2.2 取得氣象測站天氣資料

氣象局目前約有 560 個氣象測站，每天產生大量氣象資料。雖然氣象局建立了讓使用者查詢這些資料的網站，同時可讓使用者下載資料，但網站一次只能查詢或下載一個測站及一個時間點的資料，若使用者需同時下載多個測站及某段時間的資料，將是一件繁瑣的工作。例如要下載 50 個測站一星期 (7 天) 的日報表資料，將在網站重複操作 50 x 7 = 350 次操作。

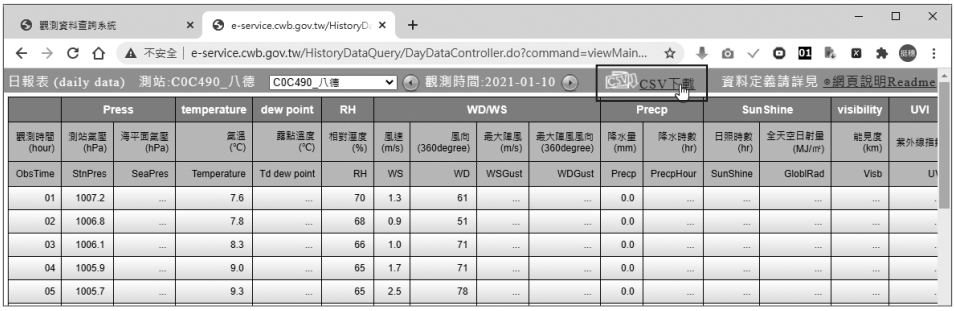
HistoricalWeatherTW 模組可設定測站、時間、資料型態等參數，將所需資料一次下載完成，並儲存為 CSV 格式檔案。

2.2.1 氣象局觀測資料查詢網站 (CODiS)

觀測資料查詢網站 (http://e-service.cwb.gov.tw/HistoryDataQuery/) 可利用提供的表單查詢及下載氣象局所有氣象測站的資料。

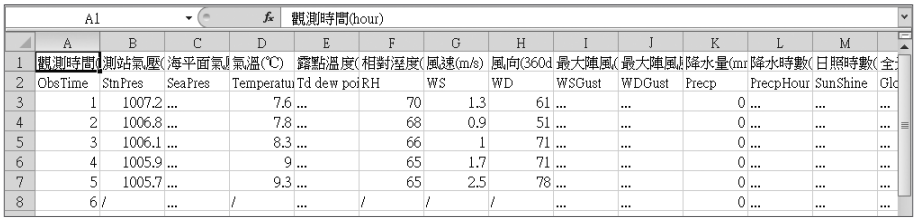
- **測站所在縣市、測站**：可在左方先點選測站所在縣市，再於 **測站** 欄選擇該縣市的測站；也可在右方先點選地區，再於地圖中點選測站圖示，該測站就會顯示於左方查詢欄位中。
- **資料格式**：有日報表、月報表及年報表三種格式。
- **時間**：點選欄位右方 圖示選擇日期。

下圖為日報表資料：每小時一筆資料，共有 24 筆資料。如果要下載此資料，按上方 **CSV 下載** 項目。



	Press		temperature	dew point	RH	WD/WS				Precp		SunShine		visibility	UVI
觀測時間 (hour)	測站氣壓 (hPa)	海平面氣壓 (hPa)	氣溫 (℃)	露點溫度 (℃)	相對濕度 (%)	風速 (m/s)	風向 (360degree)	最大陣風 (m/s)	最大陣風風向 (360degree)	降水量 (mm)	降水時數 (hr)	日照時數 (hr)	全天空日射量 (MJ/m²)	能見度 (km)	紫外線指
ObsTime	StnPres	SeaPres	Temperature	Td dew point	RH	WS	WD	WSGust	WDGust	Precp	PrecpHour	SunShine	GloblRad	Visb	UV
01	1007.2	...	7.6	...	70	1.3	61	...	...	0.0	...	...	...	...	.
02	1006.8	...	7.8	...	68	0.9	51	...	...	0.0	...	...	...	...	.
03	1006.1	...	8.3	...	66	1.0	71	...	...	0.0	...	...	...	...	.
04	1005.9	...	9.0	...	65	1.7	71	...	...	0.0	...	...	...	...	.
05	1005.7	...	9.3	...	65	2.5	78	...	...	0.0	...	...	...	...	.

下載的檔案為 <C0C490-2021-01-10.csv>：「C0C490」為資料格式，此為日報表；「2021-01-10」為日期。



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	觀測時間(	測站氣壓(	海平面氣壓	氣溫(℃)	露點溫度(	相對濕度(	風速(m/s)	風向(360d	最大陣風(	最大陣風(	降水量(mr	降水時數(	日照時數(
2	ObsTime	StnPres	SeaPres	Temperatu	Td dew poi	RH	WS	WD	WSGust	WDGust	Precp	PrecpHour	SunShine
3	1	1007.2	...	7.6	...	70	1.3	61	...	...	0	...	...
4	2	1006.8	...	7.8	...	68	0.9	51	...	...	0	...	...
5	3	1006.1	...	8.3	...	66	1	71	...	...	0	...	...
6	4	1005.9	...	9	...	65	1.7	71	...	...	0	...	...
7	5	1005.7	...	9.3	...	65	2.5	78	...	...	0	...	...
8	6	/	...	/	...	/	/	/	...	...	0	...	...

在網站上操作只能下載單一測站單日的日報表、單月的月報表或單年的年報表，若要下載多個測站的某段時間氣象資料，必須反覆多次下載操作。

2.2.2 HistoricalWeatherTW：台灣氣象測站資料爬蟲

氣象局在全台有許多測站，每個測站會定時更新氣象資料，如果要逐一下載各測站資料是一件繁瑣的工作。HistoricalWeatherTW 模組可根據使用者需求，一次下載多個測站的某段時間氣象資料。

模組名稱	`HistoricalWeatherTW`
模組功能	取得台灣氣象測站資料
官方網站	`https://github.com/CarsonSlovoka/HistoricalWeatherTW`
安裝方式	`!pip install carson-tool.HistoricalWeatherTW`

模組使用方式

1. 匯入 HistoricalWeatherTW 模組的語法：

```
from Carson.Tool.HistoricalWeatherTW import
    collect_weather_tw, QueryFormat
```

2. 設定資料儲存檔案路徑時需使用 Path 模組，匯入的語法為：

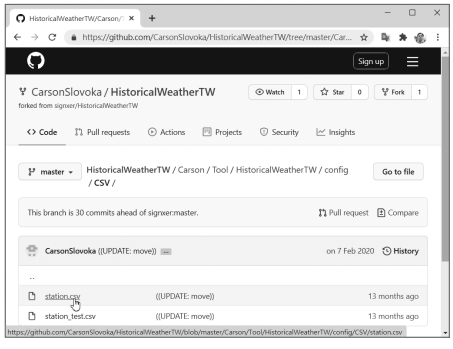
```
from pathlib import Path
```

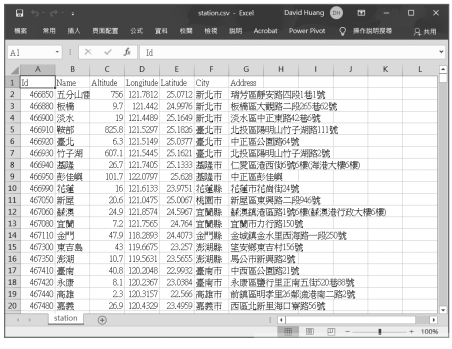
建立測站資料

如果想要爬取多個氣象測站的資料，必須將這些測站的資料整理成 CSV 檔。在觀測資料查詢網站的「https://e-service.cwb.gov.tw/wdps/obs/state.htm」頁面中提供了所有的測站資料。

站號	站名	海拔高度(m)	經度	緯度	城市	地址
466850	五分山雷達站	756.0	121.781205	25.071182	新北市	瑞芳區靜安路四段1巷1號
466880	板橋	9.7	121.442017	24.997647	新北市	板橋區大觀路二段265巷62號
466900	淡水	19.0	121.448906	25.164889	新北市	淡水區中正東路42巷6號
466910	鞍部	837.6	121.529731	25.182586	臺北市	北投區陽明山竹子湖路111號
466920	臺北	5.3	121.514853	25.037658	臺北市	中正區公園路64號

在 HistoricalWeatherTW 模組的官方網站上將所有氣象測站的資料整理成 CSV 檔 (https://github.com/CarsonSlovoka/HistoricalWeatherTW/blob/master/Carson/Tool/HistoricalWeatherTW/config/CSV/station.csv)，其中包含了 560 筆資料。

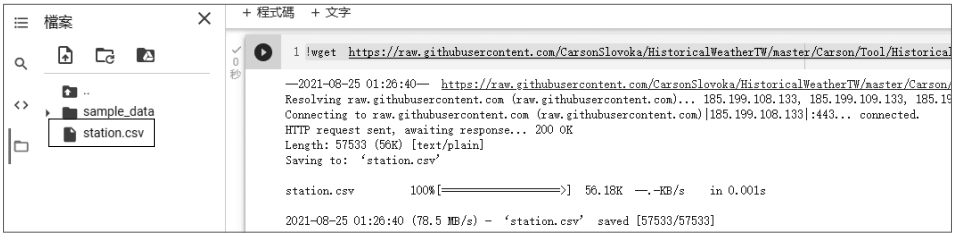




執行下面命令可下載所有測站資料檔 <station.csv>：

```
!wget https://raw.githubusercontent.com/CarsonSlovoka/HistoricalWeatherTW/
    master/Carson/Tool/HistoricalWeatherTW/config/CSV/station.csv
```

如果執行後在檔案總管 <station.csv> 沒有顯示，請按重整鈕 來顯示：



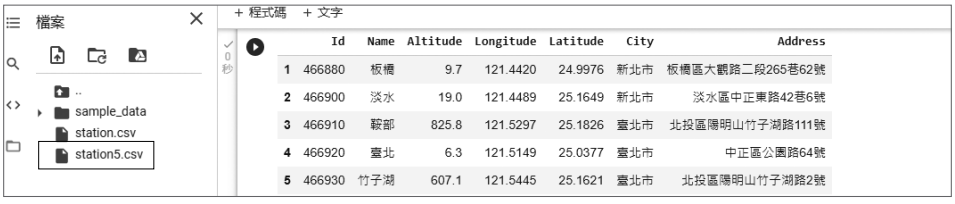
本模組需將要爬取氣象資料的測站資料存於 CSV 檔做為 HistoricalWeatherTW 模組的參數。<station.csv> 為所有測站資料，多達 560 餘個，若以此爬取資料將耗費相當長的時間；以下面程式擷取 <station.csv> 前 5 筆資料儲存為 <station5.csv>，本節將以 <station5.csv> 做為示範。

```
1 import pandas as pd
2 df = pd.read_csv('station.csv')
3 df1 = df[1:6]
4 df1.to_csv('station5.csv', index=False)
5 df1
```

程式說明

- 2　讀取所有測站資料。
- 3　因第 1 列為標題，故由第 2 列開始取 5 筆資料。
- 4　以忽略索引方式儲存為 <station5.csv>。
- 5　顯示前 5 筆資料內容。

執行結果：



	Id	Name	Altitude	Longitude	Latitude	City	Address
1	466880	板橋	9.7	121.4420	24.9976	新北市	板橋區大觀路二段265巷62號
2	466900	淡水	19.0	121.4489	25.1649	新北市	淡水區中正東路42巷6號
3	466910	鞍部	825.8	121.5297	25.1826	臺北市	北投區陽明山竹子湖路111號
4	466920	臺北	6.3	121.5149	25.0377	臺北市	中正區公園路64號
5	466930	竹子湖	607.1	121.5445	25.1621	臺北市	北投區陽明山竹子湖路2號

5.1 文字語音轉換模組

將文字轉換為語音稱為 TTS (Text To Speech)，目前 TTS 的技術已臻成熟，讀出的語音已可被大部分人們接受。

將語音轉換為文字稱為「語音辨識 (speech recognition)」，是以電腦自動將人類的語音內容轉換為相應的文字。語音辨識技術的應用很廣泛，包括語音撥號、語音導航、室內裝置控制等。

5.1.1 gTTS 模組：文字轉語音

模組名稱	gTTS
模組功能	透過線上翻譯，將文字轉換為語音，並將語音存檔。
官方網站	https://github.com/pndurette/gTTS
安裝方式	!pip install gtts

模組使用方式

1. 匯入 gTTS 模組的語法：

```
import gtts
```

2. 因為文字轉語音後需要播放語音，所以也要匯入 IPython.display 模組：

```
import IPython.display as display
```

3. 接著建立文字轉語音物件，語法為：

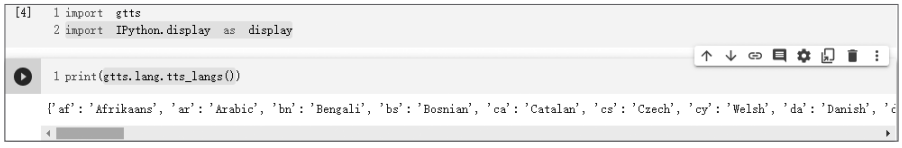
```
語音物件變數 = gtts.gTTS(text=轉換的文字 , lang=語言代碼 [,slow=布林值 ])
```

- **text**：代表要轉換的文字字串。
- **lang**：lang 參數為 ISO 639-1 語言代碼。
- **slow**：設定發音速度，預設值為 False，若設為 True 會產生一個發音速度比較慢的 mp3 檔案。

設定支援語言

gTTS 模組提供 lang.tts_langs() 方法顯示支援的語言代碼讓使用者參考，語法為：

```
gtts.lang.tts_langs()
```



傳回值是字典資料，鍵值對為「語言代碼 : 文字」：

```
{'af': 'Afrikaans', 'ar': 'Arabic', 'bn': 'Bengali', ……}
```

下表為較常用的語言代碼：

語言代碼	文字	語言代碼	文字
zh-tw	繁體中文	zh-cn	簡體中文
en	英文	de	德文
ja	日文	ko	韓文
fr	法文	es	西班牙文

例如：語音物件變數為 tts，將「線上翻譯」繁體中文轉換為語音。

```
tts = gtts.gTTS(text='線上翻譯', lang='zh-tw')
```

將語音儲存為 mp3 檔：save()

1. 呼叫語音物件的 save() 方法可以將轉換的語音儲存為 mp3 檔，語法為：

```
tts.save( 檔案路徑 )
```

「檔案路徑」非必填，若未指定檔案路徑則存於目前工作目錄中。

例如：將 tts 語音物件變數內容存於 <gtts.mp3> 檔。

```
tts.save('gtts.mp3')
```

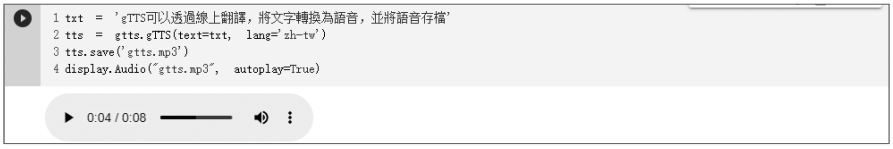
2. 最後就可用 IPython.display 模組讀出轉換後的語音，語法為：

```
display.Audio( 檔案路徑 , autoplay=True)
```

例如：想將一段文字轉換為語音後存檔，最後讀出語音。

範例：文字轉語音後播放

```
txt = 'gTTS 可以透過線上翻譯，將文字轉換為語音，並將語音存檔 '
tts = gtts.gTTS(text=txt, lang='zh-tw')
tts.save('gtts.mp3')
display.Audio("gtts.mp3", autoplay=True)
```



多段語音合併存檔：write_to_fp()

如果要轉換的文字太長，可以將文字分解為多個段落，分別轉換為語音後，共同存入同一個語音檔中。

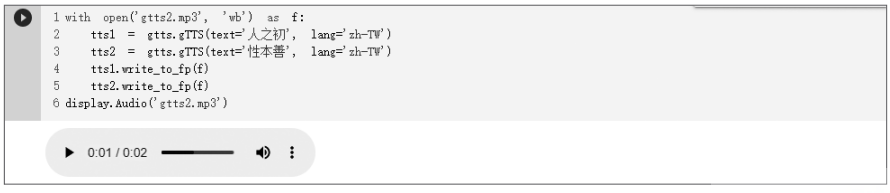
語音物件的 write_to_fp() 方法可將多個語音物件內容寫入同一個檔案，語法為：

```
with open( 檔案路徑 , "wb") as 檔案變數 :
    語音物件 1 = gtts.gTTS(text= 第 1 段文字 , lang= 語音代碼 )
    語音物件 1 .write_to_fp( 檔案變數 )
    語音物件 2 = gtts.gTTS(text= 第 2 段文字 , lang= 語音代碼 )
    語音物件 2 .write_to_fp( 檔案變數 )
.....
```

例如：想將兩段文字轉換為語音，並將兩段語音存入同一個檔案中。

範例：將多段文字轉語音進行語音合併

```
with open('gtts2.mp3', 'wb') as f:
  tts1 = gtts.gTTS(text='人之初', lang='zh-TW')
  tts2 = gtts.gTTS(text='性本善', lang='zh-TW')
  tts1.write_to_fp(f)
  tts2.write_to_fp(f)
display.Audio('gtts2.mp3')
```



6.2 twstock 模組：台灣股市資訊

twstock 是台灣股市的專用模組，可以讀取指定股票的歷史記錄、股票分析和即時股票的買賣資訊等。

模組名稱	twstock
模組功能	讀取台灣股票資訊
官方網站	https://github.com/mlouielu/twstock
安裝方式	!pip install twstock

6.2.1 查詢歷史股票資料

模組使用方式

1. 匯入 twstock 模組的語法：

```
import twstock
```

2. twstock 模組利用 Stock() 方法查詢個股歷史股票資料，語法為：

```
歷史股票變數 = twstock.Stock('股票代號')
```

例如，設定歷史股票變數為 stock，查詢鴻海股票 (代碼 2317) 的歷史資料，預設會讀取最近 31 日的歷史記錄。

```
stock = twstock.Stock('2317')
```

取得股票歷史資料

1. 利用 Stock 物件的屬性即可以讀取指定的歷史資料，Stock 物件的屬性：

屬性	說明	屬性	說明
date	日期 (datetime.datetime)	**low**	最低價
capacity	總成交股數 (單位：股)	**price**	收盤價
turnover	總成交金額 (單位：元)	**close**	收盤價
open	開盤價	**change**	漲跌價差
high	最高價	**transaction**	成交筆數

例如：顯示「鴻海」最近 31 筆收盤價資。

```
print(stock.price)
```

```
1 stock = twstock.Stock('2317')
2 print(stock.price)

[87.8, 87.7, 88.0, 87.7, 88.8, 89.6, 91.8, 91.8, 90.4, 91.6, 92.0, 99.9, 104.0, 105.0, 107.0, 108.0, 107.5, 104.0, 106.5, 116.0, 115.5,
```

傳回結果為串列，可使用串列語法擷取部分資料，例如顯示最近 1 日開盤價、最高價、最低價及收盤價：

```
print("日期：",stock.date[-1])
print("開盤價：",stock.open[-1])
print("最高價：",stock.high[-1])
print("最低價：",stock.low[-1])
print("收盤價：",stock.price[-1])
```

```
1 print('日期: ',stock.date[-1])
2 print('開盤價: ',stock.open[-1])
3 print('最高價: ',stock.high[-1])
4 print('最低價: ',stock.low[-1])
5 print('收盤價: ',stock.price[-1])

日期:  2021-01-29 00:00:00
開盤價:  119.0
最高價:  120.0
最低價:  111.5
收盤價:  111.5
```

2. Stock 物件也提供下列 fetch()、fetch_31() 及 fetch_from() 方法可以讀取指定期間的歷史資料。

方法	傳回資料
fetch(西元年 , 月)	傳回參數指定月份的資料。
fetch_31()	傳回最近 31 日的資料。
fetch_from(西元年 , 月)	傳回參數指定月份到現在的資料。

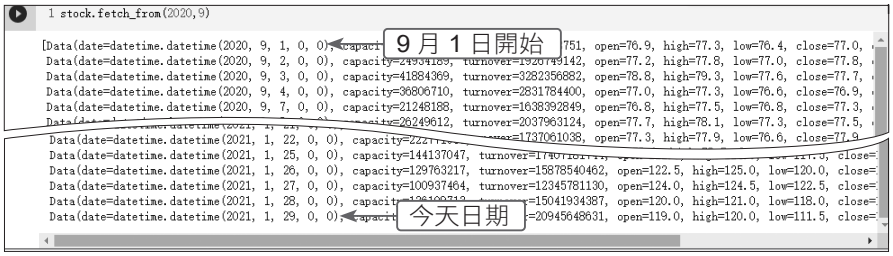
例如：以 fetch 取得 2020 年 1 月的資料

```
stock.fetch(2020,1)
```

```
1 stock.fetch(2020,1)

[Data(date=datetime.datetime(2020, 1, 2, 0, 0), capacity=20758722, turnover=1886677519, open=91.0, high=91.5, low=90.3, close=90.8, cha
 Data(date=datetime.datetime(2020, 1, 3, 0, 0), capacity=37936877, turnover=3471335594, open=91.4, high=92.2, low=90.8, close=91.6, cha
 Data(date=datetime.datetime(2020, 1, 6, 0, 0), capacity=26352522, turnover=2388785688, open=91.1, high=91.1, low=90.1, close=90.5, cha
 Data(date=datetime.datetime(2020, 1, 7, 0, 0), capacity=43978140, turnover=3935667984, open=90.5, high=91.0, low=88.3, close=89.1, cha
 Data(date=datetime.datetime(2020, 1, 8, 0, 0), capacity=56101121, turnover=4891344755, open=87.9, high=88.1, low=86.5, close=86.5, cha
 Data(date=datetime.datetime(2020, 1, 9, 0, 0), capacity=28513381, turnover=2491571248, open=87.3, high=87.7, low=87.0, close=87.1, cha
 Data(date=datetime.datetime(2020, 1, 10, 0, 0), capacity=32264863, turnover=2852056044, open=88.0, high=89.0, low=87.5, close=89.0, ch
 Data(date=datetime.datetime(2020, 1, 13, 0, 0), capacity=23369554, turnover=2084580613, open=89.7, high=89.7, low=88.6, close=89.6, ch
 Data(date=datetime.datetime(2020, 1, 14, 0, 0), capacity=19568838, turnover=1758920258, open=90.0, high=90.1, low=89.6, close=90.0, ch
 Data(date=datetime.datetime(2020, 1, 15, 0, 0), capacity=23798041, turnover=2141165898, open=90.0, high=90.3, low=89.5, close=89.9, ch
```

例如：以 fetch_from() 方法取得 2020 年 9 月至今的資料。

```
stock.fetch_from(2020,9)
```



IP 會被鎖定

當以 fetch()、fetch_31() 和 fetch_from() 方法向台灣證券交易所網頁讀取資料時，如果資料量太大，很容易會被視為攻擊而被鎖定 IP，如此就無法連上該網站，必須等一段時間才可再連上證券交易所網頁。若下載資料量多時，建議分時間、分次下載，避免一次下載太多資料，而且每次下載的時間最好有點間隔。

6.2.2 查詢股票即時交易資訊

twstock 模組利用 realtime.get() 方法查詢個股即時股票資訊，語法為：

```
即時股票變數 = twstock.realtime.get(' 股票代號 ')
```

例如：設定即時股票變數為 real，查詢鴻海股票 (代碼 2317) 的即時交易資訊。

```
real = twstock.realtime.get('2317')
```

傳回資料為：

```
{'timestamp': 1611901800.0, 'info': {'code': '2317', 'channel':
 '2317.tw', 'name': ' 鴻海 ', 'fullname': ' 鴻海精密工業股份有限公司 ',
 'time': '2021-01-29 06:30:00'}, 'realtime': {'latest_trade_
 price': '111.5000', 'trade_volume': '20289', 'accumulate_trade_
 volume': '179678', 'best_bid_price': ['111.5000', '111.0000',
 '110.5000', '110.0000', '109.5000'], 'best_bid_volume': ['1350',
 '3175', '2552', '4918', '127'], 'best_ask_price': ['112.0000',
 '112.5000', '113.0000', '113.5000', '114.0000'], 'best_ask_
 volume': ['92', '135', '577', '1278', '1844'], 'open': '119.0000',
 'high': '120.0000', 'low': '111.5000'}, 'success': True}
```

10.2 AI 聊天機器人

聊天機器人是經由對話或文字進行交談的電腦程式，是自然語言處理的重要應用之一。目前大多簡單的聊天機器人系統是擷取輸入的關鍵字，再從資料庫中找尋最合適的應答句。聊天機器人最常用於客服系統。

10.2.1 chatterbot 模組內建模式

模組名稱	ChatterBot
模組功能	基於已知會話的集合生成回應，支援多國語言。
官方網站	https://github.com/gunthercox/ChatterBot
安裝方式	!pip install chatterbot

模組使用方式

匯入 chatterbot 模組的語法：

```
from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer
```

使用 chatterbot 模組首先要建立 ChatBot 物件，語法為：

```
機器人物件變數 = ChatBot( 機器人名稱 ,  storage_adapter= 資料庫轉接器 ,
    logic_adapters= 邏輯轉接器 )
```

- **機器人名稱**：為此機器人任意指定一個名稱。
- **storage_adapter**：設定使用的資料庫，此參數非必填。可能的值有：
  - **chatterbot.storage.SQLStorageAdapter**：使用 SQLite 資料庫。
  - **chatterbot.storage.MongoDatabaseAdapter**：使用 MongoDB 資料庫。
- **logic_adapters**：設定使用的模式。此設定值是一個串列，可同時使用多個模式。常用的模式有：
  - **chatterbot.logic.BestMatch**：一般匹配模式。
  - **chatterbot.logic.MathematicalEvaluation**：數學模式。
  - **chatterbot.logic.TimeLogicAdapter**：時間模式。

例如機器人物件變數為 bot，機器人名稱為 MathTimeBot，資料庫轉接器為 SQLite 資料庫，邏輯轉接器使用數學模式：

```
bot = ChatBot('MathTimeBot',
    storage_adapter='chatterbot.storage.SQLStorageAdapter',
    logic_adapters=['chatterbot.logic.MathematicalEvaluation'])
```

chatterbot 數學與時間模式

數學模式及時間模式使用較為簡單：它們使用系統內建的資料模型進行判斷，然後給予回應。下面程式建立數學及時間模式機器人：

範例：建立數學及時間模式機器人

```
bot = ChatBot(
    'MathTimeBot',
    storage_adapter='chatterbot.storage.SQLStorageAdapter',
    logic_adapters=[
        'chatterbot.logic.MathematicalEvaluation',
        'chatterbot.logic.TimeLogicAdapter'
    ]
)

question = '14 + 19 = ?'
response = bot.get_response(question)
print('{} -> {}\n'.format(question, response))

question = '45 - 23 等於多少 ?'
response = bot.get_response(question)
print('{} -> {}\n'.format(question, response))

question = 'What time is it?'
response = bot.get_response(question)
print('{} -> {}\n'.format(question, response))

question = 'how are you?'
response = bot.get_response(question)
print('{} -> {}\n'.format(question, response))
```

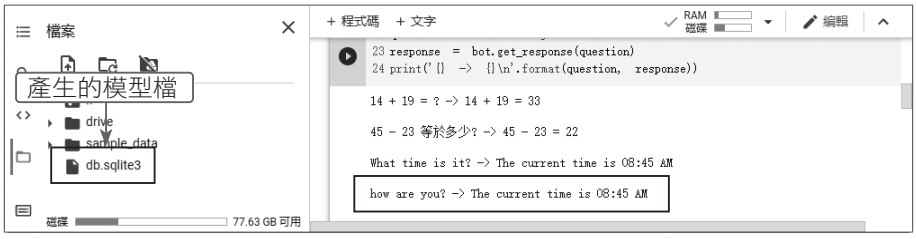
程式說明

- 3　　設定使用 SQLite 資料庫。
- 4-7　　設定使用數學及時間模式。

- 10-16　數學問答示例。
- 18-20　時間問答示例。
- 22-24　非數學及時間示例。

執行時會先檢查程式所在目錄是否有 <db.sqlite3> 資料庫檔存在，若不存在會複製系統內建的數學及時間模型到 <db.sqlite3> 資料庫檔，然後以 <db.sqlite3> 資料庫檔為模型進行處理。(有時還會產生 <db.sqlite3-sgm> 及 <db.sqlite3-wal> 檔)

執行結果：



注意：數學模式只檢查是否有數學算式存在，存在的話就計算後返回結果。時間模式經測試只能返回現在時間。若遇到看不懂的文句就返回現在時間。

chatterbot 串列訓練模式

chatterbot 模組允許使用者自行輸入文句來訓練產生模型，其中較簡單的方式是將文句置於串列中進行訓練，這就是串列訓練模式。

串列訓練模式首先要設定 ChatBot 物件的 logic_adapters 參數，語法為：

```
logic_adapters=[
    { 'import_path': 'chatterbot.logic.BestMatch',
      'default_response': 預設回應文句 ,
      'maximum_similarity_threshold': 數值 ,
    }
]
```

- **default_response**：設定機器人看不懂問句時的回應文句。
- **maximum_similarity_threshold**：機器人尋找回應相似度大於此設定值就停止尋找。預設值為 0.95。

例如預設回應文句為「我不了解你的意思」，最大相似度為 0.65：

```
logic_adapters=[
    { 'import_path': 'chatterbot.logic.BestMatch',
      'default_response': '很抱歉！我不了解你的意思。',
      'maximum_similarity_threshold': 0.65,
    }
]
```

然後建立 ListTrainer 物件，語法為：

```
訓練變數 = ListTrainer( 機器人物件變數 )
```

最後以 train() 方法進行訓練，語法為：

```
訓練變數 .train([ 文句 1, 文句 2, ·····])
```

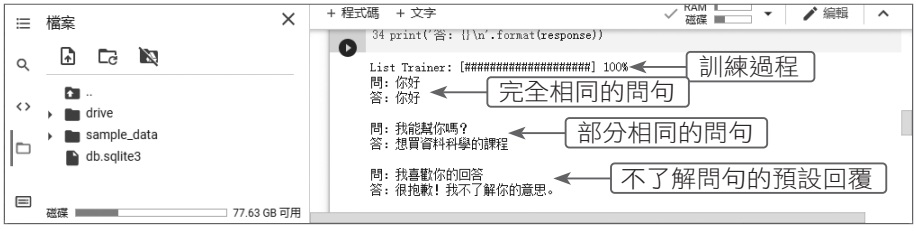
文句的安排順序是一問一答：文句 2 是文句 1 的回答，文句 4 是文句 3 的回答，依此類推。訓練完成後會將模型儲存於程式所在目錄的 <db.sqlite3> 檔。

下面程式建立串列訓練模式並進行數個問答示例：

```
bot = ChatBot(
    'SimpleBot',
    storage_adapter='chatterbot.storage.SQLStorageAdapter',
    logic_adapters=[
        { 'import_path': 'chatterbot.logic.BestMatch',
          'default_response': '很抱歉！我不了解你的意思。',
          'maximum_similarity_threshold': 0.65,
        }
    ]
)
trainer = ListTrainer(bot)
trainer.train([
    '你好',
    '你好',
    '有什麼能幫你的？',
    '想買資料科學的課程',
    '具體是資料科學哪塊呢？'
    '機器學習',
])

question = '你好'
```

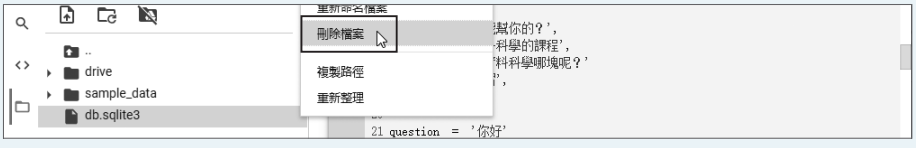
```
print('問：{}'.format(question))
response = bot.get_response(question)
print('答：{}\n'.format(response))

question = '我能幫你嗎？'
print('問：{}'.format(question))
response = bot.get_response(question)
print('答：{}\n'.format(response))

question = '我喜歡你的回答'
print('問：{}'.format(question))
response = bot.get_response(question)
print('答：{}\n'.format(response))
```

執行結果：



先移除 <db.sqlite3> 檔再訓練

經實測，第二次執行此程式其結果常會與第一次執行有差異，如果執行訓練前發現 <db.sqlite3> 檔已存在，最好先移除 <db.sqlite3> 檔再進行訓練。



11.1 Selenium 模組：瀏覽器自動化操作

一般情況下，我們都是以人工操作方式，執行瀏覽器上的各項操作。事實上，只要安裝自動化操作模組，Python 就可以代替我們自動執行。

在網頁應用程式開發時，測試使用者介面一向是相當困難的工作。如果以手動的方式進行操作，不僅會因為人力時間而受到限制，而且也容易出錯。Selenium 的出現就是為了解決這個問題，它可以藉由指令自動操作網頁，達到測試的功能。如果延伸這個功能，Selenium 也能讓許多在網頁上要大量操作的工作指令化，能在設定的時間內自動執行，功能相當強大。

雖然 Colab 也可經由安裝一些特殊模組來使用 Selenium，但因 Colab 程式是在遠端伺服器執行，無法開啟本機瀏覽器進行操作，**因此本章的 Selenium 及 pyautogui 模組的所有範例都在本機執行。**

11.1.1 使用 Selenium 模組

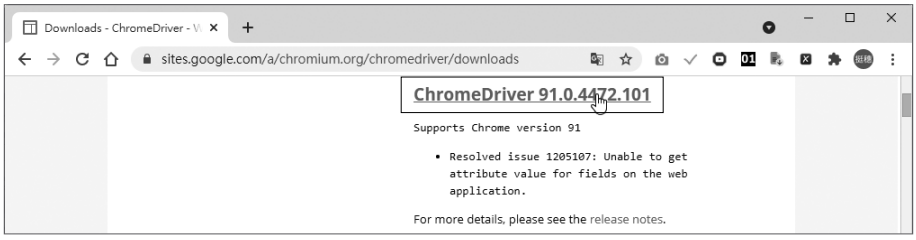
模組名稱	Selenium
模組功能	瀏覽器操作自動化
官方網站	https://github.com/SeleniumHQ/selenium/
安裝方式	pip install selenium (本機安裝)

下載 Chrome WebDriver

在 Chrome 瀏覽器操作 Selenium，還必須安裝相關的驅動程式。首先檢查 Chrome 版本：點選 ⚙ / **說明** / **關於 Google Chrome**，即可看到目前版本編號。

請到下面網址依照作業系統及 Chrome 版本下載 Chrome WebDriver 並解壓縮：(版本編號最後碼可能不同，找最接近者)

```
https://sites.google.com/a/chromium.org/chromedriver/downloads
```



以 Windows 作業系統為例，下載後解壓縮產生 <chromedrvier.exe> 檔，再複製到目前專案的工作目錄中。

建立 Google Chrome 瀏覽器物件

1. 第一步是匯入 selenium 模組，語法為：

```
from selenium import webdriver
```

2. 然後使用 Chrome() 方法建立 Google Chrome 瀏覽器物件，語法為：

```
瀏覽器物件變數 = webdriver.Chrome()
```

例如瀏覽器物件變數為 driver：

```
driver = webdriver.Chrome()
```

Selenium Webdriver 的屬性和方法

Selenium Webdriver API 常用的屬性和方法如下：

屬性及方法	說明
current_url	取得目前的網址。
page_source	讀取網頁的原始碼。
get_window_position()	取得視窗左上角的位置。
set_window_position(x,y)	設定視窗左上角的位置。
maximize_window()	瀏覽器視窗最大化。

屬性及方法	說明
get_window_size()	取得視窗的高度和寬度。
set_window_size(x,y)	設定視窗的高度和寬度。
click()	按單擊鈕。
close()	關閉瀏覽器。
get(url)	連結 url 網址。
refresh()	重新整理畫面。
back()	返回上一頁。
forward()	下一頁。
clear()	清除輸入內容。
send_keys()	以鍵盤輸入。
submit()	提交。
quit()	關閉瀏覽器並且退出驅動程序。

範例：開啟網頁並顯示瀏覽器資訊

以 Selenium 開啟指定網頁，顯示部分資訊，於 5 秒後關閉瀏覽器。

程式碼：**browse.py**

```
 1 from selenium import webdriver
 2 from time import sleep
 3
 4 driver = webdriver.Chrome()
 5 driver.get('http://www.e-happy.com.tw')
 6 driver.maximize_window()
 7 print('目前網址：', driver.current_url)
 8 print('瀏覽器尺寸：', driver.get_window_size())
 9 print('網頁原始碼：\n', driver.page_source)
10
11 sleep(5)
12 driver.quit()
```

程式說明

- 4　　建立瀏覽器物件。
- 5　　開啟文淵閣工作室首頁。

- 6　　將瀏覽器放大到全螢幕。
- 7-9　　顯示網址、瀏覽器尺寸及網頁原始碼。
- 11-12　停留 5 秒後關閉瀏覽器。

本書所有本機程式都在 <C:\example> 資料夾執行：請將本章範例複製到 <C:\example> 資料夾，開啟 **命令提示字元** 視窗，切換到 <C:\example> 資料夾，以下列命令執行程式：

```
python 程式名稱
```

例如執行本範例程式為：

```
python browse.py
```

執行結果：瀏覽器自動開啟 2 個頁籤且停留在 **設定** 頁籤，頁面會有 **要求重設您的設定** 對話方塊，可以不予理會，切換到文淵閣頁籤即可。Selenium 開啟的網頁會有 **Chrome 目前受到自動測試軟體控制** 的警告訊息。

命令提示字元 視窗顯示的訊息：

```
目前網址： http://www.e-happy.com.tw/
瀏覽器尺寸： {'width': 1382, 'height': 744}
網頁原始碼：
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
.....
```

12.1.5 人體整合偵測 (Holistic)

人體整合偵測結合上述臉部特徵網、手部及姿勢偵測。

模組使用方式

1. 建立人體整合物件，語法為：

```
人體整合變數 = mp.solutions.holistic
```

例如，人體整合變數為 mp_holistic：

```
mp_holistic = mp.solutions.holistic
```

2. 以人體整合物件的 Holistic() 方法建立姿勢偵測物件，參數與姿勢偵測完全相同，語法為：

```
偵測變數 = 人體整合變數.Holistic(static_image_mode=布林值, model_complexity=數值,
smooth_landmarks=布林值, min_detection_confidence=數值, min_tracking_confidence=數值)
```

例如，偵測變數為 holistic，偵測靜態圖片，模型複雜度為 1：

```
holistic = mp_holistic.Holistic(static_image_mode=True, model_complexity=1)
```

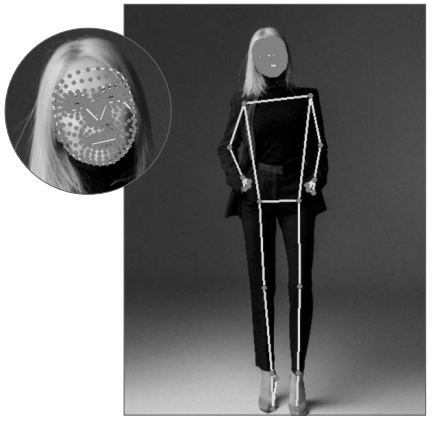
範例：偵測臉部特徵網、手部及姿勢並畫出所有特徵點

下面程式可偵測圖片中的臉部特徵網、手部及姿勢並畫出所有特徵點：

```
1 # 人體整合偵測
... (略)
16 image = resizeimg(cv2.imread('pose1.jpg'))
17 # image = resizeimg(cv2.imread('pose2.jpg'))
18 mp_drawing = mp.solutions.drawing_utils
19 mp_holistic = mp.solutions.holistic
20 holistic = mp_holistic.Holistic(static_image_mode=True, model_complexity=1)
21 results = holistic.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
22 if results.pose_landmarks:
23     mp_drawing.draw_landmarks(image, results.face_landmarks,
                                 mp_holistic.FACE_CONNECTIONS)
24     mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
                                 mp_holistic.HAND_CONNECTIONS)
25     mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
                                 mp_holistic.HAND_CONNECTIONS)
```

```
26      mp_drawing.draw_landmarks(image, results.pose_landmarks,
                                  mp_holistic.POSE_CONNECTIONS)
27  cv2_imshow(image)
```

執行結果：



▲ pose1.jpg

▲ pose2.jpg

12.1.6 3D 物體偵測 (Objectron)

3D 物體偵測可偵測圖片中的物體，並畫出物體的 3D 結構。目前物體種類僅支援鞋子、椅子、杯子及相機。

模組使用方式

1. 建立 3D 物體物件，語法為：

```
3D 物體變數 = mp.solutions.objectron
```

例如 3D 物體變數為 mp_objectron：

```
mp_objectron = mp.solutions.objectron
```

2. 以 3D 物體物件的 Objectron() 方法建立 3D 物體偵測物件，語法為：

```
偵測變數 = 3D 物體變數 .Objectron(static_image_mode= 布林值 ,
    max_num_objects= 數值 , min_detection_confidence= 數值 ,
    min_tracking_confidence= 數值 , model_name= 模型名稱 )
```

- **static_image_mode**：True 表示靜態圖片，False 表示影片 (預設值)。
- **max_num_objects**：設定偵測物體的最多數量。
- **min_detection_confidence**：最小偵測信心指數，其值 0 到 1 之間。
- **min_tracking_confidence**：最小追蹤信心指數，其值 0 到 1 之間。
- **model_name**：設定模型名稱，可能值有 Shoe (鞋子)、Chair (椅子)、Cup (杯子)、Camera (相機)。

例如偵測變數為 objectron，偵測靜態圖片，最多偵測 5 個物體，最小偵測信心指數為 0.5，用於偵測「椅子」物體：

```
objectron = mp_objectron.Objectron(static_image_mode=True,
    max_num_objects=5, min_detection_onfidence=0.5, model_name='Chair')
```

範例：偵測圖片中的椅子並畫出 3D 結構

請由 Unsplash 網站的 https://unsplash.com/photos/kvmdsTrGOBM 下載圖片並名命名「object1.jpg」；由 https://unsplash.com/photos/rhcllVy2zBU 下載圖片並名命名「object2.jpg」。

下面程式可偵測圖片中的椅子並畫出 3D 結構：

```
#3D 物體偵測
... ( 略 )
image = resizeimg(cv2.imread('object1.jpg'))
# image = resizeimg(cv2.imread('object2.jpg'))
mp_drawing = mp.solutions.drawing_utils
mp_objectron = mp.solutions.objectron
objectron = mp_objectron.Objectron(static_image_mode=True,
    max_num_objects=5, min_detection_confidence=0.5, model_name='Chair')

results = objectron.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
if results.detected_objects:
  for detected_object in results.detected_objects:
      mp_drawing.draw_landmarks(image, detected_object.landmarks_2d,
                                mp_objectron.BOX_CONNECTIONS)
      mp_drawing.draw_axis(image, detected_object.rotation,
                                detected_object.translation)
cv2_imshow(image)
```

執行結果：

▲ object1.jpg

▲ object2.jpg

12.1.7 在 MediaPipe 中使用 WebCam 攝影機

MediaPipe 最為人稱道的就是處理速度非常快，即使不使用 GPU，利用 CPU 來處理影片也有令人滿意的結果。

因 Colab 在遠端伺服器執行，雖然可以利用 Javascript 開啟本機攝影機拍攝及播放，若是要處理攝影機拍攝畫面如偵測人臉，必須將攝影機拍攝畫面傳送到伺服器，處理完成後再送回本機顯示，畫面停頓嚴重。在這裡將要介紹，在本機的環境中，如何使用 MediaPipe 模組搭配本機的 WebCam 攝影機進行臉部偵測等動作。

在開啟操作之前，**記住在本機要先安裝 MediaPipe**。下面程式會開啟攝影機進行人臉偵測，偵測過程相當流暢，按 **q** 鍵就結束程式。

程式碼： faceDetect_cam.py

```
1 import cv2
 2 import mediapipe as mp
 3
 4 cap = cv2.VideoCapture(0)
 5
 6 mp_face_detection = mp.solutions.face_detection
 7 mp_drawing = mp.solutions.drawing_utils
 8 face_detection = mp_face_detection.FaceDetection(min_detection_confidence=0.5)
 9 while cap.isOpened():
10     success, image = cap.read()
```