age'], dtype='object')
Index([1, 2, 3, 4, 5, 6], dtype='int64')
'''
```

5. 存取元素

5.1 取 row（列）

```
print(df.loc[4])  # 用 label 存取 row index 為 4 的 row
print(df.iloc[3])  # 用 integer 位置存取第 3 (從 0 開始) row
```

name	score	age
bob	80.0	26

- `.loc[]` 依據 index 標籤存取資料。
- `.iloc[]` 依據位置序號存取資料。

5.2 取 column（行）

```
print(df.loc[:, "name"])   # 用 label 存取整個 name 欄位
print(df.iloc[:, 0])       # 用 integer 位置存取第 0 個 column
```

Source	name	score	age
Row 4	bob	80.0	26
Row 4	bob	80.0	26

5.3 取單一元素

```
print(df.at[3, "score"])   # 用 label 存取指定 row 與 column 的單一元素 (快速存取)
print(df.iat[2, 1])        # 用位置索引存取指定元素 (快速存取)
print(df.loc[3, "score"])  # 用 label 存取元素（較通用但較慢）
print(df.iloc[2, 1])       # 用位置索引存取元素（較通用但較慢）
```

```
'''
30.6
30.6
30.6
30.6
'''
```

5.4 取單一 row 的某欄位資料（Series）

```
print(df["name"])        # 取出 name 欄，結果為 Series
print(type(df["name"]))  # 資料型態為 pandas Series
```

```
'''
1 tom
2 jack
3 alice
4 bob
5 allen
6 mike
Name: name, dtype: object
```

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

5.5 另一種存取欄位的語法（屬性方式）

```
print(df.name)          # 取出 name 欄，等同於 df["name"]
print(type(df.name))    # 資料型態為 pandas Series
```

```
'''
1 tom
2 jack
3 alice
4 bob
5 allen
6 mike
Name: name, dtype: object
<class 'pandas.core.series.Series'>
'''
```

5.6 取出 DataFrame 格式的欄位（雙中括號）

```
print(df["name"]("name"))      # 取出 name 欄，結果為 DataFrame（非 Series）
print(type(df["name"]("name")))  # 資料型態為 pandas DataFrame
df["name"]("name")             # 顯示結果
```

```
'''
name 1
tom 2
jack 3
alice 4
bob 5
allen 6 mike
<class 'pandas.core.frame.DataFrame'>
'''
```

小結

- DataFrame 主要是以 row index 與 column index 結合多維陣列資料的結構。
- `.loc` 與 `.iloc` 用於依標籤或位置選取資料，支援 row 或 column。
- `.at` 與 `.iat` 為快速存取單一元素。
- 欄位資料取出時，使用單中括號取得的是 Series，雙中括號則是 DataFrame。
- 轉置 `.T` 可快速交換 row 與 column。