Python 的 50+ 練習

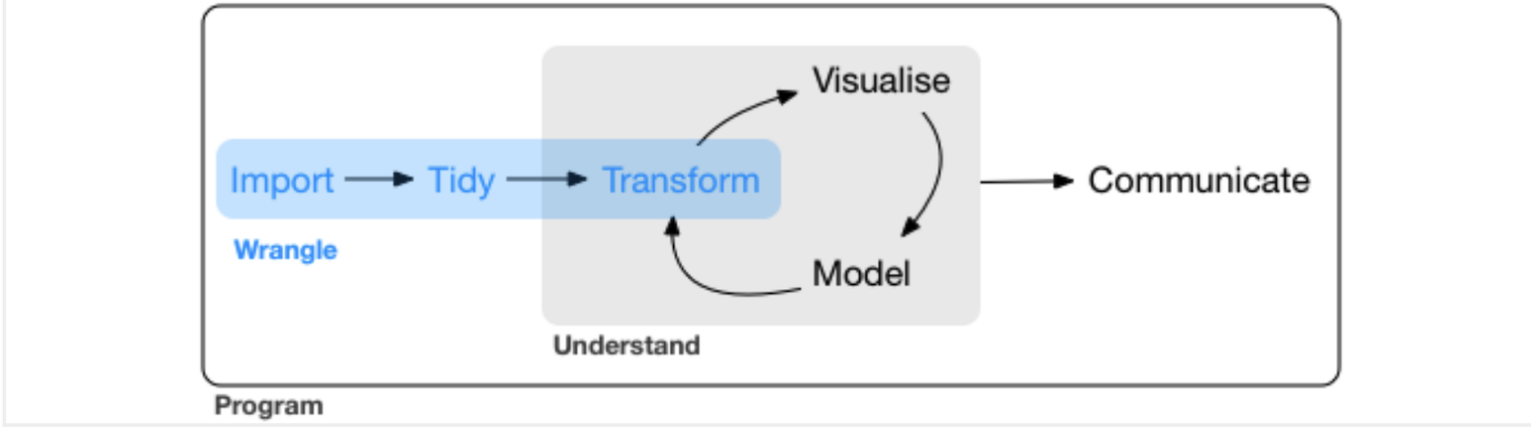
> *文字資料操作*

數據交點 | 郭耀仁 yaojenkuo@datainpoint.com

這個章節會登場的模組

- `re` 模組。
- `pandas` 模組。

（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。

常用的文字方法

（複習）使用成對的單引號 `'`、雙引號 `"` 或三個雙引號 `"""` 形成 `str`

In [1]:

```
str_with_single_quotes = 'Hello, world!'
str_with_double_quotes = "Hello, world!"
str_with_triple_double_quotes = """Hello, world!"""
print(type(str_with_single_quotes))
print(type(str_with_double_quotes))
print(type(str_with_triple_double_quotes))
```

```
<class 'str'>
<class 'str'>
<class 'str'>
```

（複習）`str` 中可能會包含單引號或雙引號，有兩種解決方式

1. 以反斜線 `\`（又稱跳脫符號）作為標註。
2. 以不同樣式的引號來形成 `str` 藉此區隔。

In [2]:

```
mcd = 'I\'m lovin\' it!'   # escape with \
mcd = "I'm lovin' it!"     # use different quotation marks
mcd = """I'm lovin' it!""" # use different quotation marks
```

`str` 與其他資料類別不同的地方

- `str` 是可迭代類別（Iterable）
- `str` 適用 indexing/slicing

In [3]:

```
print(iter(mcd))
print(mcd[4])
print(mcd[4:])
```

```
<str_iterator object at 0x7fbdc4f0f520>
l
lovin' it!
```

使用內建函數 `dir()` 檢視 `str` 的方法

In [4]:

```
print(dir(mcd))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

使用內建函數 `len()` 檢視 `str` 的長度（字元數）

In [5]:
```
len(mcd)
```

Out[5]:
```
14
```

文字操作目的

- 大小寫調整。
- 特徵判斷。
- 擷取片段。
- 移除多餘的空白字元。
- 取代特徵。
- 連接與分割。

大小寫調整

- `str.upper()` 變成大寫。
- `str.lower()` 變成小寫。
- `str.title()` 單字字首大寫。
- `str.capitalize()` 句首大寫。
- `str.swapcase()` 大小寫互換。

In [6]:

```
movie_title = "The Lord of the Rings"
print(movie_title.upper())
print(movie_title.lower())
print(movie_title.title())
print(movie_title.capitalize())
print(movie_title.swapcase())
```

```
THE LORD OF THE RINGS
the lord of the rings
The Lord Of The Rings
The lord of the rings
tHE lORD OF THE rINGS
```

特徵判斷

- 運用 `in` 關係運算符。
- `str.find()` 找出特徵出現的第一個索引值。
- `str.count()` 找出特徵出現的次數。
- `str.startswith()` 判斷第 0 個字元是否為特徵。
- `str.endswith()` 判斷第 -1 個字元是否為特徵。

In [7]:

```
movie_title = "The Lord of the Rings"
print("Ring" in movie_title)
print(movie_title.find("o"))  # The L"o"rd of the Rings
print(movie_title.count("o")) # The L"o"rd "o"f the Rings
print(movie_title.startswith("T"))
print(movie_title.endswith("s"))
```

```
True
5
2
True
True
```

擷取片段

運用 indexing/slicing

In [8]:

```
movie_title = "The Lord of the Rings"
R_index = movie_title.find("R")
print(movie_title[4:8])
print(movie_title[R_index:])
```

```
Lord
Rings
```

移除多餘的空白字元

- 空白字元包含像是空白（Space 鍵）、Tab 鍵（`"\t"`）以及換行（`"\n"`）等。
- `str.lstrip()` 移除字首左邊多餘的空白字元。
- `str.rstrip()` 移除字尾右邊多餘的空白字元。
- `str.strip()` 移除字首字尾多餘的空白字元。

In [9]:
```
movie_title = " \t\nThe Lord of the Rings \t\n"
movie_title.lstrip()
```

Out[9]:
```
'The Lord of the Rings \t\n'
```

In [10]:
```
movie_title.rstrip()
```

Out[10]:
```
' \t\nThe Lord of the Rings'
```

In [11]:
```
movie_title.strip()
```

Out[11]:

```
'The Lord of the Rings'
```

取代特徵

`str.replace(old, new)` 取代特徵 `old` 為 `new`

In [12]:

```
movie_title = "The-Lord-of-the-Rings"
movie_title.replace("-", " ") # replace "-" with space
```

Out[12]:

```
'The Lord of the Rings'
```

連接與分割

- 運用 `+` 運算符連接文字。
- 運用 f-string 或 `str.format()` 連接文字。
- 使用 `"sep".join(list)` 連接 `list` 元素、以 `"sep"` 作分隔符號為文字。
- 使用 `str.split()` 分割文字。

連接文字

In [13]:

```
trilogy = "The Lord of the Rings"
title = "The Return of the Kings"
print(trilogy + ": " + title)
print(f"{trilogy}: {title}")
print("{}: {}".format(trilogy, title))
```

```
The Lord of the Rings: The Return of the Kings
The Lord of the Rings: The Return of the Kings
The Lord of the Rings: The Return of the Kings
```

使用 `"sep".join(list)` 連接 `list` 元素、以 `"sep"` 作分隔符號為文字

In [14]:

```
trilogy = ["The", "Lord", "of", "the", "Rings"]
title = ["The", "Return", "of", "the", "Kings"]
print(" ".join(trilogy))
print(" ".join(title))
print(",".join(trilogy))
print(",".join(title))
```

```
The Lord of the Rings
The Return of the Kings
The,Lord,of,the,Rings
The,Return,of,the,Kings
```

`str.split()` 預設分割特徵為空白字元

- 空白字元包含像是空白（Space 鍵）、Tab 鍵（`"\t"`）以及換行（`"\n"`）等。
- 可以指定分割特徵。

In [15]:

```
movie_title = "The Lord of\tthe\nRings"
print(movie_title.split())
movie_title = "The-Lord-of-the-Rings"
print(movie_title.split("-"))
```

```
['The', 'Lord', 'of', 'the', 'Rings']
['The', 'Lord', 'of', 'the', 'Rings']
```

標準模組 `re`

什麼是標準模組 `re`

> `re` *模組為 Python 提供與 Perl 語言相似的正規表達式（Regular expression）操作。*

來源：https://docs.python.org/3/library/re.html

什麼是正規表達式

常見文字操作目的包含特徵判斷、取代特徵與分割等，都要仰賴特徵（Pattern）的描述。正規表達式（Regular expression）是一種用來描述文字特徵的語法，是更為強大、能夠表現文字泛用特徵的技巧，許多程式設計語言都支援整合正規表達式進行文字操作，在 Python 中可以透過 `re` 模組來應用。

透過正規表達式能夠更廣泛地表達指定特徵

- 文字的組成是數字，像是手機號碼、郵遞區號等。
- 文字的組成有特殊格式，像是電子郵件信箱、姓名等。
- 文字的組成是特定語言，像是英文、中文等。

運用正規表達式判斷整數特徵

`re.search()`

`"[0-9]"` 正規表達式可以用來廣泛表示 0 到 9 的整數特徵。

In [16]:

```
import re

integer_pattern = "[0-9]"
print(bool(re.search(integer_pattern, "5566 is my favorite boy band.")))
print(bool(re.search(integer_pattern, "F4 is my favorite boy band.")))
print(bool(re.search(integer_pattern, "NSYNC is my favorite boy group.")))
print(bool(re.search(integer_pattern, "Backstreet Boys is my favorite boy group.")))
```

```
True
True
False
False
```

運用正規表達式判斷英文特徵

`re.search()`

`"[A-z]"` 正規表達式可以用來廣泛表示英文特徵。

In [17]:

```
english_pattern = "[A-z]"
print(bool(re.search(english_pattern, "5566 是我最喜歡的男孩團體。")))
print(bool(re.search(english_pattern, "F4 是我最喜歡的男孩團體。")))
print(bool(re.search(english_pattern, "NSYNC 是我最喜歡的男孩團體。")))
print(bool(re.search(english_pattern, "Backstreet Boys 是我最喜歡的男孩團體。")))
```

```
False
True
True
True
```

運用正規表達式取代特徵 `re.sub()`

將文字中的整數取代為空字串。

In [18]:

```
numeric_pattern = "[0-9]"
print(re.sub(numeric_pattern, "", "5566 is my favorite boy band."))
print(re.sub(numeric_pattern, "", "F4 is my favorite boy band."))
print(re.sub(numeric_pattern, "", "NSYNC is my favorite boy group."))
print(re.sub(numeric_pattern, "", "Backstreet Boys is my favorite boy group."))
```

```
 is my favorite boy band.
F is my favorite boy band.
NSYNC is my favorite boy group.
Backstreet Boys is my favorite boy group.
```

運用正規表達式取代特徵 `re.sub()`

將文字中的英文取代為空字串。

In [19]:

```
english_pattern = "[A-z]"
print(re.sub(english_pattern, "", "5566 是我最喜歡的男孩團體。"))
print(re.sub(english_pattern, "", "F4 是我最喜歡的男孩團體。"))
print(re.sub(english_pattern, "", "NSYNC 是我最喜歡的男孩團體。"))
print(re.sub(english_pattern, "", "Backstreet Boys 是我最喜歡的男孩團體。"))
```

```
5566 是我最喜歡的男孩團體。
4 是我最喜歡的男孩團體。
 是我最喜歡的男孩團體。
 是我最喜歡的男孩團體。
```

運用正規表達式分割文字 `re.split()`

`"[-_ ;]"` 正規表達式可以用來表示有減號、底線、空白或分號的特徵。

In [20]:

```
split_pattern = "[-_ ;]"
print(re.split(split_pattern, "Lord-of-the-Rings"))
print(re.split(split_pattern, "The_Dark_Knight"))
print(re.split(split_pattern, "The God Father"))
print(re.split(split_pattern, "Forrest;Gump"))
```

```
['Lord', 'of', 'the', 'Rings']
['The', 'Dark', 'Knight']
['The', 'God', 'Father']
['Forrest', 'Gump']
```

特徵若是正規表達式中的特殊文字，使用 \ 跳脫

- 例如 | 是正規表達式的特殊文字，表達特徵時就改用 pattern="\|"
- 例如 * 是正規表達式的特殊文字，表達特徵時就改用 pattern="*"
- 例如 () 是正規表達式的特殊文字，表達特徵時就改用 pattern="[\(\)]"

In [21]:

```
print(re.split("\|", "The|Lord|of|the|Rings"))
print(re.split("\*", "The*Lord*of*the*Rings"))
print(re.sub("[\(\)]", "", "(The Lord of the Rings)"))
```

```
['The', 'Lord', 'of', 'the', 'Rings']
['The', 'Lord', 'of', 'the', 'Rings']
The Lord of the Rings
```

對程式設計與資料科學初學者而言，正規表達式不是在短時間就能活用的技巧

- re — Regular expression operations
- Regular Expression HOWTO

常用的 `Series` 文字方法

文字類別的 `Series`

以 `Series.dtype` 屬性檢視，文字類別顯示為 `object`

In [22]:

```
import pandas as pd

boy_bands = pd.Series(["5566", "F4", "NSYNC", "Backstreet Boys"])
print(boy_bands.dtype)
boy_bands
```

```
object
```

Out[22]:

```
0                5566
1                  F4
2               NSYNC
3    Backstreet Boys
dtype: object
```

文字類別的 `Series` 常用聚合方法

- `Series.unique()` 剔除重複值。
- `Series.nunique()` 獨一值個數。
- `Series.value_counts()` 依獨一值分組後計數並且依計數遞減排序。

In [23]:

```
movies_csv = pd.read_csv("/home/jovyan/data/internet-movie-database/movies.csv")
movies_csv["director"].unique()
```

Out[23]:

```
array(['Frank Darabont', 'Francis Ford Coppola', 'Christopher
Nolan',
       'Sidney Lumet', 'Steven Spielberg', 'Peter Jackson',
       'Quentin Tarantino', 'Sergio Leone', 'David Fincher',
       'Robert Zemeckis', 'Irvin Kershner', 'Lana Wachowski',
       'Martin Scorsese', 'Milos Forman', 'Akira Kurosawa',
       'Roberto Benigni', 'Fernando Meirelles', 'Jonathan Dem
me',
       'Frank Capra', 'George Lucas', 'Hayao Miyazaki', 'Bong
Joon Ho',
       'Luc Besson', 'Masaki Kobayashi', 'Bryan Singer', 'Rog
er Allers',
       'Roman Polanski', 'James Cameron', 'Tony Kaye', 'Charl
es Chaplin',
```

```
       'Ridley Scott', 'Alfred Hitchcock', 'Olivier Nakache',
       'Damien Chazelle', 'Isao Takahata', 'Michael Curtiz',
       'Giuseppe Tornatore', 'Florian Henckel von Donnersmarc
k',
       'Thomas Kail', 'Stanley Kubrick', 'Andrew Stanton',
       'Todd Phillips', 'Anthony Russo', 'Billy Wilder', 'Cha
n-wook Park',
       'Bob Persichetti', 'Makoto Shinkai', 'Lee Unkrich',
       'Nadine Labaki', 'Sam Mendes', 'Mel Gibson', 'Wolfgang
Petersen',
       'John Lasseter', 'Rajkumar Hirani', 'Gus Van Sant',
       'Richard Marquand', 'Aamir Khan', 'Darren Aronofsky',
       'Thomas Vinterberg', 'Fritz Lang', 'Michel Gondry', 'O
rson Welles',
       'Nitesh Tiwari', 'Stanley Donen', 'Vittorio De Sica',
       'Elem Klimov', 'Guy Ritchie', 'Brian De Palma', 'Denis
Villeneuve',
       'Asghar Farhadi', 'George Roy Hill', 'David Lean',
       'Jean-Pierre Jeunet', 'Robert Mulligan', 'Pete Docte
r',
       'Michael Mann', 'Curtis Hanson', 'Peter Farrelly',
       'John McTiernan', 'Terry Gilliam', 'Oliver Hirschbiege
l',
       'Majid Majidi', 'Clint Eastwood', 'Joseph L. Mankiewic
z',
       'Ron Howard', 'John Sturges', 'Guillermo del Toro',
       'Paul Thomas Anderson', 'Juan José Campanella', 'Stanl
```

```
ey Kramer',
       'John Huston', 'Martin McDonagh', 'Çagan Irmak', 'Etha
n Coen',
       'James McTeigue', 'John Carpenter', 'David Lynch',
       'Ingmar Bergman', 'M. Night Shyamalan', "Gavin O'Conno
r",
       'Sergio Pablos', 'Danny Boyle', 'Zack Snyder', 'Peter
Weir',
       'Victor Fleming', 'Andrei Tarkovsky', 'Joel Coen',
       'Damián Szifron', 'Lenny Abrahamson', 'Yasujirô Ozu',
'Carol Reed',
       'Elia Kazan', 'Michael Cimino', 'Jim Sheridan', 'Adam
Elliot',
       'Richard Linklater', 'Wes Anderson', 'Sriram Raghava
n',
       'Buster Keaton', 'Ernst Lubitsch', 'Clyde Bruckman',
       'Dean DeBlois', 'James Mangold', 'Steve McQueen', 'Yav
uz Turgul',
       'George Miller', 'Hrishikesh Mukherjee', 'David Yate
s',
       'Rob Reiner', 'William Wyler', 'Lasse Hallström',
       'Stuart Rosenberg', 'Oliver Stone', 'Henri-Georges Clo
uzot',
       'Sean Penn', 'Mathieu Kassovitz', 'Terry Jones',
       'François Truffaut', 'Carl Theodor Dreyer', 'Tom McCar
thy',
       'Ram Kumar', 'Terry George', 'Anurag Kashyap',
```

```
       'Alejandro G. Iñárritu', 'John G. Avildsen', 'Emir Kus
turica',
       'Kar-Wai Wong', 'Jules Dassin', 'Rakeysh Omprakash Meh
ra',
       'Wim Wenders', 'Jeethu Joseph', 'Céline Sciamma', 'Hid
eaki Anno',
       'Mehmet Ada Öztekin', 'Zaza Urushadze', 'Nishikant Kam
at',
       'Ashutosh Gowariker'], dtype=object)
```

In [24]:

```
print(movies_csv["director"].nunique())
print(movies_csv["director"].value_counts())
```

```
157
Martin Scorsese       7
Christopher Nolan     7
Stanley Kubrick       7
Akira Kurosawa        6
Steven Spielberg      6
                     ..
Vittorio De Sica      1
Stanley Donen         1
Nitesh Tiwari         1
Orson Welles          1
Ashutosh Gowariker    1
Name: director, Length: 157, dtype: int64
```

`str.method()` 與 `Series.str.method()` 不同的地方

- `str.method()` 操作一個文字；`Series.str.method()` 操作一欄文字。
- `str.method()` 不支援正規表達式，必須透過 `re` 模組；`Series.str.method()` 可以透過 `regex=True` 支援正規表達式。

常用的 `Series.str.method()`

- `Series.str.contains()` 判斷文字是否包含指定特徵。
- `Series.str.replace()` 取代文字中的指定特徵。
- `Series.str.split()` 依據文字中的指定特徵切割。
- `Series.str.cat()` 以元素操作（Elementwise）連接兩個文字 `Series`

以判斷整數特徵為例

In [25]:

```
integer_pattern = "[0-9]"
boy_bands = ["5566", "F4", "NSYNC", "Backstreet Boys"]
contains_numeric_pattern = [bool(re.search(integer_pattern, boy_band)) for boy_band in boy_bands]
contains_numeric_pattern
```

Out[25]:

```
[True, True, False, False]
```

使用 `Series.str.contains()` 判斷文字是否包含指定特徵

In [26]:

```
boy_bands = pd.Series(["5566", "F4", "NSYNC", "Backstreet Boys"])
boy_bands.str.contains(integer_pattern, regex=True)
```

Out[26]:

```
0     True
1     True
2    False
3    False
dtype: bool
```

使用 `Series.str.replace()` 取代文字中的指定特徵

In [27]:

```
boy_bands = pd.Series(["5566", "F4", "NSYNC", "Backstreet Boys"])
boy_bands.str.replace(integer_pattern, "", regex=True)
```

Out[27]:

```
0
1                   F
2               NSYNC
3    Backstreet Boys
dtype: object
```

使用 `Series.str.split()` 依據文字中的指定特徵分割

分割後的結果為包含多個 `list` 的 `Series`

In [28]:

```
split_pattern = "[-_ ;]"
movie_titles = pd.Series(["The-Lord-of-the-Rings", "The_Dark_Knight", "The God Father", "Forrest;Gump"])
movie_titles.str.split(split_pattern)
```

Out[28]:

```
0    [The, Lord, of, the, Rings]
1            [The, Dark, Knight]
2             [The, God, Father]
3                [Forrest, Gump]
dtype: object
```

In [29]:

```
movie_titles_sep_with_space = [" ".join(title) for title in movie_titles.str.split(split_pattern)]
movie_titles_sep_with_space
```

Out[29]:

```
['The Lord of the Rings', 'The Dark Knight', 'The God Fathe
r', 'Forrest Gump']
```

使用 `Series.str.cat()` 以元素操作（Elementwise）連接兩個文字 `Series`

In [30]:

```
trilogy = pd.Series(["The Lord of the Rings", "The Lord of the Rings", "The Lord of the Rings"])
titles = pd.Series(["The Fellowship of the Ring", "The Two Towers", "The Return of the King"])
trilogy.str.cat(titles, sep=": ")
```

Out[30]:

```
0    The Lord of the Rings: The Fellowship of the Ring
1                The Lord of the Rings: The Two Towers
2        The Lord of the Rings: The Return of the King
dtype: object
```

重點統整

- `str` 與其他資料類別不同的地方
    - `str` 是可迭代類別（Iterable）
    - `str` 適用 indexing/slicing
- 文字操作目的
    - 大小寫調整。
    - 特徵判斷。
    - 擷取片段。
    - 移除多餘的空白字元。
    - 取代特徵。
    - 連接與分割。

重點統整（續）

- 正規表達式（Regular expression）是一種用來描述文字特徵的語法，是更為強大、能夠表現文字泛用特徵的技巧，許多程式設計語言都支援整合正規表達式進行文字操作，在 Python 中可以透過 `re` 模組來應用。
- `str.method()` 與 `Series.str.method()` 不同的地方
    - `str.method()` 操作一個文字；`Series.str.method()` 操作一欄文字。
    - `str.method()` 不支援正規表達式，必須透過 `re` 模組；`Series.str.method()` 可以透過指定參數 `regex=True` 支援正規表達式。