天氣資料概述：

名稱：地面測站最近 30 天觀測資料-30 天觀測資料（以小時為單位）

網址：https://opendata.cwa.gov.tw/dataset/climate/C-B0024-001

主要欄位：觀測時間, 測站氣壓, 溫度, 相對濕度, 風速, 風向, 降水量, 日照時數

目的：想要做 Youbike 數量預測，希望可以有天氣資料，讓預測模型更精準

讀取檔案並將資料轉換為 DataFrame（只取臺北市測站的觀測資料）

```
# 讀取 JSON 檔案
file_path = 'C-B0024-001.json'
with open(file_path, 'r', encoding='UTF-8') as f:
    data = json.load(f)
```

[2] ✓ 0.1s

```
# 提取台北測站的數據，包含日期時間和氣象要素
taipei_data = []
for location in data['cwaopendata']['resources']['resource']['data']['surfaceObs']['location']:
    if location['station']['StationName'] == "臺北":
        for obs in location['stationObsTimes']['stationObsTime']:
            entry = {
                "DateTime": obs["DateTime"],
                "AirPressure": obs["weatherElements"].get("AirPressure"),
                "AirTemperature": obs["weatherElements"].get("AirTemperature"),
                "RelativeHumidity": obs["weatherElements"].get("RelativeHumidity"),
                "WindSpeed": obs["weatherElements"].get("WindSpeed"),
                "WindDirection": obs["weatherElements"].get("WindDirection"),
                "Precipitation": obs["weatherElements"].get("Precipitation"),
                "SunshineDuration": obs["weatherElements"].get("SunshineDuration"),
            }
            taipei_data.append(entry)

# 將數據轉換為 DataFrame
df = pd.DataFrame(taipei_data)
df['DateTime'] = pd.to_datetime(df['DateTime'])
```

[3] ✓ 0.0s

資料觀察，發現日照時數欄位有許多空值

```
df.head()
```

[4] ✓ 0.0s Python

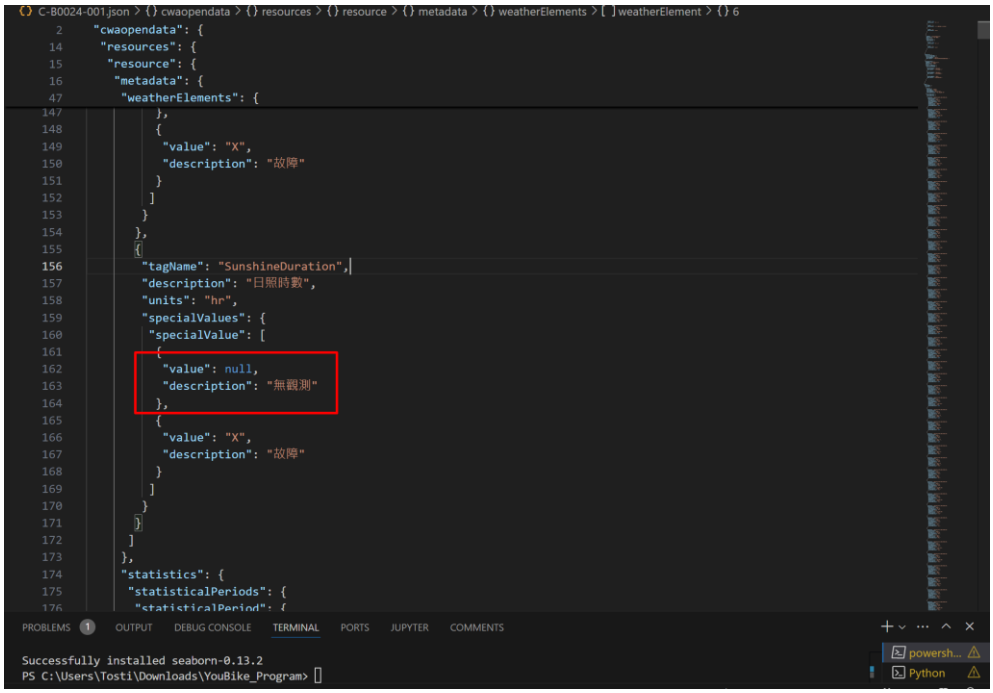
	DateTime	AirPressure	AirTemperature	RelativeHumidity	WindSpeed	WindDirection	Precipitation	SunshineDuration
0	2024-09-26 01:00:00+08:00	1007.0	27.8	82	0.9	西南,SW	0.0	None
1	2024-09-26 02:00:00+08:00	1006.9	27.7	82	0.9	北,N	0.0	None
2	2024-09-26 03:00:00+08:00	1006.7	27.7	83	0.6	西,W	0.0	None
3	2024-09-26 04:00:00+08:00	1006.8	27.7	82	1.4	西,W	0.0	None
4	2024-09-26 05:00:00+08:00	1007.1	27.6	83	0.4	風向不定,Variable	0.0	None

```
df.count()
```

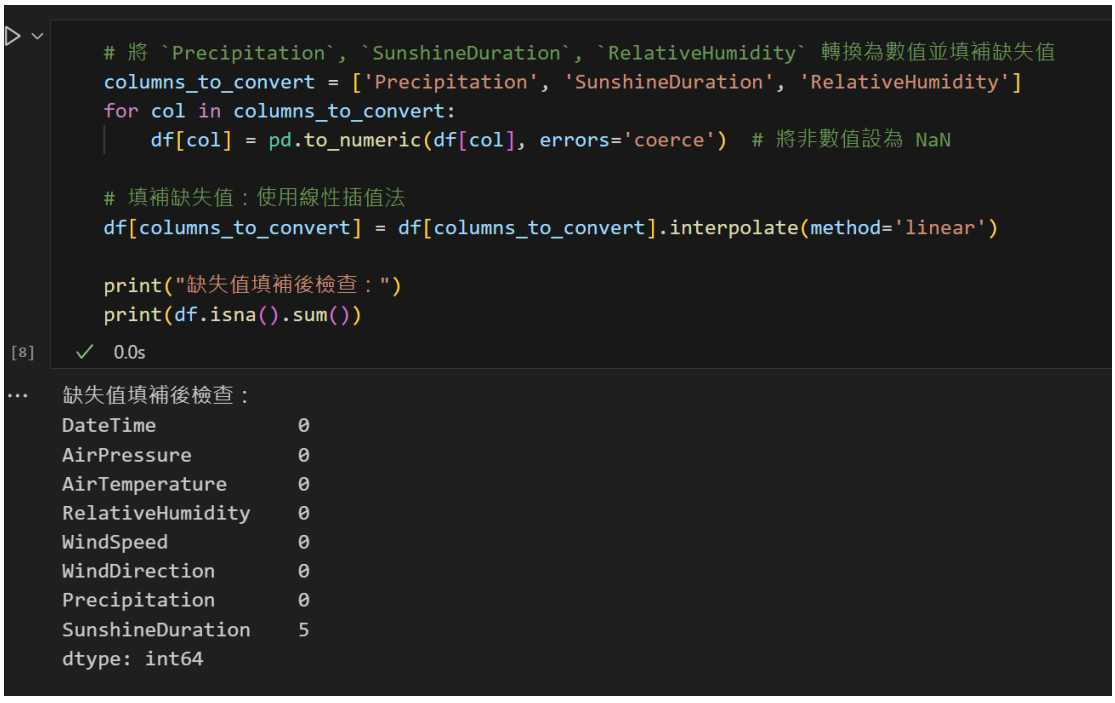
[6] ✓ 0.0s Python

```
DateTime            744
AirPressure         744
AirTemperature      744
RelativeHumidity    744
WindSpeed           744
WindDirection       744
Precipitation       744
SunshineDuration    434
dtype: int64
```

調查資料，發現是觀測測站故障，因此無觀測



對缺失值做填補處理，感覺補零不太恰當，問 ChatGPT 有何填補方法
他建議連續的資料可以使用線性插值法填補空缺，但不知為何沒有填補完全



ChatGPT：可能是因為這些缺失值位於數據的開頭或結尾，導致線性插值無法對這些位置進行填補。線性插值只會填補內部的缺失值，而不會處理位於開頭或結尾的 NaN 值。
又學到一課啦！

於是向前後填補，這時欄位確實填補完全了

```
    # 對剩餘缺失值進行向前和向後填補
    df[columns_to_convert] = df[columns_to_convert].fillna(method='ffill').fillna(method='bfill')

    print("缺失值填補後檢查：")
    print(df.isna().sum())
```

[9] ✓ 0.0s

```
缺失值填補後檢查：
DateTime            0
AirPressure         0
AirTemperature      0
RelativeHumidity    0
WindSpeed           0
WindDirection       0
Precipitation       0
SunshineDuration    0
dtype: int64
C:\Users\Tosti\AppData\Local\Temp\ipykernel_8476\427469855.py:2: FutureWarning: DataFrame.fillna with 'met
  df[columns_to_convert] = df[columns_to_convert].fillna(method='ffill').fillna(method='bfill')
```

依據以下的標準來推估天氣狀況：

晴天（Clear）：當降水量為 0 且有日照時數。
多雲（Partly Cloudy）：當降水量為 0，日照時數為 0，相對濕度小於 80%。
陰天（Cloudy）：當降水量為 0，日照時數為 0，且相對濕度超過 80%。
小雨（Light Rain）：當降水量介於 0.1mm 和 10mm 之間。
大雨（Heavy Rain）：當降水量超過 10mm。

```
def classify_weather(row):
    if row['Precipitation'] == 0:
        if row['SunshineDuration'] > 0:
            return 'Clear'
        elif row['RelativeHumidity'] >= 80:
            return 'Cloudy'
        else:
            return 'Partly Cloudy'
    elif 0.1 <= row['Precipitation'] < 10:
        return 'Light Rain'
    elif row['Precipitation'] >= 10:
        return 'Heavy Rain'
    return 'Unknown'

# 新增 WeatherCondition 欄位並儲存每筆資料的天氣狀況
df['WeatherCondition'] = df.apply(classify_weather, axis=1)
```

2] ✓ 0.0s

```
weather_counts = df['WeatherCondition'].value_counts()

print("各天氣狀況的數量：")
print(weather_counts)
```

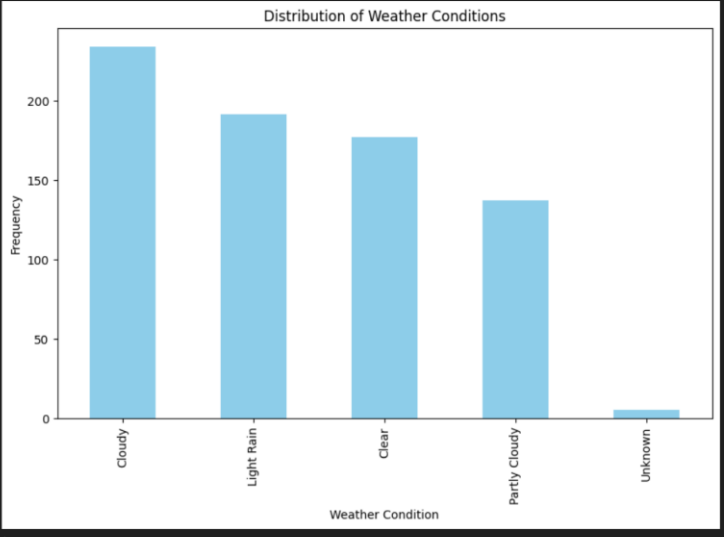
3] ✓ 0.0s

```
各天氣狀況的數量：
WeatherCondition
Cloudy          234
Light Rain      191
Clear           177
Partly Cloudy   137
Unknown           5
Name: count, dtype: int64
```

簡單資料視覺化一下

```
df['WeatherCondition'] = df.apply(classify_weather, axis=1)

plt.figure(figsize=(10, 6))
df['WeatherCondition'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Distribution of Weather Conditions')
plt.xlabel('Weather Condition')
plt.ylabel('Frequency')
plt.show()
```

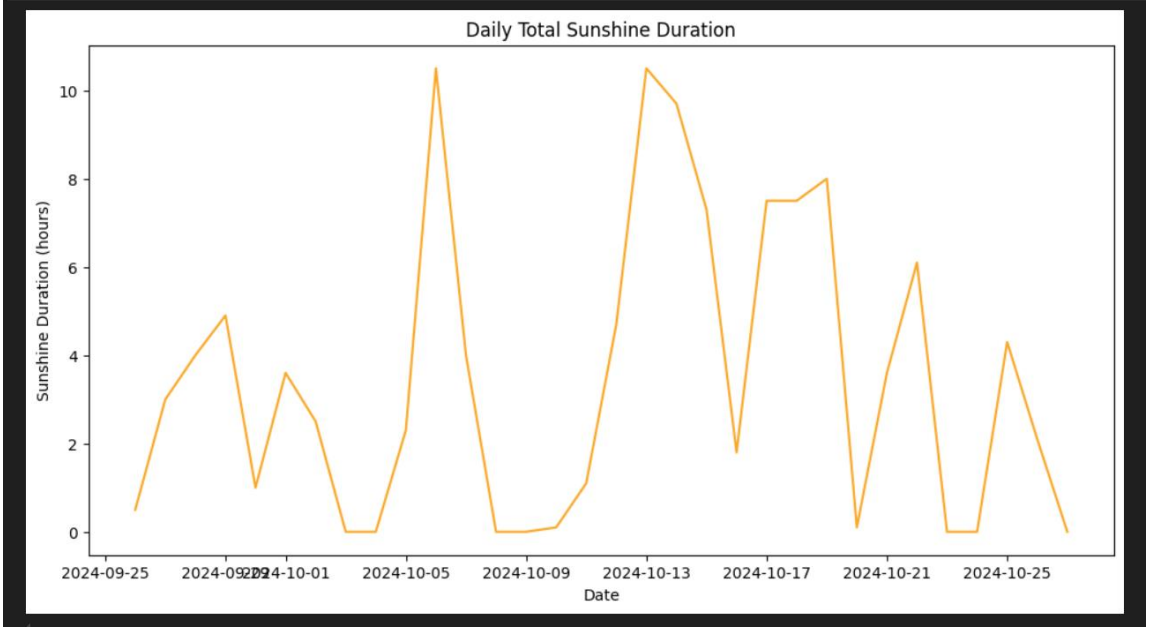
✓ 0.1s



來看一下每日總日照時數趨勢

```
# 每日總日照時數趨勢圖
# 新增日期欄位並計算每日總日照時數
df['Date'] = df['DateTime'].dt.date
daily_sunshine = df.groupby('Date')['SunshineDuration'].sum()

plt.figure(figsize=(12, 6))
daily_sunshine.plot(kind='line', color='orange')
plt.title('Daily Total Sunshine Duration')
plt.xlabel('Date')
plt.ylabel('Sunshine Duration (hours)')
plt.show()
```

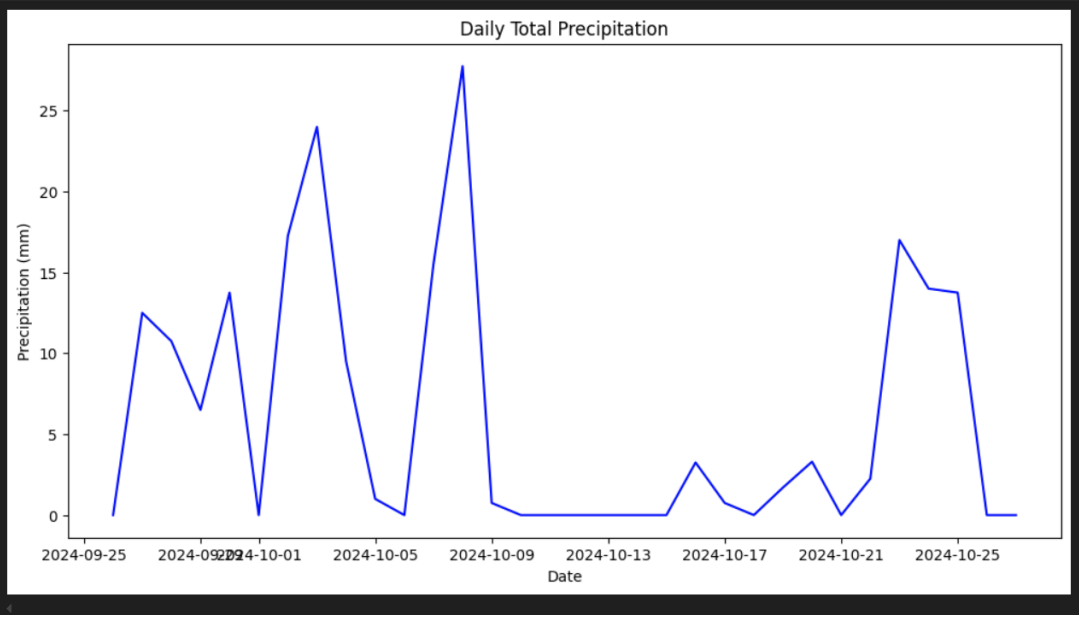
✓ 0.3s



每日總降雨量趨勢圖

```python
# 計算每日總降水量
daily_precipitation = df.groupby('Date')['Precipitation'].sum()

plt.figure(figsize=(12, 6))
daily_precipitation.plot(kind='line', color='blue')
plt.title('Daily Total Precipitation')
plt.xlabel('Date')
plt.ylabel('Precipitation (mm)')
plt.show()
```

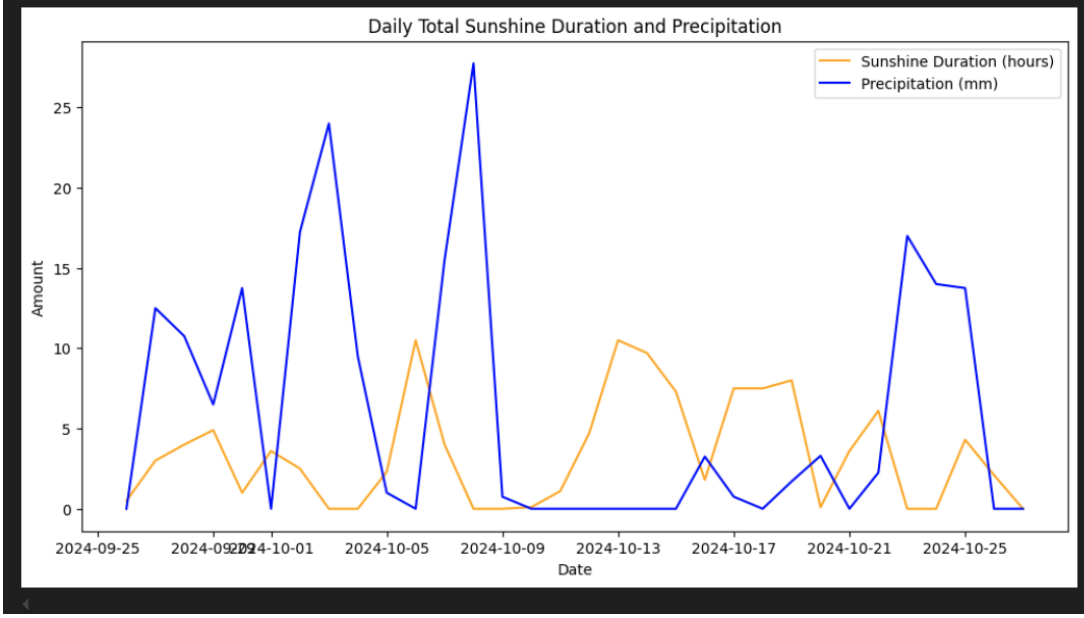
✓ 0.2s



將兩趨勢疊合看一下關聯性

```python
# 繪製在同一張圖上
plt.figure(figsize=(12, 6))
plt.plot(daily_sunshine, label='Sunshine Duration (hours)', color='orange')
plt.plot(daily_precipitation, label='Precipitation (mm)', color='blue')
plt.title('Daily Total Sunshine Duration and Precipitation')
plt.xlabel('Date')
plt.ylabel('Amount')
plt.legend()
plt.show()
```

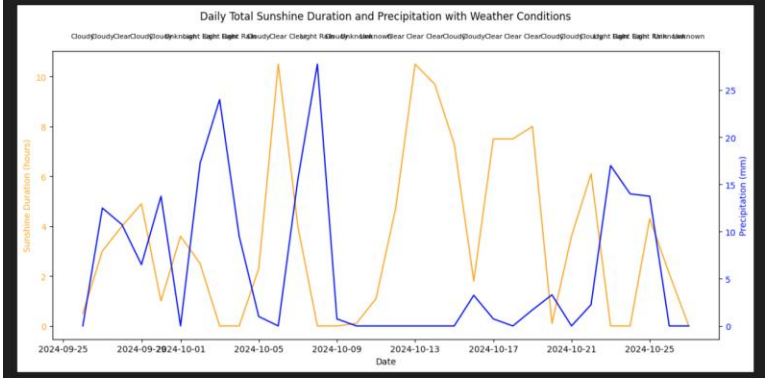
✓ 0.3s



這時突然觀察到一個現象，雖然降雨量與日照時數看得出關聯，但為何日照時數的變化幅度會那麼小呢？明明這一個月內也有許多大晴天的日子？
這時才發現單位不同共用縱軸的影響求助了 ChatGPT，學了如何畫雙縱軸圖：

```python
# 計算每日的主要天氣狀況
daily_weather = df.groupby('Date')['WeatherCondition'].agg(lambda x: x.mode()[0])

# 繪製圖表
fig, ax1 = plt.subplots(figsize=(12, 6))

# 使用左側 y 軸顯示日照時數
ax1.plot(daily_sunshine, label='Sunshine Duration (hours)', color='orange')
ax1.set_xlabel('Date')
ax1.set_ylabel('Sunshine Duration (hours)', color='orange')
ax1.tick_params(axis='y', labelcolor='orange')

# 創建右側 y 軸顯示降水量
ax2 = ax1.twinx()
ax2.plot(daily_precipitation, label='Precipitation (mm)', color='blue')
ax2.set_ylabel('Precipitation (mm)', color='blue')
ax2.tick_params(axis='y', labelcolor='blue')

# 在圖表的上方標示每日的主要天氣狀況，橫向顯示
for date, weather in daily_weather.items():
    ax1.text(date, daily_sunshine.max() + 1, weather, fontsize=8, ha='center', va='bottom', rotation=0)

# 添加圖例和標題
fig.suptitle('Daily Total Sunshine Duration and Precipitation with Weather Conditions')
fig.tight_layout()
plt.show()
```

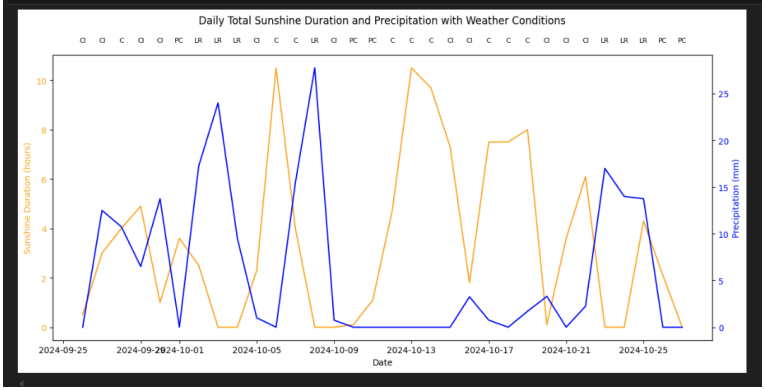
✓ 1.1s



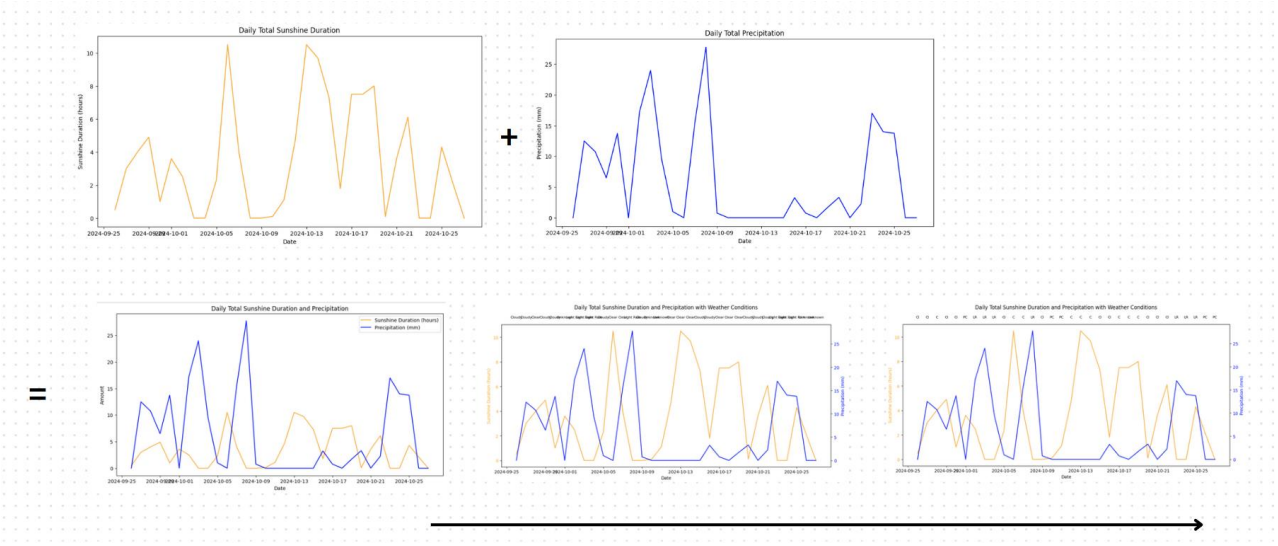
上方標籤文字過長遮擋到彼此，將標籤簡寫一下：

```python
# 計算每日的主要天氣狀況並轉換為代號
weather_code = {
    'Clear': 'C',
    'Cloudy': 'Cl',
    'Partly Cloudy': 'PC',
    'Light Rain': 'LR',
    'Heavy Rain': 'HR',
    'Unknown': 'Unk'
}
daily_weather = df.groupby('Date')['WeatherCondition'].agg(lambda x: x.mode()[0]).map(weather_code)

# 繪製圖表
fig, ax1 = plt.subplots(figsize=(12, 6))

# 使用左側 y 軸顯示日照時數
ax1.plot(daily_sunshine, label='Sunshine Duration (hours)', color='orange')
ax1.set_xlabel('Date')
ax1.set_ylabel('Sunshine Duration (hours)', color='orange')
ax1.tick_params(axis='y', labelcolor='orange')

# 創建右側 y 軸顯示降水量
ax2 = ax1.twinx()
ax2.plot(daily_precipitation, label='Precipitation (mm)', color='blue')
ax2.set_ylabel('Precipitation (mm)', color='blue')
ax2.tick_params(axis='y', labelcolor='blue')

# 在圖表的上方標示每日的主要天氣狀況代號
for date, weather in daily_weather.items():
    ax1.text(date, daily_sunshine.max() + 1, weather, fontsize=8, ha='center', va='bottom', rotation=0)

# 添加圖例和標題
fig.suptitle('Daily Total Sunshine Duration and Precipitation with Weather Conditions')
fig.tight_layout()
plt.show()
```

✓ 0.2s



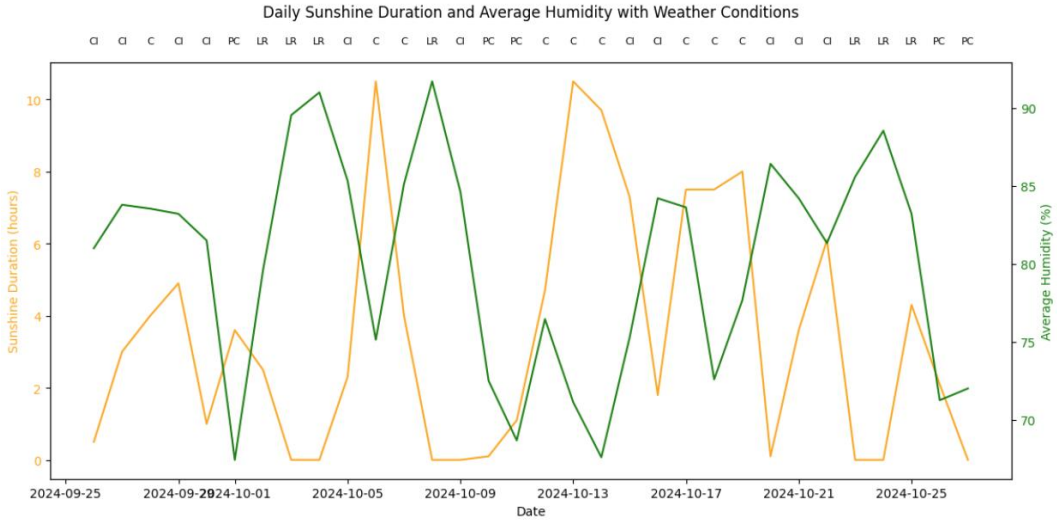
日照、降水量趨勢圖過程總結：



濕度與溫度的關聯性：

程式碼類似上面展示的範例，就不多做贅述了。

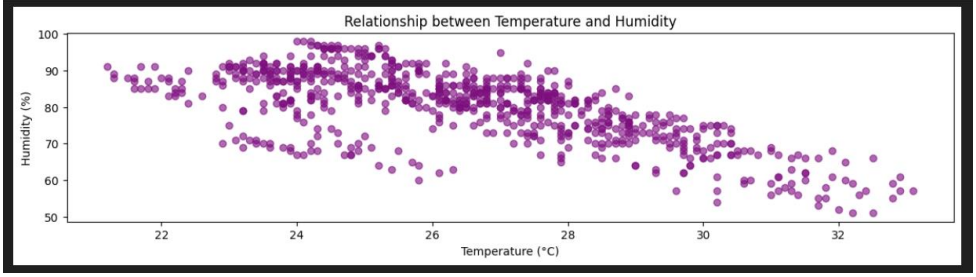
可以看出濕度與日照時數的關聯性



溫度與濕度散佈圖，大致能看出溫度越高濕度越低。

```
# 創建溫度與濕度的關係散佈圖
plt.figure(figsize=(14, 3))
plt.scatter(df['AirTemperature'], df['RelativeHumidity'], alpha=0.6, color='purple')
plt.title('Relationship between Temperature and Humidity')
plt.xlabel('Temperature (°C)')
plt.ylabel('Humidity (%)')
plt.show()
```

✓ 0.1s



結語

最後想透過 YouBike 資料，探討天氣狀況與腳踏車借還狀況的關聯性，但可惜 Youbike 的資料我們才開始儲存沒幾天，沒有能跟這三十天天氣資料對應的 Youbike 資料，因此只能先做天氣部分的資料清洗及簡單統計及視覺化的程式碼，等到 Youbike 資料收集三十天後再加入分析關聯性，若關聯性夠高的話，在進一步加入預測模型，提升預測腳踏車樹功能的精準度及成效。

*現在正在動態將 Youbike 資料收集進 Database