SiNGULAR

奇點創意

程 式 創 客 教 室

機器人 / AI人工智慧 / 程式語言

Join Singular!
Be a super inventor!

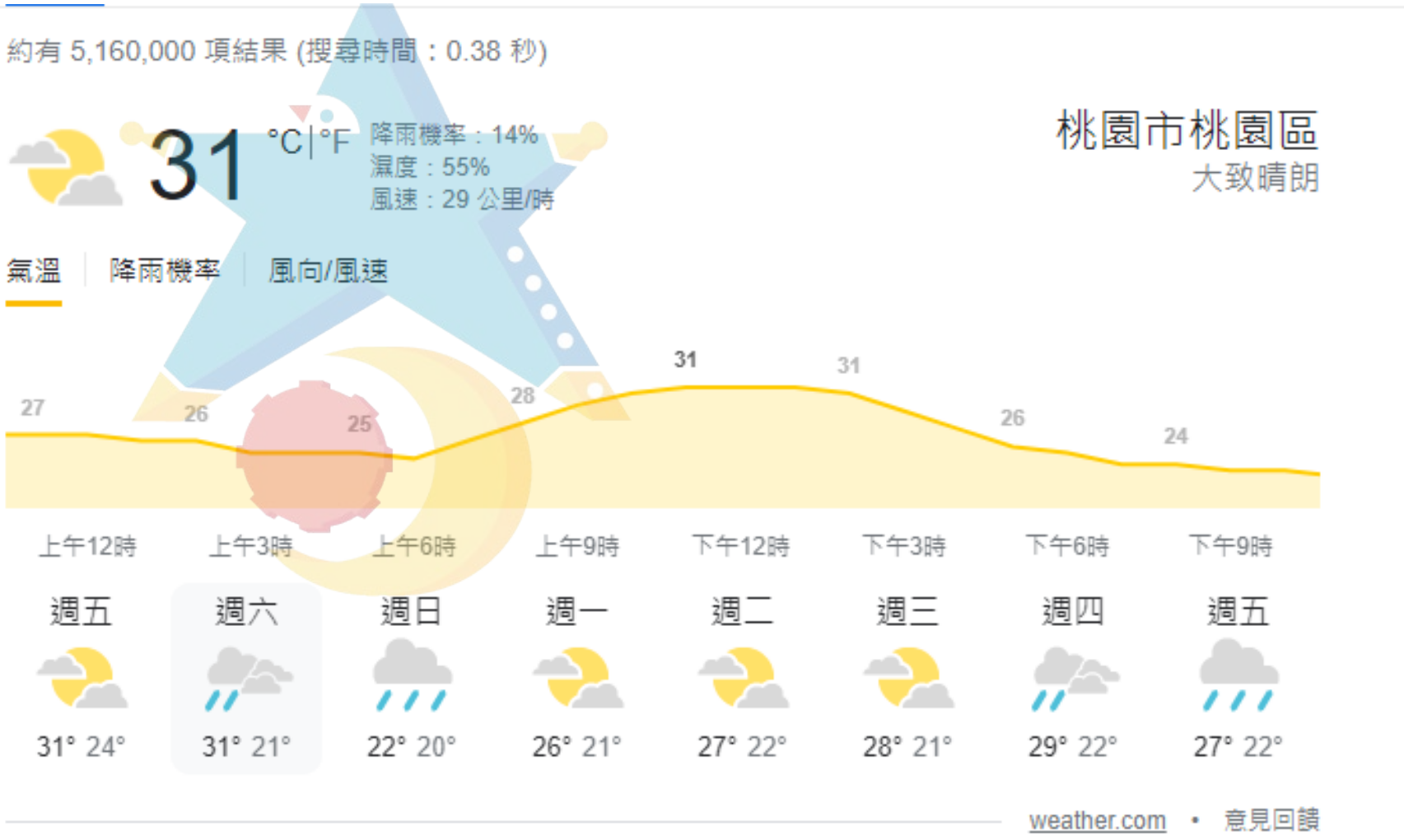
class12

Maker + Coder = Singular Super Inventor

Web API ： 天氣預報

圖表視覺化

- 新增程式


約有 5,160,000 項結果 (搜尋時間：0.38 秒)
31 °C | °F
降雨機率：14%
濕度：55%
風速：29 公里/時
桃園市桃園區
大致晴朗
氣溫
降雨機率
風向/風速
27
26
25
28
31
31
26
24
上午12時
上午3時
上午6時
上午9時
下午12時
下午3時
下午6時
下午9時
週五 31° 24°
週六 31° 21°
週日 22° 20°
週一 26° 21°
週二 27° 22°
週三 28° 21°
週四 29° 22°
週五 27° 22°
weather.com
意見回饋

獲得未來的天氣

```python
import requests

############################定義常數############################
API_KEY = "XXX"  # 替換為你的 API Key
BASE_URL = "http://api.openweathermap.org/data/2.5/forecast?"
# 5 Day / 3 Hour Forecast API URL
UNITS = "metric"  # 使用公制單位
LANG = "zh_tw"  # 使用繁體中文
```

發送請求

```
##########################主程式#############################
city_name = "Taipei"  # 可以改為任何城市名稱

# 構建請求 URL
send_url = f"{BASE_URL}q={city_name}&appid={API_KEY}&units={UNITS}&lang={LANG}"

print(f"發送的 URL：{send_url}")  # 印出發送的 URL
response = requests.get(send_url)
response.raise_for_status()  # 檢查請求是否成功
info = response.json()
```

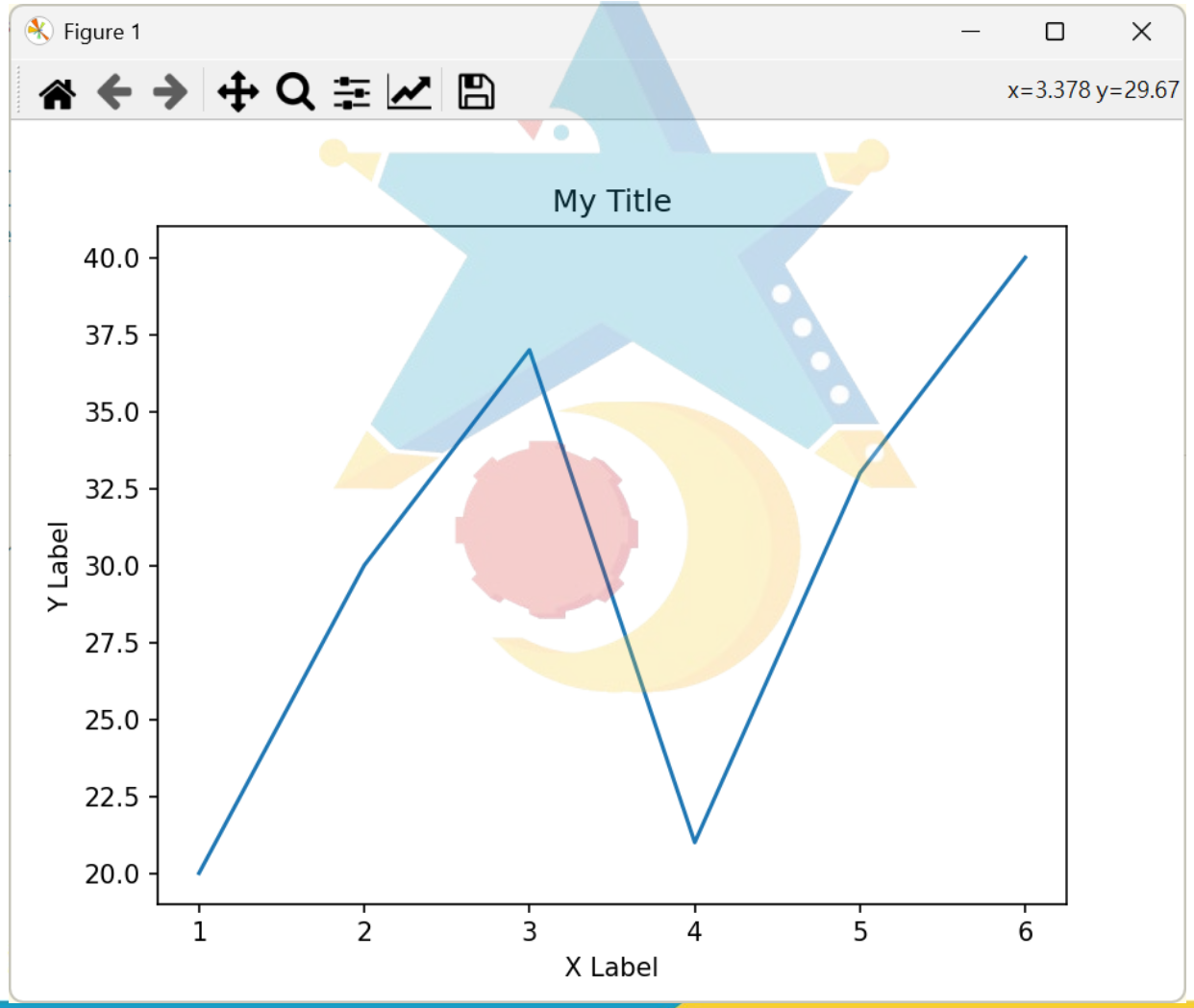
解析獲得資料內容

```
##########################主程式#############################
...省略...
info = response.json()

if "city" in info:
    # 處理並顯示天氣預報
    for forecast in info["list"]:
        dt_txt = forecast["dt_txt"]  # 預報時間
        temp = forecast["main"]["temp"]  # 預報溫度
        weather_description = forecast["weather"][0]["description"]  # 天氣描述
        print(f"{dt_txt} - 溫度: {temp} 度, 天氣狀況: {weather_description}")
else:
    print("找不到該城市或無法獲取天氣資訊")
```

繪製圖表

- 新增程式

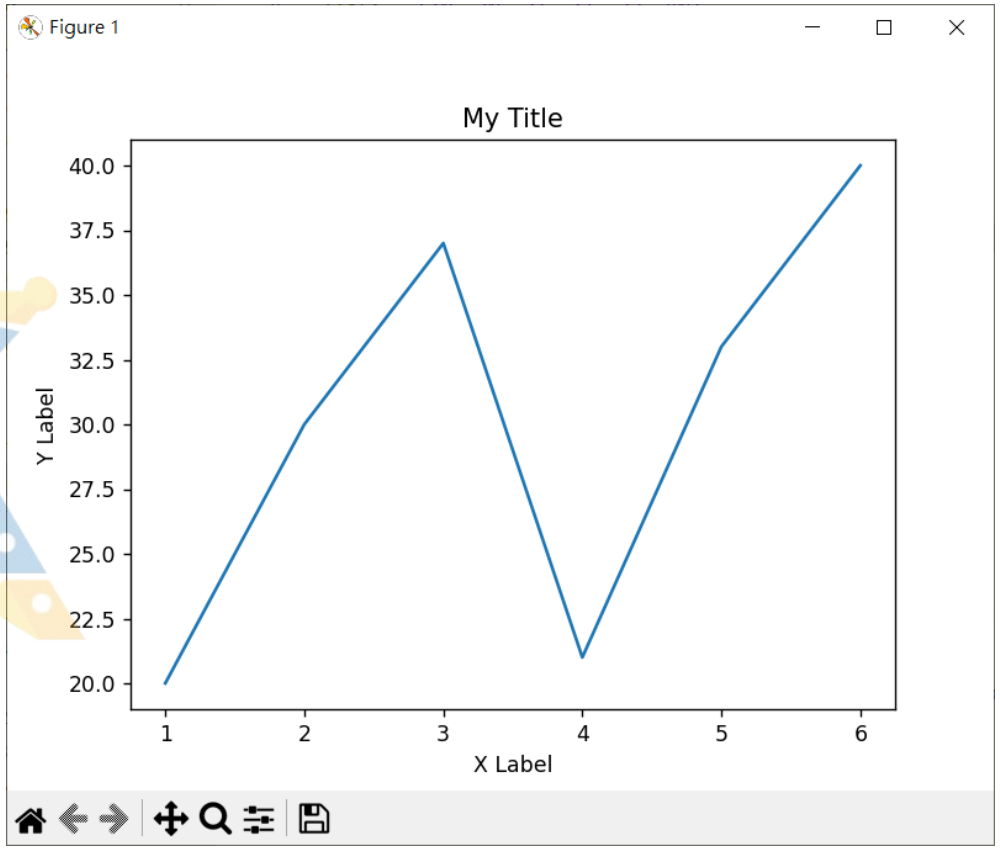


安裝Matplotlib

- pip install matplotlib -U

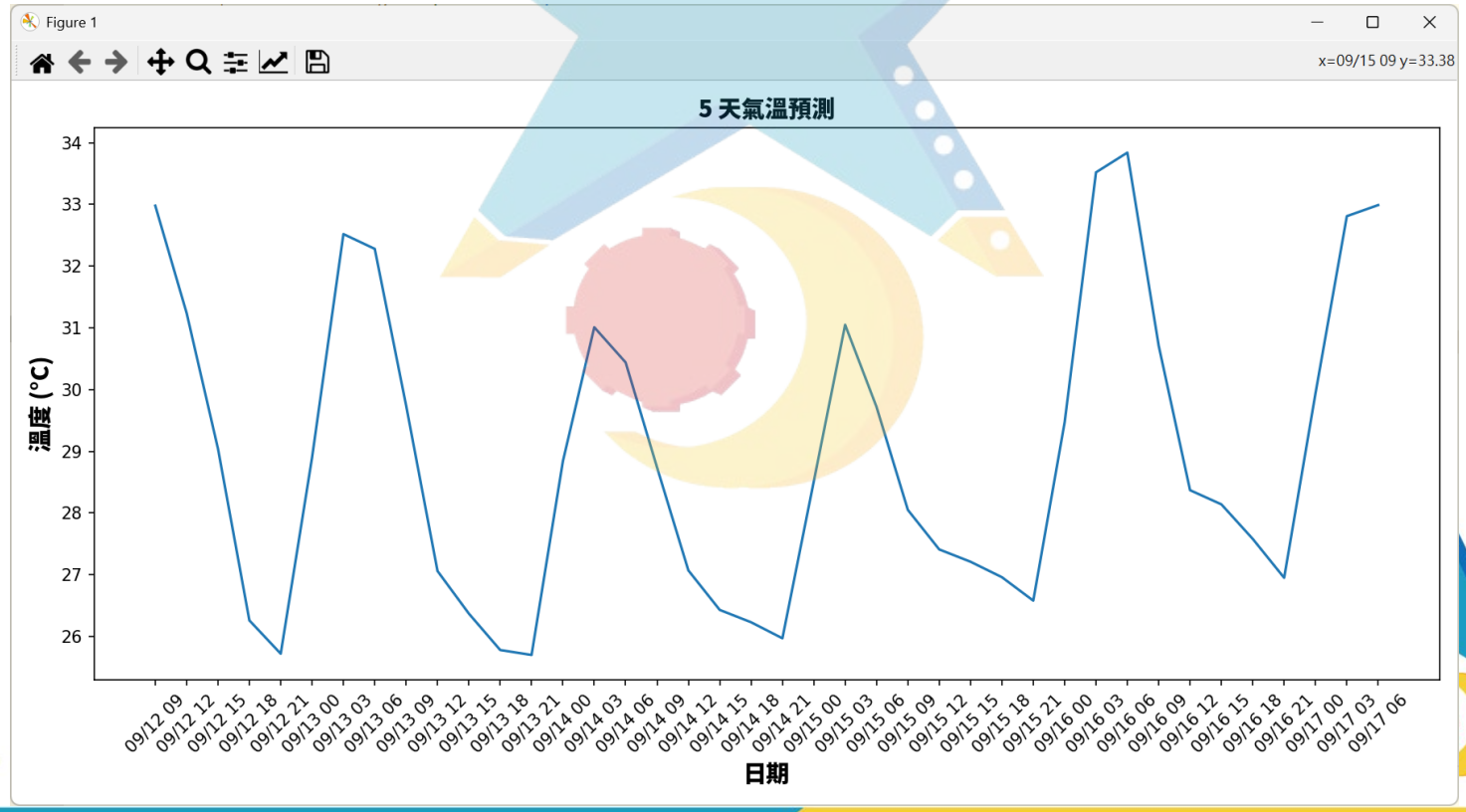
Matplotlib.pyplot

```
import matplotlib.pyplot as plt
# 定義 X 軸和 Y 軸的數據列表
listX = [1, 2, 3, 4, 5, 6]
listY = [20, 30, 37, 21, 33, 40]
# 創建圖表和軸
fig, ax = plt.subplots()  # 創建圖表和軸對象
ax.plot(listX, listY)  # 使用軸對象繪製圖表
# 設置圖表標籤和標題
ax.set_xlabel("X Label")  # 設置 X 軸標籤
ax.set_ylabel("Y Label")  # 設置 Y 軸標籤
ax.set_title("My Title")  # 設置圖表標題
plt.show()  # 顯示圖表
```



試試看

- 複製"獲得未來天氣"的程式進行修改
- 將天氣預測溫度資料轉換成折線圖



匯入模組及設定工作目錄

```
import requests
import datetime
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
import os
import sys

##########################設定工作目錄##########################
os.chdir(sys.path[0])
```

新增x，y數據清單

```
##############################主程式##############################
…省略…
info = response.json()

# 準備繪圖數據
xlist = []  # 準備 x 軸數據
ylist = []  # 準備 y 軸數據
```

將獲得的數據存入

```
##########################主程式############################
…省略…
ylist = []  # 準備 y 軸數據
if "city" in info:
    # 處理並顯示天氣預報
    for forecast in info["list"]:
        dt_txt = forecast["dt_txt"]  # 預報時間
        temp = forecast["main"]["temp"]  # 預報溫度
        # 格式化時間
        time = datetime.datetime.strptime(dt_txt, "%Y-%m-%d %H:%M:%S").strftime("%m/%d %H")
        xlist.append(time)
        ylist.append(temp)
        print(f"{time} 的溫度是 {temp} 度")
```

搜尋中文字體並顯示於圖表

```
########################主程式############################
...省略...
########################繪製圖表#########################
# https://fonts.google.com/
font = FontProperties(fname="NotoSansTC-Black.otf", size=14)  # 設定字型這樣才能顯示中文
fig, ax = plt.subplots(figsize=(12, 6))  # 設定圖表大小, 單位是英寸
ax.plot(xlist, ylist)  # 使用軸對象繪製圖表
ax.set_title("5 天氣溫預測", fontproperties=font)
ax.set_ylabel("溫度 (°C)", fontproperties=font)
ax.set_xlabel("日期", fontproperties=font)
```

旋轉標籤、將圖片存程檔案

```
##############################主程式##############################
...省略...
##############################繪製圖表##############################
...省略...
ax.set_xlabel("日期", fontproperties=font)
plt.xticks(rotation=45)  # 旋轉 x 軸標籤以避免重疊
plt.tight_layout()  # 自動調整佈局
fig.savefig("weather_forecast.png")  # 儲存圖表
plt.show()
```

將圖片放到應用程式當中

- 複製上一個程式進行修改

試試看

- 已經將圖表存為圖片
- 已經學過在畫布上顯示圖片
- 已經學過按鈕連動特定指令
- 流程：
  - 開始>按下按鈕>觸發指令>指令當中更改畫布大小為圖片大小>將圖片放到畫布當中>完成

匯入視窗模組

```
import requests
import datetime
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
import os
import sys
from ttkbootstrap import *
from PIL import Image, ImageTk
```

新增指令

- 把原本"主程式"的內容裝入draw_graph指令當中，將show改為close來節省記憶體因為我只需要用到圖片

```
##############################定義函數##############################
def draw_graph():
    city_name = "Taipei"   # 可以改為任何城市名稱
    …省略…
    fig.savefig("weather_forecast.png")   # 儲存圖表
    plt.close()  # 關閉圖表以釋放記憶體
```

加入畫布設定及讀取圖片

```
##########################定義函數#############################
def draw_graph():
    …省略…
    plt.close()  # 關閉圖表以釋放記憶體

    # 在 Canvas 上顯示圖片
    image = Image.open("weather_forecast.png")
    img = ImageTk.PhotoImage(image)

    # 重新設定畫布大小
    canvas.config(width=image.width, height=image.height)
    canvas.create_image(image.width // 2, image.height // 2, image=img)  # 在畫布上顯示圖片
    canvas.image = img  # 保持對圖片的引用，防止被垃圾回收
```

新增顯示GUI視窗

```
############################定義函數############################
...省略...
############################建立視窗############################
window = tk.Tk()
window.title("Weather App")

############################創建畫布############################
canvas = Canvas(window, width=0, height=0, bg="white")
canvas.grid(row=0, column=0, padx=10, pady=10)
```

新增顯示GUI視窗

```
########################創建畫布#############################
...省略...
########################設定字型#############################
font_size = 20
window.option_add("*font", ("Helvetica", font_size))

########################設定主題#############################
style = Style(theme="minty")
style.configure("my.TButton", font=("Helvetica", font_size))

########################建立按鈕#############################
draw_button = Button(window, text="顯示圖表", command=draw_graph, style="my.TButton")
draw_button.grid(row=1, column=0, padx=10, pady=10)

########################運行應用程式#########################
window.mainloop()
```