Data:

1. Tables, columns, data source

   Source: 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

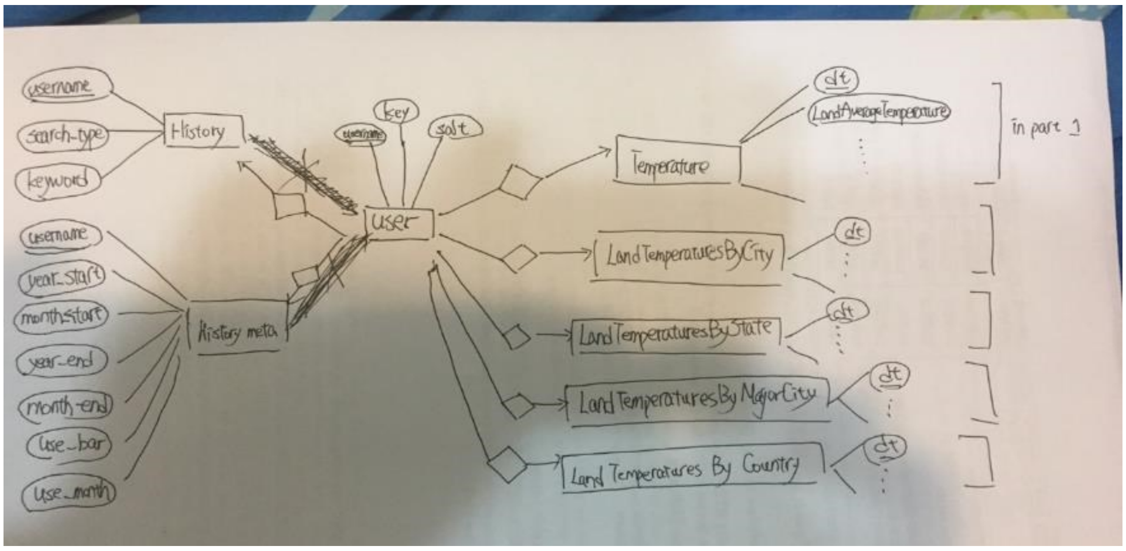
history_meta_table:

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

2. 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
3. Draw a ER model of your data



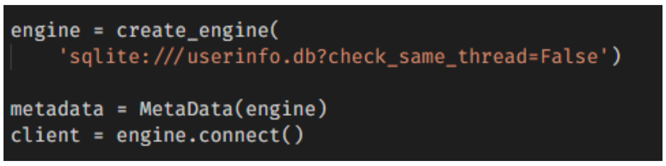
Database:

What database do you use: SQLite3.

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

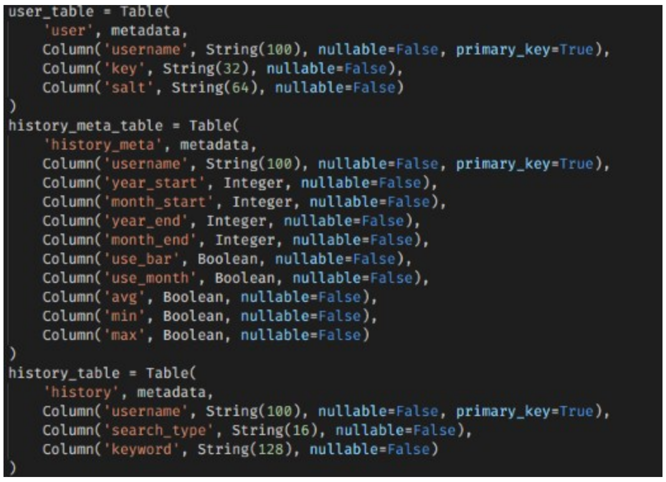
How you connect your database to your application:

首先要先初始化一個資料庫物件



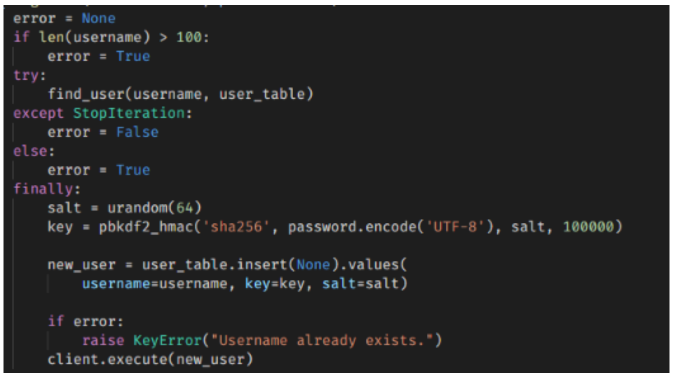
```
engine = create_engine(
    'sqlite:///userinfo.db?check_same_thread=False')

metadata = MetaData(engine)
client = engine.connect()
```

然後建立表，如果已經存在則會載入



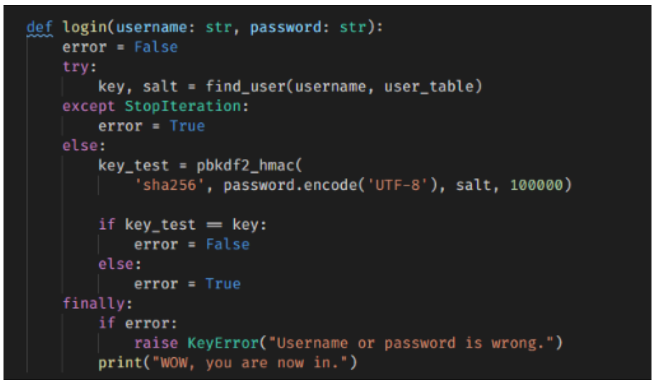
```
user_table = Table(
    'user', metadata,
    Column('username', String(100), nullable=False, primary_key=True),
    Column('key', String(32), nullable=False),
    Column('salt', String(64), nullable=False)
)
history_meta_table = Table(
    'history_meta', metadata,
    Column('username', String(100), nullable=False, primary_key=True),
    Column('year_start', Integer, nullable=False),
    Column('month_start', Integer, nullable=False),
    Column('year_end', Integer, nullable=False),
    Column('month_end', Integer, nullable=False),
    Column('use_bar', Boolean, nullable=False),
    Column('use_month', Boolean, nullable=False),
    Column('avg', Boolean, nullable=False),
    Column('min', Boolean, nullable=False),
    Column('max', Boolean, nullable=False)
)
history_table = Table(
    'history', metadata,
    Column('username', String(100), nullable=False, primary_key=True),
    Column('search_type', String(16), nullable=False),
    Column('keyword', String(128), nullable=False)
)
```

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



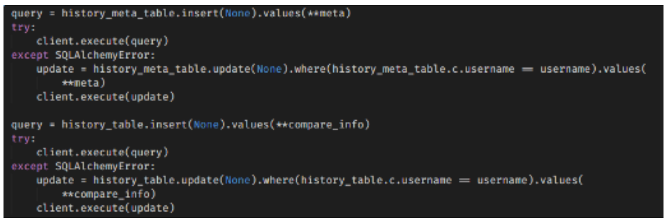
```
error = None
if len(username) > 100:
    error = True
try:
    find_user(username, user_table)
except StopIteration:
    error = False
else:
    error = True
finally:
    salt = urandom(64)
    key = pbkdf2_hmac('sha256', password.encode('UTF-8'), salt, 100000)

    new_user = user_table.insert(None).values(
        username=username, key=key, salt=salt)

    if error:
        raise KeyError("Username already exists.")
    client.execute(new_user)
```

當使用者登入時， 用鹽雜湊提供的密碼進行比對



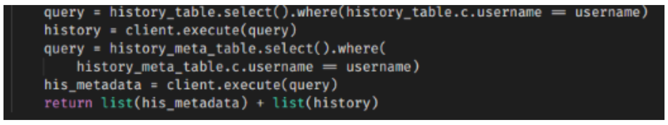
```
def login(username: str, password: str):
    error = False
    try:
        key, salt = find_user(username, user_table)
    except StopIteration:
        error = True
    else:
        key_test = pbkdf2_hmac(
            'sha256', password.encode('UTF-8'), salt, 100000)

        if key_test == key:
            error = False
        else:
            error = True
    finally:
        if error:
            raise KeyError("Username or password is wrong.")
        print("WOW, you are now in.")
```

記錄使用者當次查詢參數，簡單實作了 Upsert



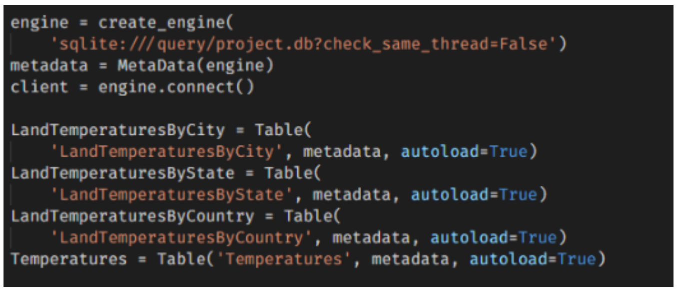
```
query = history_meta_table.insert(None).values(**meta)
try:
    client.execute(query)
except SQLAlchemyError:
    update = history_meta_table.update(None).where(history_meta_table.c.username == username).values(
        **meta)
    client.execute(update)

query = history_table.insert(None).values(**compare_info)
try:
    client.execute(query)
except SQLAlchemyError:
    update = history_table.update(None).where(history_table.c.username == username).values(
        **compare_info)
    client.execute(update)
```

讀取上次查詢參數



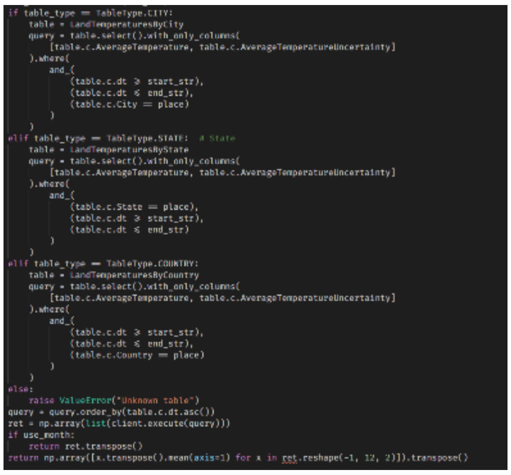
```
    query = history_table.select().where(history_table.c.username == username)
    history = client.execute(query)
    query = history_meta_table.select().where(
        history_meta_table.c.username == username)
    his_metadata = client.execute(query)
    return list(his_metadata) + list(history)
```

原始資料部分因為表先建過了，也不會修改，所以直接載入



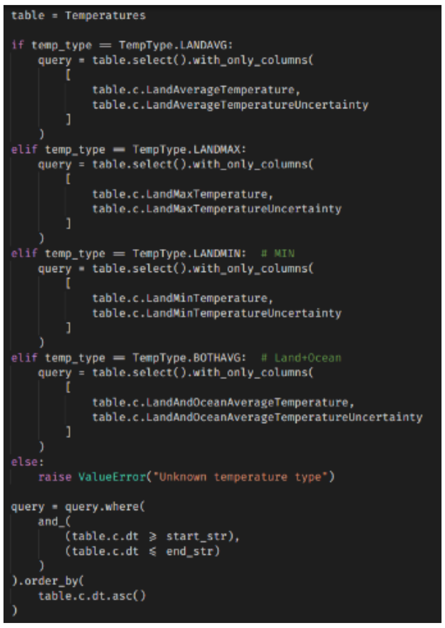
```
engine = create_engine(
    'sqlite:///query/project.db?check_same_thread=False')
metadata = MetaData(engine)
client = engine.connect()

LandTemperaturesByCity = Table(
    'LandTemperaturesByCity', metadata, autoload=True)
LandTemperaturesByState = Table(
    'LandTemperaturesByState', metadata, autoload=True)
LandTemperaturesByCountry = Table(
    'LandTemperaturesByCountry', metadata, autoload=True)
Temperatures = Table('Temperatures', metadata, autoload=True)
```

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



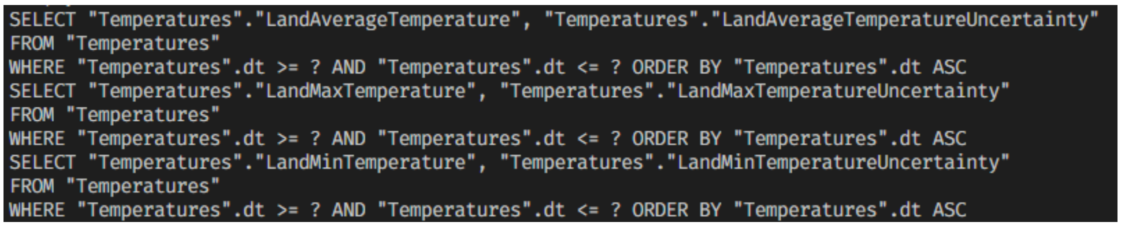
```python
if table_type == TableType.CITY:
    table = LandTemperaturesByCity
    query = table.select().with_only_columns(
        [table.c.AverageTemperature, table.c.AverageTemperatureUncertainty]
    ).where(
        and_(
            (table.c.dt >= start_str),
            (table.c.dt <= end_str),
            (table.c.City == place)
        )
    )
elif table_type == TableType.STATE:  # State
    table = LandTemperaturesByState
    query = table.select().with_only_columns(
        [table.c.AverageTemperature, table.c.AverageTemperatureUncertainty]
    ).where(
        and_(
            (table.c.State == place),
            (table.c.dt >= start_str),
            (table.c.dt <= end_str)
        )
    )
elif table_type == TableType.COUNTRY:
    table = LandTemperaturesByCountry
    query = table.select().with_only_columns(
        [table.c.AverageTemperature, table.c.AverageTemperatureUncertainty]
    ).where(
        and_(
            (table.c.dt >= start_str),
            (table.c.dt <= end_str),
            (table.c.Country == place)
        )
    )
else:
    raise ValueError("Unknown table")
query = query.order_by(table.c.dt.asc())
ret = np.array(list(client.execute(query)))
if use_month:
    return ret.transpose()
return np.array([x.transpose().mean(axis=1) for x in ret.reshape(-1, 12, 2)]).transpose()
```

這是全球資料查詢，格式稍有不同



```python
table = Temperatures

if temp_type == TempType.LANDAVG:
    query = table.select().with_only_columns(
        [
            table.c.LandAverageTemperature,
            table.c.LandAverageTemperatureUncertainty
        ]
    )
elif temp_type == TempType.LANDMAX:
    query = table.select().with_only_columns(
        [
            table.c.LandMaxTemperature,
            table.c.LandMaxTemperatureUncertainty
        ]
    )
elif temp_type == TempType.LANDMIN:  # MIN
    query = table.select().with_only_columns(
        [
            table.c.LandMinTemperature,
            table.c.LandMinTemperatureUncertainty
        ]
    )
elif temp_type == TempType.BOTHAVG:  # Land+Ocean
    query = table.select().with_only_columns(
        [
            table.c.LandAndOceanAverageTemperature,
            table.c.LandAndOceanAverageTemperatureUncertainty
        ]
    )
else:
    raise ValueError("Unknown temperature type")

query = query.where(
    and_(
        (table.c.dt >= start_str),
        (table.c.dt <= end_str)
    )
).order_by(
    table.c.dt.asc()
)
```

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

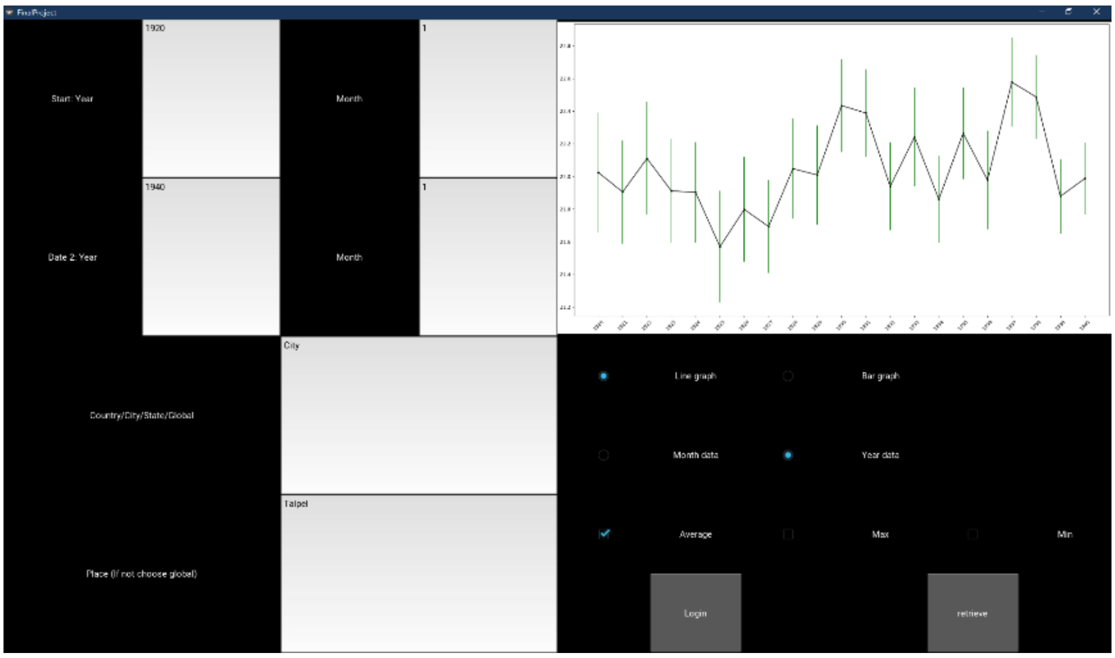


```
SELECT "Temperatures"."LandAverageTemperature", "Temperatures"."LandAverageTemperatureUncertainty"
FROM "Temperatures"
WHERE "Temperatures".dt >= ? AND "Temperatures".dt <= ? ORDER BY "Temperatures".dt ASC
SELECT "Temperatures"."LandMaxTemperature", "Temperatures"."LandMaxTemperatureUncertainty"
FROM "Temperatures"
WHERE "Temperatures".dt >= ? AND "Temperatures".dt <= ? ORDER BY "Temperatures".dt ASC
SELECT "Temperatures"."LandMinTemperature", "Temperatures"."LandMinTemperatureUncertainty"
FROM "Temperatures"
WHERE "Temperatures".dt >= ? AND "Temperatures".dt <= ? ORDER BY "Temperatures".dt ASC
```

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



FinalProject

Start Year: 1920 Month: 1

Date 2 Year: 1940 Month: 1

Country/City/State/Global: City

Place (If not choose global): Taipei

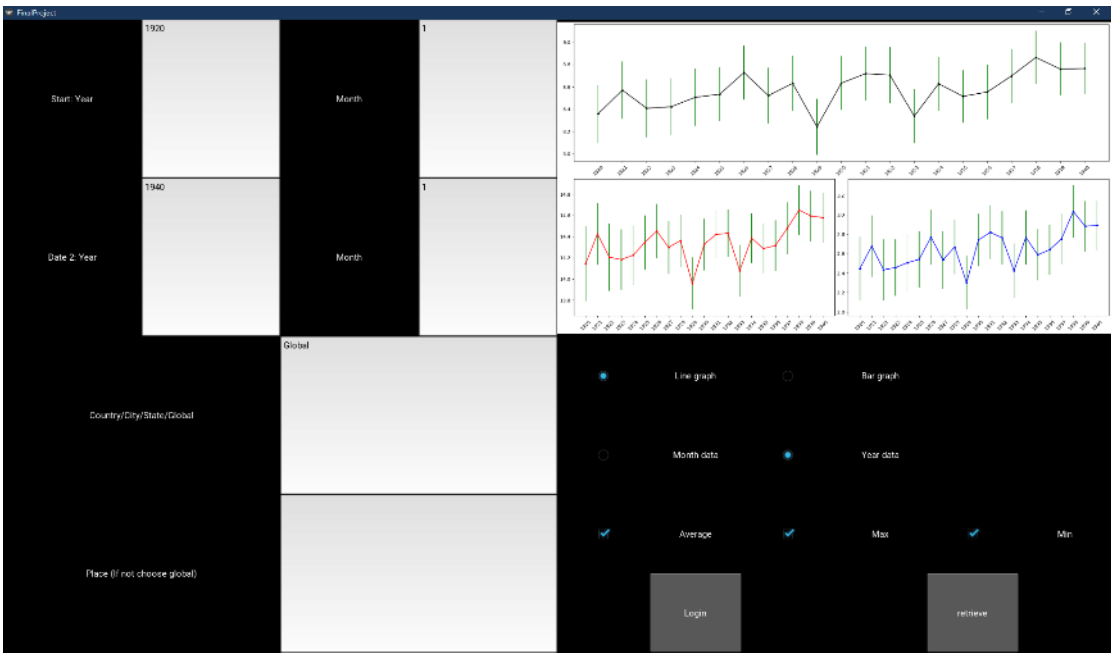
[✓ Line graph] Bar graph

Month data [✓ Year data]

[✓ Average] Max Min

Login retrieve

綠色部分為 Uncertainty



FinalProject

Start Year: 1920 Month: 1

Date 2 Year: 1940 Month: 1

Country/City/State/Global: Global

Place (If not choose global):

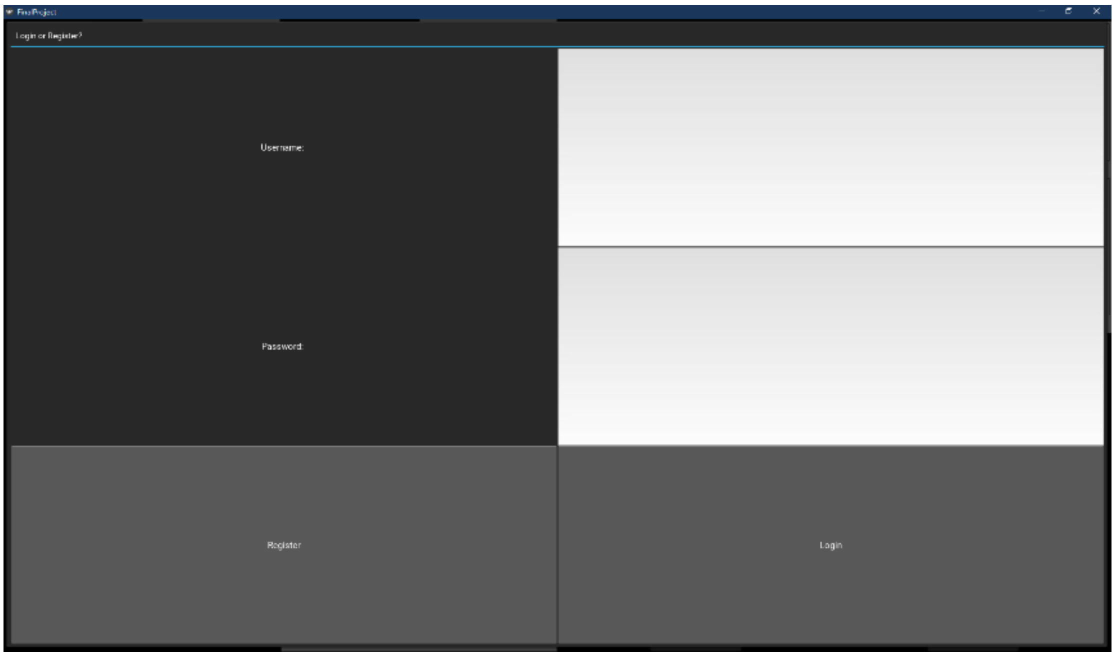
[✓ Line graph] Bar graph

Month data [✓ Year data]

[✓ Average] [✓ Max] [✓ Min]

Login retrieve

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。

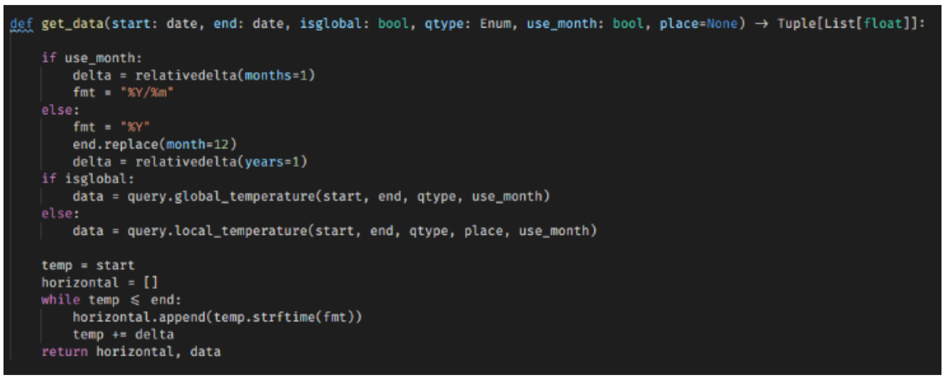
額外功能:

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:

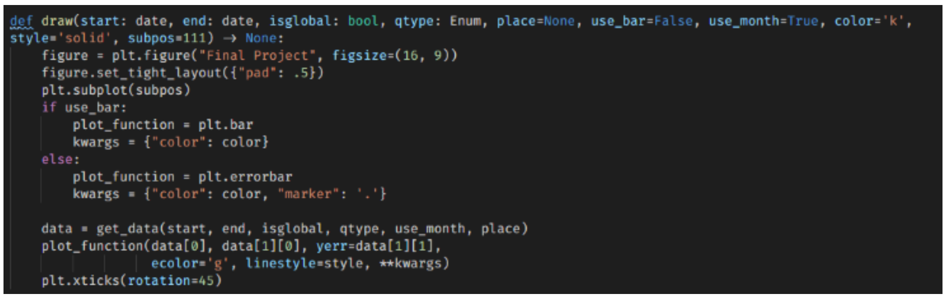
與資料庫相關的操作已經在上面介紹過了，所以這裡會以 GUI 和畫圖為主

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



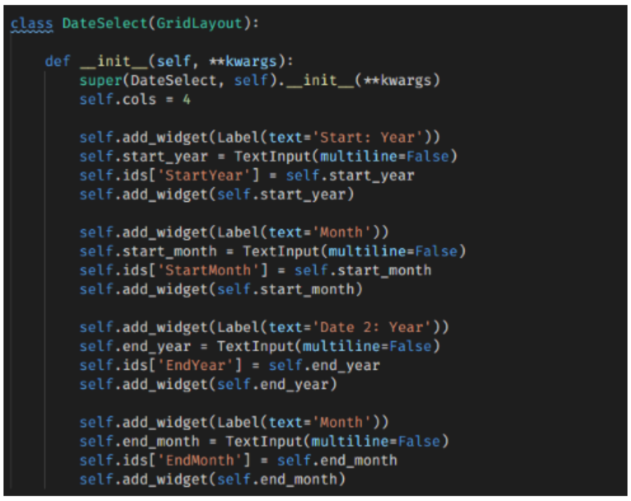
```python
def get_data(start: date, end: date, isglobal: bool, qtype: Enum, use_month: bool, place=None) → Tuple[List[float]]:

    if use_month:
        delta = relativedelta(months=1)
        fmt = "%Y/%m"
    else:
        fmt = "%Y"
        end.replace(month=12)
        delta = relativedelta(years=1)
    if isglobal:
        data = query.global_temperature(start, end, qtype, use_month)
    else:
        data = query.local_temperature(start, end, qtype, place, use_month)

    temp = start
    horizontal = []
    while temp ≤ end:
        horizontal.append(temp.strftime(fmt))
        temp += delta
    return horizontal, data
```

然後畫圖

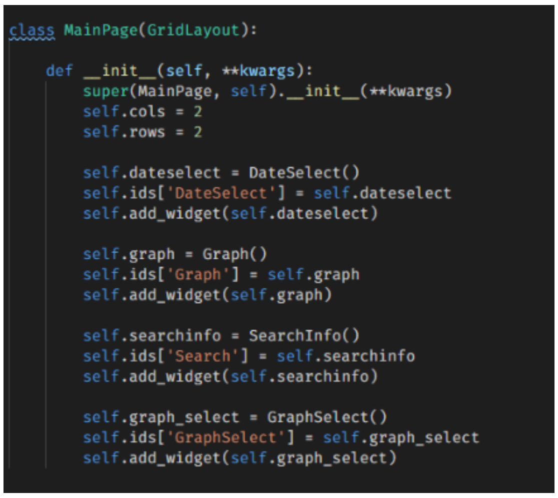


```python
def draw(start: date, end: date, isglobal: bool, qtype: Enum, place=None, use_bar=False, use_month=True, color='k',
style='solid', subpos=111) → None:
    figure = plt.figure("Final Project", figsize=(16, 9))
    figure.set_tight_layout({"pad": .5})
    plt.subplot(subpos)
    if use_bar:
        plot_function = plt.bar
        kwargs = {"color": color}
    else:
        plot_function = plt.errorbar
        kwargs = {"color": color, "marker": '.'}

    data = get_data(start, end, isglobal, qtype, use_month, place)
    plot_function(data[0], data[1][0], yerr=data[1][1],
                  ecolor='g', linestyle=style, **kwargs)
    plt.xticks(rotation=45)
```

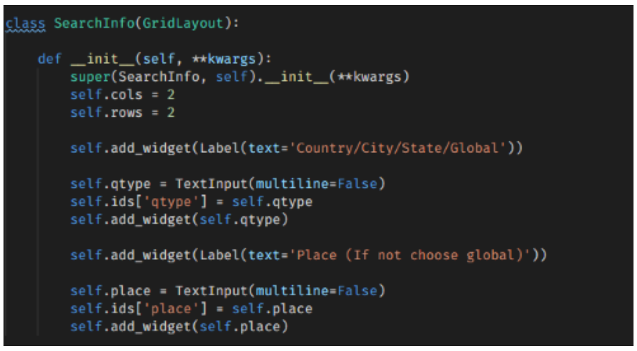
GUI 元件部分



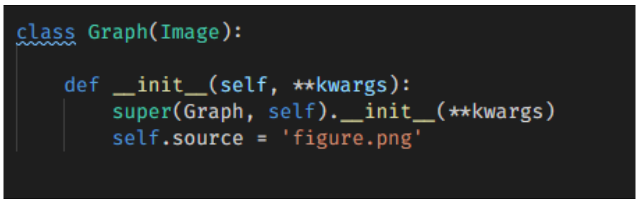
```python
class DateSelect(GridLayout):

    def __init__(self, **kwargs):
        super(DateSelect, self).__init__(**kwargs)
        self.cols = 4

        self.add_widget(Label(text='Start: Year'))
        self.start_year = TextInput(multiline=False)
        self.ids['StartYear'] = self.start_year
        self.add_widget(self.start_year)

        self.add_widget(Label(text='Month'))
        self.start_month = TextInput(multiline=False)
        self.ids['StartMonth'] = self.start_month
        self.add_widget(self.start_month)

        self.add_widget(Label(text='Date 2: Year'))
        self.end_year = TextInput(multiline=False)
        self.ids['EndYear'] = self.end_year
        self.add_widget(self.end_year)

        self.add_widget(Label(text='Month'))
        self.end_month = TextInput(multiline=False)
        self.ids['EndMonth'] = self.end_month
        self.add_widget(self.end_month)
```



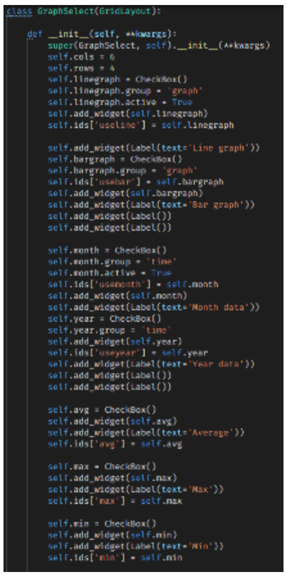
```python
class MainPage(GridLayout):

    def __init__(self, **kwargs):
        super(MainPage, self).__init__(**kwargs)
        self.cols = 2
        self.rows = 2

        self.dateselect = DateSelect()
        self.ids['DateSelect'] = self.dateselect
        self.add_widget(self.dateselect)

        self.graph = Graph()
        self.ids['Graph'] = self.graph
        self.add_widget(self.graph)

        self.searchinfo = SearchInfo()
        self.ids['Search'] = self.searchinfo
        self.add_widget(self.searchinfo)

        self.graph_select = GraphSelect()
        self.ids['GraphSelect'] = self.graph_select
        self.add_widget(self.graph_select)
```



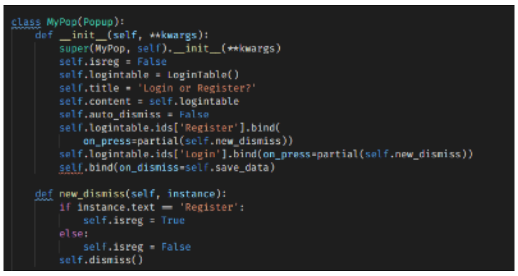
```python
class SearchInfo(GridLayout):

    def __init__(self, **kwargs):
        super(SearchInfo, self).__init__(**kwargs)
        self.cols = 2
        self.rows = 2

        self.add_widget(Label(text='Country/City/State/Global'))

        self.qtype = TextInput(multiline=False)
        self.ids['qtype'] = self.qtype
        self.add_widget(self.qtype)

        self.add_widget(Label(text='Place (If not choose global)'))

        self.place = TextInput(multiline=False)
        self.ids['place'] = self.place
        self.add_widget(self.place)
```



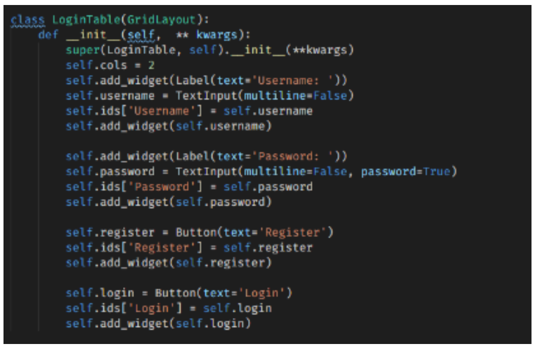
```python
class Graph(Image):

    def __init__(self, **kwargs):
        super(Graph, self).__init__(**kwargs)
        self.source = 'figure.png'
```



```python
class GraphSelect(GridLayout):

    def __init__(self, **kwargs):
        super(GraphSelect, self).__init__(**kwargs)
        self.cols = 6
        self.rows = 4
        self.linegraph = CheckBox()
        self.linegraph.group = 'graph'
        self.linegraph.active = True
        self.add_widget(self.linegraph)
        self.ids['useline'] = self.linegraph

        self.add_widget(Label(text='Line graph'))
        self.bargraph = CheckBox()
        self.bargraph.group = 'graph'
        self.ids['usebar'] = self.bargraph
        self.add_widget(self.bargraph)
        self.add_widget(Label(text='Bar graph'))
        self.add_widget(Label())
        self.add_widget(Label())

        self.month = CheckBox()
        self.month.group = 'time'
        self.month.active = True
        self.ids['usemonth'] = self.month
        self.add_widget(self.month)
        self.add_widget(Label(text='Month data'))
        self.year = CheckBox()
        self.year.group = 'time'
        self.add_widget(self.year)
        self.ids['useyear'] = self.year
        self.add_widget(Label(text='Year data'))
        self.add_widget(Label())
        self.add_widget(Label())

        self.avg = CheckBox()
        self.add_widget(self.avg)
        self.add_widget(Label(text='Average'))
        self.ids['avg'] = self.avg

        self.max = CheckBox()
        self.add_widget(self.max)
        self.add_widget(Label(text='Max'))
        self.ids['max'] = self.max

        self.min = CheckBox()
        self.add_widget(self.min)
        self.add_widget(Label(text='Min'))
        self.ids['min'] = self.min
```

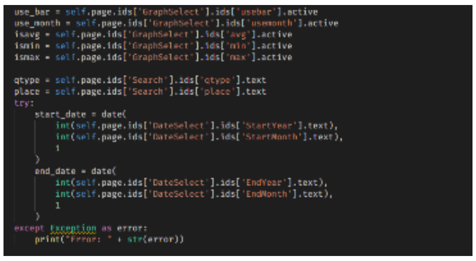


```python
class MyPop(Popup):
    def __init__(self, **kwargs):
        super(MyPop, self).__init__(**kwargs)
        self.isreg = False
        self.logintable = LoginTable()
        self.title = 'Login or Register?'
        self.content = self.logintable
        self.auto_dismiss = False
        self.logintable.ids['Register'].bind(
            on_press=partial(self.new_dismiss))
        self.logintable.ids['Login'].bind(on_press=partial(self.new_dismiss))
        self.bind(on_dismiss=self.save_data)

    def new_dismiss(self, instance):
        if instance.text == 'Register':
            self.isreg = True
        else:
            self.isreg = False
        self.dismiss()
```



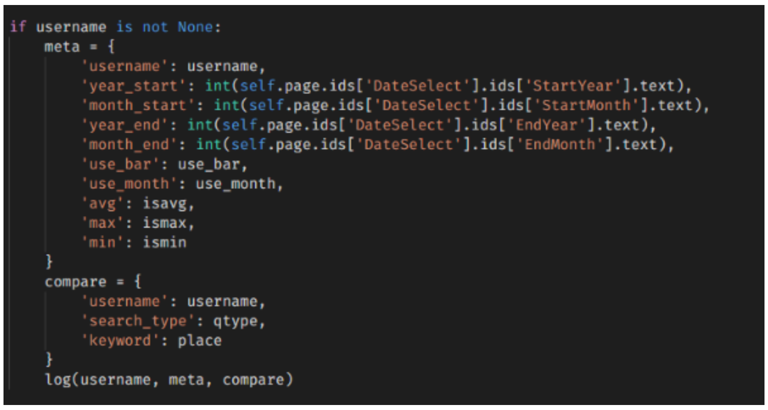
```python
class LoginTable(GridLayout):
    def __init__(self,  ** kwargs):
        super(LoginTable, self).__init__(**kwargs)
        self.cols = 2
        self.add_widget(Label(text='Username: '))
        self.username = TextInput(multiline=False)
        self.ids['Username'] = self.username
        self.add_widget(self.username)

        self.add_widget(Label(text='Password: '))
        self.password = TextInput(multiline=False, password=True)
        self.ids['Password'] = self.password
        self.add_widget(self.password)

        self.register = Button(text='Register')
        self.ids['Register'] = self.register
        self.add_widget(self.register)

        self.login = Button(text='Login')
        self.ids['Login'] = self.login
        self.add_widget(self.login)
```

送出查詢時，讀取表單資料，同時檢查日期資料正確性



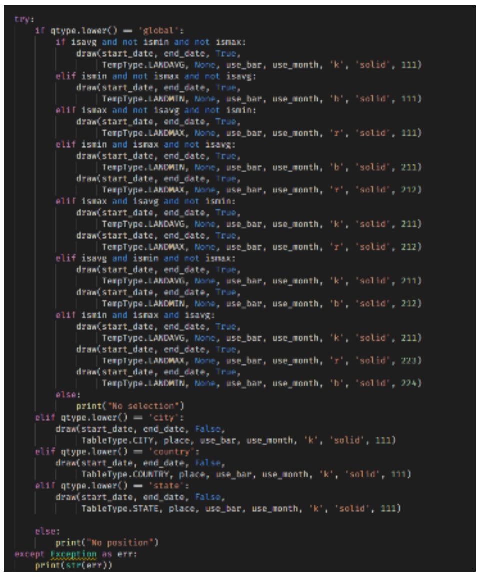
```python
use_bar = self.page.ids['GraphSelect'].ids['usebar'].active
use_month = self.page.ids['GraphSelect'].ids['usemonth'].active
isavg = self.page.ids['GraphSelect'].ids['avg'].active
ismin = self.page.ids['GraphSelect'].ids['min'].active
ismax = self.page.ids['GraphSelect'].ids['max'].active

qtype = self.page.ids['Search'].ids['qtype'].text
place = self.page.ids['Search'].ids['place'].text
try:
    start_date = date(
        int(self.page.ids['DateSelect'].ids['StartYear'].text),
        int(self.page.ids['DateSelect'].ids['StartMonth'].text),
        1
    )
    end_date = date(
        int(self.page.ids['DateSelect'].ids['EndYear'].text),
        int(self.page.ids['DateSelect'].ids['EndMonth'].text),
        1
    )
except Exception as error:
    print("Error: " + str(error))
```

若使用者有登入，則紀錄資料

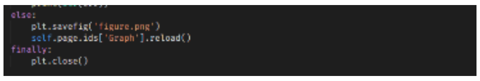


```python
if username is not None:
    meta = {
        'username': username,
        'year_start': int(self.page.ids['DateSelect'].ids['StartYear'].text),
        'month_start': int(self.page.ids['DateSelect'].ids['StartMonth'].text),
        'year_end': int(self.page.ids['DateSelect'].ids['EndYear'].text),
        'month_end': int(self.page.ids['DateSelect'].ids['EndMonth'].text),
        'use_bar': use_bar,
        'use_month': use_month,
        'avg': isavg,
        'max': ismax,
        'min': ismin
    }
    compare = {
        'username': username,
        'search_type': qtype,
        'keyword': place
    }
    log(username, meta, compare)
```

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

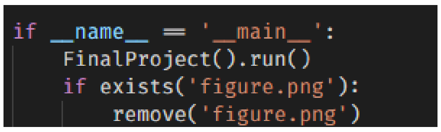


```python
try:
    if qtype.lower() == 'global':
        if isavg and not ismin and not ismax:
            draw(start_date, end_date, True,
                TempType.LANDAVG, None, use_bar, use_month, 'k', 'solid', 111)
        elif ismin and not ismax and not isavg:
            draw(start_date, end_date, True,
                TempType.LANDMIN, None, use_bar, use_month, 'b', 'solid', 111)
        elif ismax and not isavg and not ismin:
            draw(start_date, end_date, True,
                TempType.LANDMAX, None, use_bar, use_month, 'r', 'solid', 111)
        elif ismin and ismax and not isavg:
            draw(start_date, end_date, True,
                TempType.LANDMIN, None, use_bar, use_month, 'b', 'solid', 211)
            draw(start_date, end_date, True,
                TempType.LANDMAX, None, use_bar, use_month, 'r', 'solid', 212)
        elif ismax and isavg and not ismin:
            draw(start_date, end_date, True,
                TempType.LANDAVG, None, use_bar, use_month, 'k', 'solid', 211)
            draw(start_date, end_date, True,
                TempType.LANDMAX, None, use_bar, use_month, 'r', 'solid', 212)
        elif isavg and ismin and not ismax:
            draw(start_date, end_date, True,
                TempType.LANDAVG, None, use_bar, use_month, 'k', 'solid', 211)
            draw(start_date, end_date, True,
                TempType.LANDMIN, None, use_bar, use_month, 'b', 'solid', 212)
        elif ismin and ismax and isavg:
            draw(start_date, end_date, True,
                TempType.LANDAVG, None, use_bar, use_month, 'k', 'solid', 211)
            draw(start_date, end_date, True,
                TempType.LANDMAX, None, use_bar, use_month, 'r', 'solid', 223)
            draw(start_date, end_date, True,
                TempType.LANDMIN, None, use_bar, use_month, 'b', 'solid', 224)
        else:
            print("No selection")
    elif qtype.lower() == 'city':
        draw(start_date, end_date, False,
            TableType.CITY, place, use_bar, use_month, 'k', 'solid', 111)
    elif qtype.lower() == 'country':
        draw(start_date, end_date, False,
            TableType.COUNTRY, place, use_bar, use_month, 'k', 'solid', 111)
    elif qtype.lower() == 'state':
        draw(start_date, end_date, False,
            TableType.STATE, place, use_bar, use_month, 'k', 'solid', 111)

    else:
        print("No position")
except Exception as err:
    print(str(err))
```



```python
else:
    plt.savefig('figure.png')
    self.page.ids['Graph'].reload()
finally:
    plt.close()
```

程式關閉時清除圖片



```python
if __name__ == '__main__':
    FinalProject().run()
    if exists('figure.png'):
        remove('figure.png')
```

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