自動收集資料的程式

- 參考 Web Scraping with Python by Ryan Mitchell ，我個人則將擷取網頁資料的程式粗略分成三大類
  - A. 存取網路的應用程式界面 (web API，其中 API 為 Application Programming Interface 簡稱)
    - 實例：經由中央氣象局的氣象開放平臺 API 來下載海象監測資料
    - 專家講解： https://tw.alphacamp.co/blog/api-introduction-understand-web-api-http-json
  - B. 寫網路爬蟲處理靜態網頁 (static web page)
    - 實例：港灣環境資訊網的港區海象之定點歷線圖
      - 雖然該網站操作偏向動態 (即和使用者間的互動性較大)，不過因為我們只下載網頁的原始碼 (包含了所有資料)，所以歸類為靜態
    - 靜態網頁和動態網頁的講解： http://pocawsc.edu.hk/~losir/cweb_pdf/cweb10.pdf
  - C. 寫網路爬蟲處理動態網頁 (dynamic web page)
    - 實例：模擬真人操作瀏覽器來和 windy.com 互動，以獲得全台各電廠出海口 gps 座標點的海象資料
    - 特別說明：動態網頁和靜態網頁最大的區別是動態網站的資料必須透過互動才能呈現，而靜態網頁可以直接透過網址找到資料

下手寫程式前的評估

- 一般來說，若網站有提web api，就以它為優先考量，除非有所限制
  - 透過api獲得的資料一般會以結構化的方式存儲，方便程式進一步處理
    - 「中央氣象局的氣象開放平臺API提供的海象監測資料」支援xml檔案。另一種常見的檔案格式為json。
- 網頁和使用者的互動性越來越頻繁，從早期的靜態網站發展到現在的動態網站，好在即使是動態網站，仍保留了不少靜態的元素，此為第二優先
  - 即使網站的排版是方便給人看和操作的，所幸經過一些html parser處理後，資料也能結構化排列，方便程式進一步擷取
  - 在Python程式中，一般會先以requests模組下載網頁的原始html資料，之後再藉由beautiful soup4模組將原本雜亂的資料結構化
    - 當初處理「港灣環境資訊網的港區海象之定點歷線圖 」時只用到了requests模組，並沒有用到beautifulsoup4模組，反而是用到了更底層的re模組(Regular Expression，正規表示式)，原因是該網頁沒將原始資料以html方式顯現
- 最難處理的是包含過多動態元素的網頁，只能透過模仿人操作瀏覽器的方式方能讓網頁顯示資料，這是最終的解法
  - 一般藉由selenium模組和安裝瀏覽器的web driver來實現，從「windy.com」下載的資料就是使用這種方式，不過資料偶爾不穩。

Web API 回覆的資料對程式最有善


```
currentDirectionDescription><currentSpeed>0.80</currentSpeed><currentSpeedInKnots>1.55</currentSpeedInKnots></layer></seaCurrents></weatherElements></-
stationObsTime><stationObsTime><dataTime>2021-04-14T00:00:00+08:00</dataTime><weatherElements><tideHeight>None</tideHeight><tideLevel>-</tideLevel><waveHeight>1.5</waveHeight><waveDirection>90.0</-
waveDirection><waveDirectionDescription>E</waveDirectionDescription><wavePeriod>5.0</wavePeriod><seaTemperature>19.7</seaTemperature><temperature>None</temperature><stationPressure>1016.4</-
stationPressure><primaryAnemometer><windSpeed>11.5</windSpeed><windScale>6</windScale><windDirection>42.0</windDirection><windDirectionDescription>NE</windDirectionDescription><maximumWindSpeed>13.9</-
maximumWindSpeed><maximumWindScale>7</maximumWindScale></primaryAnemometer><seaCurrents><layer><layerNumber>1</layerNumber><currentDirection>273</currentDirection><currentDirectionDescription>W</-
currentDirectionDescription><currentSpeed>0.68</currentSpeed><currentSpeedInKnots>1.31</currentSpeedInKnots></layer></seaCurrents></weatherElements></-
stationObsTime><stationObsTime><dataTime>2021-04-13T23:00:00+08:00</dataTime><weatherElements><tideHeight>None</tideHeight><tideLevel>-</tideLevel><waveHeight>1.7</waveHeight><waveDirection>78.0</-
waveDirection><waveDirectionDescription>ENE</waveDirectionDescription><wavePeriod>4.9</wavePeriod><seaTemperature>19.7</seaTemperature><temperature>None</temperature><stationPressure>1016.9</-
stationPressure><primaryAnemometer><windSpeed>9.7</windSpeed><windScale>5</windScale><windDirection>43.0</windDirection><windDirectionDescription>NE</windDirectionDescription><maximumWindSpeed>12.8</-
maximumWindSpeed><maximumWindScale>6</maximumWindScale></primaryAnemometer><seaCurrents><layer><layerNumber>1</layerNumber><currentDirection>283</currentDirection><currentDirectionDescription>WNW</-
currentDirectionDescription><currentSpeed>0.46</currentSpeed><currentSpeedInKnots>0.90</currentSpeedInKnots></layer></seaCurrents></weatherElements></stationObsTime></stationObsTimes></location></-
```


```python
files_path = FOLDER_XML + '*.xml'
filenames_source = glob.glob(files_path)
for filename_source in filenames_source:
    filename_raw = os.path.basename(filename_source).split('.')[0]
    filename_csv = f'{FOLDER_CSV}{filename_raw}.csv'

    with open(filename_source, mode='rb') as f:
        # get a list of 30-day records & every record is a dict with hierarchy
        my_dict = xmltodict.parse(f)
        a = my_dict['cwbdata']['resources']['resource']['data']['seaSurfaceObs']
        records_by_month = a['location']['stationObsTimes']['stationObsTime']

    #  get header names from any day
    csv_columns = list(flatten_dict(records_by_month[0]).keys())

    try:
        with open(filename_csv, 'wt', newline='') as csvfile:
            writer = csv.DictWriter(csvfile, fieldnames=csv_columns)
            writer.writeheader()
            # records_by_month[0] = the latest (newest) record
            for record in records_by_month[::-1]:
                writer.writerow(flatten_dict(record))
    except IOError:
        print("I/O error :", filename_raw)

print('2. convert xml into csv (complete)')
```

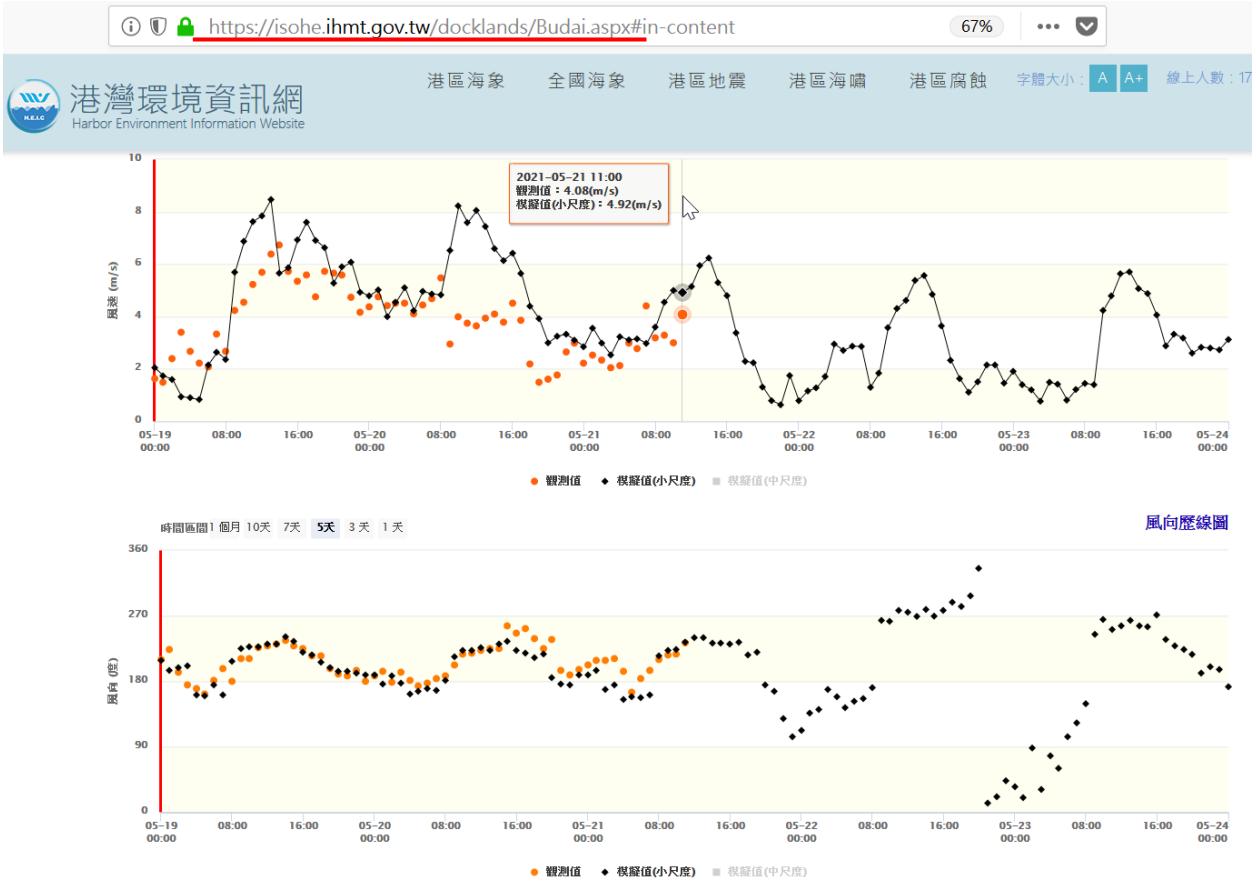
1. 官方網站刻意準備的結構化資料

2. 轉換程式 (根據書本定義，使用 api 不叫爬蟲)

3. 處理好的 csv 檔案

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	dataTime	tideHeight	tideLevel	waveHeight	waveDirection	waveDirectionDescription	wavePeriod	seaTemperature	temperature	stationPressure	windSpeed	windScale	windDirection	windDirectionDescription	maximumWindSpeed	maximumWindScale	layerNumber
2	2021-04-13T23:00:00+08:00	None	-	1.7	78	ENE	4.9	19.7	None	1016.9	9.7	5	43	NE	12.8	6	1
3	2021-04-14T00:00:00+08:00	None	-	1.5	90	E	5	19.7	None	1016.4	11.5	6	42	NE	13.9	7	1
4	2021-04-14T01:00:00+08:00	None	-	1.2	78	ENE	4.7	19.3	None	1016.2	10.4	5	43	NE	13.2	6	1
5	2021-04-14T02:00:00+08:00	None	-	1.2	67	ENE	4.8	19.1	None	1015.4	10.8	6	46	NE	13.8	6	1
6	2021-04-14T03:00:00+08:00	None	-	0.9	90	E	4.5	19.1	None	1015.4	10.1	5	39	NE	12.5	6	1
7	2021-04-14T04:00:00+08:00	None	-	1	78	ENE	4.9	19.1	None	1015.2	11.1	6	44	NE	13.4	6	1

網頁呈現的畫面是給人看的



反而不利程式收集結構化的資料

網頁的原始資料透過爬蟲轉換成資訊


```
{
    "datetime": '2021-05-21 10:00:00',
    "year": '2021',
    "MONTH_": '5',
    "day": '21',
    "HOUR": '10',
    "minute": '0',
    "wd" : '217.5',
    "ws" : '3'


,

{
    "datetime": '2021-05-21 11:00:00',
    "year": '2021',
    "MONTH_": '5',
    "day": '21',
    "HOUR": '11',
    "minute": '0',
    "wd" : '233',
    "ws" : '4.08'


,

{
    "datetime": '2021-05-21 12:00:00',
    "year": '2021',
    "MONTH_": '5',
    "day": '21',
    "HOUR": '12',
    "minute": '0',
    "wd" : '232.4',
    "ws" : '4.93'
```


1. 未加工過的網頁原始文字

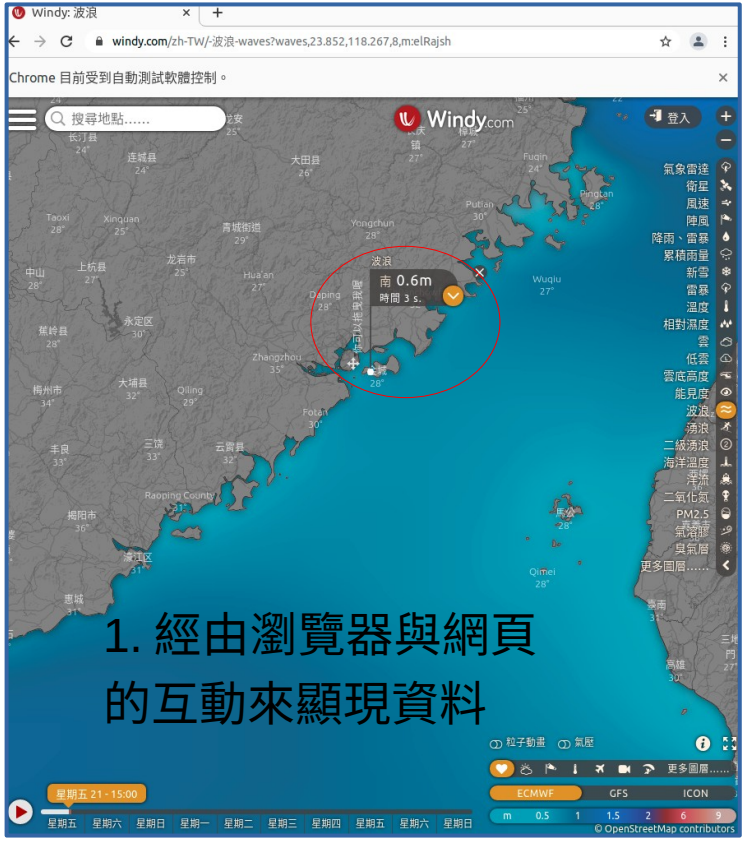
```
# handle re object without saving filtered-raw-data
raw = primitive.group(1)
tmp2 = raw.replace('\'','"') # replace ' with "

info = json.loads(tmp2) # got python object(list with dictionaries)


# save modified info as json file
filename_modifiedInfo = '{}/{}_info_{}.json'.format(folder,place,var)
with open(filename_modifiedInfo, mode='wt', encoding='utf-8') as f:
    #f.write(json.dumps(info))
    f.write(json.dumps(info, indent=2))

print('Got', filename_modifiedInfo)

# 2021/02/19 restore
# 2020/09/29 remove
#'''
# save as excel file (converted from json)
filename_excel = '{}/{}_info_{}.xlsx'.format(folder,place,var)
conv = Converter()
conv.convert(info, Writer(file=filename_excel)) # info is python object

print('Got', filename_excel)
```

2. 網路爬蟲程式

	A	B	C	D	E	F	G	H
699	2021-05-21 04:00:00	2021	5	21	4	0	193.8	2.13
700	2021-05-21 05:00:00	2021	5	21	5	0	165.2	2.99
701	2021-05-21 06:00:00	2021	5	21	6	0	183.3	2.78
702	2021-05-21 07:00:00	2021	5	21	7	0	194.6	4.41
703	2021-05-21 08:00:00	2021	5	21	8	0	210.1	3.19
704	2021-05-21 09:00:00	2021	5	21	9	0	216.1	3.29
705	2021-05-21 10:00:00	2021	5	21	10	0	217.5	3
706	2021-05-21 11:00:00	2021	5	21	11	0	233	4.08
707	2021-05-21 12:00:00	2021	5	21	12	0	232.4	4.93
708	2021-05-21 13:00:00	2021	5	21	13	0	240.6	5.21
709	2021-05-21 14:00:00	2021	5	21	14	0	242.1	4.79

3.xlsx 檔案

模仿真人操作瀏覽器來騙取資料



	A	B	C	D
1	time	海洋溫度	洋流	波浪
2	2021/05/21 15:18:03	26°C	東北 0.6 kt	南 0.6m 時間 3 s.
3				

3.csv 檔案

```
# 顯示波浪
WAVES_SHOW_CSS = '#overlay > a[data-do="set,waves"]'
waves_show = browser.find_element_by_css_selector(WAVES_SHOW_CSS)
waves_show.click()
time.sleep(AFTER_CLICK)
print(show_box.text)
info = show_box.text.split('\n')
record[info[0]] = ' '.join(info[1:])

folders = 'output/' + gps_name + '/'
os.makedirs(folders, exist_ok=True)


with open(folders + filename_csv, 'a', newline='', encoding='utf-8') as f_a:
    fieldnames = ['time', '海洋溫度', '洋流', '波浪']
    #fieldnames = ['time', 'Sea temperature', 'Currents', 'Waves'] # docker
    writer = csv.DictWriter(f_a, fieldnames=fieldnames)
    writer.writeheader() # 第一次寫入檔案時使用
    writer.writerow(record)
```

2. 網路爬蟲程式是操控瀏覽器的幕後黑手

網站擷取：使用 Python （二版）

- 解析複雜HTML網頁
- 以Scrapy架構開發爬行程序
- 學習爬行資料的儲存方式
- 從文件讀取與提煉資料
- 清理格式不良的資料
- 以自然語言讀寫
- 透過表單與登入的爬行
- JavaScript與API爬行
- 使用影像文字識別軟體
- 避開爬行陷阱與機器人阻擋程序
- 使用爬行程序測試你的網站