

# Python 的 50+ 練習：資料科學學習手冊

資料科學模組 *Pandas* 入門

數據交點 | 郭耀仁 [yaojenkuo@datainpoint.com](mailto:yaojenkuo@datainpoint.com)

# 這個章節會登場的模組

`pandas` 模組。

關於 Pandas

# 什麼是 Pandas

*Pandas 是 Python 處理表格式資料 ( Tabular data ) 的第三方模組，它創造了 `Index`、`Series` 與 `DataFrame` 的資料結構類別，讓 Python 在面對表格式資料時能夠用更直覺的觀念操作。*

來源：<https://github.com/pandas-dev/pandas>

# ( 沒什麼用的冷知識 ) Pandas 跟熊貓「沒有關係」

1. **Panel**(自從版本 0.20.0 之後棄用)
2. **DataFrame**
3. **Series**



來源：<https://media.giphy.com/media/46Zj6ze2Z2t4k/giphy.gif>

# 根據說明文件的範例載入

來源：[https://pandas.pydata.org/docs/user\\_guide/10min.html](https://pandas.pydata.org/docs/user_guide/10min.html)

In [1]:

```
import pandas as pd
```

如果環境中沒有安裝 Pandas，載入時會遭遇  
**ModuleNotFoundError**

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
ModuleNotFoundError: No module named 'pandas'
```

如果遭遇 `ModuleNotFoundError` 可以在終端機使用 `pip install pandas` 或者 `conda install pandas` 指令安裝

若要指定模組版本可以加上 `==MAJOR.MINOR.PATCH` 課程使用的模組版本為 1.3

```
pip install pandas==1.3
```

或者

```
conda install pandas==1.3
```



## 可以透過兩個屬性檢查版本號與安裝路徑

- `__version__` 屬性檢查版本號。
- `__file__` 屬性檢查安裝路徑。

In [2]:

```
print(pd.__version__)  
print(pd.__file__)
```

1.3.0

/Users/kuoyaojen/opt/miniconda3/envs/pythonfiftyplus/lib/python3.9/site-packages/pandas/\_\_init\_\_.py

入門 Pandas 的第一步就是掌握 `Index`、`ndarray`、`Series` 與 `DataFrame` 四個資料結構類別彼此之間的關係

- `Series` 由 `Index` 與 `ndarray` 組合而成。
- `DataFrame` 由數個共享同一個 `Index` 的 `Series` 組合而成。

具備 tuple 與 set 特性的 Index

# Pandas 的 Index 類別

使用 `pd.Index()` 函數創造 Index 類別的實例。

In [3]:

```
import numpy as np
```

In [4]:

```
primes_array = np.array([2, 3, 5, 7, 11, 13, 17, 19, 23, 29])
prime_indexes = pd.Index(primes_array)
print(prime_indexes)
print(type(prime_indexes))
```

```
Int64Index([2, 3, 5, 7, 11, 13, 17, 19, 23, 29], dtype='int64')
<class 'pandas.core.indexes.numeric.Int64Index'>
```

# Index 的基礎屬性

- `Index.dtype` 資料類別。
- `Index.size` 元素個數。

In [5]:

```
print(prime_indexes.dtype)  
print(prime_indexes.size)
```

```
int64  
10
```

# Index 類別結合 Python 內建的 tuple 與 set 兩種資料結構類別的特性

- 具有 tuple 無法更動的特性。
- 具有 set 集合運算的特性。

# Index 類別具有 tuple 無法更動的特性

In [6]:

```
# Index has the characteristics of a tuple
primes_array = np.array([2, 3, 5, 7, 11, 13, 17, 19, 23, 29])
prime_indexes = pd.Index(primes_array)
try:
    prime_indexes[-1] = 31
except TypeError as error_message:
    print(error_message)
```

Index does not support mutable operations

# Index 類別具有 set 集合運算的特性

In [7]:

```
# Index has the characteristics of a set
primes_array = np.array([2, 3, 5, 7, 11, 13, 17, 19, 23, 29])
prime_indexes = pd.Index(primes_array)
odd_indexes = pd.Index(np.arange(1, 30, 2))
print(prime_indexes)
print(odd_indexes)
```

```
Int64Index([2, 3, 5, 7, 11, 13, 17, 19, 23, 29], dtype='int64')
```

```
Int64Index([1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29], dtype='int64')
```



# Index 具有與 set 同樣名稱的集合運算方法

- `Index.intersection()` 交集。
- `Index.union()` 聯集。
- `Index.difference()` 差集。
- `Index.symmetric_difference()` 對稱差集。

In [8]:

```
# Set operations of Index
print(prime_indexes.intersection(odd_indexes))
print(prime_indexes.union(odd_indexes))
print(prime_indexes.difference(odd_indexes))
print(odd_indexes.difference(prime_indexes))
print(prime_indexes.symmetric_difference(odd_indexes))
```

```
Int64Index([3, 5, 7, 11, 13, 17, 19, 23, 29], dtype='int64')
Int64Index([1, 2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25,
27, 29], dtype='int64')
Int64Index([2], dtype='int64')
Int64Index([1, 9, 15, 21, 25, 27], dtype='int64')
Int64Index([1, 2, 9, 15, 21, 25, 27], dtype='int64')
```

Index 加 ndarray 等於 Series

# Pandas 的 Series 類別

使用 `pd.Series()` 函數創造 Series 類別的實例。

In [9]:

```
months_array = np.arange(1, 13)
months_series = pd.Series(months_array)
print(months_series)
print(type(months_series))
```

```
0      1
1      2
2      3
3      4
4      5
5      6
6      7
7      8
8      9
9     10
10     11
11     12
dtype: int64
<class 'pandas.core.series.Series'>
```

# Series 的基礎屬性與方法

- `Series.dtype` 資料類別。
- `Series.size` 元素個數。
- `Series.index` 取出 `Series` 的 `Index` 部分。
- `Series.values` 取出 `Series` 的 `ndarray` 部分。
- `Series.astype()` 轉換 `Series` 的資料類別。

In [10]:

```
print(months_series.dtype)
print(months_series.size)
```

```
int64
12
```

# Series 由 Index 與 ndarray 組合而成

In [11]:

```
print(months_series.index)
print(type(months_series.index))
```

```
RangeIndex(start=0, stop=12, step=1)
<class 'pandas.core.indexes.range.RangeIndex'>
```

In [12]:

```
print(months_series.values)
print(type(months_series.values))
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12]
<class 'numpy.ndarray'>
```

## 調整 Series 的 Index

- 在建立的時候指定。
- 透過 `Series.index` 更新。

# 在建立的時候指定

In [13]:

```
months_abbreviation = ["JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"]  
months_series = pd.Series(months_array, index=months_abbreviation)  
months_series
```

Out[13]:

```
JAN      1  
FEB      2  
MAR      3  
APR      4  
MAY      5  
JUN      6  
JUL      7  
AUG      8  
SEP      9  
OCT     10  
NOV     11  
DEC     12  
dtype: int64
```

# 透過 Series.index 更新

In [14]:

```
months_abbreviation = ["JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"]  
months_series = pd.Series(months_array)  
months_series.index = months_abbreviation  
months_series
```

Out[14]:

```
JAN      1  
FEB      2  
MAR      3  
APR      4  
MAY      5  
JUN      6  
JUL      7  
AUG      8  
SEP      9  
OCT     10  
NOV     11  
DEC     12  
dtype: int64
```



## 如何取出 `Series` 中的元素

- 以元素位置 indexing/slicing
- 以 `Index` indexing/slicing

## 以元素位置 indexing/slicing

In [15]:

```
print(months_series[0])  
print(months_series[:3])
```

```
1  
JAN    1  
FEB    2  
MAR    3  
dtype: int64
```

## 以 Index indexing/slicing

In [16]:

```
print(months_series["JAN"])  
print(months_series["JAN":"MAR"])
```

```
1  
JAN    1  
FEB    2  
MAR    3  
dtype: int64
```

## 轉換 `Series` 的資料類別

- 在建立的時候指定。
- 透過 `Series.astype()` 轉換 `Series` 的資料類別。

# 在建立的時候指定

In [17]:

```
months_series = pd.Series(months_array, dtype=float)  
months_series
```

Out[17]:

```
0      1.0  
1      2.0  
2      3.0  
3      4.0  
4      5.0  
5      6.0  
6      7.0  
7      8.0  
8      9.0  
9     10.0  
10     11.0  
11     12.0  
dtype: float64
```

# Series.astype() 轉換 Series 的資料類別

In [18]:

```
months_series = pd.Series(months_array)
print(months_series.dtype)
print(months_series.astype(float))
```

```
int64
0      1.0
1      2.0
2      3.0
3      4.0
4      5.0
5      6.0
6      7.0
7      8.0
8      9.0
9     10.0
10     11.0
11     12.0
```

結合多個相同 Index 的 Series 成為  
DataFrame

# Pandas 的 DataFrame 類別

使用 `pd.DataFrame()` 函數創造 DataFrame 類別的實例。

- 輸入以欄為基準 ( Column-based ) 的資料內容。
- 輸入以行為基準 ( Row-based ) 的資料內容。



# 輸入以欄為基準 ( Column-based ) 的資料內容

In [19]:

```
movie_df = pd.DataFrame()
movie_df["title"] = ["The Shawshank Redemption", "The Dark Knight", "Schindler's List", "Forrest Gump", "Inception"]
movie_df["imdb_rating"] = [9.3, 9.0, 8.9, 8.8, 8.7]
movie_df["release_year"] = [1994, 2008, 1993, 1994, 2010]
print(movie_df)
print(type(movie_df))
```

	title	imdb_rating	release_year
0	The Shawshank Redemption	9.3	1994
1	The Dark Knight	9.0	2008
2	Schindler's List	8.9	1993
3	Forrest Gump	8.8	1994
4	Inception	8.7	2010

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

## 輸入以列為基準 ( Row-based ) 的資料內容

In [20]:

```
movies = [  
    {"title": "The Shawshank Redemption", "imdb_rating": 9.3, "release_year": 1994},  
    {"title": "The Dark Knight", "imdb_rating": 9.0, "release_year": 2008},  
    {"title": "Schindler's List", "imdb_rating": 8.9, "release_year": 1993},  
    {"title": "Forrest Gump", "imdb_rating": 8.8, "release_year": 1994},  
    {"title": "Inception", "imdb_rating": 8.7, "release_year": 2010},  
]  
movie_df = pd.DataFrame(movies)  
print(movie_df)  
print(type(movie_df))
```

	title	imdb_rating	release_year
0	The Shawshank Redemption	9.3	1994
1	The Dark Knight	9.0	2008
2	Schindler's List	8.9	1993
3	Forrest Gump	8.8	1994
4	Inception	8.7	2010

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

# Jupyter Notebook 針對 DataFrame 類別有特別的顯示外觀

In [21]:

```
movie_df
```

Out[21]:

	title	imdb_rating	release_year
0	The Shawshank Redemption	9.3	1994
1	The Dark Knight	9.0	2008
2	Schindler's List	8.9	1993
3	Forrest Gump	8.8	1994
4	Inception	8.7	2010

# DataFrame 的基礎屬性

- `DataFrame.dtypes` 資料類別。
- `DataFrame.shape` 外型。
- `DataFrame.index` 取出列標籤 ( row labels ) 部分。
- `DataFrame.columns` 取出欄標籤 ( column labels ) 的部分。

In [22]:

```
print(movie_df.dtypes)
print(movie_df.shape)
print(movie_df.index)
print(movie_df.columns)
```

```
title           object
imdb_rating     float64
release_year    int64
dtype: object
(5, 3)
RangeIndex(start=0, stop=5, step=1)
Index(['title', 'imdb_rating', 'release_year'], dtype='object')
```

DataFrame 由數個 Series 共享同一個 Index 組成

In [23]:

```
print(type(movie_df.index))  
print(type(movie_df["title"]))  
print(type(movie_df["imdb_rating"]))  
print(type(movie_df["release_year"]))
```

```
<class 'pandas.core.indexes.range.RangeIndex'>  
<class 'pandas.core.series.Series'>  
<class 'pandas.core.series.Series'>  
<class 'pandas.core.series.Series'>
```

# DataFrame 的基礎方法

- `DataFrame.head(n)` 檢視前 `n` 列。
- `DataFrame.tail(n)` 檢視後 `n` 列。
- `DataFrame.describe()` 檢視數值欄位的描述性統計。
- `DataFrame.info()` 檢視詳細資訊。

## 檢視前 `n` 列、後 `n` 列

- `DataFrame.head(n)` 檢視前 `n` 列。
- `DataFrame.tail(n)` 檢視後 `n` 列。

In [24]:

```
movie_df.head(3)
```

Out[24]:

	title	imdb_rating	release_year
0	The Shawshank Redemption	9.3	1994
1	The Dark Knight	9.0	2008
2	Schindler's List	8.9	1993

In [25]:

```
movie_df.tail(2)
```

Out[25]:

	title	imdb_rating	release_year
3	Forrest Gump	8.8	1994
4	Inception	8.7	2010

## DataFrame.describe() 檢視數值欄位的描述性統計

In [26]:

```
movie_df.describe()
```

Out[26]:

	imdb_rating	release_year
count	5.000000	5.000000
mean	8.940000	1999.800000
std	0.230217	8.438009
min	8.700000	1993.000000
25%	8.800000	1994.000000
50%	8.900000	1994.000000
75%	9.000000	2008.000000
max	9.300000	2010.000000



## DataFrame.info() 檢視詳細資訊

In [27]:

```
movie_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5 entries, 0 to 4  
Data columns (total 3 columns):  
#   Column          Non-Null Count  Dtype  
---  -  
0   title           5 non-null     object  
1   imdb_rating     5 non-null     float64  
2   release_year    5 non-null     int64  
dtypes: float64(1), int64(1), object(1)  
memory usage: 248.0+ bytes
```

# 重點統整

- Pandas 創造了 `Index`、`Series` 與 `DataFrame` 的資料結構類別，讓 Python 在面對表格式資料時能夠用更直覺的觀念操作。
- 入門 Pandas 的第一步就是掌握 `Index`、`ndarray`、`Series` 與 `DataFrame` 四個資料結構類別彼此之間的關係。
- `Index` 類別結合 Python 內建的 `tuple` 與 `set` 兩種資料結構類別的特性。
- `Series` 由 `Index` 與 `ndarray` 組合而成。
- `DataFrame` 由數個共享同一個 `Index` 的 `Series` 組合而成。

