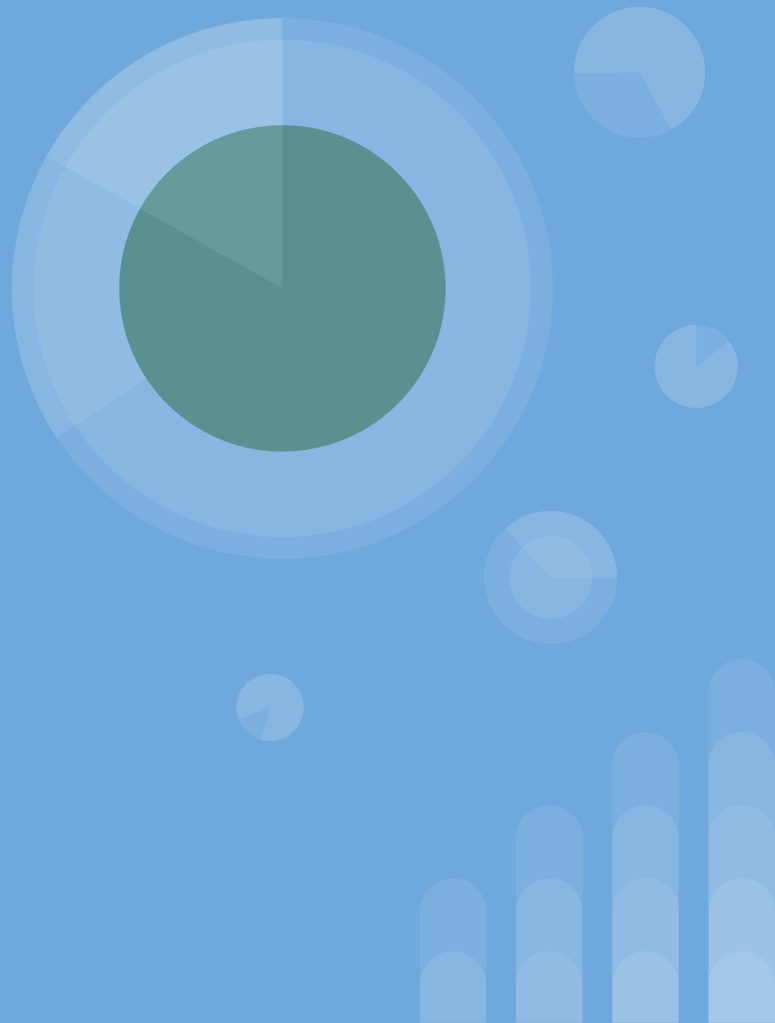


# 民生公共物聯網資料應用研習工作坊

資料取用



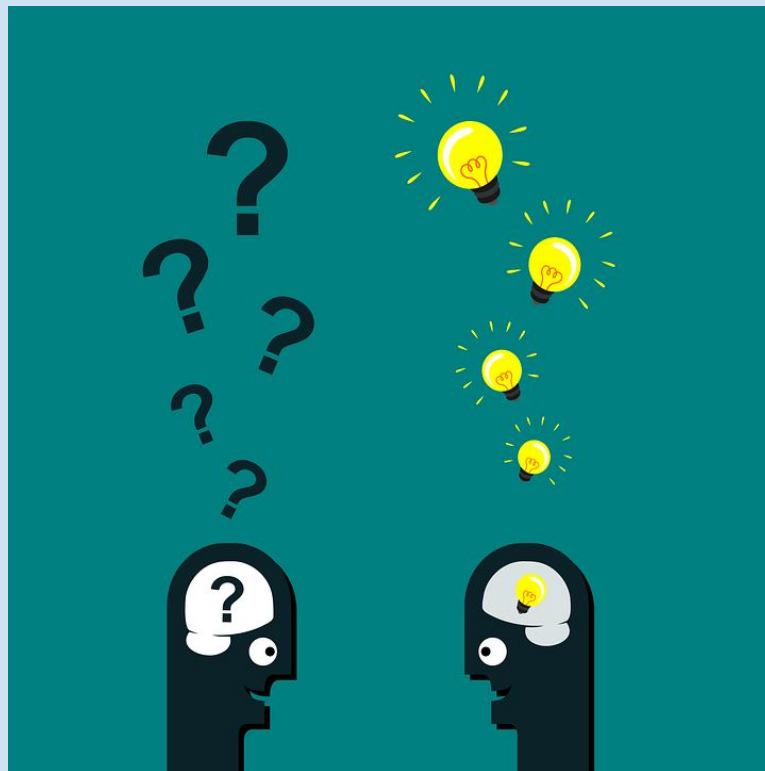
1. 開頭 大概說一下這段落包含重點(資料取用)
2. 帶到一部分歷史(?)
3. 民生公共物聯網介紹  
水/空/地/災資料
4. 民生公共物聯網資料服務平台(sensor things API)  
使用狀況+歷史  
(STA範例)
5. 本段落重點"資料取用 介紹我們的工具 使用Python語法 直接取用開放資料平台的資料"  
基本python操作+安裝lib  
使用colab+介紹
6. 基本資料存取方法  
get\_source() / get\_station() / get\_data()  
分別項的組織架構圖/對照表
7. 存取特定時空條件
8. 探索式資料分析 (Exploratory Data Analysis, EDA)+繪製簡易的圖表



## 章節介紹

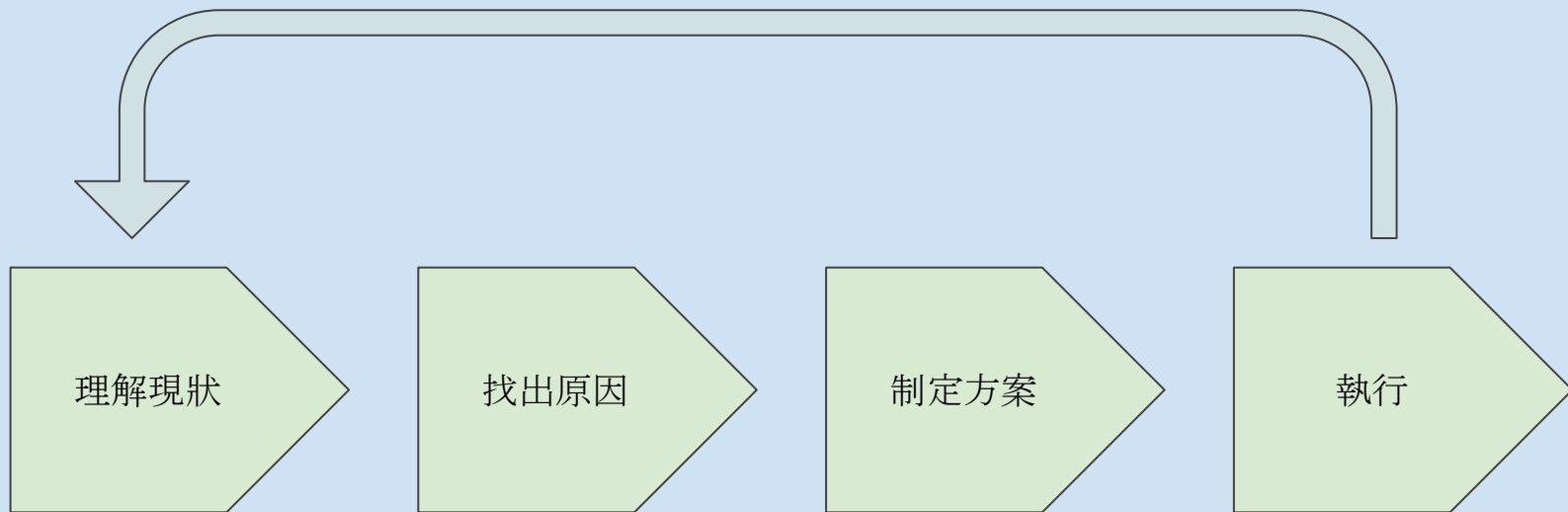
1. 資料的來源與種類
2. 民生公共物聯網資料服務平台
3. 資料工具- Google Colab與 pyCIOT API
4. 基本資料存取方法
5. 存取特定時空條件的資料

# 我們要做什麼



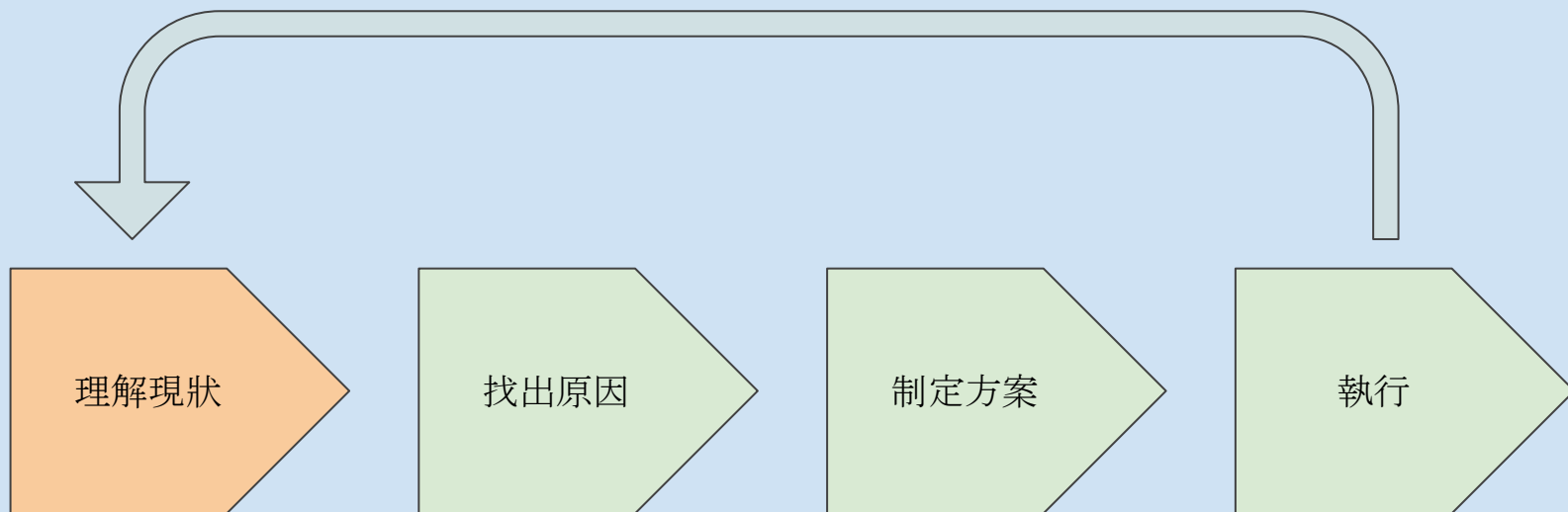


# 解決問題





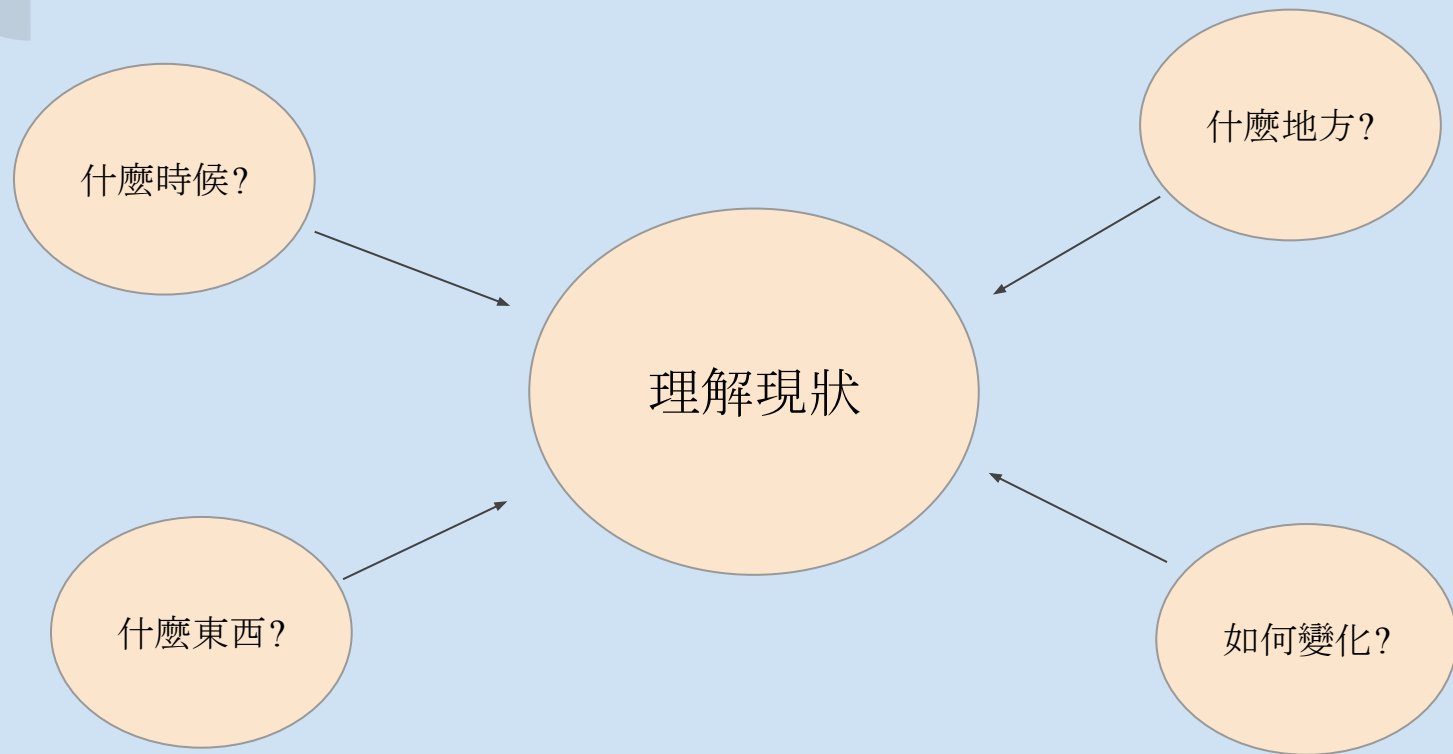
# 解決問題



簡單來說

我們想知道，**現在、這裡**發生什麼事

# 如何解決問題





我們需要資料



# 資料從哪裡來

感測器

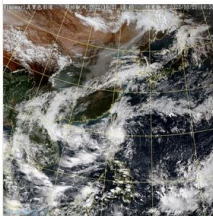
開放資料

各部門公開檔案

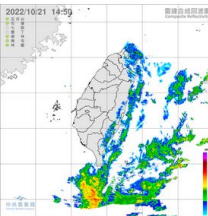


## 圖資專區

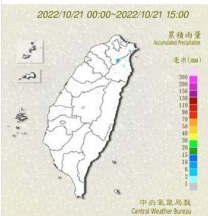
衛星



雷達



雨量



紫外線



即時閃電



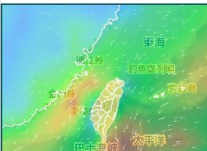
溫度



健康氣象



風場預報



## 觀測資料查詢 CODIS CWB Observation Data Inquire System

服務系統，於2022年9月6日正式上線，可由本局官網「資料」

全臺 Nationwide 北部地區 North Area 中部地區 Central Area 南部地區 South Area 東部地區 East Area 離島地區 Islands 農業 氣象

測站所在縣市:

測站:

資料類型:

資料格式:

時間:

[查詢](#) [取消](#)

[氣象局空氣品質監測網](#) [觀測雨量資料查詢](#) [站況查詢](#)

[觀測雨量資料查詢](#) [觀測雨量資料查詢](#)

[更新時間每日12:00 \(Updated Time: 12:00\)](#)

測站資訊 (station information)

測站 ANBU (466910)

經度: 121.5297

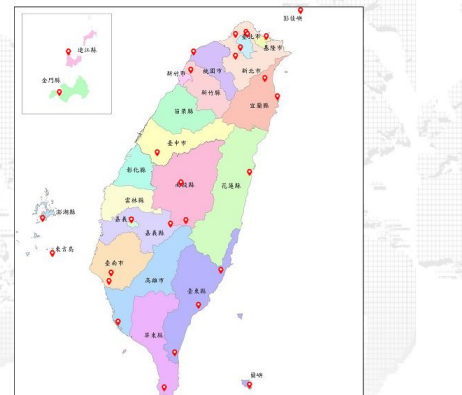
緯度: 25.1825

海拔高度: 837.5m

建站日期: 1937/01/01 ~

地址: 臺北市北投區南門山竹子湖路111號

備註:



## 空氣品質監測網

[網站導覽](#) [空氣監測](#) [任務監測](#) [空氣預報](#) [作業規範](#) [空氣科普](#) [EN](#)

[首頁](#) > [資料下載](#)

## 資料下載



近兩年小時值查詢

不良日數月報表  
(AQI)

污染物測值月報表

月平均值查詢

歷年監測資料

光化監測資料

其他環境資料

## 全國環境水質監測資訊網

[二](#) [首頁](#) | [網站導覽](#) | [English](#)





## 資料從哪裡來

非常好！

這樣問題就解決了？



# 資料的來源與種類

## 水資源相關資料集

歷史資料下載

提供單位	項目	數量	更新頻率	起始時間	API 網址	歷史資料
水利署	河川水位站	265 站	10 分鐘	2012 年 1 月	<a href="#">API 網址</a>	○
	雨量感測器	8 站	10 分鐘	2019 年 ~ 迄今	<a href="#">API 網址</a>	○
	非連續性淹水感測器	1 站	10 分鐘	2018 年 ~ 迄今	<a href="#">API 網址</a>	○
	淹水感測器	245	10 分鐘	2018 年 ~ 迄今	<a href="#">API 網址</a>	○
	堤防結構安全測站	29 站	10 分鐘	2018 年 ~ 迄今	<a href="#">API 網址</a>	○
	閘門	62 站	10 分鐘	2019 年 ~ 迄今	<a href="#">API 網址</a>	○
	地下水位站	733 站	10 分鐘	2021 年 ~ 迄今	<a href="#">API 網址</a>	○
農田水利署	流量感測器	12 站	10 分鐘	2019 年 ~ 迄今	<a href="#">API 網址</a>	○
	埤塘水位站	107 站	10 分鐘	2018 年 ~ 迄今	<a href="#">API 網址</a>	○
	農田灌溉圳路水位站	32 站	10 分鐘	2018 年 ~ 迄今	<a href="#">API 網址</a>	○
	閘門	2 站	10 分鐘	2019 年 ~ 迄今	<a href="#">API 網址</a>	○

水利署（與縣市政府合建）	雨量感測器	85 站	10 分鐘	2012 年 1 月	<a href="#">API 網址</a>	○
	非連續性淹水感測器	4 站	10 分鐘	2018 年 ~ 迄今	<a href="#">API 網址</a>	○
	流量感測器	211 站	10 分鐘	2019 年 ~ 迄今	<a href="#">API 網址</a>	○
	區域排水水位站	258 站	10 分鐘	2019 年 ~ 迄今	<a href="#">API 網址</a>	○
	淹水感測器	1519 站	10 分鐘	2018 年 ~ 迄今	<a href="#">API 網址</a>	○
	移動抽水機	93 站	10 分鐘	2019 年 ~ 迄今	<a href="#">API 網址</a>	○
	抽水站	100 站	10 分鐘	2020 年 ~ 迄今	<a href="#">API 網址</a>	○
營建署	閘門	64 站	10 分鐘	2019 年 ~ 迄今	<a href="#">API 網址</a>	○
	化學需氧量感測器	42 站	每小時	2020 年 3 月	<a href="#">API 網址</a>	○
	污水放水量感測器	43 站	每小時	2020 年 3 月	<a href="#">API 網址</a>	○
	懸浮固體感測器	42 站	每小時	2020 年 3 月	<a href="#">API 網址</a>	○
臺北市	抽水站	77 站	即時	2020 年 10 月	<a href="#">API 網址</a>	○
環保署	河川水質監測資料	335 站	每月更新	2012 年	NA	○
	區域性地下水水質監測資料	491 站	每季更新	2014 年	NA	○
	水庫水質監測資料	151 站	不定期更新	1993 年	NA	○



# 資料的來源與種類

## 空氣品質相關資料集

歷史資料下載

提供單位	項目	數量	更新頻率	起始時間	API 網址	歷史資料
環保署	國家空品測站	77 站	每小時	1998 年	<a href="#">API 網址</a>	○
	智慧城鄉空品微型感測器	10,496 站	3 分鐘	2017 年 6 月	<a href="#">API 網址</a>	○
	空品監測即時影像器	64 站	10 分鐘	NA	<a href="#">API 網址</a>	X
中研院	校園空品微型感測器	3233 站	5 分鐘	2017 年 9 月	<a href="#">API 網址</a>	○
國家科學及技術委員會	智慧園區空品測站	20 站	每小時	2018 年 9 月	<a href="#">API 網址</a>	○
大同股份有限公司	大同空品微型感測器	500 站	每 10 分鐘	2019 年 10 月	NA	○
暨南大學	在地空品微型感測器 (PM2.5 十分鐘平均值)	200 站	每 10 分鐘	2019 年 12 月	<a href="#">API 網址</a>	○
台固	台固空品微型感測器	500 站	每 3 分鐘	2020 年 01 月	NA	X



## 資料的來源與種類

所以我們有：

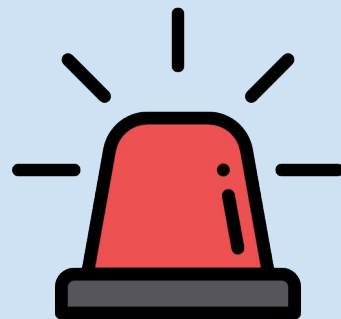
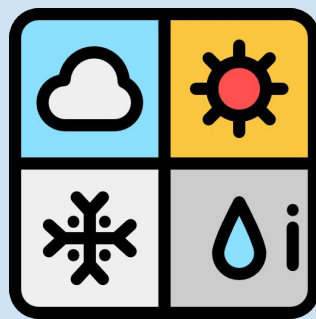
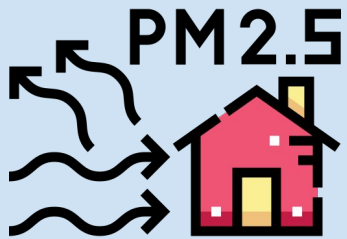




## 資料的來源與種類

民生公共物聯網:

為整合與貼近民生公共相關服務，跨部會協調與提供了水資源、空氣品質、地震、天氣，以及災防資訊等資料。





# 民生公共物聯網資料服務平台

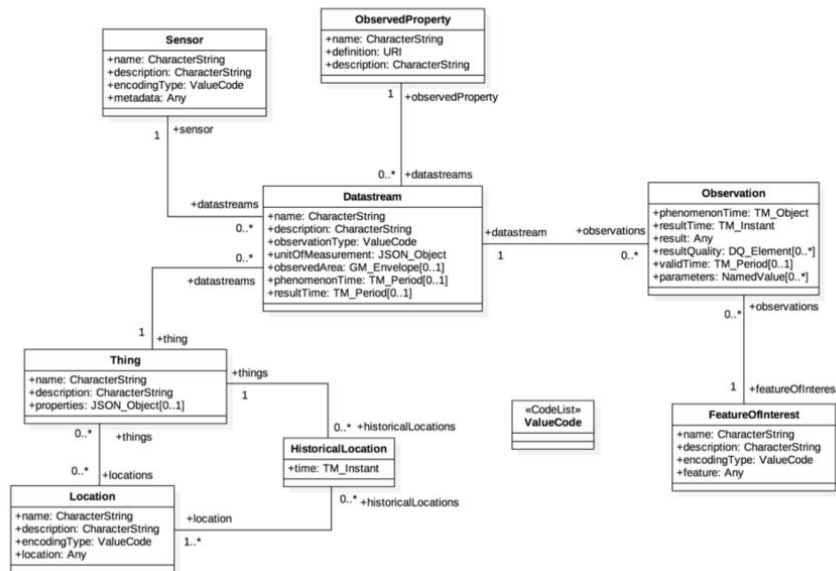


# 民生公共物聯網資料服務平台

## SensorThings API

### 觀測資料描述架構

有鑒於感測物聯網網路服務開放式標準中 OGC 國際標準制定組織所提出之 SWE 系列標準受到國內外許多相關單位之肯定及依循，本作業規範遵循 OGC SensorThings API 標準所制定的觀測資料模型，進行空氣品質、水位、地震以及 CCTV 等觀測資料的描述。下圖為 OGC 定義的 SensorThings API 資料模型。



\_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_ /  
 |                    |                    |  
 service root URI    resource path    query options



# 民生公共物聯網資料服務平台

範例:

找出特定範圍裡的空氣品質資料

[https://sta.ci.taiwan.gov.tw/STA\\_AirQuality/v1.0/Locations?\\$expand=Things&\\$filter=geo.intersects\(Locations/location,geography'POLYGON\(\(\(120.9 25.3,120.8 24.7,121.9 24.6,122.1 25.3,120.9 25.3\)\)\)'\)&\\$count=true](https://sta.ci.taiwan.gov.tw/STA_AirQuality/v1.0/Locations?$expand=Things&$filter=geo.intersects(Locations/location,geography'POLYGON(((120.9 25.3,120.8 24.7,121.9 24.6,122.1 25.3,120.9 25.3)))')&$count=true)

`http://example.org/v1.0/Observations?$orderby=ID&$top=10`

\_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_ /

service root URI

resource path

query options



# 民生公共物聯網資料服務平台

範例:

找出特定範圍裡的空氣品質資料

[https://sta.ci.taiwan.gov.tw/STA\\_AirQuality/v1.0/Locations?\\$expand=Things&\\$filter=geo.intersects\(locations/location-geometry/POLYGON\(\(120.9253,120.8247,121.0246,122.125.3,120.9253\)\)&\\$top=10](https://sta.ci.taiwan.gov.tw/STA_AirQuality/v1.0/Locations?$expand=Things&$filter=geo.intersects(locations/location-geometry/POLYGON((120.9253,120.8247,121.0246,122.125.3,120.9253))&$top=10)

聽起來不錯，對吧？

`http://example.org/v1.0/Observations?$orderby=ID&$top=10`

\_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_ /

service root URI

resource path

query options



# 資料工具- Google Colab與 pyCIOT API

pyCIOT API :

政府開放資料有非常多種資料及管道讓我們查詢，而不同的窗口有各自不同的資料存取方式。即便這些資料使用開放授權，但進行資料搜集時，因為下載方式各有差異，資料整理將變得極為麻煩。

本函式庫為了解決窗口不一的問題，將所有民生開放資料搜集至此讓開放資料獲取門檻降低，減少資料處理成本與方便自動化。



# 資料工具- Google Colab與 pyCIOT API

pyCIOT API :

我們將各資料整合為幾個工具, 讓使用者快速直觀的使用

[https://sta.ci.taiwan.gov.tw/STA\\_AirQuality/v1.0/Locations?\\$expand=Things&\\$filter=geo.intersects\(Locations/location,geography'POLYGON\(\(\(120.9 25.3,120.8 24.7,121.9 24.6,122.1 25.3,120.9 25.3\)\)\)'\)&\\$count=true](https://sta.ci.taiwan.gov.tw/STA_AirQuality/v1.0/Locations?$expand=Things&$filter=geo.intersects(Locations/location,geography'POLYGON(((120.9 25.3,120.8 24.7,121.9 24.6,122.1 25.3,120.9 25.3)))')&$count=true)



```
Air().get_data(src="OBS:EPA_IoT", location = loc)
```



# 資料工具- Google Colab與 pyCIOT API

## 程式語言 - Python

本課程將以目前資料科學界所常用的程式語言 Python 為主要程式語言, 利用淺顯易懂的方式  
逐步進入各章節

```
>>> print("Hello World!")  
Hello World!  
>>>
```



# 資料工具- Google Colab與 pyCIOT API

歡迎使用 Colaboratory

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 無法儲存變更

目錄

開始使用

數據科學

機器學習

其他資源

主要範例

區段

歡迎使用 Colab !

如果你已經熟悉 Colab，請觀看這部影片瞭解互動式表格、執行過的程式碼歷史記錄檢視畫面，以及指令區塊面板。

3 Cool Google Colab Features

```
print("hello world")
```

### Colab 是什麼？

Colab (全名為「Colaboratory」) 可讓你在瀏覽器中編寫及執行 Python 程式碼，並具有以下優點：

- 不必進行任何設定
- 免付費使用 GPU
- 輕鬆共用

無論你是學生、數據資料學家或是 AI 研究人員，Colab 都能讓你的工作事半功倍。請觀看 [Colab 的簡介影片](#) 瞭解詳情，或是直接瀏覽以下的新手入門說明！

CC BY



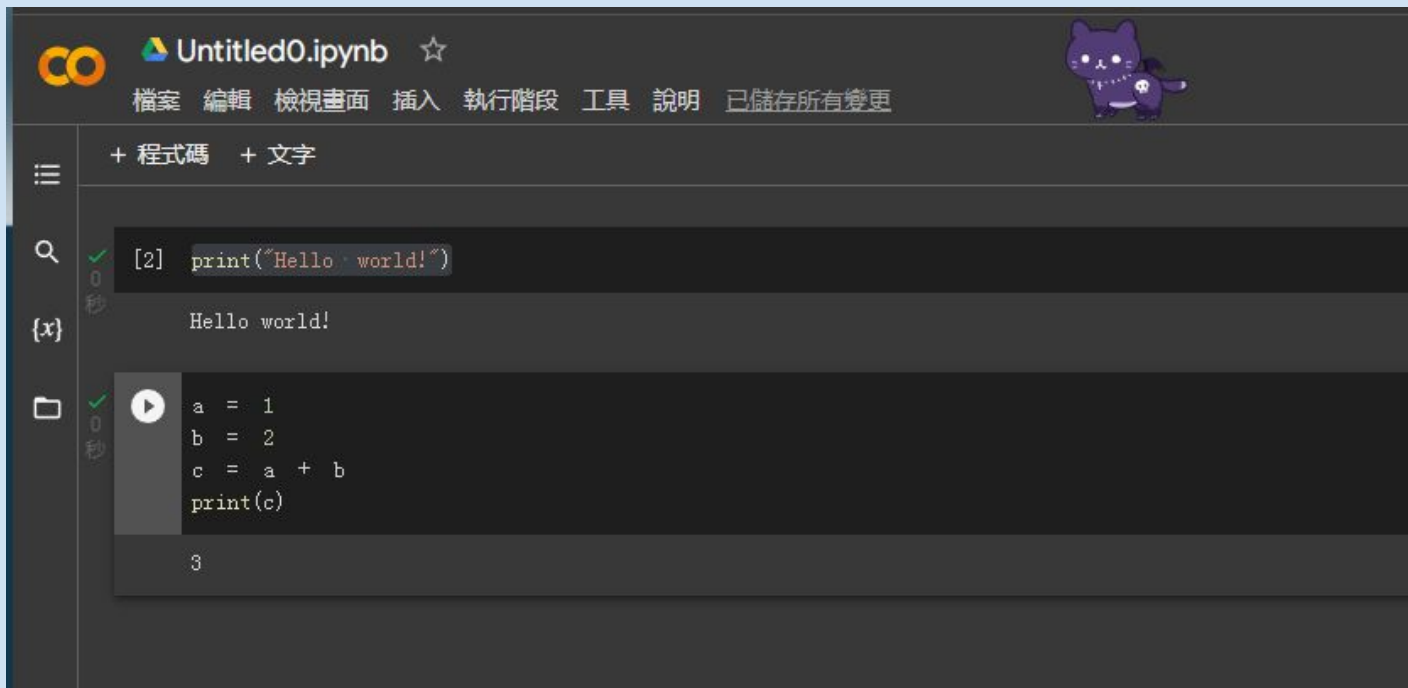
# 資料工具- Google Colab與 pyCIOT API

程式語言 - Python

第一支程式

The screenshot shows the Google Colab web interface. At the top, there's a header with the Colab logo, the file name 'Untitled0.ipynb', and a star icon. Below the header is a menu bar with options: '檔案' (File), '編輯' (Edit), '檢視畫面' (View), '插入' (Insert), '執行階段' (Runtime), '工具' (Tools), '說明' (Help), and '已儲存所有變更' (Save all changes). On the right side of the header is a purple cat icon. The main area has a sidebar on the left with icons for a menu, search, a list of cells, and a file explorer. The central workspace shows a single code cell. The cell's toolbar has a play button, a checkmark, and a timer showing '0 秒'. The code inside the cell is `print("Hello world!")`. Below the code, the output 'Hello world!' is displayed.

# 資料工具- Google Colab與 pyCIOT API





# 基本資料存取方法

請搭配此連結使用





# 基本資料存取方法

## pyCIOT說明

大部分的 API 資料都能夠透過下列方式獲取，包含空氣、水源、地震、天氣、影像資料等：

- `.get_source()` 回傳在民生公共物聯網開放資料平台中有的專案代碼，格式為 list
- `.get_station(src="")` 回傳各測站資料的基本資訊及位置，格式為 list  
可選擇性帶入 `src` 參數，指名要查詢的專案。
- `.get_data(src="", stationID="")` 回傳各測站的資訊，位置及感測資料，格式為 list  
可選擇性帶入 `src` 參數，指名所要查詢的專案代碼，或選擇性帶入 `stationIds` 參數，指名所要查詢的機器代碼。

災害警示資料則適用於以下：

- `.get_alert()` 獲取警示資料，返回格式為 json，包括該事件的相關示警資訊。
- `.get_notice()` 獲取通報歷史資料，返回格式為 json，包括該事件的相關通報資訊。



# 基本資料存取方法

功能說明：

Air()

Water()

Weather()

Quake()

CCTV()



.get\_source()

.get\_station()

.get\_data()

---

Disaster()



.get\_alert()

.get\_notice()



# 基本資料存取方法

## 安裝 pyCIOT

```
!pip install pyCIOT
```

```
+ 程式碼 + 文字

!pip install pyCIOT

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyCIOT
  Downloading pyCIOT-1.0.0-py3-none-any.whl (25 kB)
Collecting requests==2.28.0
  Downloading requests-2.28.0-py3-none-any.whl (62 kB)
| 62 kB 1.0 MB/s
Collecting xmltodict==0.13.0
  Downloading xmltodict-0.13.0-py2.py3-none-any.whl (10.0 kB)
Collecting charset-normalizer~2.0.0
  Downloading charset_normalizer-2.0.12-py3-none-any.whl (39 kB)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests==2.28.0->pyCIOT) (2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests==2.28.0->pyCIOT) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests==2.28.0->pyCIOT) (1.24.3)
Installing collected packages: charset-normalizer, xmltodict, requests, pyCIOT
  Attempting uninstall: charset-normalizer
    Found existing installation: charset-normalizer 2.1.1
    Uninstalling charset-normalizer-2.1.1:
      Successfully uninstalled charset-normalizer-2.1.1
  Attempting uninstall: requests
    Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
Successfully installed charset-normalizer-2.0.12 pyCIOT-1.0.0 requests-2.28.0 xmltodict-0.13.0
```





# 基本資料存取方法

匯入 pyCIOT 函式庫

```
from pyCIOT.data import *
```

取得專案相關代號

```
Air().get_source()
```

- OBS:EPA: 環保署國家空品測站
- OBS:EPA\_IoT: 環保署智慧城鄉空品微型感測器
- OBS:AS\_IoT: 中研院校園空品微型感測器
- OBS:MOST\_IoT: 科技部智慧園區空品測站
- OBS:NCNU\_IoT: 暨南大學在地空品微型感測器



```
# Import pyCIOT.data
from pyCIOT.data import *
# 回傳所有空氣相關的專案代碼
a = Air().get_source()
print(a)
|
```

```
[ 'OBS:EPA', 'OBS:EPA_IoT', 'OBS:AS_IoT', 'OBS:MOST_IoT', 'OBS:NCNU_IoT' ]
```

PM2.5







# 基本資料存取方法

## 取得感測站列表

```
Air().get_station()
```

```
✓ 0.0s # 獲取"環保署智慧城鄉空品微型感測站"列表
b = Air().get_station(src="OBS:EPA")

# 只顯示前面3筆資料
b[0:3]

[[{'name': '空氣品質測站-基隆',
  'description': '空氣品質測站-基隆',
  'properties': {'city': '基隆市',
    'areaName': '北部空品區',
    'township': '信義區',
    'authority': '行政院環境保護署',
    'stationID': 'EPA001',
    'englishName': 'Keelung',
    'stationName': '基隆',
    'stationType': '一般測站'},
  'location': {'latitude': 25.1292,
    'longitude': 121.7601,
    'address': '基隆市信義區東信路324號'}},
 {'name': '空氣品質測站-汐止',
  'description': '空氣品質測站-汐止',
  'properties': {'city': '新北市',
    'areaName': '北部空品區',
    'township': '汐止區',
    'authority': '行政院環境保護署',
    'stationID': 'EPA002',
    'englishName': 'Xizhi',
    'stationName': '汐止',
    'stationType': '一般測站'},
  'location': {'latitude': 25.0657,
    'longitude': 121.6408,
    'address': '新北市汐止區樟樹一路137巷26號'}},
 ...]]
```

```
{'name': '空氣品質測站-基隆',
 'description': '空氣品質測站-基隆',
 'properties': {'city': '基隆市',
  'areaName': '北部空品區',
  'township': '信義區',
  'authority': '行政院環境保護署',
  'stationID': 'EPA001',
  'englishName': 'Keelung',
  'stationName': '基隆',
  'stationType': '一般測站'},
 'location': {'latitude': 25.1292,
  'longitude': 121.7601,
  'address': '基隆市信義區東信路 324號'}}
```





# 基本資料存取方法

## 取得特定感測站資料

```
Air().get_data()
```

```
# 獲取測站資料 id:11613429495
d = Air().get_data(src="OBS:EPA_IoT", stationID="11613429495")
d

[{'name': '智慧城鄉空品微型感測器-11613429495',
  'description': '智慧城鄉空品微型感測器-11613429495',
  'properties': {'city': '新竹市',
                  'areaType': '一般社區',
                  'isMobile': 'false',
                  'township': '香山區',
                  'authority': '行政院環境保護署',
                  'isDisplay': 'true',
                  'isOutdoor': 'true',
                  'stationID': '11613429495',
                  'locationId': 'HC0154',
                  'Description': 'AQ1001',
                  'areaDescription': '新竹市香山區'},
  'data': [{'name': 'Relative humidity',
            'description': '相對溼度',
            'values': [{'timestamp': '2022-10-21T12:46:06.000Z', 'value': 100}]},
            {'name': 'PM2.5',
             'description': '細懸浮微粒 PM2.5',
             'values': [{'timestamp': '2022-10-21T12:46:06.000Z', 'value': 10.5}]},
            {'name': 'Temperature',
             'description': '溫度',
             'values': [{'timestamp': '2022-10-21T12:46:06.000Z', 'value': 24.81796}]},
            {'name': 'Location',
             'description': '位置',
             'values': [{'timestamp': '2022-10-21T12:46:06.000Z', 'value': {'latitude': 24.81796, 'longitude': 120.92664, 'address': '新竹市香山區'}}]}]}
```

```
{'name': '智慧城鄉空品微型感測器-11613429495',
  'description': '智慧城鄉空品微型感測器-11613429495',
  'properties': {'city': '新竹市',
                  'areaType': '一般社區',
                  'isMobile': 'false',
                  'township': '香山區',
                  'authority': '行政院環境保護署',
                  'isDisplay': 'true',
                  'isOutdoor': 'true',
                  'stationID': '11613429495',
                  'locationId': 'HC0154',
                  'Description': 'AQ1001',
                  'areaDescription': '新竹市香山區'},
  'data': [{'name': 'Relative humidity',
            'description': '相對溼度',
            'values': [{'timestamp': '2022-10-21T12:46:06.000Z', 'value': 100}]},
            {'name': 'PM2.5',
             'description': '細懸浮微粒 PM2.5',
             'values': [{'timestamp': '2022-10-21T12:46:06.000Z', 'value': 10.5}]},
            {'name': 'Temperature',
             'description': '溫度',
             'values': [{'timestamp': '2022-10-21T12:46:06.000Z', 'value': 24.81796}]},
            {'name': 'Location',
             'description': '位置',
             'values': [{'timestamp': '2022-10-21T12:46:06.000Z', 'value': {'latitude': 24.81796, 'longitude': 120.92664, 'address': '新竹市香山區'}}]}]}
```



# 基本資料存取方法

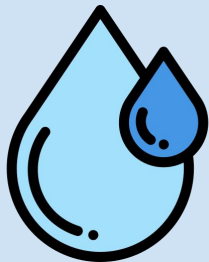
取得專案相關代號

```
Water().get_source()
```

- WRA: 水利署
- WRA2: 水利署(與縣市政府合建)
- IA: 農田水利署
- CPAMI: 營建署
- TPE: 臺北市

```
# 回傳所有水相關的專案代碼
wa = Water().get_source()
wa

[ 'WATER_LEVEL:WRA_RIVER',
  'WATER_LEVEL:WRA_GROUNDWATER',
  'WATER_LEVEL:WRA2_DRAINAGE',
  'WATER_LEVEL:IA_POND',
  'WATER_LEVEL:IA_IRRIGATION',
  'GATE:WRA',
  'GATE:WRA2',
  'GATE:IA',
  'PUMPING:WRA2',
  'PUMPING:TPE',
  'FLOODING:WRA',
  'FLOODING:WRA2' ]
```



# 基本資料存取方法

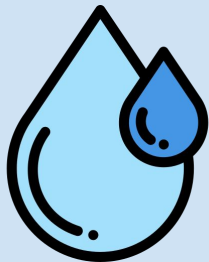
## 取得感測站列表

```
Air().get_station()
```

```
✓ 1秒 # 獲取水利署所有測站列表
wa = Water().get_station(src="WATER_LEVEL:WRA_RIVER")
# 顯示第一到第三筆
wa[0:3]

[{'name': '01790145-cd7e-4498-9240-f0fcd9061df2',
 'description': '現場觀測',
 'properties': {'authority': '水利署水文技術組',
 'stationID': '01790145-cd7e-4498-9240-f0fcd9061df2',
 'stationCode': '2200H007',
 'stationName': '延平',
 'authority_type': '水利署'},
 'location': {'latitude': 22.8983536,
 'longitude': 121.0845795,
 'address': None}},
 {'name': '030347f1-bc7e-478e-ac00-e066d6b1e0df',
 'description': '現場觀測',
 'properties': {'authority': '水利署水文技術組',
 'stationID': '030347f1-bc7e-478e-ac00-e066d6b1e0df',
 'stationCode': '1140H099',
 'stationName': '思源橋',
 'authority_type': '水利署'},
 'location': {'latitude': 24.9617977,
 'longitude': 121.7689438,
 'address': None}},
```

```
{'name':
'01790145-cd7e-4498-9240-f0fcd9061df2',
 'description': '現場觀測',
 'properties': {'authority': '水利署水文技術組',
 'stationID':
'01790145-cd7e-4498-9240-f0fcd9061df2',
 'stationCode': '2200H007',
 'stationName': '延平',
 'authority_type': '水利署'},
 'location': {'latitude': 22.8983536,
 'longitude': 121.0845795,
 'address': None}},
```



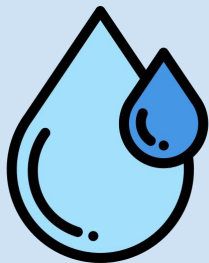
# 基本資料存取方法

## 取得特定感測站資料

```
Water().get_data()
```

```
# 獲取特定測站資料
wa = Water().get_data(src="WATER_LEVEL:WRA_RIVER", stationID="01790145-cd7e-4498-9240-f0fcd9061df2")
wa

[{'name': '01790145-cd7e-4498-9240-f0fcd9061df2',
  'description': '現場觀測',
  'properties': {'authority': '水利署水文技術組',
    'stationID': '01790145-cd7e-4498-9240-f0fcd9061df2',
    'stationCode': '2200H007',
    'stationName': '延平',
    'authority_type': '水利署'},
  'data': [{'name': '水位',
    'description': 'Datastream_id=016e5ea0-7c7f-41a2-af41-eabacdbb613f, Datastream_FullName=延平.水位, Datastream_id=016e5ea0-7c7f-41a2-af41-eabacdbb613f',
    'values': [{'timestamp': '2022-10-21T13:30:00.000Z', 'value': 157.49}]}],
  'location': {'latitude': 22.8983536,
    'longitude': 121.0845795,
    'address': None}}]
```





# 基本資料存取方法

參數介紹:

```
.get_source()
```

```
.get_station(src="")
```

```
.get_data(src="", stationID="")
```

src="專案代碼"

stationID="感測站ID"



# 基本資料存取方法

參數介紹:  
以空氣為例

執行 `Air().get_source()`

列出所有來源



`['OBS:EPA', 'OBS:EPA_IoT', 'OBS:AS_IoT',  
'OBS:MOST_IoT', 'OBS:NCNU_IoT']`

執行 `Air().get_station(src="OBS:EPA")`

找到所有站點



`{'name': '空氣品質測站-基隆',  
'description': '空氣品質測站基隆',  
'properties': {'city': '基隆市',  
'areaName': '北部空品區', 'township': '信義區',  
'authority': '行政院環境保護署',  
'stationID': 'EPA001'.....}`

執行 `Air().get_data(src="OBS:EPA", stationID="EPA001")`

取得目標測站資料



測站資料

# 基本資料存取方法

取得地震專案相關代號

```
Quake().get_source()
```

取得地震監測站列表

```
Quake().get_station()
```

取得地震資料

```
Quake().get_data()
```

```
# 取得地震相關專案代碼
q = Quake().get_source()
q

['EARTHQUAKE:CWB+NCREE']

[ ] # 取得地震監測站列表
q = Quake().get_station(src="EARTHQUAKE:CWB+NCREE")
# 顯示前兩筆

q[0:2]

[{'name': '地震監測站-Jiqi-EGC',
 'description': '地震監測站-Jiqi-EGC',
 'properties': {'authority': '中央氣象局',
 'stationID': 'EGC',
 'deviceType': 'FBA',
 'stationName': 'Jiqi'},
 'location': {'latitude': 23.708, 'longitude': 121.548, 'address': None}},
 {'name': '地震監測站-Xilin-ESL',
 'description': '地震監測站-Xilin-ESL',
 'properties': {'authority': '中央氣象局',
 'stationID': 'ESL',
 'deviceType': 'FBA',
 'stationName': 'Xilin'},
 'location': {'latitude': 23.812, 'longitude': 121.442, 'address': None}}]
```





# 基本資料存取方法

取得天氣專案相關代號

```
Weather().get_source()
```

取得天氣測站列表

```
Weather().get_station()
```

取得天氣資料

```
Weather().get_data()
```

```
[ ] # 取得天氣相關專案代碼
```

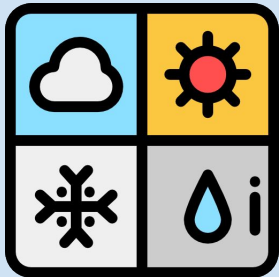
```
w = Weather().get_source()  
w
```

```
['GENERAL:CWB',  
 'GENERAL:CWB_IoT',  
 'RAINFALL:CWB',  
 'RAINFALL:WRA',  
 'RAINFALL:WRA2',  
 'RAINFALL:IA',  
 'IMAGE:CWB']
```

```
▶ # 獲取所有測站列表
```

```
w = Weather().get_station(src="RAINFALL:CWB")  
# 顯示前五筆  
w[0:5]
```

```
⌘ [{ 'name': '雨量站-CIR120-上德文',  
    'description': '雨量站-CIR120-上德文',  
    'properties': { 'city': '屏東縣',  
                    'township': '三地門鄉',  
                    'authority': '中央氣象局',  
                    'stationID': 'CIR120',  
                    'stationName': '上德文',  
                    'stationType': '局屬無人測站'},  
    'location': { 'latitude': 22.765, 'longitude': 120.6964, 'address': None}},
```





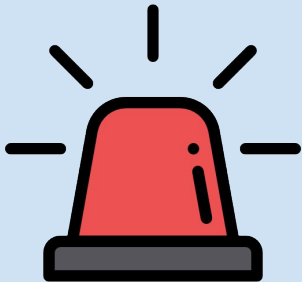
# 基本資料存取方法

## 取得災情示警

```
Disaster().get_alert("代號")
```

## 取得災情通報

```
Disaster().get_notice("代號")
```



```
# 取得降雨災害通報(10)資料
d = Disaster().get_alert("10")
d['entry'][0:3]

[{'id': 'CWB-Weather_extremely-rain_202210150020001',
  'title': '降雨',
  'updated': '2022-10-15T00:25:25+08:00',
  'author': {'name': '中央氣象局'},
  'link': {'@rel': 'alternate',
    '@href': 'https://b-alertsline.cdn.hinet.net/Capstorage/CWB/2022/Weather_warnings_RAIN/fifows_extremely-rain_202210150025.cap'},
  'summary': {'@type': 'html',
    '#text': '\n東北季風影響，今（15）日基隆北海岸及大臺北地區有局部大雨發生的機率，請注意。'},
  'category': {'@term': '降雨'}},
 {'id': 'CWB-Weather_extremely-rain_202210150330001',
  'title': '降雨',
  'updated': '2022-10-15T03:41:33+08:00',
  'author': {'name': '中央氣象局'},
  'link': {'@rel': 'alternate',
    '@href': 'https://b-alertsline.cdn.hinet.net/Capstorage/CWB/2022/Weather_warnings_RAIN/fifows_extremely-rain_202210150341.cap'},
  'summary': {'@type': 'html',
    '#text': '\n熱帶性低氣壓外圍環流影響，今（15）日基隆北海岸、大臺北地區及宜蘭縣山區有局部大雨發生的機率，請注意。'},
  'category': {'@term': '降雨'}},
 {'id': 'CWB-Weather_extremely-rain_202210150515001',
  'title': '降雨',
  'updated': '2022-10-15T05:21:15+08:00',
  'author': {'name': '中央氣象局'},
  'link': {'@rel': 'alternate',
    '@href': 'https://b-alertsline.cdn.hinet.net/Capstorage/CWB/2022/Weather_warnings_RAIN/fifows_extremely-rain_2022101505150015001.cap'},
  'summary': {'@type': 'html'}}
```





# 基本資料存取方法

功能說明:

Air()

Water()

Weather()

Quake()

CCTV()



.get\_source()

.get\_station()

.get\_data()

---

Disaster()



.get\_alert()

.get\_notice()



# 存取特定時空條件的資料

## 獲取特定時間之資料

使用 `.get_data(src="", stationID="")` 時可以再加入參數 `time_range` 來取得特定時間段的資料

```
from datetime import datetime, timedelta

end_date = datetime.now() # 獲取現在時間
isodate_end = end_date.isoformat().split(".")[0]+"Z" # 將格式轉換為 ISO8601 格式
start_date = datetime.now() + timedelta(days = -1) # 獲取前一天的時間
isodate_start = start_date.isoformat().split(".")[0]+"Z" # 將格式轉換為 ISO8601 格式

time = {
    "start": isodate_start,
    "end": isodate_end,
    "num_of_data": 15
}

# 從「智慧城鄉空品微型感測器-11613429495」獲得距離現在一天、最多 15 筆資料
data = Air().get_data("OBS:EPA_IoT", stationID="11613429495", time_range=time)
data
```



# 存取特定時空條件的資料

## 獲取特定時間之資料

參數`time_range` 為一個事先設定好的格式

```
time = {  
  "start": "2022-10-22T00:00:00Z",  
  "end": "2022-10-22T23:59:59Z",  
  "num_of_data": 15  
}
```

`start` 與 `end` 指資料搜集的開始及結束時間  
格式為 ISO8601 或 Datetime。`num_of_data`  
則是會控制獲取資料的筆數不會超過此數字。

例如 `.get_data(src="", stationID="", time_range=time)`

# 存取特定時空條件的資料

執行結果就會如圖

以 List 的格式儲存在 data 中，  
並依照不同種類的數值一起存放。

溫度、相對濕度、PM2.5 的資料會分別  
存放在對應名字下的 'values' list 下  
並標上每筆資料紀錄的時間，  
以 ISO8601 顯示。

```
[{"name": "智慧城鄉空氣品質監測站-11613429495",  
  "description": "智慧城鄉空氣品質監測站-11613429495",  
  "properties": {"city": "新竹市",  
    "areaType": "一般社區",  
    "isMobile": "false",  
    "township": "香山區",  
    "authority": "行政院環境保護署",  
    "isDisplay": "true",  
    "isOutdoor": "true",  
    "stationID": "11613429495",  
    "locationId": "HC0154",  
    "Description": "AQ1001",  
    "areaDescription": "新竹市香山區"},  
  "data": [{"name": "Temperature",  
    "description": "溫度",  
    "values": [{"timestamp": "2022-08-27T12:53:10.000Z", "value": 30.6},  
      {"timestamp": "2022-08-27T12:52:09.000Z", "value": 30.6},  
      {"timestamp": "2022-08-27T12:51:09.000Z", "value": 30.6},  
      {"timestamp": "2022-08-27T12:50:09.000Z", "value": 30.6},  
      {"timestamp": "2022-08-27T12:49:09.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:48:10.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:47:10.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:46:10.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:45:10.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:44:10.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:43:09.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:42:10.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:41:09.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:40:10.000Z", "value": 30.7},  
      {"timestamp": "2022-08-27T12:39:10.000Z", "value": 30.7}]],  
    {"name": "Relative humidity",  
      "description": "相對濕度",  
      "values": [{"timestamp": "2022-08-27T12:54:10.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:53:10.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:52:09.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:51:09.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:50:09.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:49:09.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:48:10.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:47:10.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:46:10.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:45:10.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:44:10.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:43:09.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:42:10.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:41:09.000Z", "value": 100},  
        {"timestamp": "2022-08-27T12:40:10.000Z", "value": 100}]],  
    {"name": "PM2.5",  
      "description": "智慧手機 PM2.5",  
      "values": [{"timestamp": "2022-08-27T12:53:10.000Z", "value": 11.9},  
        {"timestamp": "2022-08-27T12:52:09.000Z", "value": 12.15},  
        {"timestamp": "2022-08-27T12:51:09.000Z", "value": 12.2},  
        {"timestamp": "2022-08-27T12:50:09.000Z", "value": 12.22},  
        {"timestamp": "2022-08-27T12:49:09.000Z", "value": 12.54},  
        {"timestamp": "2022-08-27T12:48:10.000Z", "value": 12.54},  
        {"timestamp": "2022-08-27T12:47:10.000Z", "value": 12.31},  
        {"timestamp": "2022-08-27T12:46:10.000Z", "value": 12.19},  
        {"timestamp": "2022-08-27T12:45:10.000Z", "value": 12.26},  
        {"timestamp": "2022-08-27T12:44:10.000Z", "value": 12.17},  
        {"timestamp": "2022-08-27T12:43:09.000Z", "value": 12.04},  
        {"timestamp": "2022-08-27T12:42:10.000Z", "value": 11.7},  
        {"timestamp": "2022-08-27T12:41:09.000Z", "value": 11.67},  
        {"timestamp": "2022-08-27T12:40:10.000Z", "value": 11.56},  
        {"timestamp": "2022-08-27T12:39:10.000Z", "value": 11.56}]]],  
  "location": {"latitude": 24.81796, "longitude": 120.92664, "address": None}]}
```





# 存取特定時空條件的資料

## 獲取特定特定區域之資料

使用 `.get_data(src="", stationID="")` 時可以再加入  
參數 `location` 來取得特定區域的資料

```
loc = {
    "latitude": 24.990550, # 緯度
    "longitude": 121.507532, # 經度
    "distance": 3.0 # 半徑(km)
}
c = Air().get_data(src="OBS:EPA_IoT", location = loc)
c[0:3]
```

```
[{'name': '智慧城鄉空品微型感測器-10382640142',
  'description': '智慧城鄉空品微型感測器-10382640142',
  'properties': {'city': '新北市',
    'areaType': '交通',
    'isMobile': 'false',
    'township': '中和區',
    'authority': '行政院環境保護署',
    'isDisplay': 'true',
    'isOutdoor': 'true',
    'stationID': '10382640142',
    'locationId': 'TW040203A0506917',
    'Description': '廣域SAQ-210',
    'areaDescription': '中和區'},
  'data': [{'name': 'Relative humidity',
    'description': '相對濕度',
    'values': [{'timestamp': '2022-10-21T18:07:58.000Z', 'value': 94.66}]},
    {'name': 'PM2.5',
    'description': '細懸浮微粒 PM2.5',
    'values': [{'timestamp': '2022-10-21T18:07:58.000Z', 'value': 0.73}]},
    {'name': 'Temperature',
    'description': '溫度',
    'values': [{'timestamp': '2022-10-21T18:07:58.000Z', 'value': 25.45}]},
```



# 存取特定時空條件的資料

## 獲取特定特定區域之資料

參數 `location` 為一個事先設定好的格式

```
loc = {  
    "latitude": 24.990550, # 緯度  
    "longitude": 121.507532, # 經度  
    "distance": 3.0 # 半徑(km)  
}
```

`longitude` 與 `latitude` 指資料搜尋的經度及緯度  
`distance` 則是會控制搜尋半徑

例如 `.get_data(src="", stationID="", location = loc)`



# 存取特定時空條件的資料

執行結果就會如圖

程式會找出所有範圍內的站點，並回傳感測值



```
loc = {  
    "latitude": 24.990550, # 緯度  
    "longitude": 121.507532, # 經度  
    "distance": 3.0 # 半徑(km)  
}  
c = Air().get_data(src="OBS:EPA_IoT", location = loc)  
  
# 計算資料筆數  
print("資料筆數:", len(c))  
  
# 顯示前三筆  
c[0:3]
```



```
資料筆數: 70  
[{'name': '智慧城鄉空品微型感測器-10382640142',  
  'description': '智慧城鄉空品微型感測器-10382640142',  
  'value': 100,  
  'time': '2017-01-01 00:00:00',  
  'location': '10382640142'}]
```



## 進階應用

實作：所在地有比附近的空氣糟嗎？

- 匯入資料：環保署智慧城鄉空品微型感測器(OBS:EPA\_IoT)
- 測試地點：中和南勢角捷運站 1 號出口：(24.990550, 121.507532)
- 比較環境：新北市中和區

## 進階應用

### 獲取檢測站資料

首先，需要獲得測試地點和比較環境的所有資料。我們可以利用「獲取特定區域之資料」的方法，將經度緯度設定在南勢角捷運站一號出口，將距離設定為三公里，即可簡單的將資料利用 `Air().get_data()` 獲取：

```
# 獲取檢測站的資料 ( 溫度、濕度、 PM2.5 )

loc = {
    "latitude": 24.990550, # 緯度
    "longitude": 121.507532, # 經度
    "distance": 3.0 # (km)
}

EPA_IoT_zhonghe_data_raw = Air().get_data(src="OBS:EPA_IoT", location = loc)
print("len:", len(EPA_IoT_zhonghe_data_raw)) # 印出測站個數
EPA_IoT_zhonghe_data_raw[0]
```

# 進階應用

## 去除無效資料

在每個範圍內的測站中，不一定每個測站都還在順利運行。為了將這些測站去除，我們觀察無效測站的資料會有什麼特徵，發現三個資料（溫度、濕度、PM2.5 濃度）都會是 0。只要挑出並刪除這些資料便能夠進行下一步驟。

```
# Data cleaning

for datajson in EPA_IoT_zhonghe_data_raw:
    # 確認資料存在
    if "data" not in datajson:
        continue;
    # 將格式轉換為 Temperature, Relative_Humidity 和 PM2_5
    for rawdata_array in datajson['data']:
        if(rawdata_array['name'] == 'Temperature'):
            datajson['Temperature'] = rawdata_array['values'][0]['value']
        if(rawdata_array['name'] == 'Relative humidity'):
            datajson['Relative_Humidity'] = rawdata_array['values'][0]['value']
        if(rawdata_array['name'] == 'PM2.5'):
            datajson['PM2_5'] = rawdata_array['values'][0]['value']
    datajson.pop('data')
    # 確認所有資料皆為有效，同時去除無資料之檢測站
    if "Relative_Humidity" not in datajson.keys():
        continue
    if "PM2_5" not in datajson.keys():
        continue
    if "Temperature" not in datajson.keys():
        continue
    if(datajson['Relative_Humidity'] == 0 and datajson['PM2_5'] == 0 and datajson['Temperature'] == 0):
        continue
    EPA_IoT_zhonghe_data.append(datajson)

print("len:", len(EPA_IoT_zhonghe_data))
EPA_IoT_zhonghe_data[0]
```

# 進階應用

## 計算距離

假設每個測站的資料沒有誤差，那最接近目標地點的測站資料即為要比較的資料。為了找到最接近的測站，我們要算出將每個測站和目標地點的距離。

我們可以利用點到點距離公式計算並排序找到最接近目標地點的測站，可以直接使用在 `math` 內的 `pow()` 函式，計算平方及平方根距離。但在這裡我們使用比較標準的 Haversine 公式計算地球上兩點間的球面距離，以下為在 WGS84 坐標系下的實作：

```
# 增加與南勢角站距離欄位
import math
def LLS2Dist(lat1, lon1, lat2, lon2):
    R = 6371
    dLat = (lat2 - lat1) * math.pi / 180.0
    dLon = (lon2 - lon1) * math.pi / 180.0
    a = math.sin(dLat / 2) * math.sin(dLat / 2) + math.cos(lat1 * math.pi / 180.0) * math.cos(lat2 *
    math.pi / 180.0) * math.sin(dLon / 2) * math.sin(dLon / 2)
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    dist = R * c
    return dist

for data in EPA_IoT_zhonghe_data:
    data['distance'] = LLS2Dist(data['location']['latitude'], data['location']['longitude'], 24.990550,
    121.507532)# (24.990550, 121.507532)
EPA_IoT_zhonghe_data[0]
```

## 進階應用

### Pandas 函式庫

Pandas 是用於資料操縱和分析的函式庫，其中的 DataFrame 格式用來儲存雙維度或多欄位的資料格式，非常適合用來進行資料分析。我們將處理好的資料轉換成 DataFrame，並挑選出需要的欄位並根據先前計算的距離排序由小到大。


```
# 轉換成 Pandas.DataFrame 格式
import pandas as pd

df = pd.json_normalize(EPA_IoT_zhonghe_data)
#Results contain the required data
df

EPA_IoT_zhonghe_data_raw = df[['distance', 'data.PM2_5', 'data.Temperature', 'data.Relative_Humidity',
'properties.stationID', 'location.latitude', 'location.longitude', 'properties.areaType']]
EPA_IoT_zhonghe_data_raw = EPA_IoT_zhonghe_data_raw.sort_values(by=['distance', 'data.PM2_5'],
ascending=True)
EPA_IoT_zhonghe_data_raw
```


# 進階應用

## 使用pandas進行資料排序



	distance	PM2_5	Temperature	Relative_Humidity	properties.stationID	location.latitude	location.longitude	properties.areaType
16	0.297714	2.33	25.67	89.25	10364644999	24.988435	121.509343	社區
13	0.360686	2.52	25.81	90.69	10364231669	24.992348	121.504553	交通
31	0.401112	0.16	25.49	90.51	10363449767	24.987007	121.508280	社區
15	0.411062	2.27	25.82	89.47	10386461949	24.991235	121.511540	社區
40	0.428510	3.56	25.70	91.74	12276258465	24.992796	121.510987	交通區
...	...	...	...	...	...	...	...	...
57	2.801352	3.62	26.04	90.70	12272069509	25.004655	121.484500	交通區
55	2.843923	1.35	25.12	93.77	12276605031	25.012077	121.522770	交通區
51	2.853700	2.20	25.56	89.26	12272329005	25.008680	121.487490	工業區
10	2.991552	0.58	25.33	92.46	10356488656	24.978528	121.534085	交通
8	3.016178	0.55	25.09	97.94	10354510861	24.985350	121.536903	交通

70 rows × 8 columns





# 進階應用

## 顯示結果

為了知道目標區域的空氣品質相較附近區域的好壞，可以大致上利用所有測站空氣品質的分佈得知。可以利用在 Python 中常利用的 numpy 資料科學處理常用函式庫等工具，或直接計算出平均及標準差，便可得到答案。

```
import numpy as np
zhonghe_target = EPA_IoT_zhonghe_data_raw.iloc[0,1]
zhonghe_ave = np.mean(EPA_IoT_zhonghe_data_raw.iloc[:,1].values)
zhonghe_std = np.std(EPA_IoT_zhonghe_data_raw.iloc[:,1].values)
result = (zhonghe_target-zhonghe_ave)/zhonghe_std

print('mean:', zhonghe_ave, 'std:', zhonghe_std)
print('最近測站 PM2.5 濃度:', zhonghe_target)
print('目標離平均', result, '個標準差\n')

if(result>0):
    print('Result: 現在家裡附近的空氣比附近糟')
else:
    print('Result: 現在家裡附近的空氣比附近好')
```

```
mean: 6.71 std: 3.18
最近測站 PM2.5 濃度: 7.38
目標離平均 0.21 個標準差
```

```
Result: 現在家裡附近的空氣比附近糟
```





## 參考資料

- <https://learnciot.github.io>
- Python pyCIOT package (<https://pypi.org/project/pyCIOT/>)
- pandas - Python Data Analysis Library (<https://pandas.pydata.org/>)
- 10 minutes to pandas — pandas documentation ([https://pandas.pydata.org/pandas-docs/stable/user\\_guide/10min.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html))
- NumPy (<https://numpy.org/>)
- NumPy quickstart (<https://numpy.org/doc/stable/user/quickstart.html>)

The icons has been using resources from Flaticon.com