Data:

1. Tables, columns, data source

Source: [https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data/data#](https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data/data)

Temperatures:

dt:

日期 格式 XXXX-XX-XX

LandAverageTemperature:

陸地部分的平均溫度

LandAverageTemperatureUncertainty

陸地部分的平均溫度的95%信任區間

LandMaxTemperature

陸地部分的最大溫度

LandMaxTemperatureUncertainty

陸地部分的最大溫度的95%信任區間

LandMinTemperature

陸地部分的最小溫度

LandMinTemperatureUncertainty

陸地部分的最小溫度的95%信任區間

LandAndOceanAverageTemperature

全球的平均溫度

LandAndOceanAverageTemperatureUncertainty

全球的平均溫度的95%信任區間

LandTemperaturesByCity:

dt:

日期 格式 XXXX-XX-XX

AverageTemperature:

城市的平均溫度

AverageTemperatureUncertainty:

城市的平均溫度的95%信任區間

City:

該城市

Country:

城市所在國家

Latitude:

城市所在經度

Longitude

城市所在緯度

LandTemperaturesByState:

dt:

日期 格式 XXXX-XX-XX

AverageTemperature:

州/省的平均溫度

AverageTemperatureUncertainty:

州/省的平均溫度的95%信任區間

State:

該州/省

Country:

州/省所在國家

LandTemperaturesByMajorCity:

dt:

日期 格式 XXXX-XX-XX

AverageTemperature:

城市的平均溫度

AverageTemperatureUncertainty:

城市的平均溫度的95%信任區間

City:

該城市

Country:

城市所在國家

Latitude:

城市所在經度

Longitude

城市所在緯度

LandTemperaturesByCountry

dt:

日期 格式 XXXX-XX-XX

AverageTemperature:

國家的平均溫度

AverageTemperatureUncertainty:

國家的平均溫度的95%信任區間

Country:

國家

user\_table:

username:

使用者名稱

key: salt:

登入驗證相關

history\_meta\_table:

username:

使用者名稱

year\_start:

查詢的年份起始

month\_start:

查詢的月份起始

year\_end:

查詢的年份結束

month\_end:

查詢的月份結束

use\_bar:

是否使用柱狀圖

use\_month:

用年做橫軸或月做橫軸的條件

avg:

上次查詢是否包含均溫

max:

上次查詢是否包含最高溫

min:

上次查詢是否包含最低溫

history\_table:

username:

使用者名稱

search\_type:

keyword屬於哪一種類型(City,Country..etc.)

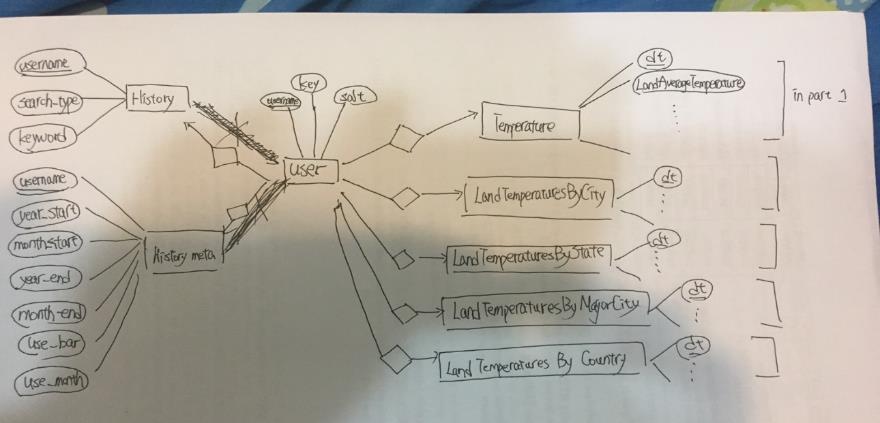
keyword:

輸入的字串

1. Normalization

我們認為原始資料已經經過標準化了，因為他已經照城市、國家等等做出區分，有符合3NF的條件

1. Draw a ER model of your data



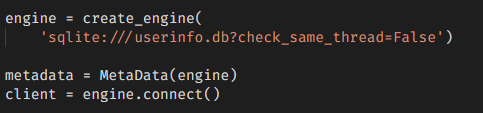
Database:

What database do you use: SQLite3.

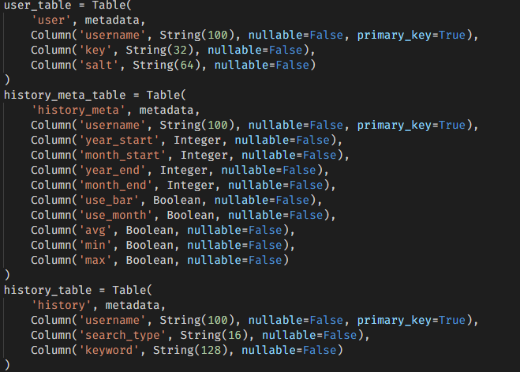
How do you maintain your database: 使用SQL Alchemy + python

How you connect your database to your application:

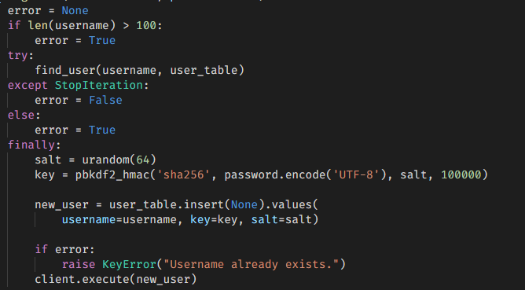
首先要先初始化一個資料庫物件



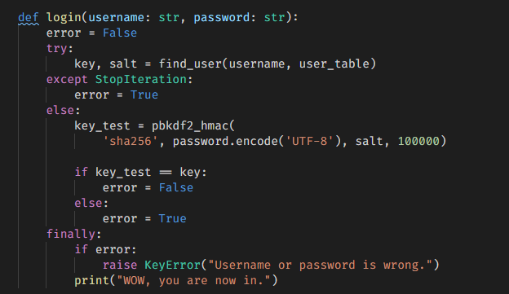
然後建立表，如果已經存在則會載入



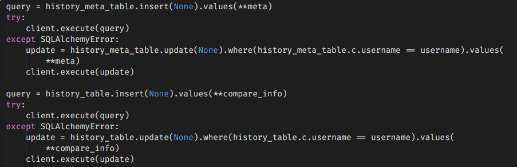
當使用者註冊時，會先檢查名稱沒被用過，然後將密碼雜湊和鹽存進資料庫中



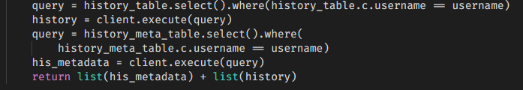
當使用者登入時， 用鹽雜湊提供的密碼進行比對



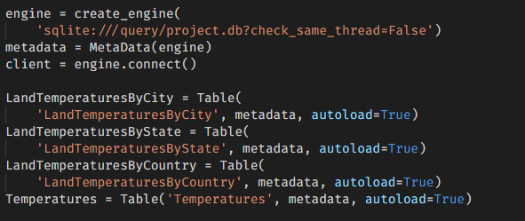
記錄使用者當次查詢參數，簡單實作了Upsert



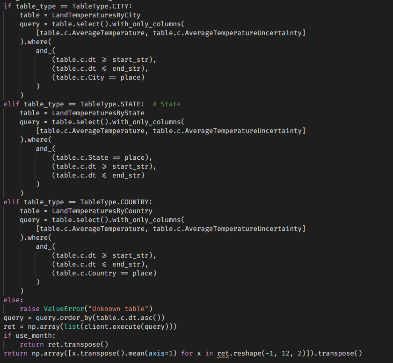
讀取上次查詢參數



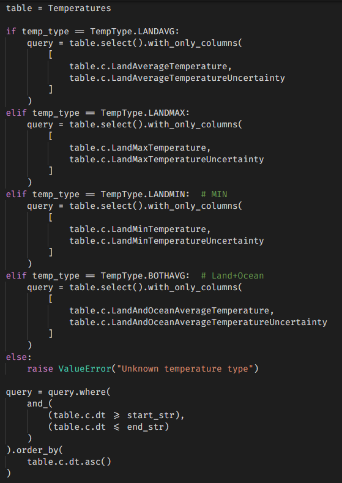
原始資料部分因為表先建過了，也不會修改，所以直接載入



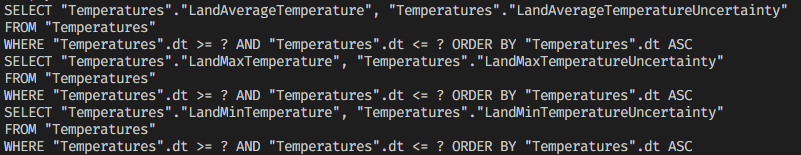
幾個主要有關資料的查詢，SQL Alchemy的語法應該很容易轉成SQL，這裡就不貼原始SQL語法。where(table.c.State == place) 就是WHERE table.State = place



這是全球資料查詢，格式稍有不同



Exception handling: 對於未知的類型都會raise一個對應的Exception, 會在 App 裡面進行處理(大多是不做任何操作)，而我們對資料庫查詢不是使用拼接SQL 字串，我們是使用SQL Alchemy提供的API，在這前提下他會使用參數化查詢(可以print(query)出來看)，那麼就可以避免SQLi的攻擊



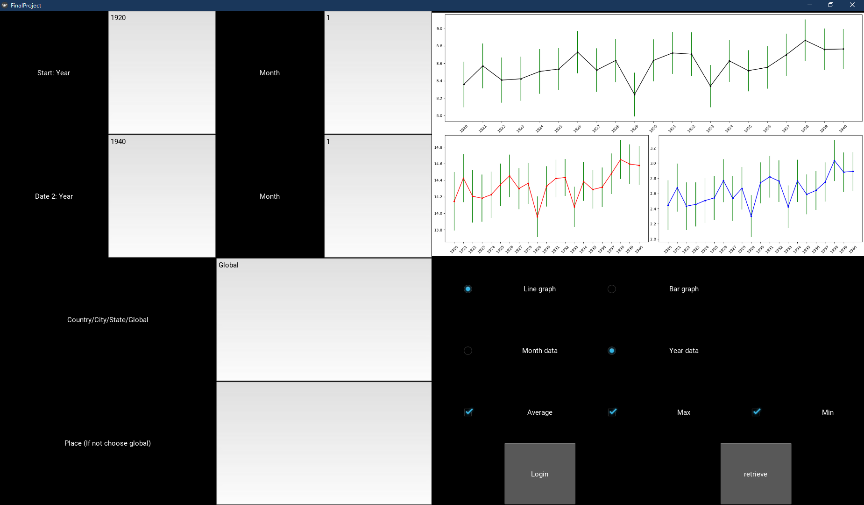
Application:

Interface:

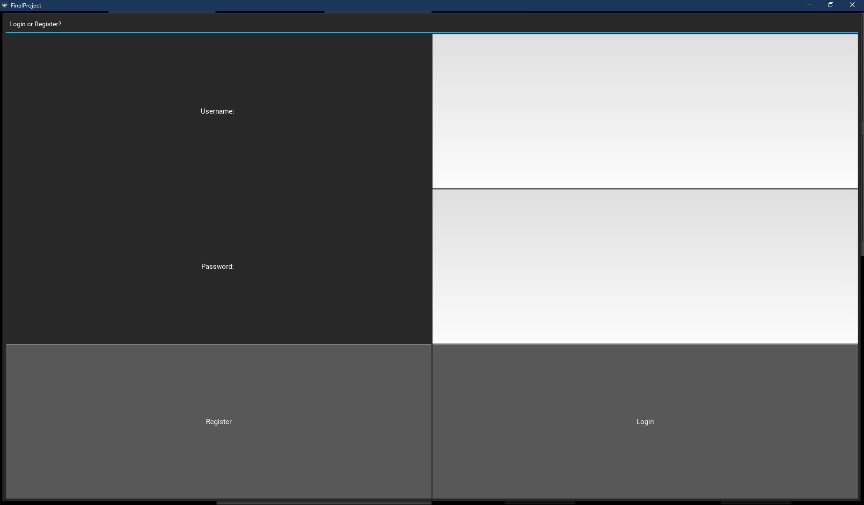
給予使用者選取及輸入的畫面，在選取完畢後送出，就會在右上方顯示所選資料圖片，根據查詢不同可能只有平均，或同時有1到3張圖(平均、最高以及最低)，不過東西要手輸有點不是很友善，但是因為介面不算分就沒弄下拉選單了



綠色部分為Uncertainty



簡易登入的彈出視窗



Function:

Register: 註冊自己的ID 及密碼

Login: 使用者利用帳號密碼登入，密碼以雜湊和鹽的方式儲存於資料庫中，以避免明文儲存密碼的風險。

History keep: 使用者的每筆查詢都會自動被insert(或是update )到table，使之能在下次登入後看到之前的查詢紀錄。

Retrieve: 若使用者有登入，按下後自動填入上次的查詢參數

Temperature search: 選取起始年月及終止年月，選出要搜尋的型態(City, Country ,State等) ，再輸入要搜尋的地名，就可以產出一張上述資料形成的圖表。

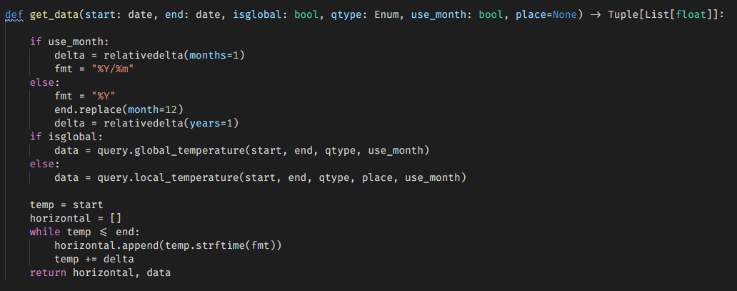
額外功能:

1. 可以複選圖表類型，已進行比較
2. 預設是由月當橫軸，可以改以年來做觀察
3. 可以改折線圖為長條圖

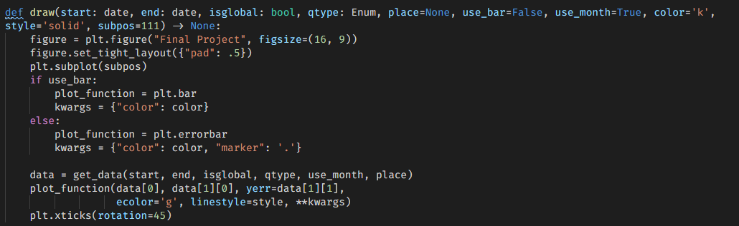
How make it possible:

與資料庫相關的操作已經在上面介紹過了，所以這裡會以GUI和畫圖為主

首先透過上面敘述的query取得資料，在這裡整理成時間軸+數值及誤差

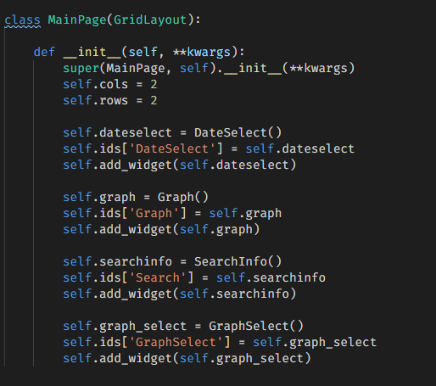


然後畫圖

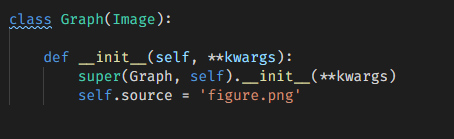


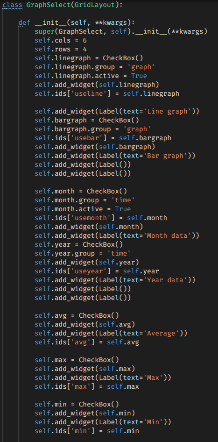
GUI 元件部分

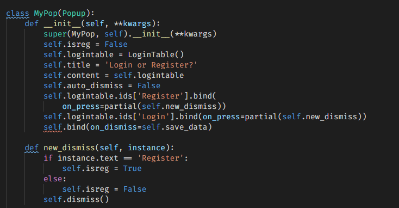






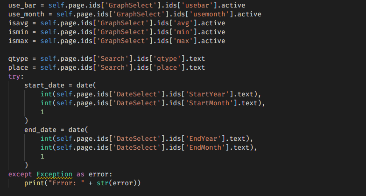




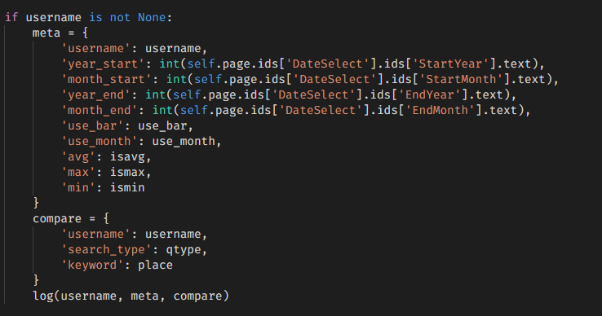




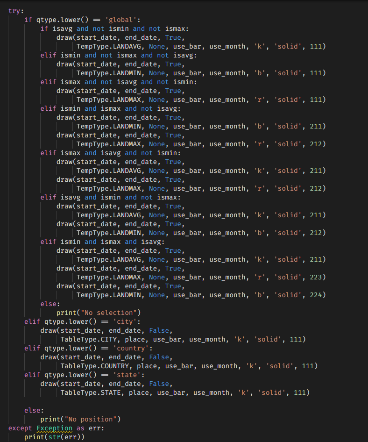
送出查詢時，讀取表單資料，同時檢查日期資料正確性



若使用者有登入，則紀錄資料

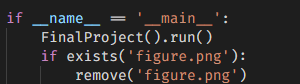


這裡應該要用其他方法做的..，若沒錯誤則儲存圖片並重新讀取至GUI中





程式關閉時清除圖片



Other:

Progress Compare:

what was the expected progress:

從6月開始做，到6月中差不多做完

the actual progress

6月做電腦動畫與特效期末，做到6月底馬上要考計組期末，所以考完計組期末後的7月初才開始準備和查資料。

細節可以參考commit history

<https://github.com/TheLurkingCat/Database-term-project/commits/master>

Problem meet:

1. 畫GUI真麻煩，還好 kivy 提供的文檔還不錯，可以慢慢讀慢慢弄
2. 期末考剛好卡到原先deadline，還好後來有延期，不然可能就沒有GUI

Contribution:

0716061: GUI, graph plot, query, user info, debug, report, video

0716014: query, report, proposal, presentation

Repo:

<https://github.com/TheLurkingCat/Database-term-project>

Discussion:

<https://hackmd.io/_9CXKGTLSt-wpB4xG372yA>