Python 的 50+ 練習

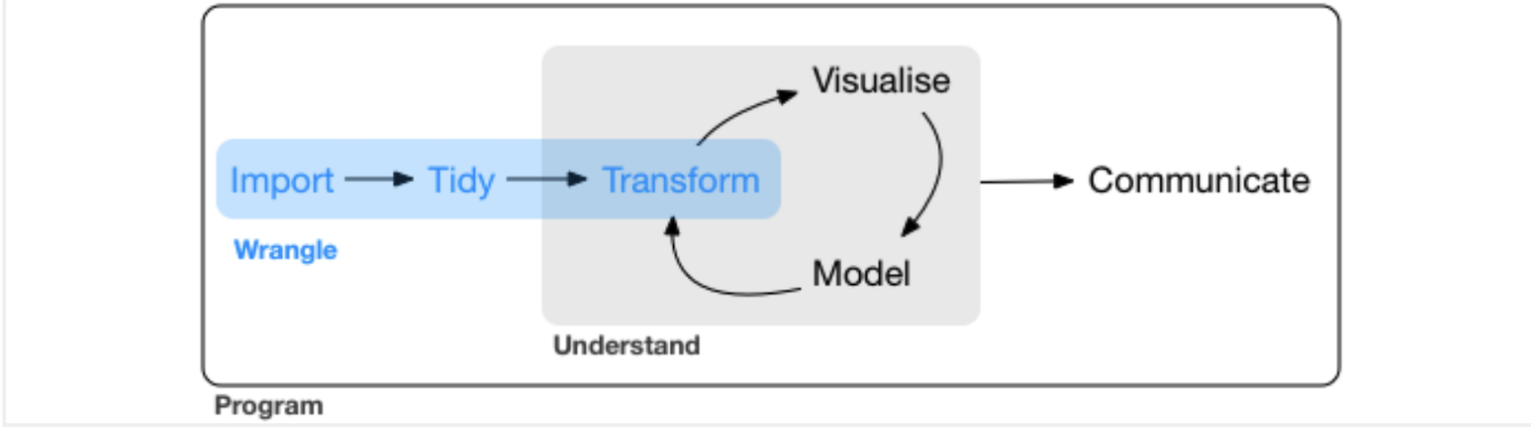
> *日期時間資料操作*

數據交點 | 郭耀仁 yaojenkuo@datainpoint.com

這個章節會登場的模組

- `datetime` 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。

關於 Unix 時間

什麼是 Unix 時間

> *Unix 時間，也稱為 Posix 時間是類 UNIX 作業系統（泛指 Linux 或 macOS）使用的時間表示方式：從 1970-01-01 00:00:00 UTC 起一直到此時此刻的總秒數。*

來源：https://en.wikipedia.org/wiki/Unix_time

不同國家、地區有各自習慣的日期、時間以及日期時間格式

- 常見的日期格式有 1970-01-01、1970/1/1、Jan, 1, 1970 等格式。
- 常見的時間格式有 00:00:00、12:00:00 a.m. 等格式。
- 常見的日期時間格式有 1970-01-01 00:00:00 等格式。

ISO 8601

> *ISO 8601 是國際標準化組織的日期、時間和日期時間的表示法。*

來源：https://en.wikipedia.org/wiki/ISO_8601

ISO 8601 的表示法

- 日期：YYYY-MM-DD，例如 1970-01-01、2022-01-01
- 時間：hh:mm:ss，例如 00:00:00、23:59:59
- 日期時間：YYYY-MM-DD hh:mm:ss，例如 1970-01-01 00:00:00、2022-01-01 00:00:01

標準模組 `datetime`

什麼是標準模組 `datetime`

> *`datatime` 模組為 Python 提供能夠操作日期、時間以及日期時間的類別與函數。*

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

`datetime` 模組提供的常用類別

- `date` 日期類別。
- `time` 時間類別。
- `datetime` 日期時間類別，模組與類別名稱相同，避免混淆。
- `timezone` 時區類別。
- `timedelta` 日期、時間與日期時間之間的差距類別。

`date` 日期類別

In [1]:

```
from datetime import date

the_origin_date_of_unix_time = date(1970, 1, 1)
print(the_origin_date_of_unix_time)
print(type(the_origin_date_of_unix_time))
```

```
1970-01-01
<class 'datetime.date'>
```

`time` 時間類別

In [2]:

```
from datetime import time

midnight = time(0, 0, 0)
print(midnight)
print(type(midnight))
```

```
00:00:00
<class 'datetime.time'>
```

`datetime` 日期時間類別

In [3]:

```
from datetime import datetime # from module import class

the_origin_of_unix_time = datetime(1970, 1, 1, 0, 0, 0)
print(the_origin_of_unix_time)
print(type(the_origin_of_unix_time))
```

```
1970-01-01 00:00:00
<class 'datetime.datetime'>
```

`timezone` 時區類別

In [4]:

```
from datetime import datetime # from module import class
from datetime import timezone

the_origin_of_unix_time = datetime(1970, 1, 1, 0, 0, 0, tzinfo=timezone.utc)
print(the_origin_of_unix_time)
print(type(the_origin_of_unix_time))
```

```
1970-01-01 00:00:00+00:00
<class 'datetime.datetime'>
```

`timedelta` 日期、時間與日期時間之間的差距類別

In [5]:

```
from datetime import timedelta

print(the_origin_of_unix_time - timedelta(seconds=1))
```

```
1969-12-31 23:59:59+00:00
```

In [6]:

```
taiwan_timezone = timedelta(hours=8)
the_origin_of_unix_time_taiwan = datetime(1970, 1, 1, 0, 0, 0, tzinfo=timezone(taiwan_timezone))
print(the_origin_of_unix_time_taiwan)
print(type(the_origin_of_unix_time_taiwan))
```

```
1970-01-01 00:00:00+08:00
<class 'datetime.datetime'>
```

由 `str` 轉換成為日期類別

使用 `date.fromisoformat()`

In [7]:

```
the_origin_of_unix_time = "1970-01-01"
print(type(the_origin_of_unix_time))
print(type(date.fromisoformat(the_origin_of_unix_time)))
```

```
<class 'str'>
<class 'datetime.date'>
```

由 `str` 轉換成為時間類別

使用 `time.fromisoformat()`

In [8]:

```
midnight = "00:00:00"
print(type(midnight))
print(type(time.fromisoformat(midnight)))
```

```
<class 'str'>
<class 'datetime.time'>
```

由 `str` 轉換成為日期時間類別

使用 `datetime.fromisoformat()`

In [9]:

```
the_origin_of_unix_time = "1970-01-01 00:00:00+00:00"
print(type(the_origin_of_unix_time))
print(type(datetime.fromisoformat(the_origin_of_unix_time)))
```

```
<class 'str'>
<class 'datetime.datetime'>
```

顯示與解析非 ISO 8601 的格式

- 使用 `date.strftime()`、`time.strftime()` 或 `datetime.strftime()` 搭配格式文字顯示。
- 使用 `datetime.strptime()` 搭配格式文字解析。

常用的格式文字

- `"%a"`：縮寫的星期幾，從 Sun 至 Sat
- `"%A"`：全稱的星期幾，從 Sunday 至 Saturday
- `"%b"`：縮寫的月份，從 Jan 至 Dec
- `"%B"`：全稱的月份，從 January 至 December
- `"%d"`：月份中的第幾天，從 01 至 31
- `"%m"`：以兩位數字表示的月份，從 01 至 12
- `"%Y"`：以四位數字表示的西元年份，從 0 至 9999
- `"%H"`：以兩位數字表示的小時，從 00 至 23
- `"%M"`：以兩位數字表示的分鐘，從 00 至 59
- `"%S"`：以兩位數字表示的秒數，從 00 至 61

來源：https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

使用 `date.strftime()`、`time.strftime()` 或 `datetime.strftime()` 搭配格式文字顯示

In [10]:

```
the_origin_of_unix_time = datetime(1970, 1, 1, 0, 0, 0, tzinfo=timezone.utc)
print(the_origin_of_unix_time)
print(the_origin_of_unix_time.strftime("%a %Y %b %d %H:%M"))
```

```
1970-01-01 00:00:00+00:00
Thu 1970 Jan 01 00:00
```

In [11]:

```
the_first_day_of_2022 = date(2022, 1, 1)
print(the_first_day_of_2022)
print(the_first_day_of_2022.strftime("%A %B %d, %Y"))
```

```
2022-01-01
Saturday January 01, 2022
```

使用 `datetime.strptime()` 搭配格式文字解析

In [12]:

```
the_origin_of_unix_time = "Thu 1970 Jan 01 00:00"
print(type(the_origin_of_unix_time))
the_origin_of_unix_time = datetime.strptime(the_origin_of_unix_time, "%a %Y %b %d %H:%M")
print(the_origin_of_unix_time)
print(type(the_origin_of_unix_time))
```

```
<class 'str'>
1970-01-01 00:00:00
<class 'datetime.datetime'>
```

In [13]:

```
the_first_day_of_2022 = "Saturday January 01, 2022"
print(type(the_first_day_of_2022))
the_first_day_of_2022 = datetime.strptime(the_first_day_of_2022, "%A %B %d, %Y")
print(the_first_day_of_2022)
print(type(the_first_day_of_2022))
```

```
<class 'str'>
2022-01-01 00:00:00
<class 'datetime.datetime'>
```

Pandas 的日期時間功能

Pandas 的日期時間功能與 `datetime` 模組的不同

- Pandas 採用的日期時間類別、差距類別是 NumPy 的 `datetime64` 與 `timedelta64`
- Pandas 操作一欄日期時間的序列資料；`datetime` 模組操作一個日期時間的資料。

Pandas 支援日期時間的轉換與生成

- `pd.to_datetime()` 函數可以轉換文字 `Series` 為日期時間類別。
- `pd.date_range()` 函數生成日期時間 `Index`

`pd.to_datetime()` 函數可以轉換文字 `Series` 為日期時間類別

In [14]:

```
import pandas as pd

first_day_of_years = pd.Series(["1970-01-01", "2022-01-01"])
print(first_day_of_years.dtype)
print(pd.to_datetime(first_day_of_years).dtype)
```

```
object
datetime64[ns]
```

pd.date_range() 函數生成日期時間 Index

In [15]:

```
first_day_of_years = pd.date_range("1970-01-01", "2022-01-01", freq="AS") # AS as in Annual Start
first_day_of_years
```

Out[15]:

```
DatetimeIndex(['1970-01-01', '1971-01-01', '1972-01-01', '197
3-01-01',
               '1974-01-01', '1975-01-01', '1976-01-01', '197
7-01-01',
               '1978-01-01', '1979-01-01', '1980-01-01', '198
1-01-01',
               '1982-01-01', '1983-01-01', '1984-01-01', '198
5-01-01',
               '1986-01-01', '1987-01-01', '1988-01-01', '198
9-01-01',
               '1990-01-01', '1991-01-01', '1992-01-01', '199
3-01-01',
               '1994-01-01', '1995-01-01', '1996-01-01', '199
7-01-01',
               '1998-01-01', '1999-01-01', '2000-01-01', '200
1-01-01',
```

```
               '2002-01-01', '2003-01-01', '2004-01-01', '200
5-01-01',
               '2006-01-01', '2007-01-01', '2008-01-01', '200
9-01-01',
               '2010-01-01', '2011-01-01', '2012-01-01', '201
3-01-01',
               '2014-01-01', '2015-01-01', '2016-01-01', '201
7-01-01',
               '2018-01-01', '2019-01-01', '2020-01-01', '202
1-01-01',
               '2022-01-01'],
              dtype='datetime64[ns]', freq='AS-JAN')
```

Pandas 支援差距的轉換與生成

- `pd.to_timedelta()` 函數可以轉換文字 `Series` 為差距類別。
- `pd.timedelta_range()` 函數生成差距 `Index`

`pd.to_timedelta()` 函數可以轉換文字 `Series` 為差距類別

In [16]:

```
first_day_of_years = pd.to_datetime(["1970-01-01 00:00:00", "2022-01-01 00:00:00"])
eight_hour_delta = pd.to_timedelta("08:00:00")
print(type(eight_hour_delta))
print(first_day_of_years + eight_hour_delta)
```

```
<class 'pandas._libs.tslibs.timedeltas.Timedelta'>
DatetimeIndex(['1970-01-01 08:00:00', '2022-01-01 08:00:00'],
dtype='datetime64[ns]', freq=None)
```

`pd.timedelta_range()` 函數生成差距 `Index`

In [17]:

```
the_first_day_of_2022 = pd.to_datetime("2022-01-01")
seven_days_delta_range = pd.timedelta_range("1 days", "7 days")
print(type(seven_days_delta_range))
print(the_first_day_of_2022 + seven_days_delta_range)
```

```
<class 'pandas.core.indexes.timedeltas.TimedeltaIndex'>
DatetimeIndex(['2022-01-02', '2022-01-03', '2022-01-04', '202
2-01-05',
               '2022-01-06', '2022-01-07', '2022-01-08'],
              dtype='datetime64[ns]', freq='D')
```

重點統整

- Unix 時間是類 UNIX 作業系統（泛指 Linux 或 macOS）使用的時間表示方式：從 `1970-01-01 00:00:00 UTC` 起一直到此時此刻的總秒數。
- ISO 8601 是國際標準化組織的日期、時間和日期時間的表示方法。
    - 日期：YYYY-MM-DD
    - 時間：hh:mm:ss
    - 日期時間：YYYY-MM-DD hh:mm:ss

重點統整（續）

- `datetime` 模組提供的常用類別
    - `date` 日期類別。
    - `time` 時間類別。
    - `datetime` 日期時間類別，模組與類別名稱相同，避免混淆。
    - `timezone` 時區類別。
    - `timedelta` 日期、時間與日期時間之間的差距類別。

重點統整（續）

- 使用 `date.strftime()`、`time.strftime()` 或 `datetime.strftime()` 搭配格式文字顯示。
- 使用 `datetime.strptime()` 搭配格式文字解析。
- Pandas 的日期時間功能與 `datetime` 模組的不同
    - Pandas 採用的日期時間類別、差距類別是 NumPy 的 `datetime64` 與 `timedelta64`
    - Pandas 操作一欄日期時間的序列資料；`datetime` 模組操作一個日期時間的資料。