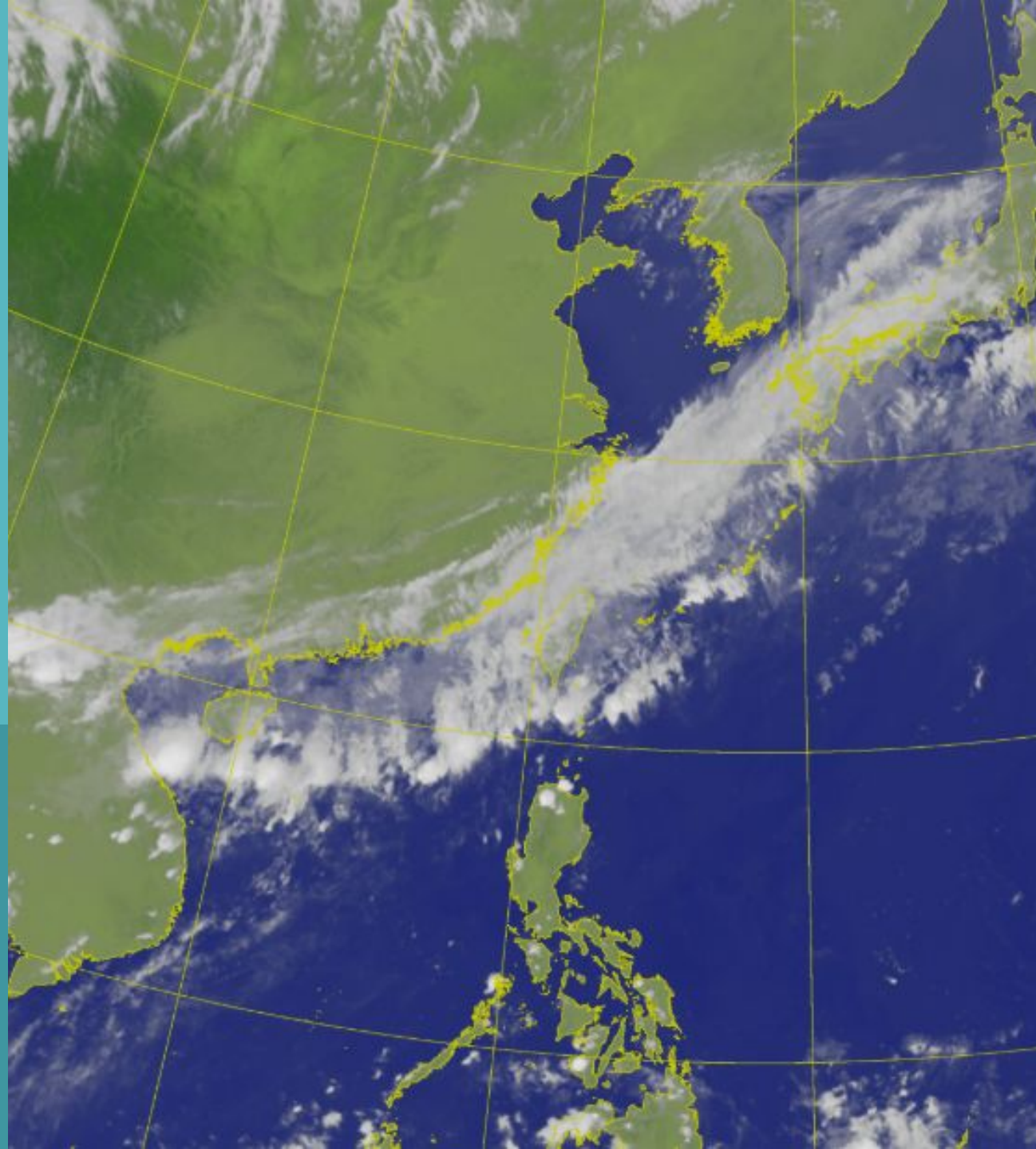
第7組 數據小尖兵

===專題主題===

氣象預報系統

===簡報者姓名===

陳銘泓



小組成員及分工

組　　長

陳銘泓

負責內容

提案發想
會議主持及記錄
github管理
面授報告

姓　　名

詹秉蒼

負責內容

程式撰寫:
資料抓取
(RequestApi.py)

姓　　名

劉彥翎

負責內容

程式撰寫:
資料呈現
(WeatherGUI.py
、
主體框架、
氣象預報)

姓　　名

傅柏凱

負責內容

程式撰寫:
資料呈現
(WeatherGUI.py
、
地震紀錄)

姓　　名

張心齡

負責內容

程式撰寫:
資料整理
(app.py)

姓　　名

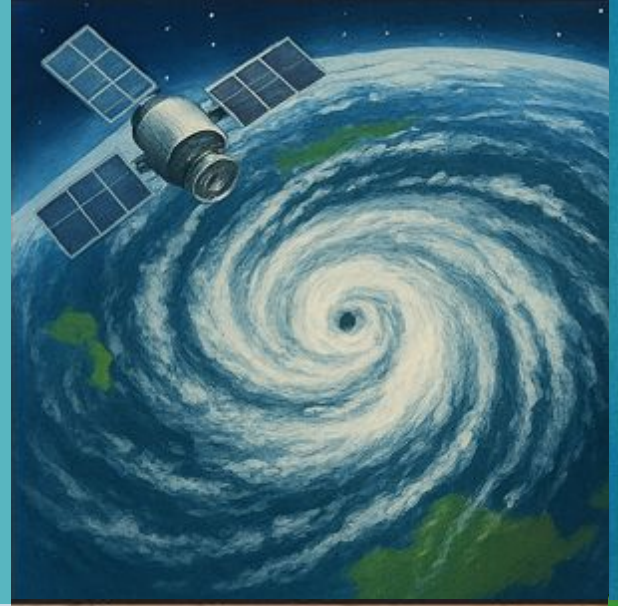
羅若嘉

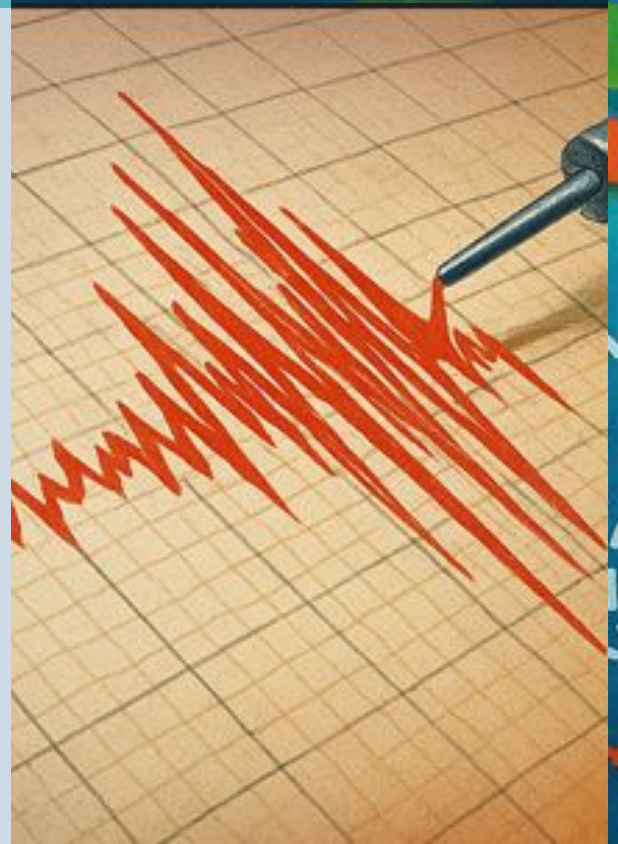
負責內容

投影片製作

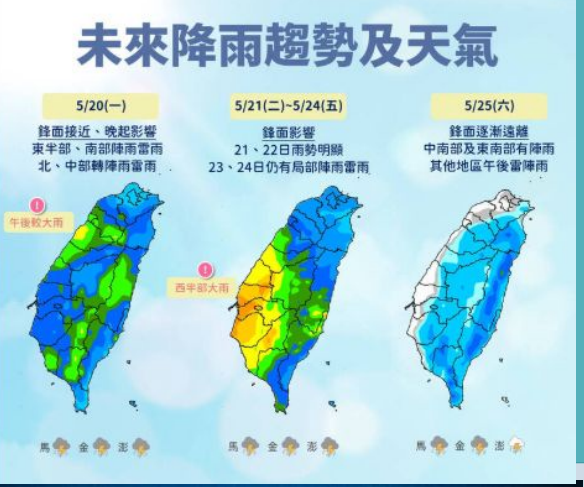
大綱內容

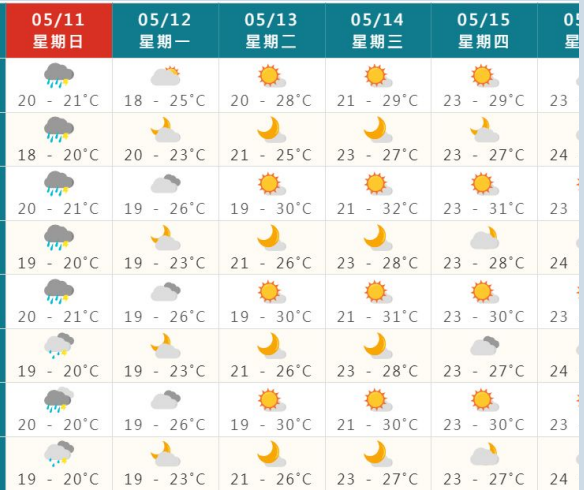
- ✚專案簡介
- ✚執行結果
- ✚流程圖
- ✚相關套件
- ✚原始碼重點解說





專案簡介





簡介

本專題開發一個台灣氣象與地震資訊的視覺化查詢系統，透過使用者友好的圖形介面，為使用者提供即時、準確的氣象預報與地震資訊。系統結合了中央氣象署的開放資料API，讓使用者能夠便捷地獲取台灣各地區的天氣狀況與地震報告。

資料來源


```
<?xml version="1.0" encoding="UTF-8"?>
<cwaopendata xmlns="urn:cwa:gov:tw:cwacommon:0.1">
    <identifier >d740fbaf-7db9-9752-673b-88605ddeeb5f</identifier>
    <sender >weather@cwa.gov.tw</sender>
    <sent >2025-05-20T22:33:02+08:00</sent>
    <status >Actual</status>
    <msgType >Issue</msgType>
    <source >MFC</source>
    <dataid >C0032-003</dataid>
    <scope >Public</scope>
    <dataset >
        <datasetInfo>
            <datasetDescription>七天天氣預報</datasetDescription>
            <issueTime>2025-05-20T23:00:00+08:00</issueTime>
            <update>2025-05-20T22:33:02+08:00</update>
</datasetInfo>
        <location>
            <locationName>臺北市</locationName>
            <weatherElement>
                <elementName>Wx</elementName>
                <time>
                    <startTime>2025-05-21T00:00:00+08:00</startTime>
                    <endTime>2025-05-22T00:00:00+08:00</endTime>
                    <parameter>
                        <parameterName>多雲午後短暫雷陣雨</parameterName>
                        <parameterValue>22</parameterValue>
</parameter>
</time>
                <time>
                    <startTime>2025-05-22T00:00:00+08:00</startTime>
                    <endTime>2025-05-23T00:00:00+08:00</endTime>
                    <parameter>
                        <parameterName>多雲午後短暫雷陣雨</parameterName>
                        <parameterValue>22</parameterValue>
</parameter>
```

執 行 結 果

WINDOW版


氣象預報系統
氣象預報系統
氣象預報系統
氣象預報系統
氣象預報系統
Dark
天氣預報
地震資訊
縣市地
宜蘭縣
宜蘭市
新北市
板橋區
小區域有感
小區域有
顯著有感
小區域有感地震
查詢
選擇完後
發生時間
震央位置
規模
深度
最大震度
地區
描述
2025-05-20 02:23:06
臺南市政府東北方 42.3 公里 (位於臺南市東山區)
3.7
7.2 公里
3級
臺南市
5/20-02:23臺南市東山區發生規模3.7有感地震，最大震度臺南市曾
級。
2025-05-19 07:38:19
臺東縣政府西南方 39.1 公里 (位於臺東縣太麻里鄉)
3.6
10.0 公里
2級
臺東縣
5/19-07:38臺東縣太麻里鄉發生規模3.6有感地震，最大震度臺東縣
及2級。
2025-05-19 03:55:34
臺東縣政府東北方 30.8 公里 (位於臺東縣近海)
4.3
39.4 公里
2級
臺東縣
5/19-03:55臺東縣近海發生規模4.3有感地震，最大震度臺東縣東河
及。
2025-05-19 03:35:31
花蓮縣政府西南方 25.4 公里 (位於花蓮縣秀林鄉)
3.5
19.0 公里
2級
南投縣
5/19-03:35花蓮縣秀林鄉發生規模3.5有感地震，最大震度花蓮縣西
南投縣奧萬大2級。
2025-05-19 03:18:03
花蓮縣政府西南西方 21.0 公里 (位於花蓮縣秀林鄉)
3.1
18.8 公里
2級
南投縣
5/19-03:18花蓮縣秀林鄉發生規模3.1有感地震，最大震度花蓮縣西
南投縣奧萬大2級。
2025-05-18 15:40:18
臺東縣政府東北方 30.2 公里 (位於臺東縣近海)
3.9
39.5 公里
1級
花蓮縣
5/18-15:40臺東縣近海發生規模3.9有感地震，最大震度臺東縣東河
花蓮縣富里1級。
2025-05-16 18:41:10
臺南市政府東北方 39.6 公里 (位於臺南市楠西區)
3.6
8.8 公里
2級
臺南市、嘉義縣
5/16-18:41臺南市楠西區發生規模3.6有感地震，最大震度臺南市曾
嘉義縣大埔2級。
2025-05-16 12:43:54
花蓮縣政府西北西方 30.8 公里 (位於花蓮縣秀林鄉)
3.4
5.5 公里
2級
南投縣
5/16-12:43花蓮縣秀林鄉發生規模3.4有感地震，最大震度花蓮縣西
臺中市梨山、南投縣合歡山、宜蘭縣南山2級。

流　程　圖

流程圖

資料呈現 WeatherGUI.py

開始：開啟 GUI

載入主畫面

外觀模式切換

切換至 WeatherTab

選擇城市

更新行政區

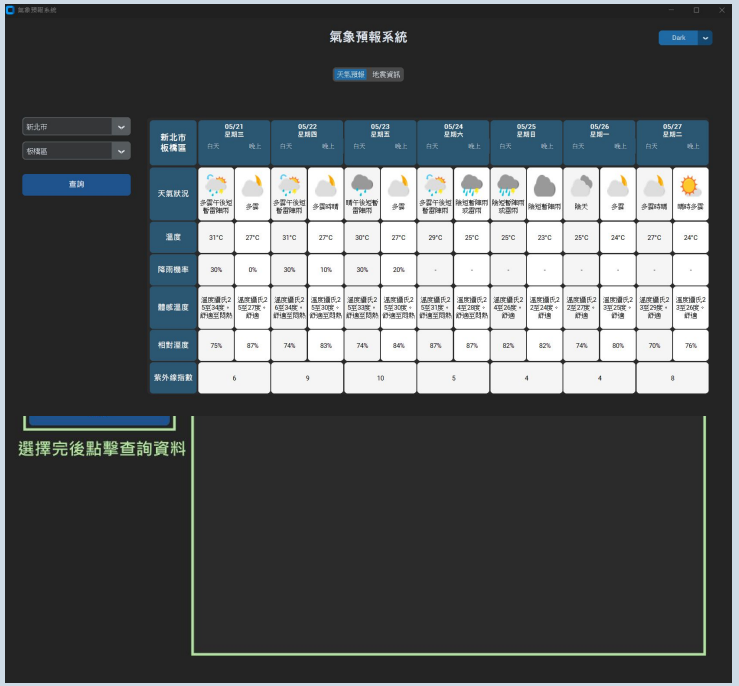
選擇行政區

執行 get_weather_by_loction

切換至 EarthquakeTab

選擇地震類型

執行 getEarthquake



資料處理 app.py

呼叫 getWeatherByCity

篩選與整理氣象資料

資料蒐集 RequestApi.py

存取氣象資料開放平臺 API

依 mode_type 取得地震資料

顯示與結束

從 weather_icons 載入圖示

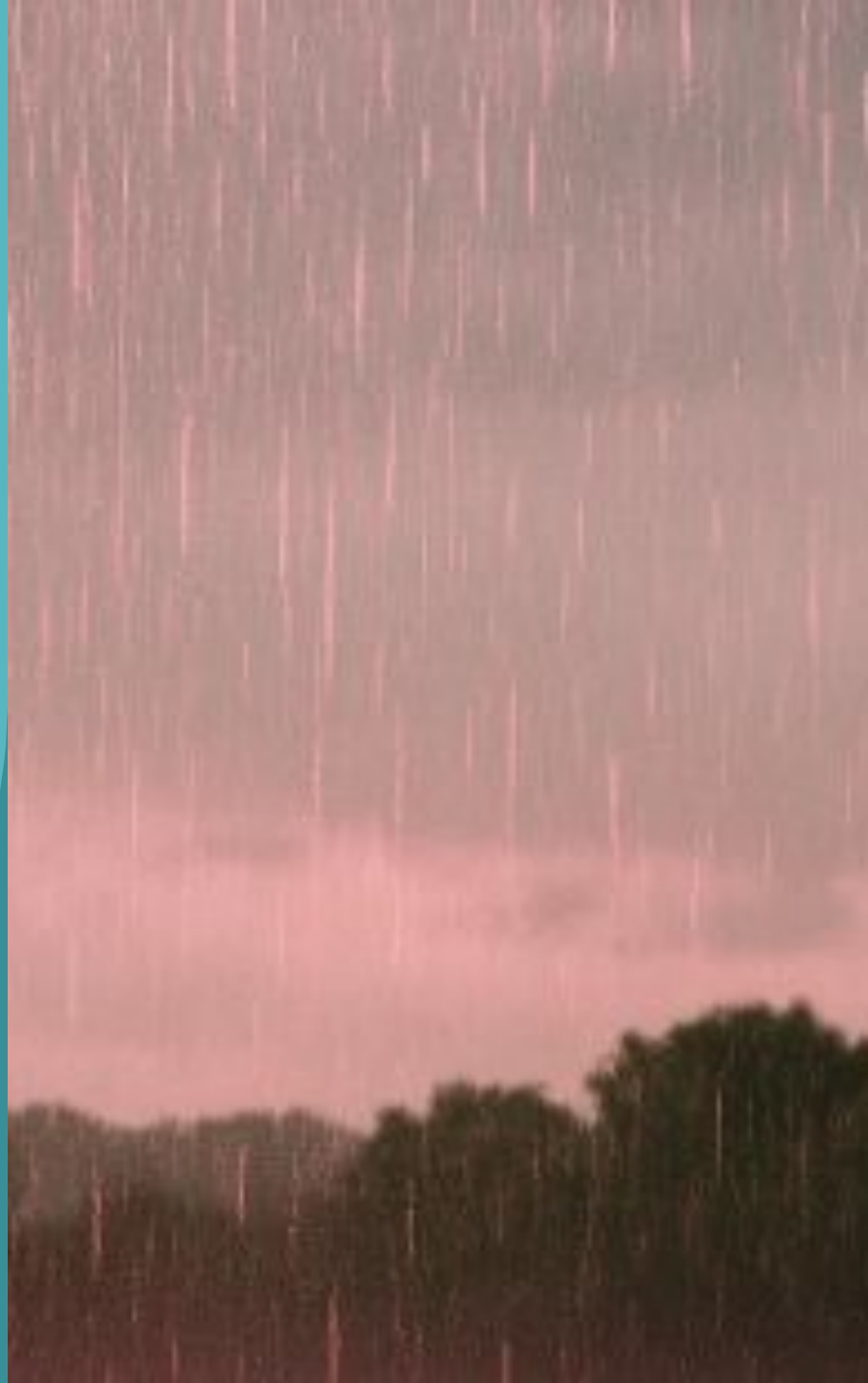
顯示天氣結果

顯示地震結果

回主畫面

結束

相關套件



相關套件

- API 請求：requests
- GUI 介面：customtkinter、tkinter
- 圖片處理：Pillow（PIL）
- 資料處理：datetime、os
- 其他：functools.partial（用於事件綁定）

原始碼重點解說

--資料蒐集--

數據小尖兵-氣象預報系統

1.採用屬性方式將固定資料寫入，提高重複使用率

```
class RequestApi:
    # 要呼叫的API URL位置
    url = 'https://opendata.cwa.gov.tw'
    # API版本
    version = 'v1'
    # API路由
    path = f'api/{version}/rest/datastore'
    # token
    apiAuth ={'Authorization': 'CWA-5B306D7D-A5ED-4639-8532-D1C274899F48'}
    # 定義請求標頭
    headers = {
        'User-Agent': 'python work',
        'Accept': '*/*',
        'Connection': 'keep-alive'
    }
    '''
    城市對應的API代碼陣列中第一個位置是3天預報，第二個位置是1週預報
    '''
    cityApi = {
        '宜蘭縣': ['F-D0047-001', 'F-D0047-003'],
        '金門縣': ['F-D0047-085', 'F-D0047-087']
    }
    '''
    地震API
    '''
    earthquakeApi = [
        'E-A0016-001', # 小區域有感地震
        'E-A0015-001', # 顯著有感地震
    ]

    # 要呼叫的API端點
    apis = [
        'O-A0003-001', # 現在天氣觀測
        'F-C0032-001', # 臺灣各縣市天氣預報資料及國際都市天氣預報
        'A-B0062-001', # 日出日落
        'E-A0014-001', # 海嘯
    ]
```

2.提供可配置
HTTP Request Header屬性，提高可擴充靈活度

```
def setHeader(self, params = {}):
    """
    配置request http header要傳遞的內容
    """
        self.headers = {**self.headers, **params}
        return None
```

3.封裝發送請求函數，並返回統一的字典格式

{status:"狀態", code:"HTTP CODE", message:"訊息", data:"內容"}

```
def send(self, method, api, params):
    """
    proxy 代理方法，對應方法發調用equest對應的方法
    """
    response = {"status": False, "code":0, "message":None, "data": None}
    try:
        if(method.upper() == 'GET'):
            _r = requests.get(api, headers=self.headers, params={**params, **self.apiAuth})
        elif(method.upper() == 'POST'):
            _r = requests.post(api, headers=self.headers, data=params,
params=self.apiAuth)
        else:
            response["message"] = f"錯誤的方法{method}"
            return response

        response["status"] = _r.status_code == 200
        response["code"] = _r.status_code
        response["message"] = "呼叫成功" if _r.status_code == 200 else "呼叫失敗"
            response["data"] = _r.json()
            return response

    except requests.exceptions.HTTPError as errh:
        response["message"] = f"HTTP錯誤: {errh}"
        return response
    except requests.exceptions.ConnectionError as errc:
        response["message"] = f"連接錯誤: {errc}"
        return response
    except requests.exceptions.Timeout as errt:
        response["message"] = f"超時錯誤: {errt}"
        return response
    except requests.exceptions.RequestException as err:
        response["message"] = f"無法預期的錯誤: {err}"
        return response
```

```python
def getWeatherByCity(self, city, days = 3):
    """
    取得指定城市的天氣資料
    Args:
        self
        city (string): 要取得的程式名稱()
        days (int): 要取得的天數(3或7)
    Returns:
        json
    """
    index = 0 if days == 3 else 1
    #取得縣市與API端點
    api = self.cityApi.get(city)
    if api is None:
        return {"status": False, "message": f"無法取得{city}的API端點"}
    #取得API端點
    if index >= len(api):
        return {"status": False, "message": f"無法取得{city}的{days}天預報API端點"}
    api = api[index]
    #取得API資料
    return self.get(api)
```

4.提供取得不同城市天氣的快速方法，方便其他程式調用

5.提供取得地震資訊的快速方法，方便其他程式調用

```
def getEarthquake(self, modeType = 0):
    """
    取得地震資料
    Args:
        self
        type (int): 0:小區域有感地震 1:顯著有感地震
    """
    #取得API端點
    api = self.earthquakeApi[modeType]
    #取得API資料
    return self.get(api)
```

原始碼重點解說

--資料處理--

1. classify_period_by_end_time(start_time, end_time)

根據時間區段區分白天/晚上，並回傳日期與時段。
白天為 6:00 ~18:00，晚上為18:00~隔日6:00。

```
if end_date.hour == 18:
    period = "白天"
elif end_date.hour == 6:
    period = "晚上"
```

2. extract_element_value(element_value, allowed_element_type)

從單一氣象元素中抽出時間區段與值，整理為統一格式。

透過allowed_element_type過濾不必要元素，留下目標氣象資訊。

回傳每個目標元素的日期、時間與值。

```
        "date": date,
        "period": period,
        "values": values
```

3. parse_weather_elements(WeatherElement, allowed_element_type)

解析氣象元素列表(WeatherElement)
取得這個氣象項目的名稱(ElementName)

並將extract_element_value回傳的資料合併

保留指定的氣象元素(平均溫度、濕度)

整理為標準格式一併回傳給get_weather_by_loction

```
element_type = element.get("ElementName")

element_values = extract_element_value(element,

allowed_element_type)
```

4. get_weather_by_loction(city, district, target_elements)

主功能

透過 RequestApi 取得天氣資料

並依需求項目過濾、解析、回傳

```
    response = app.getWeatherByCity(city, 7)

    #取得目標城市並過濾行政區
    locations = data["records"]["Locations"]
    location_list = locations[0].get("Location", [ ])
    for loc in location_list:
        if loc.get("LocationName") != district:
            continue

    #解析氣象元素，使用parse_weather_elements()來整理格式
    weather_element = loc.get("WeatherElement", [ ])
    weather_data = parse_weather_elements(weather_element,
    target_elements)

    #回傳指定城市、行政區以及目標元素 值
    return {
        "city": locations[0].get("LocationsName", [ ]),
        "district": location_name,
        "weather": weather_data
    }
```

5. calc_water_vapor_pressure(RH, temp)

計算水氣壓(vapor press), 輸入參數為相對濕度(RH)、溫度(temp)

公式:$e=RH \times 0.01 \times 6.105 \times \exp((17.27 \times T) / (237.7 + T))$

e:水氣壓(hPa)

RH:相對溼度(%)

exp :指數函式

T:溫度(℃)

6. calc_apparent_temperature(temp, humd, wind_speed)

計算體感溫度

參數為溫度(temp)、濕度(humd)以及風速(wind_speed)

取值到小數點第一位

公式：$AT = 1.04 \times T + 0.2 \times e - 0.65 \times V - 2.7$

AT：體感溫度(℃)

T：溫度(℃)

e：水氣壓(hPa)

V：風速(m/sec)

原始碼重點解說

--資料呈現--

數據小尖兵-氣象預報系統

1. 主要類別結構

```
class WeatherApp:

    def __init__(self, root):

        # 初始化設定

        self.root = root

        self.root.title("氣象預報系統")

        screen_width = self.root.winfo_screenwidth()

        screen_height = self.root.winfo_screenheight()

        self.root.geometry(f"{screen_width}x{screen_height}")
```

- 使用 WeatherApp 類別封裝所有功能
- 初始化時自動調整視窗至全螢幕大小

2. 天氣圖示載入機制

```
    def load_weather_icons(self):
        icons = {}
        icon_size = (40, 40)
        icon_path = "weather_icons"

        # 定義天氣狀況和對應的圖示檔案名稱
        weather_icon_mapping = {
            "多雲": "多雲.png",
            "陰": "陰.png",
            # ... 其他天氣狀況
        }
```

- 使用字典映射天氣狀況和圖檔
- 自動載入並調整圖檔大小
- 支援多種天氣狀況的圖示顯示

3. 天氣資料處理

```
    def get_weather(self):
        # 請求所有需要的氣象資料
        target_elements = [
            "平均溫度",
            "體感溫度",
            "相對濕度",
            "天氣現象",
            "降雨機率",
            "風向",
            "蒲福風級",
            "紫外線指數",
            "天氣預報綜合描述"
        ]
```

- 使用正則表達式解析天氣描述
- 智能補全缺失資料
- 格式化數值顯示

4. 資料呈現設計

```
def format_weather_data(self, data):
    # 定義星期對照表
    weekday_map = {
        0: "一", 1: "二", 2: "三", 3: "四", 4: "五", 5: "六", 6: "日"
    }
        # 整理數據按日期分組
    daily_data = {}
    for weather in data['weather']:
        date = weather['time'].split(' ')[0]
        period = weather['time'].split(' ')[1]
        if date not in daily_data:
            daily_data[date] = {'白天': None, '晚上': None}
        daily_data[date][period] = weather
```

- 使用字典結構組織天氣資料
- 按日期和時段分類資料
- 自動處理日期格式轉換

6. 地震資料處理

```
def format_earthquake_data(self, data):
    # 表格標題
    headers = ["發生時間", "震央位置", "規模", "深度", "最大震度", "地區", "描述"]
    header_widths = [100, 110, 50, 70, 40, 70, 200]
```

- 使用固定寬度設計表格
- 自動計算單元格高度
- 支援長文字換行顯示

7. 錯誤處理機制

```
try:
    result = self.api.getEarthquake(mode_type)
    if result['status']:
        self.format_earthquake_data(result['data'])
    else:
        messagebox.showerror("錯誤", result['message'])
except TypeError as e:
    print(f"TypeError 發生: {str(e)}")
    messagebox.showerror("錯誤", f"處理資料時發生錯誤: {str(e)}")
```

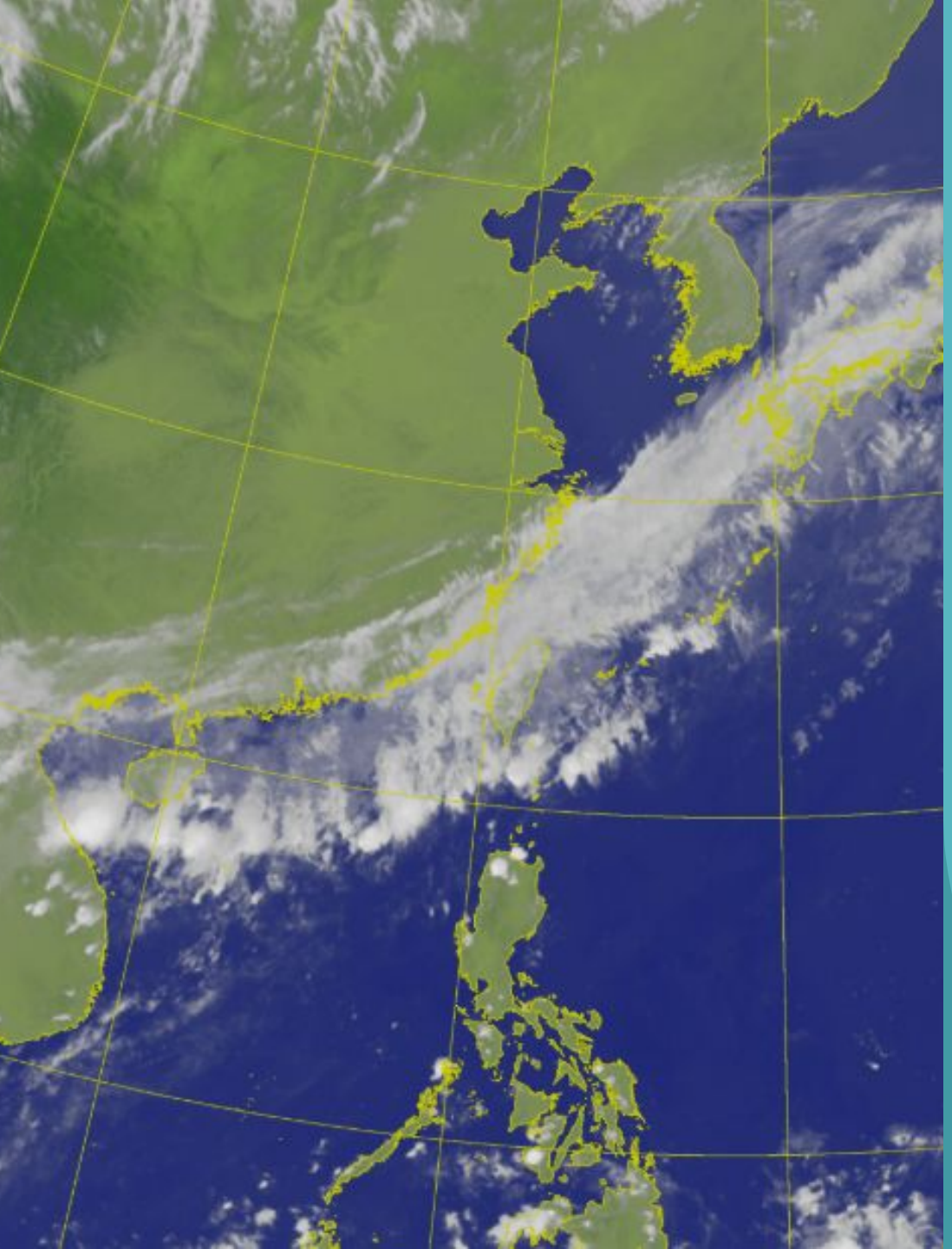
- 完整的異常處理流程
- 使用者友善的錯誤提示
- 錯誤日誌記錄

8. 主題切換功能

```
    def change_appearance_mode(self, new_mode):
        ctk.set_appearance_mode(new_mode)
        self.root.update_idletasks()
        self.root.update()


- 支援系統/淺色/深色主題
- 即時更新介面外觀
- 保持使用者設定
```

9. 打包設定

```
# 匯出命令(無控制台)
# pyinstaller --onefile --name "氣象預報系統 Windows/iOS
v0.1.0" --noconsole <your_script.py>

# 匯出命令(有控制台)
# pyinstaller --onefile --name "氣象預報系統 Windows/iOS
v0.1.0 debug" <your_script.py>

- 提供兩種打包模式
- 支援跨平台打包
- 可自訂版本號和檔名
```



報告完畢
感謝您的聆聽