Python 的 50+ 練習

基礎資料框操作

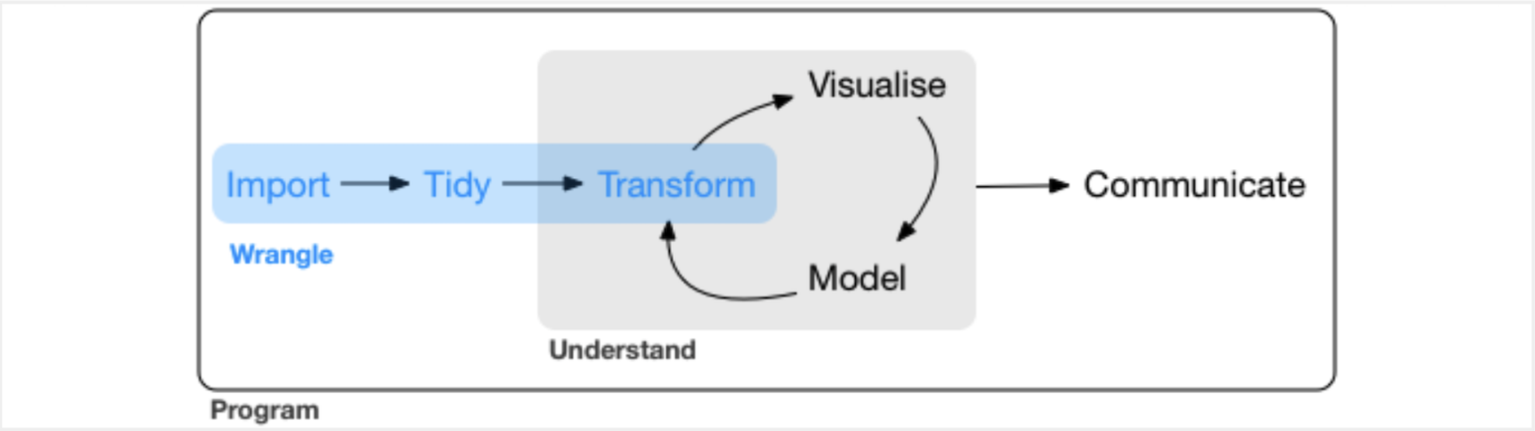
數據交點 | 郭耀仁 yaojenkuo@datainpoint.com

這個章節會登場的模組

`pandas` 模組。

關於基礎的資料框操作

（複習）現代資料科學：以程式設計做資料科學的應用



來源：R for Data Science

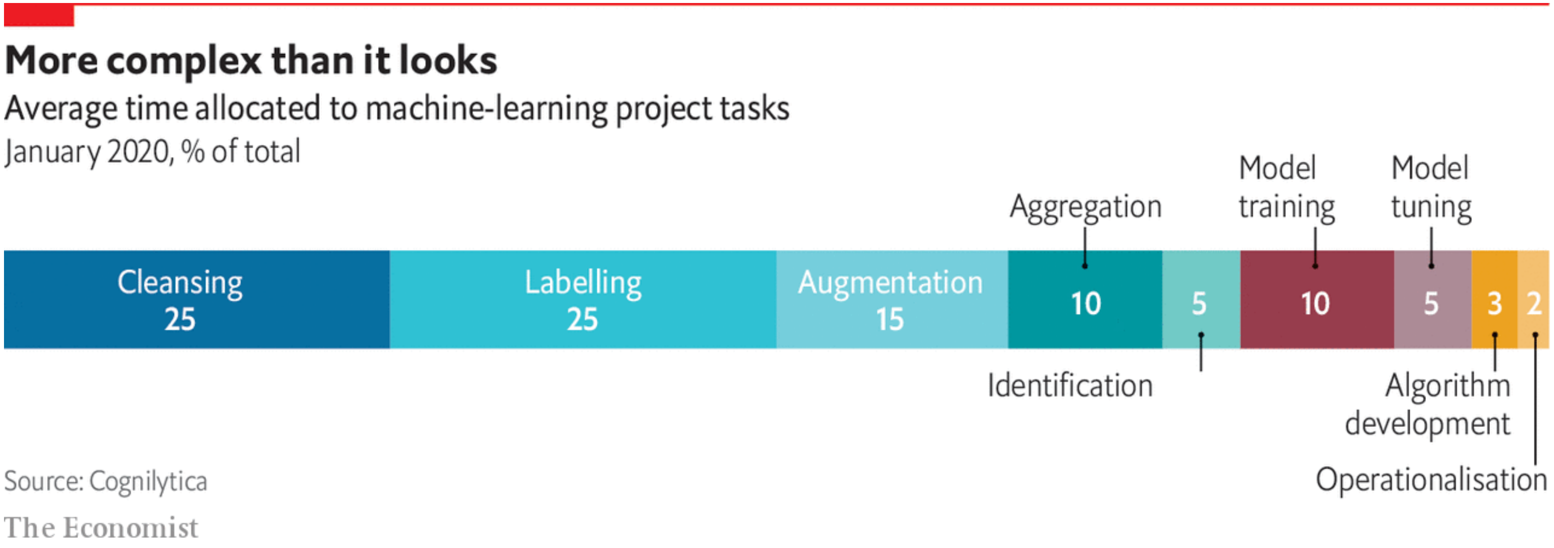
（複習）什麼是資料科學的應用場景

- Import 資料的載入。
- **Tidy 資料清理**。
- **Transform 資料外型與類別的轉換**。
- Visualise 探索性分析。
- Model 分析與預測模型。
- Communicate 溝通分享。

（沒什麼用的冷知識）Wrangle

來源：https://media.giphy.com/media/MnlZWRFHR4xruE4N2Z/giphy.gif

機器學習專案花費 50% 的時間處理 Data Wrangle 的相關任務



來源：https://www.economist.com/technology-quarterly/2020/06/11/for-ai-data-are-harder-to-come-by-than-you-think

多數的資料清理、資料外型與類別的轉換是面對 `DataFrame`

（複習）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。

`DataFrame` 是有兩個維度的資料結構

- 第一個維度稱為觀測值（Observations），有時亦稱為列（Rows）
- 第二個維度稱為變數（Variables），有時亦稱為欄（Columns）
- 我們習慣以 `(m, n)` 或者 `m x n` 來描述一個具有 `m` 列觀測值、`n` 欄變數的 `DataFrame`

In [1]:

```
import pandas as pd

movies = pd.read_csv("/home/jovyan/data/internet-movie-database/movies.csv")
print(movies.shape)
print(movies.head().shape)
movies.head() # just show the first 5 rows
```

```
(250, 6)
(5, 6)
```

Out[1]:

	id	title	release_year	rating	director	runtime
0	1	The Shawshank Redemption	1994	9.3	Frank Darabont	142
1	2	The Godfather	1972	9.2	Francis Ford Coppola	175
2	3	The Godfather: Part II	1974	9.0	Francis Ford Coppola	202
3	4	The Dark Knight	2008	9.0	Christopher Nolan	152
4	5	12 Angry Men	1957	9.0	Sidney Lumet	96

DataFrame 與二維 ndarray 不同的地方

- DataFrame 的每個變數可以是異質的。
- DataFrame 的觀測值具有列標籤（row-label）、變數具有欄標籤（column-label）

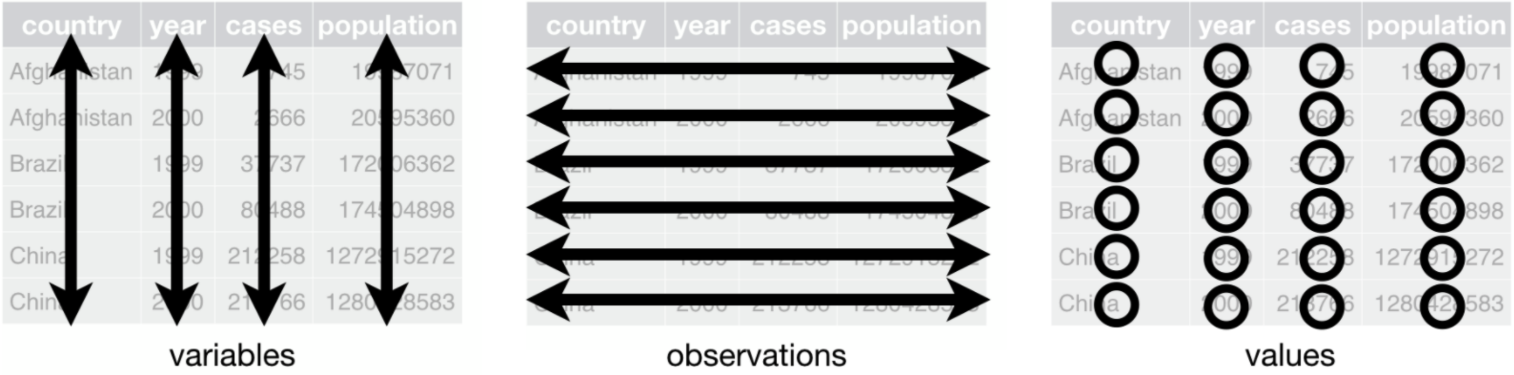
In [2]:

```python
print(movies.dtypes)  # heterogeneous
print(movies.index)   # row-label
print(movies.columns) # column-label
```

```
id                  int64
title              object
release_year        int64
rating            float64
director           object
runtime             int64
dtype: object
RangeIndex(start=0, stop=250, step=1)
Index(['id', 'title', 'release_year', 'rating', 'director',
'runtime'], dtype='object')
```

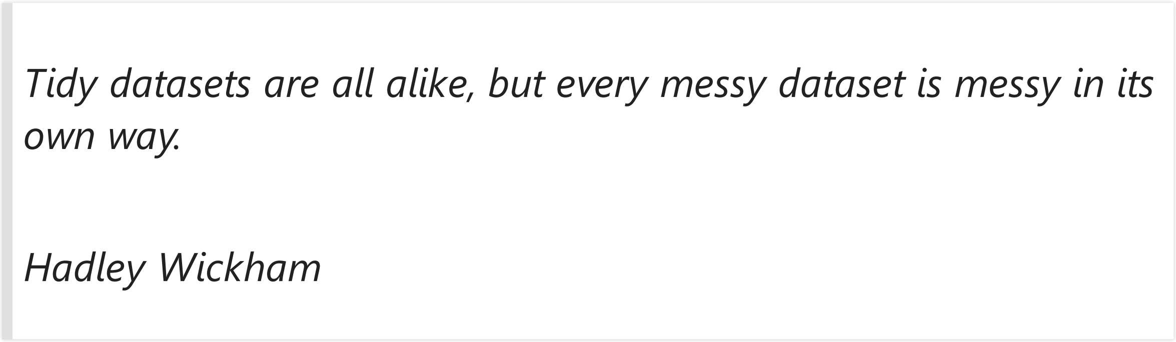
什麼是乾淨資料

1. 每個變數有自己的欄位。
2. 每個觀測值有自己的資料列。
3. 每個列、欄標籤與值的對應有自己的儲存格。



來源：https://r4ds.had.co.nz/tidy-data.html

不乾淨資料有著各自的樣態



Tidy datasets are all alike, but every messy dataset is messy in its own way.

Hadley Wickham

來源：https://r4ds.had.co.nz/tidy-data.html

不乾淨資料

In [4]:

```
messy_data
```

Out[4]:

	column_0	column_1
0	Luke Skywalker\nAnakinSkywalker	Luke Skywalker, Mark Hamill
1	None	Anakin Skywalker, Hayden Christensen

乾淨資料

In [6]:

```
tidy_data
```

Out[6]:

	type	first_name	last_name
0	character	Luke	Skywalker
1	character	Anakin	Skywalker
2	actor	Mark	Hamill
3	actor	Hayden	Christensen

Pandas 使用更直觀的概念操作資料

- 如何定義「更直觀」？
    - 像操作試算表一般（Spreadsheet-like）
    - 像使用結構化查詢語言一般（SQL-like）

選擇欄位

以 `DataFrame["column"]` 選擇欄位成為外型 `(m,)` 的 `Series`

In [7]:

```
print(type(movies["title"]))
print(movies["title"].shape)
movies["title"]
```

```
<class 'pandas.core.series.Series'>
(250,)
```

Out[7]:

```
0                            The Shawshank Redemption
1                                       The Godfather
2                              The Godfather: Part II
3                                     The Dark Knight
4                                        12 Angry Men
                              ...
245    Neon Genesis Evangelion: The End of Evangelion
246                                7 Kogustaki Mucize
247                                        Tangerines
248                                          Drishyam
```

```
249                                              Swades
Name: title, Length: 250, dtype: object
```

以 `DataFrame[["column"]]` 選擇欄位成為外型 `(m, 1)` 的 `DataFrame`

In [8]:

```
print(type(movies[["title"]]))
print(movies[["title"]].shape)
movies[["title"]]
```

```
<class 'pandas.core.frame.DataFrame'>
(250, 1)
```

Out[8]:

	title
0	The Shawshank Redemption
1	The Godfather
2	The Godfather: Part II
3	The Dark Knight
4	12 Angry Men
...	...
245	Neon Genesis Evangelion: The End of Evangelion
246	7 Kogustaki Mucize
247	Tangerines
248	Drishyam
249	Swades

250 rows × 1 columns

以 `DataFrame[["column_0", "column_1", ...]]` 選擇多個欄位成為外型 `(m, n)` 的 `DataFrame`

運用 Fancy indexing 於欄位的選擇。

In [9]:

```
movies[["title", "director", "release_year", "rating"]]
```

Out[9]:

	title	director	release_year	rating
0	The Shawshank Redemption	Frank Darabont	1994	9.3
1	The Godfather	Francis Ford Coppola	1972	9.2
2	The Godfather: Part II	Francis Ford Coppola	1974	9.0
3	The Dark Knight	Christopher Nolan	2008	9.0
4	12 Angry Men	Sidney Lumet	1957	9.0
...	...	...	...	...
245	Neon Genesis Evangelion: The End of Evangelion	Hideaki Anno	1997	8.1
246	7 Kogustaki Mucize	Mehmet Ada Öztekin	2019	8.2
247	Tangerines	Zaza Urushadze	2013	8.2
248	Drishyam	Nishikant Kamat	2015	8.2
249	Swades	Ashutosh Gowariker	2004	8.2

250 rows × 4 columns

以 `DataFrame.loc[:, [column_0, column_1, ...]]` 選擇多個欄位成為外型 `(m, n)` 的 `DataFrame`

- `loc` 是透過資料位置（Location）指定，也就是根據欄標籤選擇。
- `:` 表示不針對資料列篩選。

In [10]:

```
movies.loc[:, ["title", "director", "release_year", "rating"]]
```

Out[10]:

	title	director	release_year	rating
0	The Shawshank Redemption	Frank Darabont	1994	9.3
1	The Godfather	Francis Ford Coppola	1972	9.2
2	The Godfather: Part II	Francis Ford Coppola	1974	9.0
3	The Dark Knight	Christopher Nolan	2008	9.0
4	12 Angry Men	Sidney Lumet	1957	9.0
...	...	...	...	...
245	Neon Genesis Evangelion: The End of Evangelion	Hideaki Anno	1997	8.1
246	7 Kogustaki Mucize	Mehmet Ada Öztekin	2019	8.2
247	Tangerines	Zaza Urushadze	2013	8.2
248	Drishyam	Nishikant Kamat	2015	8.2
249	Swades	Ashutosh Gowariker	2004	8.2

250 rows × 4 columns

以 `DataFrame.iloc[:, [0, 1, ...]]` 選擇多個欄位成為外型 `(m, n)` 的 `DataFrame`

- `iloc` 是透過資料整數位置（Integer location）指定，也就是根據欄的整數位置選擇。
- `:` 表示不針對資料列篩選。

In [11]:

```python
movies.iloc[:, [1, 4, 2, 3]]
```

Out[11]:

	title	director	release_year	rating
0	The Shawshank Redemption	Frank Darabont	1994	9.3
1	The Godfather	Francis Ford Coppola	1972	9.2
2	The Godfather: Part II	Francis Ford Coppola	1974	9.0
3	The Dark Knight	Christopher Nolan	2008	9.0
4	12 Angry Men	Sidney Lumet	1957	9.0
...	...	...	...	...
245	Neon Genesis Evangelion: The End of Evangelion	Hideaki Anno	1997	8.1
246	7 Kogustaki Mucize	Mehmet Ada Öztekin	2019	8.2
247	Tangerines	Zaza Urushadze	2013	8.2
248	Drishyam	Nishikant Kamat	2015	8.2
249	Swades	Ashutosh Gowariker	2004	8.2

250 rows × 4 columns

篩選資料列

善用 `Series` 的特性

- `Series` 是由 `Index` 與 `ndarray` 組合而成，具備了 `ndarray` 的特性。
- 善用元素操作（Elementwise）運算。
- 善用特殊的 indexing 語法：Fancy indexing/Boolean indexing

透過列標籤（row-label）篩選資料列

- 在 `movies` 中魔戒三部曲分別位於列標籤 6, 9, 13
    - `loc` 是透過資料位置（Location）指定，也就是根據列標籤篩選。
    - `:` 表示不針對欄位選擇。
- 運用 Fancy indexing 於資料列的篩選。

In [12]:

```
movies.loc[[6, 9, 13], :]
```

Out[12]:

	id	title	release_year	rating	director	runtime
6	7	The Lord of the Rings: The Return of the King	2003	8.9	Peter Jackson	201
9	10	The Lord of the Rings: The Fellowship of the Ring	2001	8.8	Peter Jackson	178
13	14	The Lord of the Rings: The Two Towers	2002	8.7	Peter Jackson	179

以 `DataFrame.iloc[[0, 1, ...], :]` 選擇多個資料列

- 在 `lord_of_the_rings` 中第一集與第三集分別位於第 0, 1 列。
  - `iloc` 是透過資料整數位置（Integer location）指定，也就是根據列的整數位置篩選。
  - `:` 表示不針對欄位選擇。
- 運用 Fancy indexing 於資料列的篩選。

In [13]:

```
lord_of_the_rings = movies.loc[[6, 9, 13], :]
lord_of_the_rings.iloc[[0, 1], :]
```

Out[13]:

	id	title	release_year	rating	director	runtime
6	7	The Lord of the Rings: The Return of the King	2003	8.9	Peter Jackson	201
9	10	The Lord of the Rings: The Fellowship of the Ring	2001	8.8	Peter Jackson	178

區分 `DataFrame` 的兩種索引語法

- `DataFrame.loc[row-label, column-label]` 以列、欄標籤為準。
- `DataFrame.iloc[row-integer-location, column-integer-location]` 以列、欄整數位置為準。

透過條件敘述以 `DataFrame[booleans]` 篩選資料列

- 運用 Boolean indexing 於資料列的篩選。
- 熟悉之後會直接將條件敘述寫在中括號裡。

In [14]:

```
boolean_series = movies["director"] == "Peter Jackson"
movies[boolean_series] # movies[movies["director"] == "Peter Jackson"]
```

Out[14]:

	id	title	release_year	rating	director	runtime
6	7	The Lord of the Rings: The Return of the King	2003	8.9	Peter Jackson	201
9	10	The Lord of the Rings: The Fellowship of the Ring	2001	8.8	Peter Jackson	178
13	14	The Lord of the Rings: The Two Towers	2002	8.7	Peter Jackson	179

如何負面表列（ negate ）由 `bool` 組成的 Series

- 使用相反的關係運算符 `==` vs. `!=`
- 使用 `~` 運算符。

In [15]:

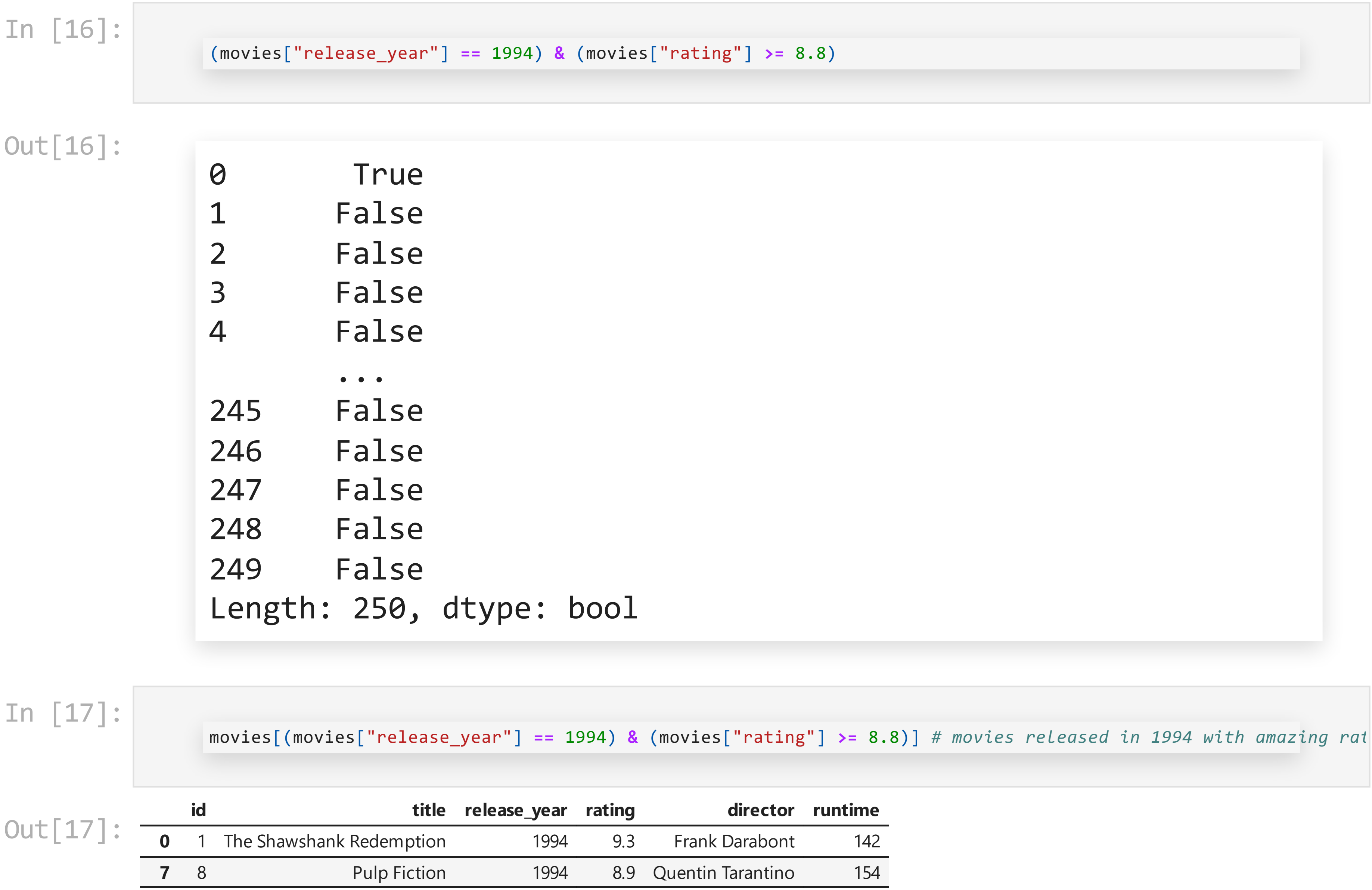
```
boolean_series = movies["director"] != "Peter Jackson"
print(boolean_series.sum())
boolean_series = movies["director"] == "Peter Jackson"
print((~boolean_series).sum())
```

```
247
247
```

加入多個條件敘述篩選資料列

- 運用 `&` 運算符**交集**多個條件敘述。
- 運用 `|` 運算符**聯集**多個條件敘述。

運用 & 運算符交集多個條件敘述



```
In [16]: (movies["release_year"] == 1994) & (movies["rating"] >= 8.8)
```

```
Out[16]:
0       True
1      False
2      False
3      False
4      False
       ...
245    False
246    False
247    False
248    False
249    False
Length: 250, dtype: bool
```

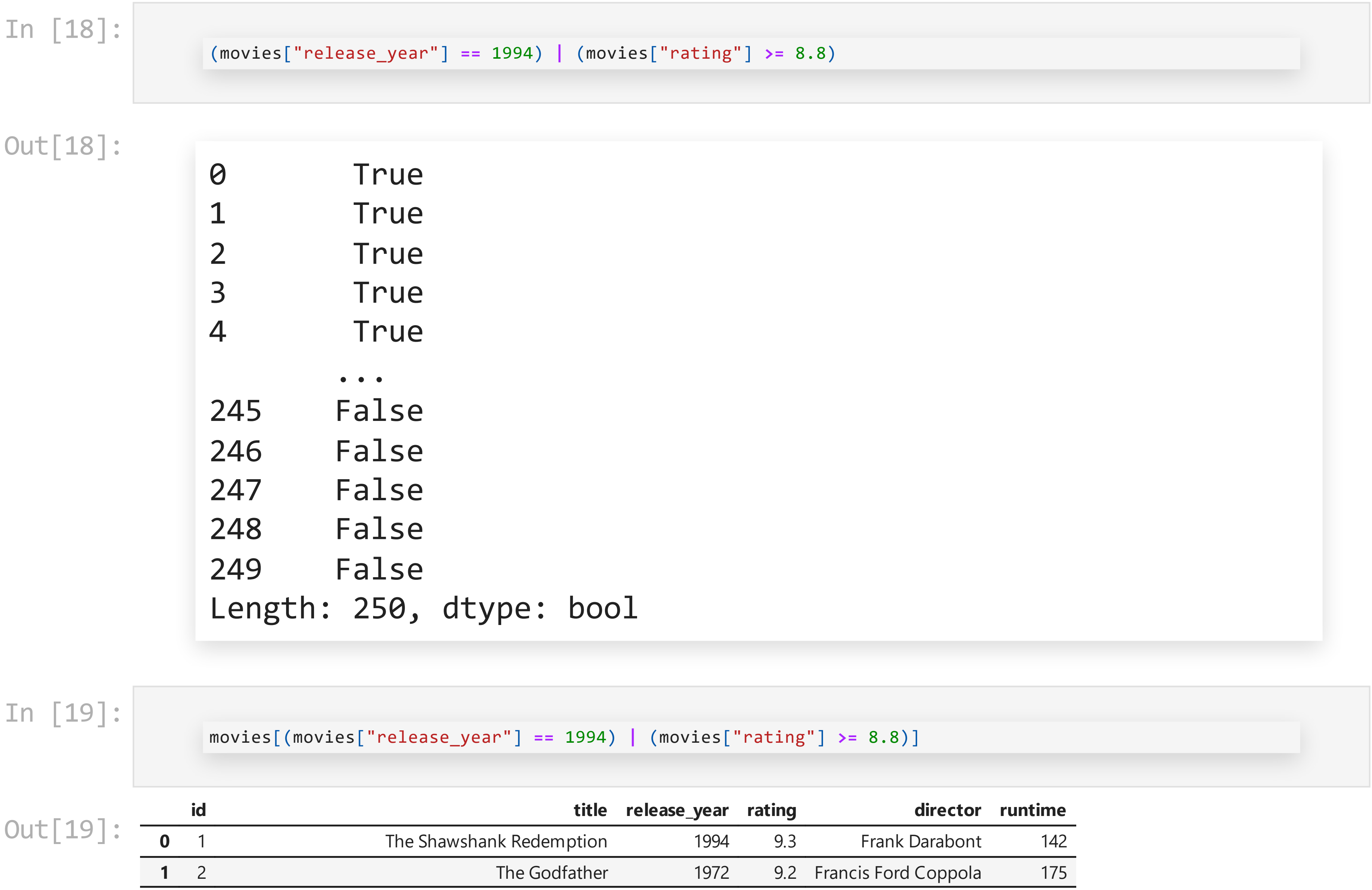
```
In [17]: movies[(movies["release_year"] == 1994) & (movies["rating"] >= 8.8)] # movies released in 1994 with amazing rat
```

Out[17]:

	id	title	release_year	rating	director	runtime
0	1	The Shawshank Redemption	1994	9.3	Frank Darabont	142
7	8	Pulp Fiction	1994	8.9	Quentin Tarantino	154

	id	title	release_year	rating	director	runtime
11	12	Forrest Gump	1994	8.8	Robert Zemeckis	142

運用 | 運算符聯集多個條件敘述



In [18]:

```
(movies["release_year"] == 1994) | (movies["rating"] >= 8.8)
```

Out[18]:

```
0       True
1       True
2       True
3       True
4       True
       ...
245    False
246    False
247    False
248    False
249    False
Length: 250, dtype: bool
```

In [19]:

```
movies[(movies["release_year"] == 1994) | (movies["rating"] >= 8.8)]
```

Out[19]:

	id	title	release_year	rating	director	runtime
0	1	The Shawshank Redemption	1994	9.3	Frank Darabont	142
1	2	The Godfather	1972	9.2	Francis Ford Coppola	175

	id	title	release_year	rating	director	runtime
2	3	The Godfather: Part II	1974	9.0	Francis Ford Coppola	202
3	4	The Dark Knight	2008	9.0	Christopher Nolan	152
4	5	12 Angry Men	1957	9.0	Sidney Lumet	96
5	6	Schindler's List	1993	8.9	Steven Spielberg	195
6	7	The Lord of the Rings: The Return of the King	2003	8.9	Peter Jackson	201
7	8	Pulp Fiction	1994	8.9	Quentin Tarantino	154
8	9	The Good, the Bad and the Ugly	1966	8.8	Sergio Leone	178
9	10	The Lord of the Rings: The Fellowship of the Ring	2001	8.8	Peter Jackson	178
10	11	Fight Club	1999	8.8	David Fincher	139
11	12	Forrest Gump	1994	8.8	Robert Zemeckis	142
12	13	Inception	2010	8.8	Christopher Nolan	148
30	31	Léon: The Professional	1994	8.5	Luc Besson	110
33	34	The Lion King	1994	8.5	Roger Allers	88

運用 `Series.isin()` 聯集單一變數的多個條件

In [20]:

```
movies[movies["director"].isin(["Peter Jackson", "Quentin Tarantino"])]
```

Out[20]:

	id	title	release_year	rating	director	runtime
6	7	The Lord of the Rings: The Return of the King	2003	8.9	Peter Jackson	201
7	8	Pulp Fiction	1994	8.9	Quentin Tarantino	154
9	10	The Lord of the Rings: The Fellowship of the Ring	2001	8.8	Peter Jackson	178
13	14	The Lord of the Rings: The Two Towers	2002	8.7	Peter Jackson	179
56	57	Django Unchained	2012	8.4	Quentin Tarantino	165
84	85	Inglourious Basterds	2009	8.3	Quentin Tarantino	153
88	89	Reservoir Dogs	1992	8.3	Quentin Tarantino	99
173	174	Kill Bill: Vol. 1	2003	8.1	Quentin Tarantino	111

In [21]:

```
movies[movies["release_year"].isin([1994, 2008])]
```

Out[21]:

	id	title	release_year	rating	director	runtime
0	1	The Shawshank Redemption	1994	9.3	Frank Darabont	142
3	4	The Dark Knight	2008	9.0	Christopher Nolan	152
7	8	Pulp Fiction	1994	8.9	Quentin Tarantino	154
11	12	Forrest Gump	1994	8.8	Robert Zemeckis	142
30	31	Léon: The Professional	1994	8.5	Luc Besson	110
33	34	The Lion King	1994	8.5	Roger Allers	88
60	61	WALL·E	2008	8.4	Andrew Stanton	98
181	182	Gran Torino	2008	8.1	Clint Eastwood	116

如何判斷條件敘述的交集或聯集

- 運用語言邏輯思考條件的結合為「和」還是「或」，「和」為交集、「或」為聯集。
- 運用資料列數思考條件的結合要「縮減」還是「擴增」，「縮減」為交集、「擴增」為聯集。

排序資料框

兩個排序方式

1. 遞增（又稱升冪）排序，預設的排序方式。
2. 遞減（又稱降冪）排序。

使用 `DataFrame` 的兩個方法排序

- `DataFrame.sort_index()` 依列標籤排序。
- `DataFrame.sort_values()` 依欄位排序。

`DataFrame.sort_index()` 依列標籤排序

預設 `ascending=True`

In [22]:

```
movies.sort_index()
```

Out[22]:

	id	title	release_year	rating	director	runtime
0	1	The Shawshank Redemption	1994	9.3	Frank Darabont	142
1	2	The Godfather	1972	9.2	Francis Ford Coppola	175
2	3	The Godfather: Part II	1974	9.0	Francis Ford Coppola	202
3	4	The Dark Knight	2008	9.0	Christopher Nolan	152
4	5	12 Angry Men	1957	9.0	Sidney Lumet	96
...	...	...	...	...	...	...
245	246	Neon Genesis Evangelion: The End of Evangelion	1997	8.1	Hideaki Anno	87
246	247	7 Kogustaki Mucize	2019	8.2	Mehmet Ada Öztekin	132
247	248	Tangerines	2013	8.2	Zaza Urushadze	87
248	249	Drishyam	2015	8.2	Nishikant Kamat	163
249	250	Swades	2004	8.2	Ashutosh Gowariker	189

250 rows × 6 columns

```
In [23]:  movies.sort_index(ascending=False)
```

```
Out[23]:
       id                                          title  release_year  rating             director  runtime
249   250                                         Swades          2004     8.2    Ashutosh Gowariker      189
248   249                                       Drishyam          2015     8.2      Nishikant Kamat      163
247   248                                     Tangerines          2013     8.2       Zaza Urushadze       87
246   247                               7 Kogustaki Mucize          2019     8.2    Mehmet Ada Öztekin      132
245   246   Neon Genesis Evangelion: The End of Evangelion          1997     8.1          Hideaki Anno       87
...   ...                                            ...           ...     ...                   ...      ...
4       5                                   12 Angry Men          1957     9.0          Sidney Lumet       96
3       4                                The Dark Knight          2008     9.0    Christopher Nolan      152
2       3                         The Godfather: Part II          1974     9.0  Francis Ford Coppola      202
1       2                                  The Godfather          1972     9.2  Francis Ford Coppola      175
0       1                       The Shawshank Redemption          1994     9.3        Frank Darabont      142

250 rows × 6 columns
```

`DataFrame.sort_values()` 依欄位排序

- 預設 `ascending=True`
- 數值由小到大、英文由 A 到 Z

In [24]:

```python
movies.sort_values("release_year")
```

Out[24]:

	id	title	release_year	rating	director	runtime
98	99	The Kid	1921	8.3	Charles Chaplin	68
194	195	Sherlock Jr.	1924	8.2	Buster Keaton	45
152	153	The Gold Rush	1925	8.2	Charles Chaplin	95
198	199	The General	1926	8.1	Clyde Bruckman	67
115	116	Metropolis	1927	8.3	Fritz Lang	153
...	...	...	...	...	...	...
246	247	7 Kogustaki Mucize	2019	8.2	Mehmet Ada Öztekin	132
164	165	Klaus	2019	8.2	Sergio Pablos	96
230	231	Soul	2020	8.1	Pete Docter	100
58	59	Hamilton	2020	8.5	Thomas Kail	160
166	167	Zack Snyder's Justice League	2021	8.3	Zack Snyder	242

250 rows × 6 columns

In [25]:

```
movies.sort_values("release_year", ascending=False)
```

Out[25]:

	id	title	release_year	rating	director	runtime
166	167	Zack Snyder's Justice League	2021	8.3	Zack Snyder	242
58	59	Hamilton	2020	8.5	Thomas Kail	160
230	231	Soul	2020	8.1	Pete Docter	100
75	76	Avengers: Endgame	2019	8.4	Anthony Russo	181
246	247	7 Kogustaki Mucize	2019	8.2	Mehmet Ada Öztekin	132
...	...	...	...	...	...	...
115	116	Metropolis	1927	8.3	Fritz Lang	153
198	199	The General	1926	8.1	Clyde Bruckman	67
152	153	The Gold Rush	1925	8.2	Charles Chaplin	95
194	195	Sherlock Jr.	1924	8.2	Buster Keaton	45
98	99	The Kid	1921	8.3	Charles Chaplin	68

250 rows × 6 columns

傳入 `list` 指定多個欄位與排序方式

In [26]:

```
movies.sort_values(["release_year", "title"], ascending=[False, True])
```

Out[26]:

	id	title	release_year	rating	director	runtime
166	167	Zack Snyder's Justice League	2021	8.3	Zack Snyder	242
58	59	Hamilton	2020	8.5	Thomas Kail	160
230	231	Soul	2020	8.1	Pete Docter	100
107	108	1917	2019	8.3	Sam Mendes	119
246	247	7 Kogustaki Mucize	2019	8.2	Mehmet Ada Öztekin	132
...	...	...	...	...	...	...
115	116	Metropolis	1927	8.3	Fritz Lang	153
198	199	The General	1926	8.1	Clyde Bruckman	67
152	153	The Gold Rush	1925	8.2	Charles Chaplin	95
194	195	Sherlock Jr.	1924	8.2	Buster Keaton	45
98	99	The Kid	1921	8.3	Charles Chaplin	68

250 rows × 6 columns

分組聚合欄位

使用 `Series` 的聚合方法取得欄位摘要

- `Series.count()` 不含未定義值的列數
- `Series.sum()` 加總
- `Series.max()` 最大值
- `Series.min()` 最小值
- `Series.mean()` 平均
- …等。

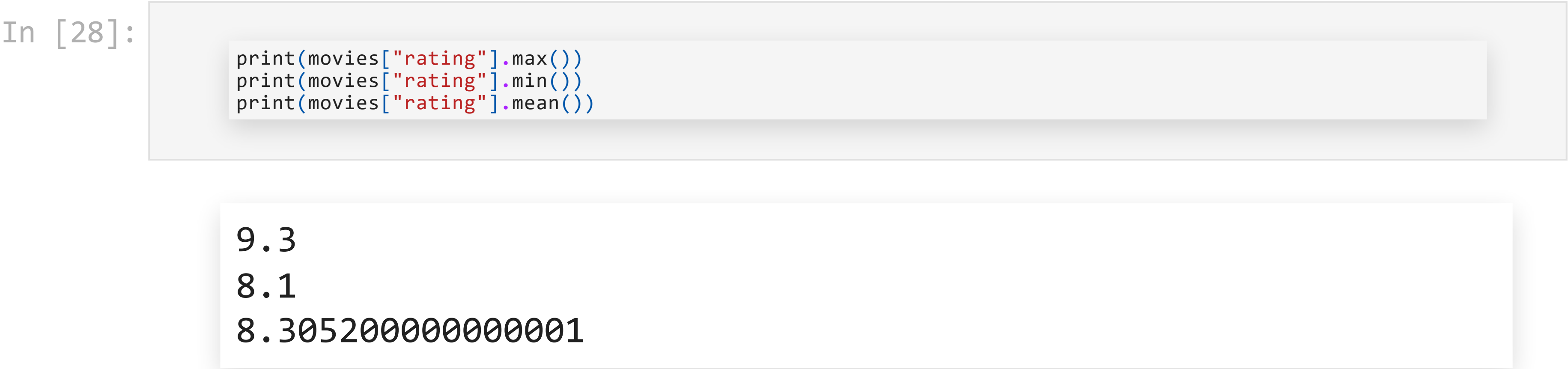
片長 `runtime` 的摘要

In [27]:

```
print(movies["runtime"].max())
print(movies["runtime"].min())
```

```
321
45
```

IMDb 評等 `rating` 的摘要



In [28]:

```
print(movies["rating"].max())
print(movies["rating"].min())
print(movies["rating"].mean())
```

```
9.3
8.1
8.305200000000001
```

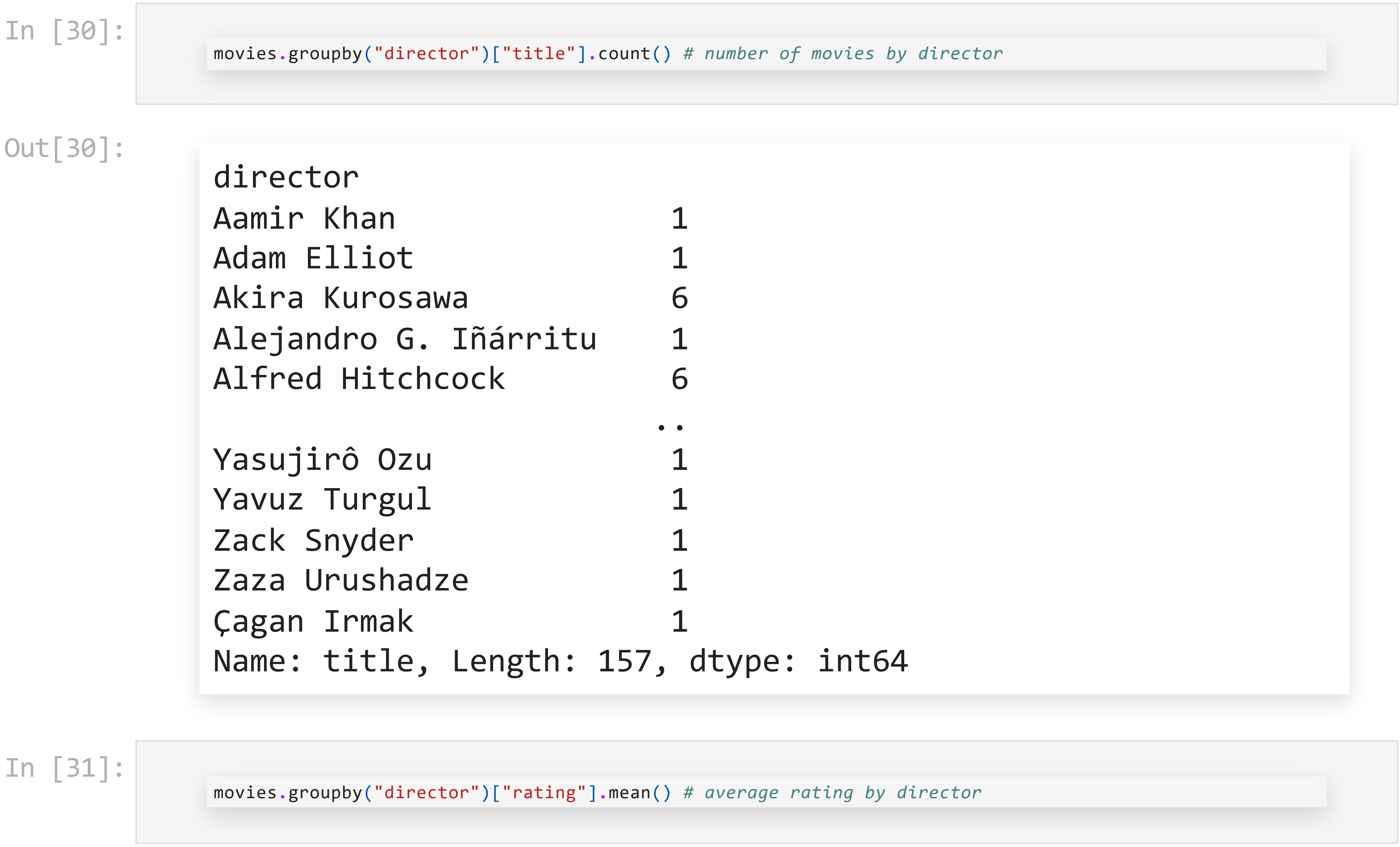
使用 `DataFrame.groupby()` 獲得排序後的獨一值 `DataFrameGroupBy` 類別

In [29]:

```
print(movies.groupby("director"))
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f9
49df20df0>
```

分組後接續選擇欄位以及聚合方法



```
In [30]: movies.groupby("director")["title"].count() # number of movies by director
```

```
Out[30]:
director
Aamir Khan               1
Adam Elliot              1
Akira Kurosawa           6
Alejandro G. Iñárritu    1
Alfred Hitchcock         6
                        ..
Yasujirô Ozu             1
Yavuz Turgul             1
Zack Snyder              1
Zaza Urushadze           1
Çagan Irmak              1
Name: title, Length: 157, dtype: int64
```

```
In [31]: movies.groupby("director")["rating"].mean() # average rating by director
```

Out[31]:

```
director
Aamir Khan                 8.400000
Adam Elliot                8.100000
Akira Kurosawa             8.316667
Alejandro G. Iñárritu      8.100000
Alfred Hitchcock           8.300000
                             ...
Yasujirô Ozu               8.200000
Yavuz Turgul               8.200000
Zack Snyder                8.300000
Zaza Urushadze             8.200000
Çagan Irmak                8.300000
Name: rating, Length: 157, dtype: float64
```

衍生計算欄位

善用三個技巧衍生計算欄位

1. 元素操作（Elementwise）運算。
2. 使用函數或 `Series` 的方法。
3. 使用 `Series.map()`

元素操作（Elementwise）運算

In [32]:

```
print(movies["runtime"] // 60) # hours
print(movies["runtime"] % 60)  # minutes
```

```
0      2
1      2
2      3
3      2
4      1
      ..
245    1
246    2
247    1
248    2
249    3
Name: runtime, Length: 250, dtype: int64
0      22
1      55
2      22
3      32
4      36
       ..
```

```
245    27
246    12
247    27
248    43
249     9
Name: runtime, Length: 250, dtype: int64
```

使用函數或 `Series` 的方法

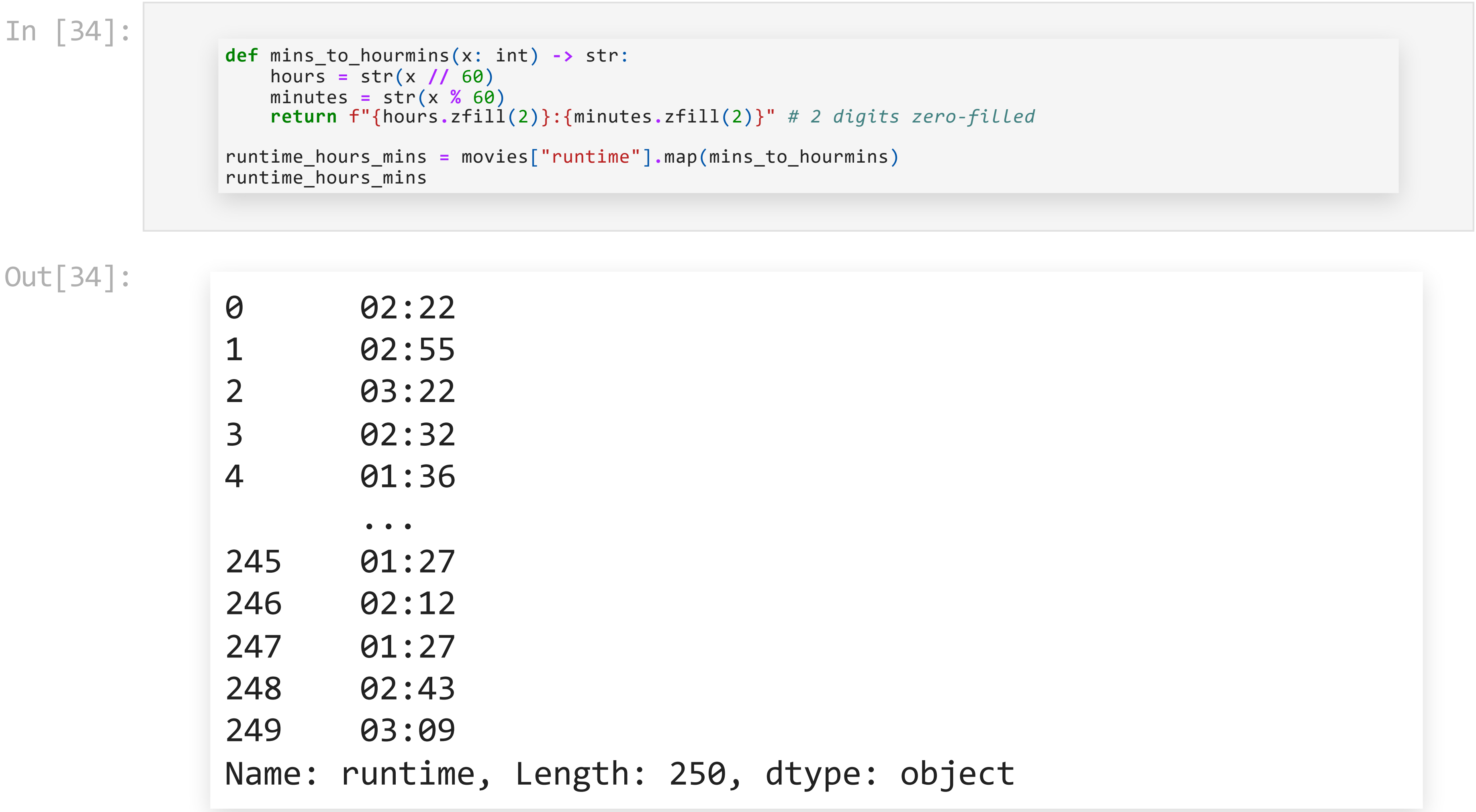
In [33]:

```
hours = (movies["runtime"] // 60).astype(str)
minutes = (movies["runtime"] % 60).astype(str)
hours.str.cat(minutes, sep=":") # hours:minutes
```

Out[33]:

```
0      2:22
1      2:55
2      3:22
3      2:32
4      1:36
       ... 
245    1:27
246    2:12
247    1:27
248    2:43
249     3:9
Name: runtime, Length: 250, dtype: object
```

使用 `Series.map()`



In [34]:

```python
def mins_to_hourmins(x: int) -> str:
    hours = str(x // 60)
    minutes = str(x % 60)
    return f"{hours.zfill(2)}:{minutes.zfill(2)}" # 2 digits zero-filled

runtime_hours_mins = movies["runtime"].map(mins_to_hourmins)
runtime_hours_mins
```

Out[34]:

```
0      02:22
1      02:55
2      03:22
3      02:32
4      01:36
       ...
245    01:27
246    02:12
247    01:27
248    02:43
249    03:09
Name: runtime, Length: 250, dtype: object
```

使用 `DataFrame.insert()` 新增變數

留意 `DataFrame.insert()` 更新的機制是更新物件本身並回傳 `None`

In [35]:

```
print(movies.columns)
n = movies.shape[1]
movies.insert(n, "runtime_hours_mins", runtime_hours_mins)
print(movies.columns)
movies.head()
```

```
Index(['id', 'title', 'release_year', 'rating', 'director',
'runtime'], dtype='object')
Index(['id', 'title', 'release_year', 'rating', 'director',
'runtime',
       'runtime_hours_mins'],
      dtype='object')
```

Out[35]:

	id	title	release_year	rating	director	runtime	runtime_hours_mins
0	1	The Shawshank Redemption	1994	9.3	Frank Darabont	142	02:22
1	2	The Godfather	1972	9.2	Francis Ford Coppola	175	02:55
2	3	The Godfather: Part II	1974	9.0	Francis Ford Coppola	202	03:22
3	4	The Dark Knight	2008	9.0	Christopher Nolan	152	02:32
4	5	12 Angry Men	1957	9.0	Sidney Lumet	96	01:36

調整列標籤與欄標籤

如何調整列標籤與欄標籤

- 使用 `DataFrame.set_index()` 指定欄位取代目前的列標籤。
- 使用 `DataFrame.reset_index()` 重設列標籤。
- 指定 `DataFrame.columns` 調整欄標籤。

使用 `DataFrame.set_index()` 指定欄位取代目前的列標籤

In [36]:

```
movies.set_index("title")
```

Out[36]:

	id	release_year	rating	director	runtime	runtime_hours_mins
title						
The Shawshank Redemption	1	1994	9.3	Frank Darabont	142	02:22
The Godfather	2	1972	9.2	Francis Ford Coppola	175	02:55
The Godfather: Part II	3	1974	9.0	Francis Ford Coppola	202	03:22
The Dark Knight	4	2008	9.0	Christopher Nolan	152	02:32
12 Angry Men	5	1957	9.0	Sidney Lumet	96	01:36
...	...	...	...	...	...	...
Neon Genesis Evangelion: The End of Evangelion	246	1997	8.1	Hideaki Anno	87	01:27
7 Kogustaki Mucize	247	2019	8.2	Mehmet Ada Öztekin	132	02:12
Tangerines	248	2013	8.2	Zaza Urushadze	87	01:27
Drishyam	249	2015	8.2	Nishikant Kamat	163	02:43
Swades	250	2004	8.2	Ashutosh Gowariker	189	03:09

250 rows × 6 columns

使用 `DataFrame.reset_index()` 重設列標籤

預設以 `RangeIndex` 重設後將原本的列標籤變為第零個欄位。

In [37]:

```
movies.set_index("title").reset_index()
```

Out[37]:

	title	id	release_year	rating	director	runtime	runtime_hours_mins
0	The Shawshank Redemption	1	1994	9.3	Frank Darabont	142	02:22
1	The Godfather	2	1972	9.2	Francis Ford Coppola	175	02:55
2	The Godfather: Part II	3	1974	9.0	Francis Ford Coppola	202	03:22
3	The Dark Knight	4	2008	9.0	Christopher Nolan	152	02:32
4	12 Angry Men	5	1957	9.0	Sidney Lumet	96	01:36
...	...	...	...	...	...	...	...
245	Neon Genesis Evangelion: The End of Evangelion	246	1997	8.1	Hideaki Anno	87	01:27
246	7 Kogustaki Mucize	247	2019	8.2	Mehmet Ada Öztekin	132	02:12
247	Tangerines	248	2013	8.2	Zaza Urushadze	87	01:27
248	Drishyam	249	2015	8.2	Nishikant Kamat	163	02:43
249	Swades	250	2004	8.2	Ashutosh Gowariker	189	03:09

250 rows × 7 columns

使用 `DataFrame.reset_index()` 重設列標籤（續）

設定參數 `drop=True` 以 `RangeIndex` 重設後捨棄原本的列標籤。

In [38]:

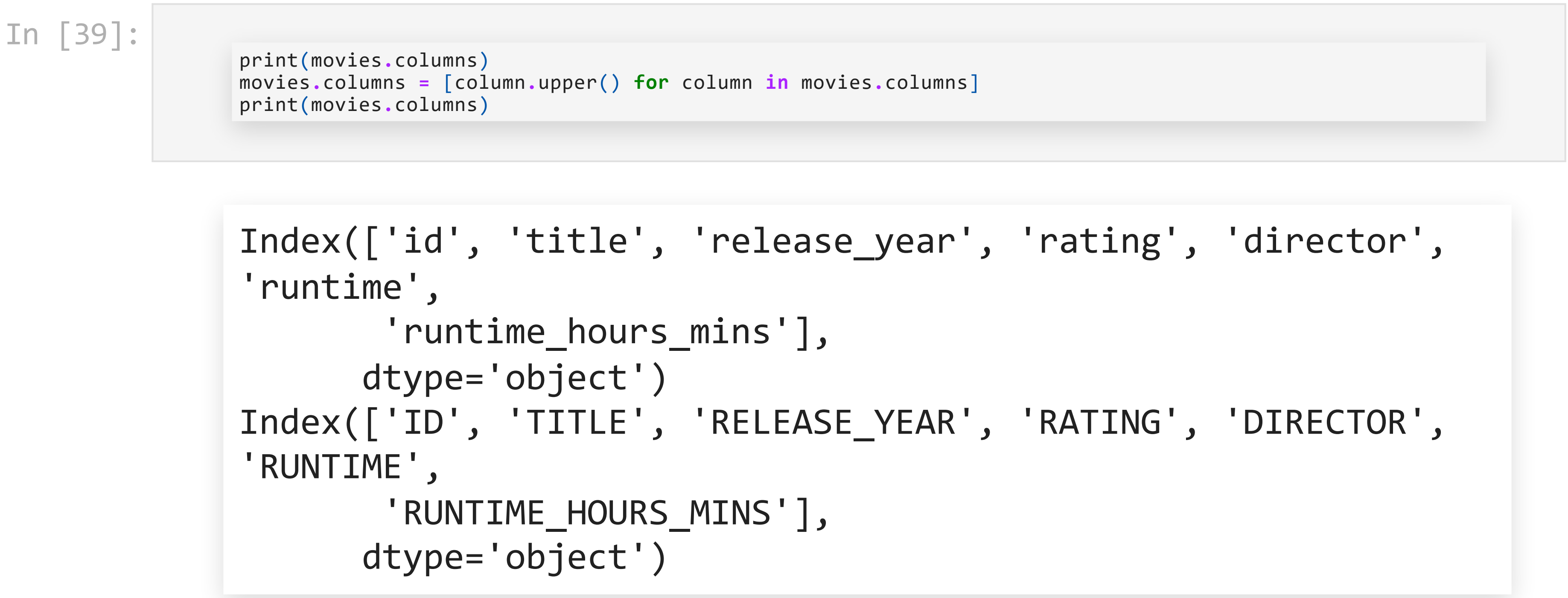
```
movies.set_index("title").reset_index(drop=True)
```

Out[38]:

	id	release_year	rating	director	runtime	runtime_hours_mins
0	1	1994	9.3	Frank Darabont	142	02:22
1	2	1972	9.2	Francis Ford Coppola	175	02:55
2	3	1974	9.0	Francis Ford Coppola	202	03:22
3	4	2008	9.0	Christopher Nolan	152	02:32
4	5	1957	9.0	Sidney Lumet	96	01:36
...	...	...	...	...	...	...
245	246	1997	8.1	Hideaki Anno	87	01:27
246	247	2019	8.2	Mehmet Ada Öztekin	132	02:12
247	248	2013	8.2	Zaza Urushadze	87	01:27
248	249	2015	8.2	Nishikant Kamat	163	02:43
249	250	2004	8.2	Ashutosh Gowariker	189	03:09

250 rows × 6 columns

指定 `DataFrame.columns` 調整欄標籤



In [39]:

```
print(movies.columns)
movies.columns = [column.upper() for column in movies.columns]
print(movies.columns)
```

```
Index(['id', 'title', 'release_year', 'rating', 'director',
'runtime',
       'runtime_hours_mins'],
      dtype='object')
Index(['ID', 'TITLE', 'RELEASE_YEAR', 'RATING', 'DIRECTOR',
'RUNTIME',
       'RUNTIME_HOURS_MINS'],
      dtype='object')
```

處理多個變數的分組聚合結果

如果在 `DataFrame.groupby()` 輸入了多個變數作為分組依據 `[column_0, column_1, ...]` 輸出的 `Series` 會有特殊的 `MultiIndex` 類別。

In [40]:

```
movies = pd.read_csv("/home/jovyan/data/internet-movie-database/movies.csv")
movies_by_year_director = movies.groupby(["release_year", "director"])["title"].count()
movies_by_year_director.index
```

Out[40]:

```
MultiIndex([(1921,      'Charles Chaplin'),
            (1924,         'Buster Keaton'),
            (1925,      'Charles Chaplin'),
            (1926,        'Clyde Bruckman'),
            (1927,            'Fritz Lang'),
            (1928, 'Carl Theodor Dreyer'),
            (1931,      'Charles Chaplin'),
            (1931,            'Fritz Lang'),
            (1934,           'Frank Capra'),
            (1936,      'Charles Chaplin'),
            ...
            (2019,          'Bong Joon Ho'),
            (2019,        'Céline Sciamma'),
            (2019,         'James Mangold'),
            (2019,   'Mehmet Ada Öztekin'),
            (2019,          'Sam Mendes'),
            (2019,       'Sergio Pablos'),
            (2019,       'Todd Phillips'),
            (2020,         'Pete Docter'),
            (2020,         'Thomas Kail'),
            (2021,         'Zack Snyder')],
           names=['release_year', 'director'], length=247)
```

如何操作 `MultiIndex` 類別

- 以操作 `tuple` 的方式面對 `MultiIndex` 類別。
- 使用 `DataFrame.reset_index()` 重設列標籤。

以操作 `tuple` 的方式面對 `MultiIndex` 類別

In [41]:

```
movies_by_year_director[(2008, )]
```

Out[41]:

```
director
Andrew Stanton       1
Christopher Nolan    1
Clint Eastwood       1
Name: title, dtype: int64
```

In [42]:

```
movies_by_year_director[(2008, "Christopher Nolan")]
```

Out[42]:

```
1
```

使用 `DataFrame.reset_index()` 重設列標籤

In [43]:

```
movies_by_year_director.reset_index()
```

Out[43]:

	release_year	director	title
0	1921	Charles Chaplin	1
1	1924	Buster Keaton	1
2	1925	Charles Chaplin	1
3	1926	Clyde Bruckman	1
4	1927	Fritz Lang	1
...	...	...	...
242	2019	Sergio Pablos	1
243	2019	Todd Phillips	1
244	2020	Pete Docter	1
245	2020	Thomas Kail	1
246	2021	Zack Snyder	1

247 rows × 3 columns

重點統整

- `DataFrame` 是有兩個維度的資料結構。
    - 第一個維度稱為觀測值（Observations），有時亦稱為列（Rows）
    - 第二個維度稱為變數（Variables），有時亦稱為欄（Columns）
    - 我們習慣以 `(m, n)` 或者 `m x n` 來描述一個具有 `m` 列觀測值、`n` 欄變數的 `DataFrame`
- 區分 `DataFrame` 的兩種索引語法。
    - `DataFrame.loc[row-label, column-label]` 以列、欄標籤為準。
    - `DataFrame.iloc[row-integer-location, column-integer-location]` 以列、欄整數位置為準。

重點統整（續）

- 善用三個技巧衍生計算欄位
    - 元素操作（Elementwise）運算。
    - 使用函數或 `Series` 的方法。
    - 使用 `Series.map()`