網路程式設計
Web API

Instructor: 馬豪尚

氣象資料來源

- 氣象資料開放平台
  - https://opendata.cwb.gov.tw/index

氣象資料開放平台

- 使用氣象資料平台api
  - 註冊帳號
  - 取得api key

氣象資料開放平台

- 取得氣象預報資料
  - 一般天氣預報-今明36小時
  - 臺灣各縣市天氣預報資料及國際都市天氣預報
  - 臺灣各鄉鎮市區預報資料-臺灣各鄉鎮市區未來2天(逐3小時)及未來1週天氣預報
  - 臺灣各鄉鎮市區預報資料-臺灣各鄉鎮市區未來1週天氣預報
  - …

氣象資料開放平台- 取得預報資料

- 使用一般天氣預報-今明36小時的api
- 輸入參數
  - authorization → api key
  - limit → 限制最多回傳的資料筆數
  - offset → 指定從第幾筆後開始回傳
  - format → 回傳資料格式 json/xml
  - locationname → 指定哪一個(全部)縣市
  - elementname → 天氣因子
  - sort →對starttime和endtime做排序
  - starttime →時間因子，格式為「yyyy-mm-ddthh:mm:ss」
  - timefrom →時間區段，從什麼時間開始
  - timeto →時間區段，到什麼時間結束

氣象資料開放平台- 取得預報資料

- Api請求資料的位置(url)
  - https://opendata.cwb.gov.tw/api/v1/rest/datastore/F-C0032-001?Authorization=你的apikey
- 輸入參數會加在請求位置後面
  - 設定參數limit=100
    - https://opendata.cwb.gov.tw/api/v1/rest/datastore/F-C0032-001?Authorization=你的apikey&limit=100
  - 選擇縣市(宜蘭縣)
    - &locationName=%E5%AE%9C%E8%98%AD%E7%B8%A3

氣象資料開放平台- 取得預報資料

- 使用python中的requests模組請求資料
  - requests.get(url)
- 回傳為json格式的資料
  - success → 伺服器回應狀態
  - result → 儲存資料格式和型態
  - records → 資料內容

氣象資料開放平台- 取得預報資料

- result物件
  - resource_id → 請求資源的資料表名稱
  - fields → 定義回傳資料格式和型態
    - {'id': 'datasetDescription', 'type': 'String'}
    - {'id': 'locationName', 'type': 'String'}
    - {'id': 'parameterName', 'type': 'String'}
    - {'id': 'parameterValue', 'type': 'String'}
    - {'id': 'parameterUnit', 'type': 'String'}
    - {'id': 'startTime', 'type': 'Timestamp'}
    - {'id': 'endTime', 'type': 'Timestamp'}

氣象資料開放平台- 取得預報資料

- records資料內容物件
  - datasetDescription → 資料概要描述
    - 三十六小時天氣預報
  - Location[]→ 回傳的縣市天氣預報資料，會以list方式儲存
    - locationName → 縣市區域名稱
    - weatherElement[] → 儲存不同天氣因子的資料
      - elementName → 定義是哪個天氣因子
        - Wx →天氣現象的描述
        - PoP → 降雨機率
        - CI → 舒適度
        - MinT → 最低溫度
        - MaxT → 最高溫度
      - time[] → 36個小時，不同時間區間的預測資料(12小時為單位)

氣象資料開放平台- 取得預報資料

- time[] →不同時間區間的預測資料
  - 基本上會分成三筆資料，每12小時為一筆，每筆資料內又包含以下
    - startTime: 標示這筆資料的時間區間的開始
    - endTime : 標示這筆資料的時間區間的結束
    - parameter
      - parameterName: 實際該天氣因子的值
      - parameterValue: 該天氣因子的值所對應的代號(只有Wx有這個值)
      - parameterUnit: 該天氣因子所使用的表示單位(百分比、C)

取得預報資料 – 實際案例

- 取得台中市未來天氣預報(36小時)
  - 指定api輸入參數
    - 輸入api key → Authorization=apikey
    - 指定縣市為台中市 → locationName=%E8%87%BA%E4%B8%AD%E5%B8%82
  - 向api資源位址請求
    - requests.get(url)
  - 取得天氣描述
    - 第一個時間區間
      - res['records']['location'][0]['weatherElement'][0]['time'][0][startTime']
      - res['records']['location'][0]['weatherElement'][0]['time'][0][endTime']
    - 描述
      - res['records']['location'][0]['weatherElement'][0]['time'][0]['parameter']['parameterName']

氣象資料開放平台 – 實際案例

- 取得台中市未來的天氣預報
  - 取得未來24-36小時(第3個時間區間)的最高溫(第5個天氣因子)
    - res['records']['location'][0]['weatherElement'][4]['time'][2]['parameter']['parameterName']
  - 取得未來12-24小時(第2個時間區間)的舒適度(第4個天氣因子)
    - res['records']['location'][0]['weatherElement'][3]['time'][1]['parameter']['parameterName']

氣象資料開放平台- 取得即時資料

- 使用觀測→自動氣象站資料-無人自動站氣象資料api
  - 用測站站號或測站站名來搜尋
    - Authorization → api key
    - stationId → 測站站號
    - locationName → 測站站名
    - elementName → 氣象因子
      - TEMP: 溫度
      - Weather: 天氣描述

氣象資料開放平台- 取得即時資料

- 使用python中的requests模組請求資料
  - requests.get(url)
- 回傳為json格式的資料
  - success → 伺服器回應狀態
  - result → 儲存資料格式和型態
  - records → 資料內容

氣象資料開放平台- 取得即時資料

- records資料內容物件
  - Location[] → 回傳的縣市天氣預報資料，會以list方式儲存
    - Lat: 緯度
    - Lon: 精度
    - locationName: 測站名稱
    - stationId: 測站編號
    - Time: 時間
      - obsTime
    - weatherElement[]: 天氣因子陣列
      - elementName: 天氣因子名稱
      - elementValue: 天氣因子值
    - parameter[]: 地區資訊陣列

取得即時資料 – 實際案例

- 要取得烏日測站(地區)的即時溫度資訊
  - 指定api輸入參數
    - 輸入api key → Authorization=apikey
    - 指定測站名稱=烏日 → locationName=%E7%83%8F%E6%97%A5
    - 指定天氣因子溫度 → elementName=TEMP
  - 向api資源位址請求
    - requests.get(url)
  - 取得測站溫度資訊
    - res['records']['location'][0]['weatherElement'][0]['elementValue']

Python 自動排程

Python 自動排程

- APScheduler 套件
  - pip install apscheduler
- APScheduler 有四個主要的組件
  - Trigger: 觸發器，任務指定的觸發方式
    - Date 就是指定時間執行一次
    - Interval 就是隔多久執行一次
    - Cron 就是 linux 的排程方式，cron 可針對 分, 時, 日, 月, 星期幾 去做設定何時去進行排程
  - Job store: 儲存工作的地方，預設是存在記憶體中，也可以存在資料庫裡
  - Executor: 如何執行排程的工作，一般就是 thread pool 或是 process pool
  - Scheduler:排程調度器，對於自動排程工作的操作(開始/結束/暫停/繼續)

Python 自動排程

- BlockingScheduler
  - 基本的排程調度器，當這個排程程序是整個程式唯一在執行的程序時使用
  - 開始執行排程之後，程式就無法做其他事情
- 載入調度器模組
  - from apscheduler.schedulers.blocking import BlockingScheduler
- 宣告BlockingScheduler物件
  - scheduler = BlockingScheduler()

Python 自動排程

- BackgroundScheduler
  - 可以背景執行的排程調度器，當排程開始執行時程是仍然可以執行其他程序
  - 適合在一個應用程式裡背景執行
- 載入調度器模組
  - from apscheduler.schedulers.background import BackgroundScheduler
- 宣告BackgroundScheduler物件
  - scheduler = BackgroundScheduler()

Python 自動排程

- 建立一個間隔時間的Trigger觸發器來新增任務
  - scheduler.add_job(task, "interval", seconds=3)
    - Task為要執行的程序/函數

參數	說明
weeks	週，整數
days	一個月中的第幾天，整數
hours	小時，整數
minutes	分鐘，整數
seconds	秒，整數
start_date	間隔觸發的起始時間，date、datetime物件
end_date	間隔觸發的結束時間，date、datetime物件

Python 自動排程

- 建立一個指定日期/時間的Trigger觸發器來新增任務
  - 指定日期/時間
    - date = datetime.date(2023, 5, 25)
    - date = datetime.datetime(2023, 5, 25, 10, 0, 0)
  - 建立工作
    - scheduler.add_job(task, "date", run_date=date, args=["工作1"])
      - task為要執行的程序/函數
      - run_date為指定時間參數
      - args為傳入的參數

Python 自動排程

- 建立一個在某個確切的時間週期性的觸發事件
  - scheduler.add_job(task, “cron", day_of_week='mon-fri', hour='0-23', minute='*', second='*/4)

參數	說明
year	4 位數年份，(int|str)
month	月 (1-12) ，(int|str)
day	月中的第幾天 (1-31)，(int|str)
week	週，(int|str)
day_of_week	工作日的數字或名稱（0-6 或mon,tue,wed,thu,fri,sat,sun），(int|str)
hour	小時(0-23)，(int|str)
minute	分鐘(0-59)，(int|str)
second	秒(0-59)，(int|str)
start_date	最早觸發的日期/時間
end_date	觸發的最晚日期/時間

Python 自動排程的操作

- 開始執行排程工作
  - scheduler.start()
- 結束執行排程工作
  - scheduler.shutdown()
- 暫停工作
  - scheduler.pause()
- 恢復工作
  - scheduler.resume()

練習

- 使用氣象資料開放平台的api
  - 取得全台測站的即時溫度和天氣描述
  - 將資料依照以下欄位存成dataframe並輸出成csv檔案
  - 包含三個欄位，測站名稱、溫度、天氣描述
- 可以搭配自動排程每個小時自動取得新的即時資料