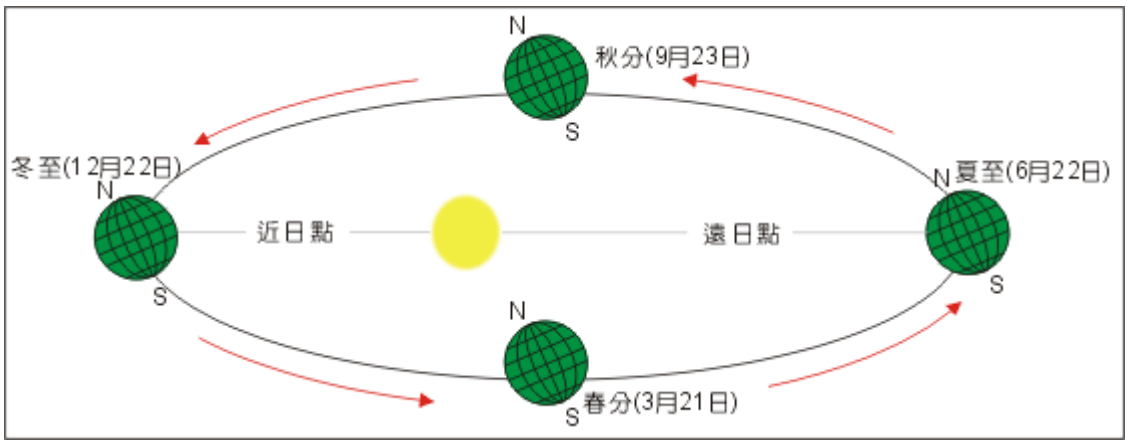
Python 微積分報告 --- 日照時間與日期之間的關係

目錄

前言

太陽作為地球生命的起源，他的**"一舉一動"**與人們的生活作息息息相關。



最受影響當然是依賴作物收成的農夫，而農夫的作物更是現今社會的推動力；沒有食物整個社會便會失去前進的動力，停滯不前。

挫折

我們一開始選定以太陽影子關係作為題目，但因為參考的document沒有足夠的公式及範例以致所繪畫出來的圖形與實際的圖形不相乎；故此我了改選另一題目。即**日照時間與日期之間的關係**

計算

計算日出日落

參考了[Sunrise/Sunset Algorithm]() 及 使用[Sunrise/Sunset Algorithm Example]()的計算這種驗證程式的正確性

程式

程式設計

- 我採用了模組化設計，將各個小程式寫成`function`以方便dubug。
- 並利用`jpyter`作為python程式的editior，藉此方便組員解釋程式碼執行那些動作。
- 並且我們近來學習了`object oriented`的概念，故此希望oo的概念編寫程式，並參考了網路上的
- 以及user輸入的參數有機會出錯，導致程式無法程式執行，而python只會顯示出錯程式行數，而沒有顯示出錯原因，故使用了`try ... except`語句顯示出錯的內容；但時間關係，故輸入地址出錯導致取得的`json` object出錯有使用`try...except`語句

  參考了8. 程式錯誤與例外(Exceptions)情形、程式語言教學誌 FB, YouTube: PYDOING: Python 3.1 快速導覽 - 例外處理 try-except陳述
- 因使用了Google提供的API故使用了JSON及urllib這個library

程式解說

經過網路取得資料

這篇文中的API key為我所申請

透過Google Maps Geocoding API 取得地點經緯度

好處是使用者能輸入**"任何語言"**也可取得該地點的準確經緯度

遇到的困難：

1. 當使用者輸入中文時`url` urlopen 將會出現error message，原因是未將ascii code外的文字作URL encoding

```
# -*- coding: utf-8 -*-
import json #我們將會處理json資料
from urllib.request import urlopen #use urllib to open website
from urllib.parse import quote #處理使用者輸入英文外地址的問題

api_key = "AIzaSyBWr7l1Mv_2aFrjx2OTwVBmS78L5Dtfruo" #Geocoding API key
location = quote(input("地點： ")) #使用quote將文字作URL encoding
language = "zh-TW" #可用語言請參考 https://developers.google.com/maps/faq?authuser=1&hl=zh-
tw#languagesupport

url = "https://maps.googleapis.com/maps/api/geocode/json?
key="+api_key+"&address="+location+"&language="+language
json_data = urlopen(url).read().decode("utf8") #將取得的data順帶編碼

#print(json_data) #pring json

jsonObj = json.loads(json_data) #create json object

address = jsonObj.get("results")[0].get("formatted_address")

lat = jsonObj.get("results")[0].get("geometry").get("location").get("lat")
```

```
lng = jsonObj.get("results")[0].get("geometry").get("location").get("lng")
print("地點全稱：", address)
print("lat:", lat)
print("lng:", lng)
```

```
地點： Beijing
地點全稱： 中國北京市北京
lat: 39.90419989999999
lng: 116.4073963
```

利用**Sunset and sunrise times API**取得日出日落時間

我們將透過Sunset and sunrise times API的API去取得地球上任何一個角落的日出日落時間

> Parameters(其參數)

- lat (float): Latitude in decimal degrees. Required.
- lng (float): Longitude in decimal degrees. Required.
- date (string): Date in YYYY-MM-DD format. Also accepts other date formats and even relative date formats. If not present, date defaults to current date. Optional.
- callback (string): Callback function name for JSONP response. Optional.
- formatted (integer): 0 or 1 (1 is default). Time values in response will be expressed following ISO 8601 and day_length will be expressed in seconds. Optional.

```
# -*- coding: utf-8 -*-
import json #我們將會處理json資料
from urllib.request import urlopen #use urllib to open website

date = "today" #date format: YYYY-MM-DD
#以中央大學的地址作為測試取得日出日落時間的例子
lat = str(24.9694808)
lng = str(121.1925163)
formatted = str(0) #不進行格式處理，以方便後續處理

url = "https://api.sunrise-sunset.org/json?
lat="+lat+"&lng="+lng+"&date="+date+"&formatted="+formatted
json_data = urlopen(url).read().decode("utf8")

#print(json_data) #print json data

jsonObj = json.loads(json_data)

sunrise_time_str = jsonObj.get("results").get("sunrise")
sunset_time_str = jsonObj.get("results").get("sunset")
print("日出時間(UTC):", sunrise_time_str)
print("日落時間(UTC):", sunset_time_str)

import datetime #處理時間用
sunrise_time = datetime.datetime.strptime(sunrise_time_str, "%Y-%m-%dT%H:%M:%S+00:00")
sunset_time = datetime.datetime.strptime(sunset_time_str, "%Y-%m-%dT%H:%M:%S+00:00")
print("sunrise_time : ", sunrise_time)
```

```
print("sunset_time : ", sunset_time)
```

```
日出時間(UTC): 2017-12-23T22:37:17+00:00
日落時間(UTC): 2017-12-24T09:12:39+00:00
sunrise_time :  2017-12-23 22:37:17
sunset_time :  2017-12-24 09:12:39
```

timezone offset

因為datetime貌似不能保存時間的時區 故又使用google timezone api取得該經緯度的時間 為了方便程式設計，讓使用者直接輸入經緯度也能取得該地點的時區

並且參考了Convert datetime to unix timestamp將datetime object 轉換成unix time 以作為API的`timestamp`的參數

本人並不打算處理日光節約時間，所以`timestamp`的時間其實並非重點 並可以用`time`內的`time.time()`取得當前的unix time

而timezone offset則參考了python time offset - Stack Overflow

- 而Google Timezone API請見開發人員指南 | Google Maps Time Zone API | Google Developers

```
# -*- coding: utf-8 -*-
import json #我們將會處理json資料
from urllib.request import urlopen #use urllib to open website

#以中央大學的地址作為例子
lat = str(24.9694808)
lng = str(121.1925163)
language = "zh-TW"
key = "AIzaSyBWr7l1Mv_2aFrjx2OTwVBmS78L5Dtfruo" #timezone API


import datetime
import time

sunrise_time = datetime.datetime(2017, 12, 23, 22, 37, 17)
sunrise_unix_time = time.mktime(sunrise_time.timetuple())
sunrise_unix_time = str(sunrise_unix_time)

url = "https://maps.googleapis.com/maps/api/timezone/json?
location="+lat+","+lng+"&timestamp="+sunrise_unix_time+"&key="+key+"&language="+language
json_data = urlopen(url).read().decode("utf8")
#print("json data: \n", json_data)

jsonObj = json.loads(json_data)
rawOffset = jsonObj.get("rawOffset") #與UTC時間相差秒數
timeZoneID = jsonObj.get("timeZoneId")
timeZoneName = jsonObj.get("timeZoneName")
print("rawOffset", rawOffset, sep=": ")
print("timeZoneID", timeZoneID, sep=": ")
print("timeZoneName", timeZoneName, sep=": ")
```

```
print("\nUTC time: ", sunrise_time)
local_time = sunrise_time + datetime.timedelta(seconds = int(rawOffset)) #使用 "+" 是因為取得的
rawOffset會帶正負號
print("local time: ", local_time)
```

```
rawOffset: 28800
timeZoneID: Asia/Taipei
timeZoneName: 台北標準時間

UTC time:  2017-12-23 22:37:17
local time:  2017-12-24 06:37:17
```

計算日出日落時間

參考了Sunrise/Sunset Algorithm 及 Sunrise/Sunset Algorithm Example 下方程式碼主要用來計算日出及日落的UTC時間

```
#input variable
year = 1990
month = 6
day = 25

#latitude, longitude =  40.9, -74.3  #location for sunrise/sunset
latitude, longitude = 24.9936281, 121.3009798
zenith = 0          #Sun's zenith for sunrise/sunset
offical_zeith = 90 + (50 / 60)
#cos_zeith = cos(offical_zeith * pi / 180) #Just for test
#print("cos_zeith: ", cos_zeith)
civil = 96
nautical = 102
astronomical = 108

offset_value = 8 #taiwan is utc+8

from math import *

def the_day_of_year(year, month, day):
    N1 = floor(275 * month / 9)
    N2 = floor((month + 9) / 12)
    N3 = (1 + floor((year - 4 * floor(year / 4) + 2) /3))
    N = N1 - (N2 * N3) + day - 30
    return N

def calculate_approximate_time(longitude, the_day_of_year_result):
    lngHour = longitude / 15
    t_sun_rise = the_day_of_year_result + ((6 - lngHour) / 24)
    t_sun_set = the_day_of_year_result + ((18 - lngHour) / 24)
    return [t_sun_rise, t_sun_set] # First value in the set is sun rise time, and second value
is sun set time

def sun_anomaly(calculate_approximate_time_result):

    M = (0.9856 * calculate_approximate_time_result) - 3.289
```

```
    return M

def sun_true_longitude(sun_anomaly_result):
    L = sun_anomaly_result + (1.916 * sin(sun_anomaly_result * pi / 180)) + (0.020 * sin(2 *
sun_anomaly_result * pi / 180)) + 282.634
    L = L - int(L / 360) * 360 #make sure return value in the range [0,360)
    return L

def sun_right_ascension(sun_true_longitude_result):
    RA = atan(0.91764 * tan(sun_true_longitude_result * pi / 180)) * 180 / pi
    RA = RA - int(RA / 360) * 360
    return RA

def same_quadrant(sun_true_longitude_result, sun_right_ascension_result):
    sun_true_longitude_result_quadrant = (floor(sun_true_longitude_result / 90)) * 90
    sun_right_ascension_result_quadrant = (floor(sun_right_ascension_result / 90)) * 90
    sun_right_ascension_result = sun_right_ascension_result +
(sun_true_longitude_result_quadrant - sun_right_ascension_result_quadrant)
    return sun_right_ascension_result

def right_ascension_converted_hours(same_quadrant_result):
    same_quadrant_result = same_quadrant_result / 15
    return same_quadrant_result

def sun_declination(sun_true_longitude_result):
    sinDec = 0.39782 * sin(sun_true_longitude_result * pi / 180)
    cosDec = cos((asin(sinDec) * 180 / pi) * pi / 180 )
    return [sinDec, cosDec]

def sun_local_hour_angle(offical_zeith, sun_declination_result, latitude):
    sinDec = sun_declination_result[0]
    cosDec = sun_declination_result[1]
    cosH = (cos(offical_zeith * pi / 180) - (sinDec * sin(latitude * pi / 180))) / (cosDec *
cos(latitude * pi / 180))
    if cosH > 1:
        print("the sun never rises on this location (on the specified date)")
    elif cosH < - 1:
        print("the sun never sets on this location (on the specified date)")
    return cosH

def h_convert_hours(sun_local_hour_angle_result, opition): #opition have two opition
    #"r" means sun rise, and "s" means sun set
    if opition == "r":
        H = 360 - acos(sun_local_hour_angle_result) * 180 / pi
    elif opition == "s":
        H = acos(sun_local_hour_angle_result) * 180 / pi
    H = H / 15
    return H

def local_mean_time(h_convert_hours_results, right_ascension_converted_hours_result,
calculate_approximate_time_result):


    T = h_convert_hours_results + right_ascension_converted_hours_result - (0.06571 *
```

```
calculate_approximate_time_result) - 6.622
    return T

def convert_utc(local_mean_time_result, longitude):
    UT = local_mean_time_result - longitude / 15
    UT = UT - int(UT / 24) * 24
    utc_hour = int(UT / 1)
    utc_minute = UT % 1
    if utc_hour < 0:
        utc_hour += 24
    return [utc_hour, utc_minute]

def utc_offset(convert_utc_result, offset_value):
    convert_utc_result[0] += offset_value
    if convert_utc_result[0] >= 24:
        convert_utc_result[0] -= 24
    return convert_utc_result

#用作分隔用
times = 75
horizon = "*" * times
hr = "-" * times

print(horizon)
#print basic info
print("year: ", year, "month: ", month, "day: ", day)
print("latitude: ", latitude, "longitude: ", longitude)
print(hr)
#calculate the day of year
the_day_of_year_value = the_day_of_year(year, month, day)
print("the_day_of_year_value", the_day_of_year_value, sep = ": ")
print(hr)
#return approximate time
approximate_sun_rise_time_value = calculate_approximate_time(longitude, the_day_of_year_value)
[0]
print("approximate_sun_rise_time_value", approximate_sun_rise_time_value, sep = ": ")
approximate_sun_set_time_value = calculate_approximate_time(longitude, the_day_of_year_value)[1]
print("approximate_sun_set_time_value", approximate_sun_set_time_value, sep = ": ")
print(hr)
#calculate sun anomaly
sun_anomaly_sun_rise_time_value = sun_anomaly(approximate_sun_rise_time_value)
print("sun_anomaly_sun_rise_time_value", sun_anomaly_sun_rise_time_value, sep = ": ")
sun_anomaly_sun_set_time_value = sun_anomaly(approximate_sun_set_time_value)
print("sun_anomaly_sun_set_time_value", sun_anomaly_sun_set_time_value, sep = ": ")
print(hr)
#calculate Sun's true longitude
sun_true_longitude_sun_rise_time_value = sun_true_longitude(sun_anomaly_sun_rise_time_value)
print("sun_true_longitude_sun_rise_time_value", sun_true_longitude_sun_rise_time_value, sep = ":
")
sun_true_longitude_sun_set_time_value = sun_true_longitude(sun_anomaly_sun_set_time_value)
print("sun_true_longitude_sun_set_time_value", sun_true_longitude_sun_set_time_value, sep = ":
")

print(hr)
```

```
#calculate Sun's right ascension
sun_right_ascension_longitude_sun_rise_time_value =
sun_right_ascension(sun_true_longitude_sun_rise_time_value)
print("sun_right_ascension_sun_rise_time_value",
sun_right_ascension_longitude_sun_rise_time_value, sep = ": ")
sun_right_ascension_longitude_sun_set_time_value =
sun_right_ascension(sun_true_longitude_sun_set_time_value)
print("sun_right_ascension_sun_set_time_value",
sun_right_ascension_longitude_sun_set_time_value, sep = ": ")
print(hr)
#same quadrant
same_quadrant_sun_rise_time_value = same_quadrant(sun_true_longitude_sun_rise_time_value,
sun_right_ascension_longitude_sun_rise_time_value)
print("same_quadrant_sun_rise_time_value", same_quadrant_sun_rise_time_value, sep = ": ")
same_quadrant_sun_set_time_value = same_quadrant(sun_true_longitude_sun_set_time_value,
sun_right_ascension_longitude_sun_set_time_value)
print("same_quadrant_sun_set_time_value", same_quadrant_sun_set_time_value, sep = ": ")
print(hr)
#right ascension value needs to be converted into hours
right_ascension_converted_hours_sun_rise_time_value =
right_ascension_converted_hours(same_quadrant_sun_rise_time_value)
print("right_ascension_converted_hours_sun_rise_time_value",
right_ascension_converted_hours_sun_rise_time_value, sep = ": ")
right_ascension_converted_hours_sun_set_time_value =
right_ascension_converted_hours(same_quadrant_sun_set_time_value)
print("right_ascension_converted_hours_sun_set_time_value",
right_ascension_converted_hours_sun_set_time_value, sep = ": ")
print(hr)
#calculate the Sun's declination
sun_declination_sun_rise_time_value = sun_declination(sun_true_longitude_sun_rise_time_value)[1]
print("sun_declination_sun_rise_time_value", sun_declination_sun_rise_time_value, sep = ": ")
sun_declination_sun_set_time_value = sun_declination(sun_true_longitude_sun_set_time_value)[1]
print("sun_declination_sun_set_time_value", sun_declination_sun_set_time_value, sep = ": ")
print(hr)
#calculate the Sun's local hour angle
Sun_local_hour_angle_sun_rise_time_value = sun_local_hour_angle(offical_zeith,
sun_declination(sun_true_longitude_sun_rise_time_value), latitude)
print("Sun_local_hour_angle_sun_rise_time_value", Sun_local_hour_angle_sun_rise_time_value, sep
= ": ")
Sun_local_hour_angle_sun_set_time_value = sun_local_hour_angle(offical_zeith,
sun_declination(same_quadrant_sun_set_time_value), latitude)
print("Sun_local_hour_angle_sun_set_time_value", Sun_local_hour_angle_sun_set_time_value, sep =
": ")
print(hr)
#H and convert into hours
h_convert_hours_sun_rise_time_value = h_convert_hours(Sun_local_hour_angle_sun_rise_time_value,
"r")
print("h_convert_hours_sun_rise_time_value", h_convert_hours_sun_rise_time_value, sep = ": ")
h_convert_hours_sun_set_time_value = h_convert_hours(Sun_local_hour_angle_sun_set_time_value,
"s")
print("h_convert_hours_sun_set_time_value", h_convert_hours_sun_set_time_value, sep = ": ")
print(hr)

#local_mean_time
```

```
local_mean_time_sun_rise_time_value = local_mean_time(h_convert_hours_sun_rise_time_value,
right_ascension_converted_hours_sun_rise_time_value, approximate_sun_rise_time_value)
print("local_mean_time_sun_rise_time_value", local_mean_time_sun_rise_time_value, sep = ": ")
local_mean_time_sun_set_time_value = local_mean_time(h_convert_hours_sun_set_time_value,
right_ascension_converted_hours_sun_set_time_value, approximate_sun_set_time_value)
print("local_mean_time_sun_set_time_value", local_mean_time_sun_set_time_value, sep = ": ")
print(hr)
#convert to UTC time
convert_utc_sun_rise_time_value = convert_utc(local_mean_time_sun_rise_time_value, longitude)
print("convert_utc_sun_rise_time_value", convert_utc_sun_rise_time_value, sep = ": ")
convert_utc_sun_set_time_value = convert_utc(local_mean_time_sun_set_time_value, longitude)
print("convert_utc_sun_set_time_value", convert_utc_sun_set_time_value, sep = ": ")
print(hr)
#covert to local time
utc_offset_sun_rise_time_value = utc_offset(convert_utc_sun_rise_time_value, offset_value)
print("utc_offset_sun_rise_time_value", utc_offset_sun_rise_time_value, sep = ": ")
utc_offset_sun_set_time_value = utc_offset(convert_utc_sun_set_time_value, offset_value)
print("utc_offset_sun_set_time_value", utc_offset_sun_set_time_value, sep = ": ")
```

```
**********************************************************************
year:  1990 month:  6 day:  25
latitude:  24.9936281 longitude:  121.3009798
----------------------------------------------------------------------
the_day_of_year_value: 176
----------------------------------------------------------------------
approximate_sun_rise_time_value: 175.91305283388888
approximate_sun_set_time_value: 176.41305283388888
----------------------------------------------------------------------
sun_anomaly_sun_rise_time_value: 170.09090487308092
sun_anomaly_sun_set_time_value: 170.5837048730809
----------------------------------------------------------------------
sun_true_longitude_sun_rise_time_value: 93.0478399095229
sun_true_longitude_sun_set_time_value: 93.52471892226038
----------------------------------------------------------------------
sun_right_ascension_sun_rise_time_value: -86.67919752027603
sun_right_ascension_sun_set_time_value: -86.15983833890009
----------------------------------------------------------------------
same_quadrant_sun_rise_time_value: 93.32080247972397
same_quadrant_sun_set_time_value: 93.84016166109991
----------------------------------------------------------------------
right_ascension_converted_hours_sun_rise_time_value: 6.221386831981598
right_ascension_converted_hours_sun_set_time_value: 6.256010777406661
----------------------------------------------------------------------
sun_declination_sun_rise_time_value: 0.9177072814387344
sun_declination_sun_set_time_value: 0.9177894222768794
----------------------------------------------------------------------
Sun_local_hour_angle_sun_rise_time_value: -0.2192822439396342
Sun_local_hour_angle_sun_set_time_value: -0.2190802354656302
----------------------------------------------------------------------
h_convert_hours_sun_rise_time_value: 17.155541384218623
h_convert_hours_sun_set_time_value: 6.843667770173764
----------------------------------------------------------------------
local_mean_time_sun_rise_time_value: 5.195681514485381
```

```
local_mean_time_sun_set_time_value: -5.114423154134414
--------------------------------------------------------------------------------
convert_utc_sun_rise_time_value: [22, 0.10894952781871403]
convert_utc_sun_set_time_value: [11, 0.798844859198919]
--------------------------------------------------------------------------------
utc_offset_sun_rise_time_value: [6, 0.10894952781871403]
utc_offset_sun_set_time_value: [19, 0.798844859198919]
```

而這個部分則是將日出日落時間轉化為秒 並將日落的秒數減去日出秒數即可得出日照秒數

```
def daytime_second(sun_rise, sun_set):
    sun_rise_second = sun_rise[0] * 3600 + sun_rise[1] * 60
    sun_set_second = sun_set[0] * 3600 + sun_set[1] * 60
    return sun_set_second - sun_rise_second

#calculate daytime (return value is second)
daytime_value = daytime_second(utc_offset_sun_rise_time_value, utc_offset_sun_set_time_value)
print("daytime_value", daytime_value, sep = ": ")
```

```
daytime_value: 46841.39371988282
```

繪畫daytime圖形

這裏我們利用較為貪功近利的方法測試 下方程式碼中的`the_day_of_year`所return的N為年積日 故此我們藉由年積日的變化即可得出daytime在一年內的變化

```
#input variable
#year = 1990
#month = 6
#day = 25

#latitude, longitude =  40.9, -74.3  #location for sunrise/sunset
latitude, longitude = 24.9936281, 121.3009798
#zenith = 0          #Sun's zenith for sunrise/sunset
offical_zeith = 90 + (50 / 60)
#cos_zeith = cos(offical_zeith * pi / 180) #Just for test
#print("cos_zeith: ", cos_zeith)
#civil = 96
#nautical = 102
#astronomical = 108

offset_value = 8 #taiwan is utc+8

from math import *

def the_day_of_year(year, month, day):
    N1 = floor(275 * month / 9)
    N2 = floor((month + 9) / 12)
    N3 = (1 + floor((year - 4 * floor(year / 4) + 2) /3))
    N = N1 - (N2 * N3) + day - 30

    return N
```

```
def calculate_approximate_time(longitude, the_day_of_year_result):
    lngHour = longitude / 15
    t_sun_rise = the_day_of_year_result + ((6 - lngHour) / 24)
    t_sun_set = the_day_of_year_result + ((18 - lngHour) / 24)
    return [t_sun_rise, t_sun_set] # First value in the set is sun rise time, and second value
is sun set time

def sun_anomaly(calculate_approximate_time_result):
    M = (0.9856 * calculate_approximate_time_result) - 3.289
    return M

def sun_true_longitude(sun_anomaly_result):
    L = sun_anomaly_result + (1.916 * sin(sun_anomaly_result * pi / 180)) + (0.020 * sin(2 *
sun_anomaly_result * pi / 180)) + 282.634
    L = L - int(L / 360) * 360 #make sure return value in the range [0,360)
    return L

def sun_right_ascension(sun_true_longitude_result):
    RA = atan(0.91764 * tan(sun_true_longitude_result * pi / 180)) * 180 / pi
    RA = RA - int(RA / 360) * 360
    return RA

def same_quadrant(sun_true_longitude_result, sun_right_ascension_result):
    sun_true_longitude_result_quadrant = (floor(sun_true_longitude_result / 90)) * 90
    sun_right_ascension_result_quadrant = (floor(sun_right_ascension_result / 90)) * 90
    sun_right_ascension_result = sun_right_ascension_result +
(sun_true_longitude_result_quadrant - sun_right_ascension_result_quadrant)
    return sun_right_ascension_result

def right_ascension_converted_hours(same_quadrant_result):
    same_quadrant_result = same_quadrant_result / 15
    return same_quadrant_result

def sun_declination(sun_true_longitude_result):
    sinDec = 0.39782 * sin(sun_true_longitude_result * pi / 180)
    cosDec = cos((asin(sinDec) * 180 / pi) * pi / 180 )
    return [sinDec, cosDec]

def sun_local_hour_angle(offical_zeith, sun_declination_result, latitude):
    sinDec = sun_declination_result[0]
    cosDec = sun_declination_result[1]
    cosH = (cos(offical_zeith * pi / 180) - (sinDec * sin(latitude * pi / 180))) / (cosDec *
cos(latitude * pi / 180))
    if cosH > 1:
        print("the sun never rises on this location (on the specified date)")
    elif cosH < - 1:
        print("the sun never sets on this location (on the specified date)")
    return cosH

def h_convert_hours(sun_local_hour_angle_result, opition): #opition have two opition
    #"r" means sun rise, and "s" means sun set

    if opition == "r":
```

```
        H = 360 - acos(sun_local_hour_angle_result) * 180 / pi
    elif opition == "s":
        H = acos(sun_local_hour_angle_result) * 180 / pi
    H = H / 15
    return H

def local_mean_time(h_convert_hours_results, right_ascension_converted_hours_result,
calculate_approximate_time_result):
    T = h_convert_hours_results + right_ascension_converted_hours_result - (0.06571 *
calculate_approximate_time_result) - 6.622
    return T

def convert_utc(local_mean_time_result, longitude):
    UT = local_mean_time_result - longitude / 15
    UT = UT - int(UT / 24) * 24
    utc_hour = int(UT / 1)
    utc_minute = UT % 1
    if utc_hour < 0:
        utc_hour += 24
    return [utc_hour, utc_minute]

def utc_offset(convert_utc_result, offset_value):
    convert_utc_result[0] += offset_value
    if convert_utc_result[0] >= 24:
        convert_utc_result[0] -= 24
    return convert_utc_result

def daytime_second(sun_rise, sun_set):
    sun_rise_second = sun_rise[0] * 3600 + sun_rise[1] * 60
    sun_set_second = sun_set[0] * 3600 + sun_set[1] * 60
    return sun_set_second - sun_rise_second

ys = []

for x in range(366):
    print(x)
    the_day_of_year_value = x
    #return approximate time
    approximate_sun_rise_time_value = calculate_approximate_time(longitude,
the_day_of_year_value)[0]
    approximate_sun_set_time_value = calculate_approximate_time(longitude,
the_day_of_year_value)[1]
    #calculate sun anomaly
    sun_anomaly_sun_rise_time_value = sun_anomaly(approximate_sun_rise_time_value)
    sun_anomaly_sun_set_time_value = sun_anomaly(approximate_sun_set_time_value)
    #calculate Sun's true longitude
    sun_true_longitude_sun_rise_time_value = sun_true_longitude(sun_anomaly_sun_rise_time_value)
    sun_true_longitude_sun_set_time_value = sun_true_longitude(sun_anomaly_sun_set_time_value)
    #calculate Sun's right ascension
    sun_right_ascension_longitude_sun_rise_time_value =
sun_right_ascension(sun_true_longitude_sun_rise_time_value)


    sun_right_ascension_longitude_sun_set_time_value =
```

```
sun_right_ascension(sun_true_longitude_sun_set_time_value)
    #same quadrant
    same_quadrant_sun_rise_time_value = same_quadrant(sun_true_longitude_sun_rise_time_value,
sun_right_ascension_longitude_sun_rise_time_value)
    same_quadrant_sun_set_time_value = same_quadrant(sun_true_longitude_sun_set_time_value,
sun_right_ascension_longitude_sun_set_time_value)
    #right ascension value needs to be converted into hours
    right_ascension_converted_hours_sun_rise_time_value =
right_ascension_converted_hours(same_quadrant_sun_rise_time_value)
    right_ascension_converted_hours_sun_set_time_value =
right_ascension_converted_hours(same_quadrant_sun_set_time_value)
    #calculate the Sun's declination
    sun_declination_sun_rise_time_value =
sun_declination(sun_true_longitude_sun_rise_time_value)[1]
    sun_declination_sun_set_time_value = sun_declination(sun_true_longitude_sun_set_time_value)
[1]
    #calculate the Sun's local hour angle
    Sun_local_hour_angle_sun_rise_time_value = sun_local_hour_angle(offical_zeith,
sun_declination(sun_true_longitude_sun_rise_time_value), latitude)
    Sun_local_hour_angle_sun_set_time_value = sun_local_hour_angle(offical_zeith,
sun_declination(same_quadrant_sun_set_time_value), latitude)
    #H and convert into hours
    h_convert_hours_sun_rise_time_value =
h_convert_hours(Sun_local_hour_angle_sun_rise_time_value, "r")
    h_convert_hours_sun_set_time_value =
h_convert_hours(Sun_local_hour_angle_sun_set_time_value, "s")
    #local_mean_time
    local_mean_time_sun_rise_time_value = local_mean_time(h_convert_hours_sun_rise_time_value,
right_ascension_converted_hours_sun_rise_time_value, approximate_sun_rise_time_value)
    local_mean_time_sun_set_time_value = local_mean_time(h_convert_hours_sun_set_time_value,
right_ascension_converted_hours_sun_set_time_value, approximate_sun_set_time_value)
    #convert to UTC time
    convert_utc_sun_rise_time_value = convert_utc(local_mean_time_sun_rise_time_value,
longitude)
    convert_utc_sun_set_time_value = convert_utc(local_mean_time_sun_set_time_value, longitude)
    #covert to local time
    utc_offset_sun_rise_time_value = utc_offset(convert_utc_sun_rise_time_value, offset_value)
    utc_offset_sun_set_time_value = utc_offset(convert_utc_sun_set_time_value, offset_value)
    #daytime (return second)
    daytime_value = daytime_second(utc_offset_sun_rise_time_value,
utc_offset_sun_set_time_value)
    ys += [daytime_value]
    print("daytime_value: ", daytime_value)
```

```
0
daytime_value:  39576.734861677134
1
daytime_value:  39577.097693266456
2
daytime_value:  39577.496105704166
3
daytime_value:  39577.92963778885
4
```

```
daytime_value:  39578.397792847
5
daytime_value:  39578.90004080289
6
daytime_value:  39579.435820341896
7
daytime_value:  39580.00454114986
8
daytime_value:  39580.6055862114
9
daytime_value:  39581.2383141499
10
daytime_value:  39581.902061592584
11
daytime_value:  39582.59614554451
12
daytime_value:  39583.319865756115
13
daytime_value:  39584.07250706988
14
daytime_value:  39584.853341732276
15
daytime_value:  39585.66163165886
16
daytime_value:  39586.49663064093
17
daytime_value:  39587.35758648366
18
daytime_value:  39588.24374306676
19
daytime_value:  39589.15434231995
20
daytime_value:  39590.088626106575
21
daytime_value:  39591.04583801025
22
daytime_value:  39592.02522502009
23
daytime_value:  39593.026039111486
24
daytime_value:  39594.04753872046
25
daytime_value:  39595.0889901102
26
daytime_value:  39596.1496686298
27
daytime_value:  39597.228859865536
28
daytime_value:  39598.325860686135
29
daytime_value:  39599.43998018387
30

daytime_value:  39600.57054051396
```

```
31
daytime_value:  39601.71687763536
32
daytime_value:  39602.87834195624
33
daytime_value:  39604.05429888793
34
daytime_value:  39605.24412931137
35
daytime_value:  39606.44722996012
36
daytime_value:  39607.663013724516
37
daytime_value:  39608.89090988133
38
daytime_value:  39610.13036425336
39
daytime_value:  39611.38083930356
40
daytime_value:  39612.64181416818
41
daytime_value:  39613.912784633256
42
daytime_value:  39615.193263058914
43
daytime_value:  39616.48277825542
44
daytime_value:  39617.78087531529
45
daytime_value:  39619.08711540524
46
daytime_value:  39620.40107552167
47
daytime_value:  39621.72234821332
48
daytime_value:  39623.05054127435
49
daytime_value:  39624.38527741114
50
daytime_value:  39625.72619388568
51
daytime_value:  39627.07294213842
52
daytime_value:  39628.42518739306
53
daytime_value:  39629.782608245914
54
daytime_value:  39631.144896241734
55
daytime_value:  39632.5117554384
56
daytime_value:  39633.88290196209

57
```

```
daytime_value:  39635.25806355475
58
daytime_value:  39636.636979115385
59
daytime_value:  39638.0193982366
60
daytime_value:  39639.40508073765
61
daytime_value:  39640.793796194936
62
daytime_value:  39642.18532347132
63
daytime_value:  39643.57945024475
64
daytime_value:  39644.97597253716
65
daytime_value:  43186.37469424421
66
daytime_value:  43187.775426666514
67
daytime_value:  43189.17798804278
68
daytime_value:  43190.582203085156
69
daytime_value:  43191.987902517256
70
daytime_value:  43193.39492261507
71
daytime_value:  43194.80310475075
72
daytime_value:  43196.21229493966
73
daytime_value:  43197.622343390656
74
daytime_value:  43199.0331040595
75
daytime_value:  43200.44443420562
76
daytime_value:  43201.856193951986
77
daytime_value:  43203.268245848056
78
daytime_value:  43204.68045443586
79
daytime_value:  46746.092685818774
80
daytime_value:  50347.504807233185
81
daytime_value:  46748.91668662263
82
daytime_value:  46750.32819221442
83

daytime_value:  46751.739192098496
```

```
84
daytime_value:  46753.149553808384
85
daytime_value:  46754.55914390405
86
daytime_value:  46755.967827556575
87
daytime_value:  46757.375468134494
88
daytime_value:  46758.781926791475
89
daytime_value:  46760.18706205563
90
daytime_value:  46761.59072942001
91
daytime_value:  46762.99278093428
92
daytime_value:  46764.39306479782
93
daytime_value:  46765.79142495384
94
daytime_value:  46767.187700684866
95
daytime_value:  46768.58172620957
96
daytime_value:  46769.97333028111
97
daytime_value:  46771.36233578692
98
daytime_value:  46772.74855935067
99
daytime_value:  46774.13181093614
100
daytime_value:  46775.51189345382
101
daytime_value:  46776.88860237023
102
daytime_value:  46778.26172532086
103
daytime_value:  46779.631041726985
104
daytime_value:  46780.99632241708
105
daytime_value:  46782.357329253675
106
daytime_value:  46783.71381476632
107
daytime_value:  46785.06552179166
108
daytime_value:  46786.412183121436
109
daytime_value:  46787.75352115993

110
```

```
daytime_value:  46789.08924759158
111
daytime_value:  46790.41906306031
112
daytime_value:  46791.74265686207
113
daytime_value:  46793.059706652
114
daytime_value:  46794.369878167825
115
daytime_value:  46795.672824971465
116
daytime_value:  46796.96818821042
117
daytime_value:  46798.25559640127
118
daytime_value:  46799.53466523715
119
daytime_value:  46800.80499742144
120
daytime_value:  46802.06618253015
121
daytime_value:  46803.31779690536
122
daytime_value:  46804.55940358201
123
daytime_value:  46805.79055225103
124
daytime_value:  46807.01077926118
125
daytime_value:  46808.21960766251
126
daytime_value:  46809.41654729404
127
daytime_value:  46810.60109491863
128
daytime_value:  46811.772734407896
129
daytime_value:  46812.93093697971
130
daytime_value:  46814.0751614913
131
daytime_value:  46815.20485479052
132
daytime_value:  46816.319452127966
133
daytime_value:  46817.418377632195
134
daytime_value:  46818.50104485077
135
daytime_value:  46819.56685735863
136

daytime_value:  46820.61520943629
```

```
137
daytime_value:  46821.6454868188
138
daytime_value:  46822.65706751733
139
daytime_value:  46823.64932271378
140
daytime_value:  46824.621617729106
141
daytime_value:  46825.57331306536
142
daytime_value:  46826.50376552091
143
daytime_value:  46827.41232937793
144
daytime_value:  46828.2983576603
145
daytime_value:  46829.16120346011
146
daytime_value:  46830.00022132951
147
daytime_value:  46830.81476873468
148
daytime_value:  46831.60420756746
149
daytime_value:  46832.367905710125
150
daytime_value:  46833.10523864713
151
daytime_value:  46833.81559111802
152
daytime_value:  46834.49835880421
153
daytime_value:  46835.15295004209
154
daytime_value:  46835.77878755413
155
daytime_value:  46836.37531018906
156
daytime_value:  46836.94197466171
157
daytime_value:  46837.47825728284
158
daytime_value:  46837.98365566836
159
daytime_value:  46838.45769041771
160
daytime_value:  46838.89990675049
161
daytime_value:  46839.30987609057
162
daytime_value:  46839.687197586754

163
```

```
daytime_value:  46840.03149955958
164
daytime_value:  46840.34244086352
165
daytime_value:  46840.61971215482
166
daytime_value:  46840.86303705525
167
daytime_value:  46841.07217320299
168
daytime_value:  46841.24691318254
169
daytime_value:  46841.38708532603
170
daytime_value:  46841.49255437993
171
daytime_value:  46841.56322203159
172
daytime_value:  46841.599027291406
173
daytime_value:  46841.59994672747
174
daytime_value:  46841.565994550954
175
daytime_value:  46841.49722255144
176
daytime_value:  46841.39371988282
177
daytime_value:  46841.25561270155
178
daytime_value:  46841.0830636604
179
daytime_value:  46840.876271261615
180
daytime_value:  46840.63546907506
181
daytime_value:  46840.3609248274
182
daytime_value:  46840.052939369736
183
daytime_value:  46839.71184553175
184
daytime_value:  46839.33800687107
185
daytime_value:  46838.93181632747
186
daytime_value:  46838.493694791934
187
daytime_value:  46838.02408960053
188
daytime_value:  46837.52347296431
189

daytime_value:  46836.99234034549
```

```
190
daytime_value:  46836.43120879086
191
daytime_value:  46835.840615233334
192
daytime_value:  46835.221114771746
193
daytime_value:  46834.57327893948
194
daytime_value:  46833.89769397152
195
daytime_value:  46833.19495907938
196
daytime_value:  46832.46568474297
197
daytime_value:  46831.71049102741
198
daytime_value:  46830.93000593243
199
daytime_value:  46830.124863781806
200
daytime_value:  46829.295703658645
201
daytime_value:  46828.44316789238
202
daytime_value:  46827.567900602444
203
daytime_value:  46826.67054630277
204
daytime_value:  46825.75174857073
205
daytime_value:  46824.81214878331
206
daytime_value:  46823.852384922946
207
daytime_value:  46822.873090454435
208
daytime_value:  46821.8748932742
209
daytime_value:  46820.858414732305
210
daytime_value:  46819.82426872738
211
daytime_value:  46818.77306087387
212
daytime_value:  46817.70538774092
213
daytime_value:  46816.62183616175
214
daytime_value:  46815.522982611765
215
daytime_value:  46814.40939265392

216
```

```
daytime_value:  46813.281620448935
217
daytime_value:  46812.14020832848
218
daytime_value:  46810.98568642868
219
daytime_value:  46809.81857238135
220
daytime_value:  46808.639371060606
221
daytime_value:  46807.448574381735
222
daytime_value:  46806.246661149984
223
daytime_value:  46805.03409695604
224
daytime_value:  46803.81133411588
225
daytime_value:  46802.57881165187
226
daytime_value:  46801.336955312596
227
daytime_value:  46800.08617762881
228
daytime_value:  46798.82687800262
229
daytime_value:  46797.55944282796
230
daytime_value:  46796.28424563921
231
daytime_value:  46795.00164728638
232
daytime_value:  46793.71199613405
233
daytime_value:  46792.415628282324
234
daytime_value:  46791.11286780772
235
daytime_value:  46789.80402702199
236
daytime_value:  46788.48940674724
237
daytime_value:  46787.16929660556
238
daytime_value:  46785.843975321906
239
daytime_value:  46784.51371103825
240
daytime_value:  46783.17876163838
241
daytime_value:  46781.83937508149
242

daytime_value:  46780.495789743756
```

```
243
daytime_value:  46779.14823476694
244
daytime_value:  46777.79693041283
245
daytime_value:  46776.44208842298
246
daytime_value:  46775.08391238269
247
daytime_value:  46773.722598088934
248
daytime_value:  46772.358333921235
249
daytime_value:  46770.99130121536
250
daytime_value:  46769.62167463919
251
daytime_value:  46768.249622570336
252
daytime_value:  46766.87530747549
253
daytime_value:  46765.498886290865
254
daytime_value:  46764.12051080384
255
daytime_value:  46762.74032803562
256
daytime_value:  46761.358480624614
257
daytime_value:  46759.97510721069
258
daytime_value:  43218.590342820295
259
daytime_value:  43217.20431925229
260
daytime_value:  43215.817165464774
261
daytime_value:  43214.429007962724
262
daytime_value:  43213.039971186954
263
daytime_value:  43211.650177904055
264
daytime_value:  43210.25974959787
265
daytime_value:  43208.868806862476
266
daytime_value:  43207.47746979685
267
daytime_value:  43206.085858401624
268
daytime_value:  43204.694092977734

269
```

```
daytime_value:  43203.302294527675
270
daytime_value:  43201.91058515912
271
daytime_value:  43200.5190884914
272
daytime_value:  43199.127930064846
273
daytime_value:  43197.7372377534
274
daytime_value:  43196.34714218035
275
daytime_value:  43194.95777713774
276
daytime_value:  43193.56928000917
277
daytime_value:  43192.18179219652
278
daytime_value:  43190.79545955019
279
daytime_value:  43189.410432803415
280
daytime_value:  43188.02686801018
281
daytime_value:  43186.64492698703
282
daytime_value:  43185.26477775851
283
daytime_value:  43183.88659500609
284
daytime_value:  43182.51056052042
285
daytime_value:  43181.136863656546
286
daytime_value:  43179.76570179182
287
daytime_value:  43178.397280785975
288
daytime_value:  43177.03181544287
289
daytime_value:  43175.669529973224
290
daytime_value:  43174.31065845775
291
daytime_value:  43172.95544530958
292
daytime_value:  43171.60414573527
293
daytime_value:  43170.25702619322
294
daytime_value:  43168.91436484807
295

daytime_value:  43167.57645202006
```

```
296
daytime_value:  43166.24359062745
297
daytime_value:  43164.916096620625
298
daytime_value:  43163.59429940581
299
daytime_value:  43162.27854225646
300
daytime_value:  43160.96918271003
301
daytime_value:  43159.666592947935
302
daytime_value:  39618.371160155744
303
daytime_value:  39617.08328686124
304
daytime_value:  39615.80339124703
305
daytime_value:  39614.53190743466
306
daytime_value:  39613.26928573691
307
daytime_value:  39612.015992874585
308
daytime_value:  39610.77251215413
309
daytime_value:  39609.539343602155
310
daytime_value:  39608.31700405264
311
daytime_value:  39607.10602718279
312
daytime_value:  39605.906963493006
313
daytime_value:  39604.72038022659
314
daytime_value:  39603.546861224575
315
daytime_value:  39602.38700671114
316
daytime_value:  39601.24143300506
317
daytime_value:  39600.110772152475
318
daytime_value:  39598.9956714767
319
daytime_value:  39597.89679304068
320
daytime_value:  39596.81481301791
321
daytime_value:  39595.75042096808

322
```

```
daytime_value:  39594.70431901378
323
daytime_value:  39593.67722091531
324
daytime_value:  39592.6698510409
325
daytime_value:  39591.68294323007
326
daytime_value:  39590.71723954914
327
daytime_value:  39589.77348893776
328
daytime_value:  39588.85244574682
329
daytime_value:  39587.95486816865
330
daytime_value:  39587.08151656149
331
daytime_value:  39586.23315167127
332
daytime_value:  39585.410532754664
333
daytime_value:  39584.61441560883
334
daytime_value:  39583.845550514074
335
daytime_value:  39583.10468009736
336
daytime_value:  39582.39253712518
337
daytime_value:  39581.70984223621
338
daytime_value:  39581.05730162489
339
daytime_value:  39580.43560468838
340
daytime_value:  39579.84542165047
341
daytime_value:  39579.28740117697
342
daytime_value:  39578.76216799789
343
daytime_value:  39578.27032055275
344
daytime_value:  39577.81242867551
345
daytime_value:  39577.38903133644
346
daytime_value:  39577.00063445828
347
daytime_value:  39576.647708824006
348

daytime_value:  39576.33068809332
```

```
349
daytime_value:  39576.04996694476
350
daytime_value:  39575.805899359315
351
daytime_value:  39575.59879706075
352
daytime_value:  39575.42892812674
353
daytime_value:  39575.29651578341
354
daytime_value:  39575.2017373946
355
daytime_value:  39575.14472365506
356
daytime_value:  39575.12555799562
357
daytime_value:  39575.144276205494
358
daytime_value:  39575.20086627564
359
daytime_value:  39575.295268464404
360
daytime_value:  39575.42737558491
361
daytime_value:  39575.59703351122
362
daytime_value:  39575.80404189828
363
daytime_value:  39576.048155109005
364
daytime_value:  39576.329083339224
365
daytime_value:  39576.6464939303
```

這裏使用pylab進行繪圖

```
import pylab
xs = []
length = len(ys)
for x in range(len(ys)):
    xs += [x]
    print("xs: ", xs[x], "ys: ", ys[x])

pylab.plot(xs, ys)
pylab.show()
```

```
xs:  0 ys:  39576.734861677134
xs:  1 ys:  39577.097693266456
xs:  2 ys:  39577.496105704166
xs:  3 ys:  39577.92963778885

xs:  4 ys:  39578.397792847
```

```
xs:  5 ys:  39578.90004080289
xs:  6 ys:  39579.435820341896
xs:  7 ys:  39580.00454114986
xs:  8 ys:  39580.6055862114
xs:  9 ys:  39581.2383141499
xs:  10 ys:  39581.902061592584
xs:  11 ys:  39582.59614554451
xs:  12 ys:  39583.319865756115
xs:  13 ys:  39584.07250706988
xs:  14 ys:  39584.853341732276
xs:  15 ys:  39585.66163165886
xs:  16 ys:  39586.49663064093
xs:  17 ys:  39587.35758648366
xs:  18 ys:  39588.24374306676
xs:  19 ys:  39589.15434231995
xs:  20 ys:  39590.088626106575
xs:  21 ys:  39591.04583801025
xs:  22 ys:  39592.02522502009
xs:  23 ys:  39593.026039111486
xs:  24 ys:  39594.04753872046
xs:  25 ys:  39595.0889901102
xs:  26 ys:  39596.1496686298
xs:  27 ys:  39597.228859865536
xs:  28 ys:  39598.325860686135
xs:  29 ys:  39599.43998018387
xs:  30 ys:  39600.57054051396
xs:  31 ys:  39601.71687763536
xs:  32 ys:  39602.87834195624
xs:  33 ys:  39604.05429888793
xs:  34 ys:  39605.24412931137
xs:  35 ys:  39606.44722996012
xs:  36 ys:  39607.663013724516
xs:  37 ys:  39608.89090988133
xs:  38 ys:  39610.13036425336
xs:  39 ys:  39611.38083930356
xs:  40 ys:  39612.64181416818
xs:  41 ys:  39613.912784633256
xs:  42 ys:  39615.193263058914
xs:  43 ys:  39616.48277825542
xs:  44 ys:  39617.78087531529
xs:  45 ys:  39619.08711540524
xs:  46 ys:  39620.40107552167
xs:  47 ys:  39621.72234821332
xs:  48 ys:  39623.05054127435
xs:  49 ys:  39624.38527741114
xs:  50 ys:  39625.72619388568
xs:  51 ys:  39627.07294213842
xs:  52 ys:  39628.42518739306
xs:  53 ys:  39629.782608245914
xs:  54 ys:  39631.144896241734
xs:  55 ys:  39632.5117554384
xs:  56 ys:  39633.88290196209

xs:  57 ys:  39635.25806355475
```

```
xs:  58 ys:  39636.636979115385
xs:  59 ys:  39638.0193982366
xs:  60 ys:  39639.40508073765
xs:  61 ys:  39640.793796194936
xs:  62 ys:  39642.18532347132
xs:  63 ys:  39643.57945024475
xs:  64 ys:  39644.97597253716
xs:  65 ys:  43186.37469424421
xs:  66 ys:  43187.775426666514
xs:  67 ys:  43189.17798804278
xs:  68 ys:  43190.582203085156
xs:  69 ys:  43191.987902517256
xs:  70 ys:  43193.39492261507
xs:  71 ys:  43194.80310475075
xs:  72 ys:  43196.21229493966
xs:  73 ys:  43197.622343390656
xs:  74 ys:  43199.0331040595
xs:  75 ys:  43200.44443420562
xs:  76 ys:  43201.856193951986
xs:  77 ys:  43203.268245848056
xs:  78 ys:  43204.68045443586
xs:  79 ys:  46746.092685818774
xs:  80 ys:  50347.504807233185
xs:  81 ys:  46748.91668662263
xs:  82 ys:  46750.32819221442
xs:  83 ys:  46751.739192098496
xs:  84 ys:  46753.149553808384
xs:  85 ys:  46754.55914390405
xs:  86 ys:  46755.967827556575
xs:  87 ys:  46757.375468134494
xs:  88 ys:  46758.781926791475
xs:  89 ys:  46760.18706205563
xs:  90 ys:  46761.59072942001
xs:  91 ys:  46762.99278093428
xs:  92 ys:  46764.39306479782
xs:  93 ys:  46765.79142495384
xs:  94 ys:  46767.187700684866
xs:  95 ys:  46768.58172620957
xs:  96 ys:  46769.97333028111
xs:  97 ys:  46771.36233578692
xs:  98 ys:  46772.74855935067
xs:  99 ys:  46774.13181093614
xs:  100 ys:  46775.51189345382
xs:  101 ys:  46776.88860237023
xs:  102 ys:  46778.26172532086
xs:  103 ys:  46779.631041726985
xs:  104 ys:  46780.99632241708
xs:  105 ys:  46782.357329253675
xs:  106 ys:  46783.71381476632
xs:  107 ys:  46785.06552179166
xs:  108 ys:  46786.412183121436
xs:  109 ys:  46787.75352115993

xs:  110 ys:  46789.08924759158
```

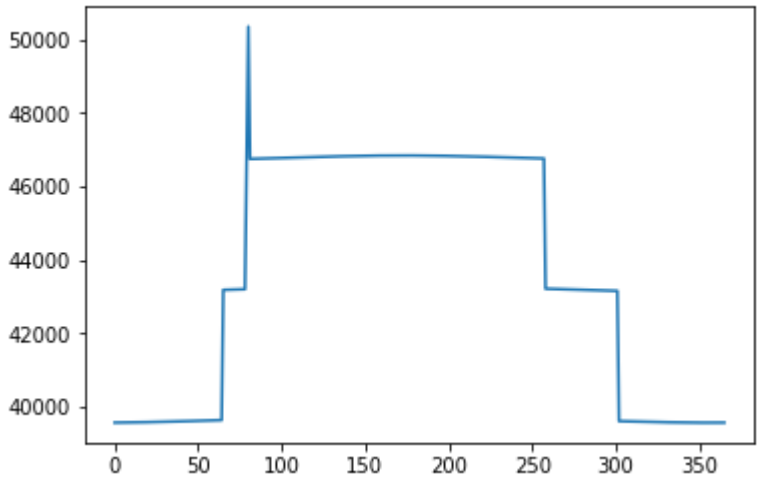
```
xs:  111 ys:  46790.41906306031
xs:  112 ys:  46791.74265686207
xs:  113 ys:  46793.059706652
xs:  114 ys:  46794.369878167825
xs:  115 ys:  46795.672824971465
xs:  116 ys:  46796.96818821042
xs:  117 ys:  46798.25559640127
xs:  118 ys:  46799.53466523715
xs:  119 ys:  46800.80499742144
xs:  120 ys:  46802.06618253015
xs:  121 ys:  46803.31779690536
xs:  122 ys:  46804.55940358201
xs:  123 ys:  46805.79055225103
xs:  124 ys:  46807.01077926118
xs:  125 ys:  46808.21960766251
xs:  126 ys:  46809.41654729404
xs:  127 ys:  46810.60109491863
xs:  128 ys:  46811.772734407896
xs:  129 ys:  46812.93093697971
xs:  130 ys:  46814.0751614913
xs:  131 ys:  46815.20485479052
xs:  132 ys:  46816.319452127966
xs:  133 ys:  46817.418377632195
xs:  134 ys:  46818.50104485077
xs:  135 ys:  46819.56685735863
xs:  136 ys:  46820.61520943629
xs:  137 ys:  46821.6454868188
xs:  138 ys:  46822.65706751733
xs:  139 ys:  46823.64932271378
xs:  140 ys:  46824.621617729106
xs:  141 ys:  46825.57331306536
xs:  142 ys:  46826.50376552091
xs:  143 ys:  46827.41232937793
xs:  144 ys:  46828.2983576603
xs:  145 ys:  46829.16120346011
xs:  146 ys:  46830.00022132951
xs:  147 ys:  46830.81476873468
xs:  148 ys:  46831.60420756746
xs:  149 ys:  46832.367905710125
xs:  150 ys:  46833.10523864713
xs:  151 ys:  46833.81559111802
xs:  152 ys:  46834.49835880421
xs:  153 ys:  46835.15295004209
xs:  154 ys:  46835.77878755413
xs:  155 ys:  46836.37531018906
xs:  156 ys:  46836.94197466171
xs:  157 ys:  46837.47825728284
xs:  158 ys:  46837.98365566836
xs:  159 ys:  46838.45769041771
xs:  160 ys:  46838.89990675049
xs:  161 ys:  46839.30987609057
xs:  162 ys:  46839.687197586754

xs:  163 ys:  46840.03149955958
```

```
xs:  164 ys:  46840.34244086352
xs:  165 ys:  46840.61971215482
xs:  166 ys:  46840.86303705525
xs:  167 ys:  46841.07217320299
xs:  168 ys:  46841.24691318254
xs:  169 ys:  46841.38708532603
xs:  170 ys:  46841.49255437993
xs:  171 ys:  46841.56322203159
xs:  172 ys:  46841.599027291406
xs:  173 ys:  46841.59994672747
xs:  174 ys:  46841.565994550954
xs:  175 ys:  46841.49722255144
xs:  176 ys:  46841.39371988282
xs:  177 ys:  46841.25561270155
xs:  178 ys:  46841.0830636604
xs:  179 ys:  46840.876271261615
xs:  180 ys:  46840.63546907506
xs:  181 ys:  46840.3609248274
xs:  182 ys:  46840.052939369736
xs:  183 ys:  46839.71184553175
xs:  184 ys:  46839.33800687107
xs:  185 ys:  46838.93181632747
xs:  186 ys:  46838.493694791934
xs:  187 ys:  46838.02408960053
xs:  188 ys:  46837.52347296431
xs:  189 ys:  46836.99234034549
xs:  190 ys:  46836.43120879086
xs:  191 ys:  46835.840615233334
xs:  192 ys:  46835.221114771746
xs:  193 ys:  46834.57327893948
xs:  194 ys:  46833.89769397152
xs:  195 ys:  46833.19495907938
xs:  196 ys:  46832.46568474297
xs:  197 ys:  46831.71049102741
xs:  198 ys:  46830.93000593243
xs:  199 ys:  46830.124863781806
xs:  200 ys:  46829.295703658645
xs:  201 ys:  46828.44316789238
xs:  202 ys:  46827.567900602444
xs:  203 ys:  46826.67054630277
xs:  204 ys:  46825.75174857073
xs:  205 ys:  46824.81214878331
xs:  206 ys:  46823.852384922946
xs:  207 ys:  46822.873090454435
xs:  208 ys:  46821.8748932742
xs:  209 ys:  46820.858414732305
xs:  210 ys:  46819.82426872738
xs:  211 ys:  46818.77306087387
xs:  212 ys:  46817.70538774092
xs:  213 ys:  46816.62183616175
xs:  214 ys:  46815.522982611765
xs:  215 ys:  46814.40939265392

xs:  216 ys:  46813.281620448935
```

```
xs:  217 ys:  46812.14020832848
xs:  218 ys:  46810.98568642868
xs:  219 ys:  46809.81857238135
xs:  220 ys:  46808.639371060606
xs:  221 ys:  46807.448574381735
xs:  222 ys:  46806.246661149984
xs:  223 ys:  46805.03409695604
xs:  224 ys:  46803.81133411588
xs:  225 ys:  46802.57881165187
xs:  226 ys:  46801.336955312596
xs:  227 ys:  46800.08617762881
xs:  228 ys:  46798.82687800262
xs:  229 ys:  46797.55944282796
xs:  230 ys:  46796.28424563921
xs:  231 ys:  46795.00164728638
xs:  232 ys:  46793.71199613405
xs:  233 ys:  46792.415628282324
xs:  234 ys:  46791.11286780772
xs:  235 ys:  46789.80402702199
xs:  236 ys:  46788.48940674724
xs:  237 ys:  46787.16929660556
xs:  238 ys:  46785.843975321906
xs:  239 ys:  46784.51371103825
xs:  240 ys:  46783.17876163838
xs:  241 ys:  46781.83937508149
xs:  242 ys:  46780.495789743756
xs:  243 ys:  46779.14823476694
xs:  244 ys:  46777.79693041283
xs:  245 ys:  46776.44208842298
xs:  246 ys:  46775.08391238269
xs:  247 ys:  46773.722598088934
xs:  248 ys:  46772.358333921235
xs:  249 ys:  46770.99130121536
xs:  250 ys:  46769.62167463919
xs:  251 ys:  46768.249622570336
xs:  252 ys:  46766.87530747549
xs:  253 ys:  46765.498886290865
xs:  254 ys:  46764.12051080384
xs:  255 ys:  46762.74032803562
xs:  256 ys:  46761.358480624614
xs:  257 ys:  46759.97510721069
xs:  258 ys:  43218.590342820295
xs:  259 ys:  43217.20431925229
xs:  260 ys:  43215.817165464774
xs:  261 ys:  43214.429007962724
xs:  262 ys:  43213.039971186954
xs:  263 ys:  43211.650177904055
xs:  264 ys:  43210.25974959787
xs:  265 ys:  43208.868806862476
xs:  266 ys:  43207.47746979685
xs:  267 ys:  43206.085858401624
xs:  268 ys:  43204.694092977734

xs:  269 ys:  43203.302294527675
```

```
xs:  270 ys:  43201.91058515912
xs:  271 ys:  43200.5190884914
xs:  272 ys:  43199.127930064846
xs:  273 ys:  43197.7372377534
xs:  274 ys:  43196.34714218035
xs:  275 ys:  43194.95777713774
xs:  276 ys:  43193.56928000917
xs:  277 ys:  43192.18179219652
xs:  278 ys:  43190.79545955019
xs:  279 ys:  43189.410432803415
xs:  280 ys:  43188.02686801018
xs:  281 ys:  43186.64492698703
xs:  282 ys:  43185.26477775851
xs:  283 ys:  43183.88659500609
xs:  284 ys:  43182.51056052042
xs:  285 ys:  43181.136863656546
xs:  286 ys:  43179.76570179182
xs:  287 ys:  43178.397280785975
xs:  288 ys:  43177.03181544287
xs:  289 ys:  43175.669529973224
xs:  290 ys:  43174.31065845775
xs:  291 ys:  43172.95544530958
xs:  292 ys:  43171.60414573527
xs:  293 ys:  43170.25702619322
xs:  294 ys:  43168.91436484807
xs:  295 ys:  43167.57645202006
xs:  296 ys:  43166.24359062745
xs:  297 ys:  43164.916096620625
xs:  298 ys:  43163.59429940581
xs:  299 ys:  43162.27854225646
xs:  300 ys:  43160.96918271003
xs:  301 ys:  43159.666592947935
xs:  302 ys:  39618.371160155744
xs:  303 ys:  39617.08328686124
xs:  304 ys:  39615.80339124703
xs:  305 ys:  39614.53190743466
xs:  306 ys:  39613.26928573691
xs:  307 ys:  39612.015992874585
xs:  308 ys:  39610.77251215413
xs:  309 ys:  39609.539343602155
xs:  310 ys:  39608.31700405264
xs:  311 ys:  39607.10602718279
xs:  312 ys:  39605.906963493006
xs:  313 ys:  39604.72038022659
xs:  314 ys:  39603.546861224575
xs:  315 ys:  39602.38700671114
xs:  316 ys:  39601.24143300506
xs:  317 ys:  39600.110772152475
xs:  318 ys:  39598.9956714767
xs:  319 ys:  39597.89679304068
xs:  320 ys:  39596.81481301791
xs:  321 ys:  39595.75042096808

xs:  322 ys:  39594.70431901378
```

```
xs:  323 ys:  39593.67722091531
xs:  324 ys:  39592.6698510409
xs:  325 ys:  39591.68294323007
xs:  326 ys:  39590.71723954914
xs:  327 ys:  39589.77348893776
xs:  328 ys:  39588.85244574682
xs:  329 ys:  39587.95486816865
xs:  330 ys:  39587.08151656149
xs:  331 ys:  39586.23315167127
xs:  332 ys:  39585.410532754664
xs:  333 ys:  39584.61441560883
xs:  334 ys:  39583.845550514074
xs:  335 ys:  39583.10468009736
xs:  336 ys:  39582.39253712518
xs:  337 ys:  39581.70984223621
xs:  338 ys:  39581.05730162489
xs:  339 ys:  39580.43560468838
xs:  340 ys:  39579.84542165047
xs:  341 ys:  39579.28740117697
xs:  342 ys:  39578.76216799789
xs:  343 ys:  39578.27032055275
xs:  344 ys:  39577.81242867551
xs:  345 ys:  39577.38903133644
xs:  346 ys:  39577.00063445828
xs:  347 ys:  39576.647708824006
xs:  348 ys:  39576.33068809332
xs:  349 ys:  39576.04996694476
xs:  350 ys:  39575.805899359315
xs:  351 ys:  39575.59879706075
xs:  352 ys:  39575.42892812674
xs:  353 ys:  39575.29651578341
xs:  354 ys:  39575.2017373946
xs:  355 ys:  39575.14472365506
xs:  356 ys:  39575.12555799562
xs:  357 ys:  39575.144276205494
xs:  358 ys:  39575.20086627564
xs:  359 ys:  39575.295268464404
xs:  360 ys:  39575.42737558491
xs:  361 ys:  39575.59703351122
xs:  362 ys:  39575.80404189828
xs:  363 ys:  39576.048155109005
xs:  364 ys:  39576.329083339224
xs:  365 ys:  39576.6464939303
```



難點

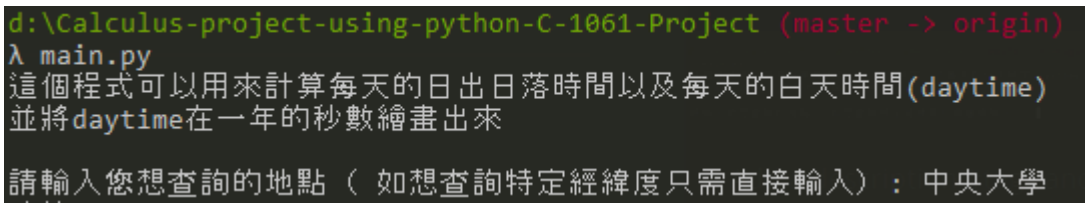
- 因為這是我第一次利用python開發程式，故對syntax並不熟悉，經常犯下語法錯誤。
- 因為涉及時間的offset以及取得時間字串，故需不斷尋找資料去解決問題

  參考了 Hank to hanker - Learning Note: [Python] 時間格式轉換(strtime & strftime)、8.1. datetime — Basic date and time types — Python 2.7.14 documentation
- 為了減輕使用者理解參數的壓力，故在某些function的參數預先填入了數值

  參考了How do you get Python to detect for no input - Stack Overflow
- 因為Google API使用上有限額而且API key附在`main.py`故有機會被他人濫用導致程式無法正常運作

執行畫面

1. 主程式為`main.py`



2. 程式具有下列的功能

> 1. 經緯度相關 (列出輸入地點的經緯度)
> 2. 時區相關 (列出與時區相關的資訊)
> 3. 日出日落相關 (使用Google 提供的 API)
> 4. 日出日落相關 (使用參考的公式計算)
> 5. 計算daytime (輸出白天長度)
> 6. 輸出特定時間內的 daytime 秒數之間關係的圖形

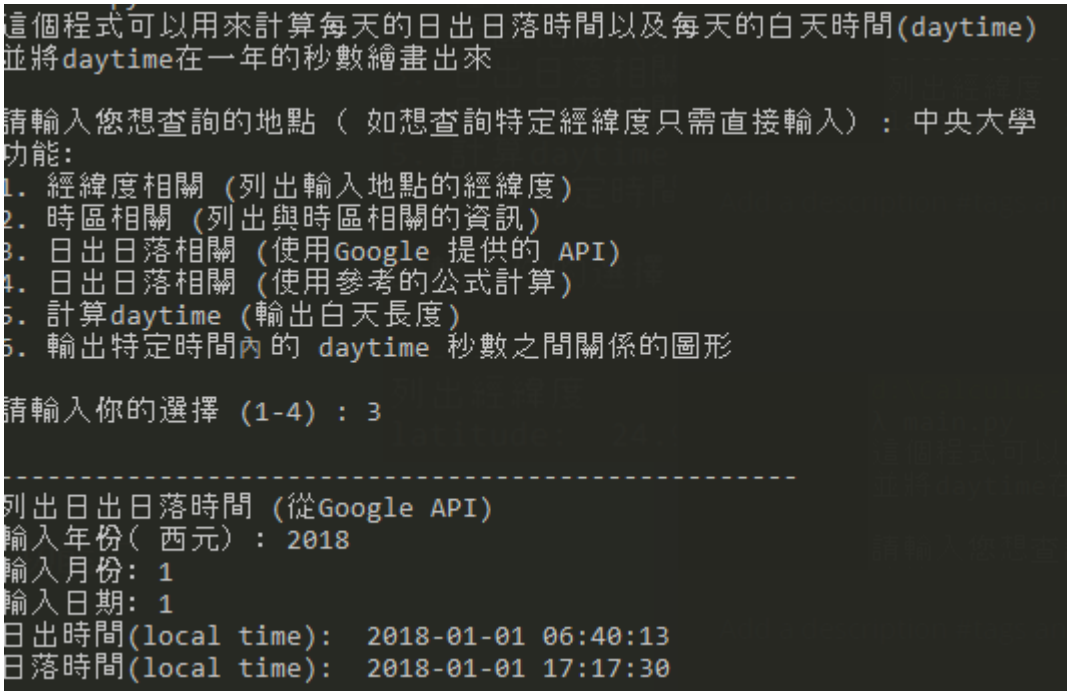
```
d:\Calculus-project-using-python-C-1061-Project (master -> origin)
λ main.py
這個程式可以用來計算每天的日出日落時間以及每天的白天時間(daytime)
並將daytime在一年的秒數繪畫出來

請輸入您想查詢的地點 ( 如想查詢特定經緯度只需直接輸入) : 中央大學
功能:
1. 經緯度相關 (列出輸入地點的經緯度)
2. 時區相關 (列出與時區相關的資訊)
3. 日出日落相關 (使用Google 提供的 API)
4. 日出日落相關 (使用參考的公式計算)
5. 計算daytime (輸出白天長度)
6. 輸出特定時間內的 daytime 秒數之間關係的圖形

請輸入你的選擇 (1-4) :
```

功能一 (列出輸入地點的經緯度)

- 可透過輸入地點而得出經緯度

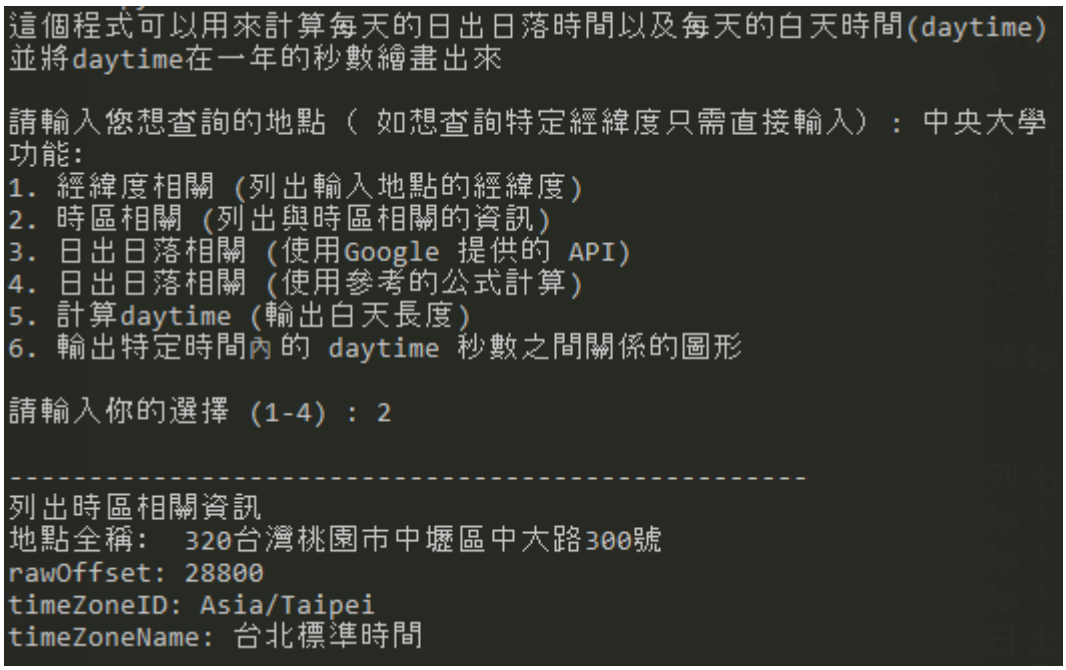
```
這個程式可以用來計算每天的日出日落時間以及每天的白天時間(daytime)
並將daytime在一年的秒數繪畫出來

請輸入您想查詢的地點 ( 如想查詢特定經緯度只需直接輸入) : 中央大學
功能:
1. 經緯度相關 (列出輸入地點的經緯度)
2. 時區相關 (列出與時區相關的資訊)
3. 日出日落相關 (使用Google 提供的 API)
4. 日出日落相關 (使用參考的公式計算)
5. 計算daytime (輸出白天長度)
6. 輸出特定時間內的 daytime 秒數之間關係的圖形

請輸入你的選擇 (1-4) : 1

-------------------------------------------------
列出經緯度
latitude:  24.9694808 longitude:  121.1925163
```

功能二 (藉由Google 提供的API去求得該地點的日出日落時間)



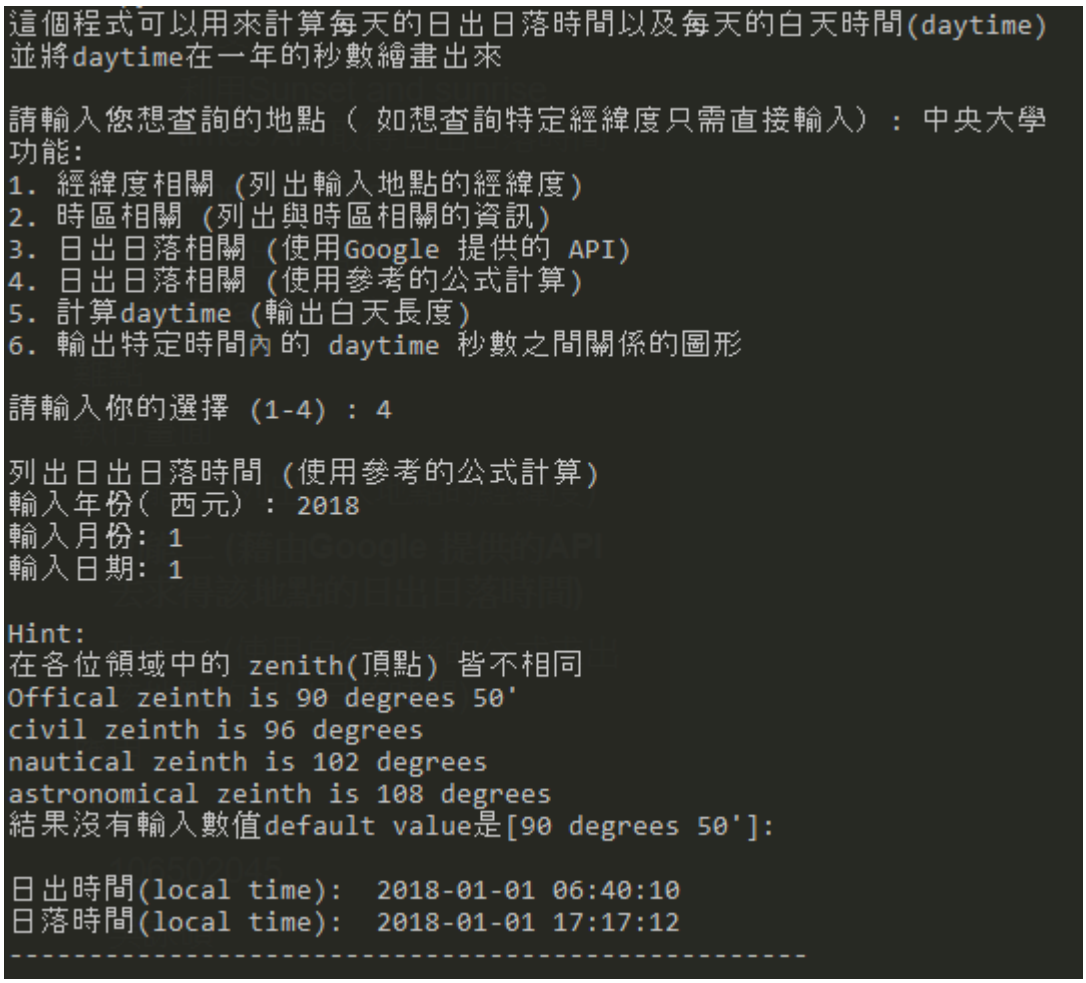
功能三 (求出該地點的時區為何)

- 藉由Google Timezone API取得特定經緯度的時區



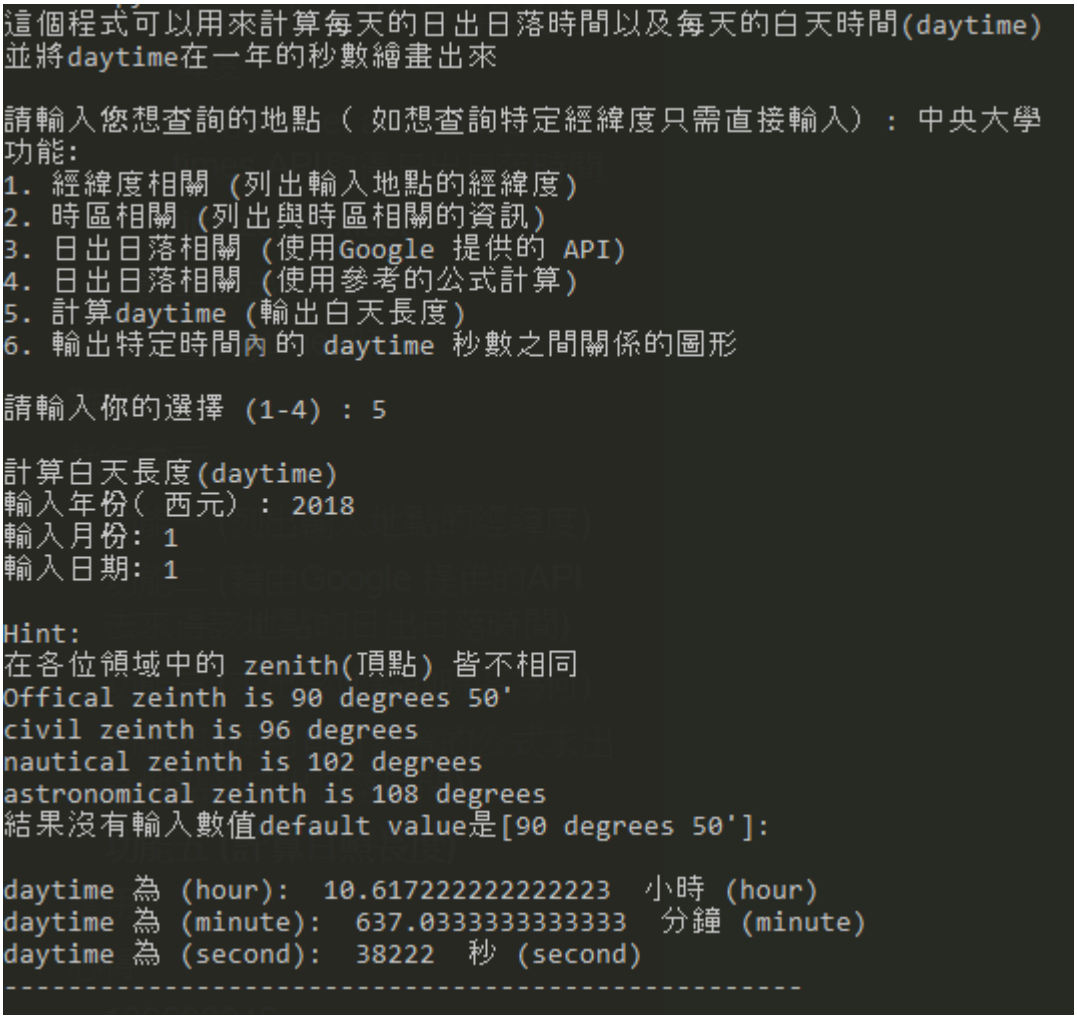
> 因為直接藉由公式所求出的日出日落時間為`UTC`時間，故所藉由取得offset方向轉換為當地實際時間

功能四 (使用自行參考的公式求出該地點的日出日落時間)

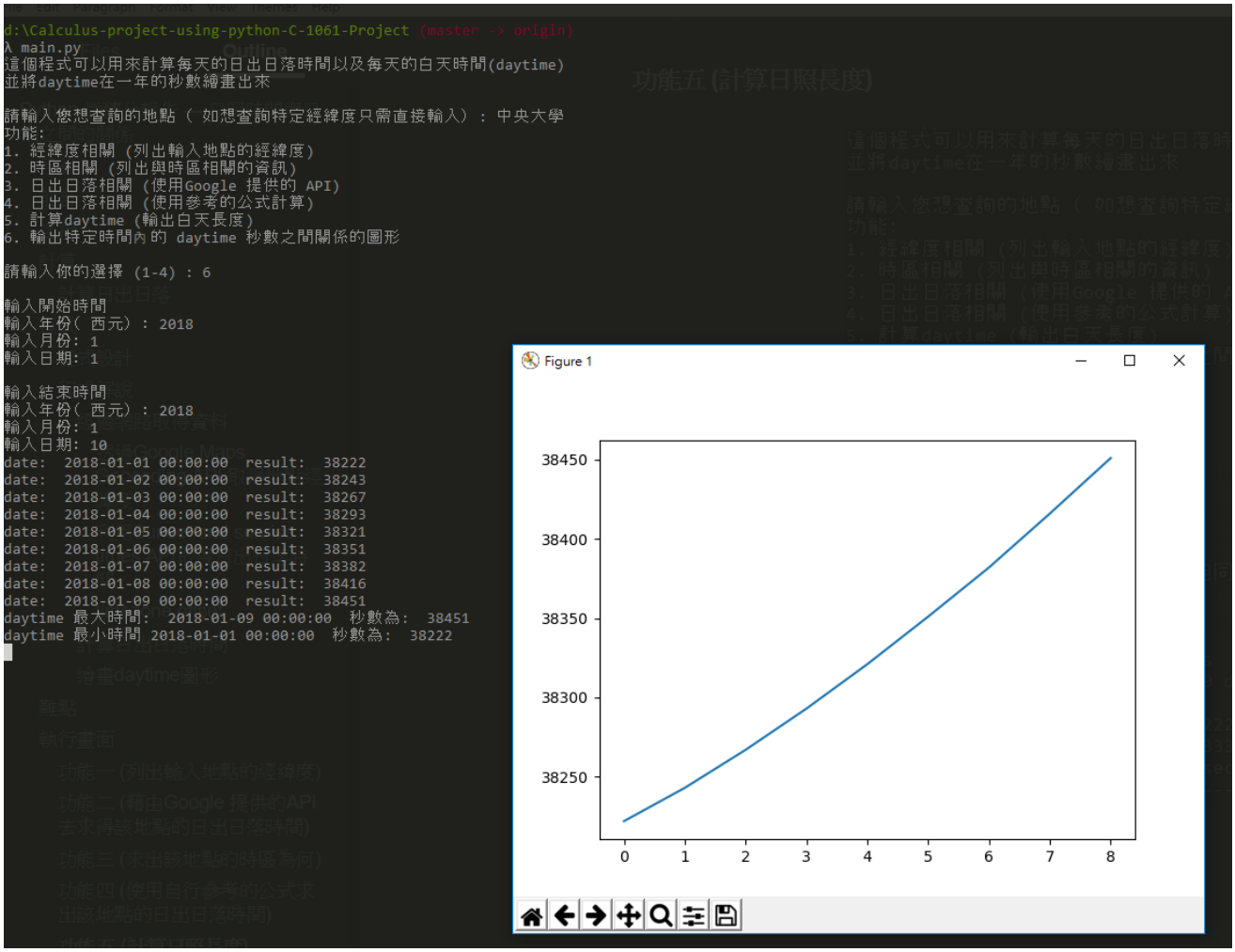


> 並可看到實際日出日落時間與計算求出的日出日落時間相差不大

功能五 (計算日照長度)



功能六 **(**將特定日期內的日照長度繪畫出來**)**

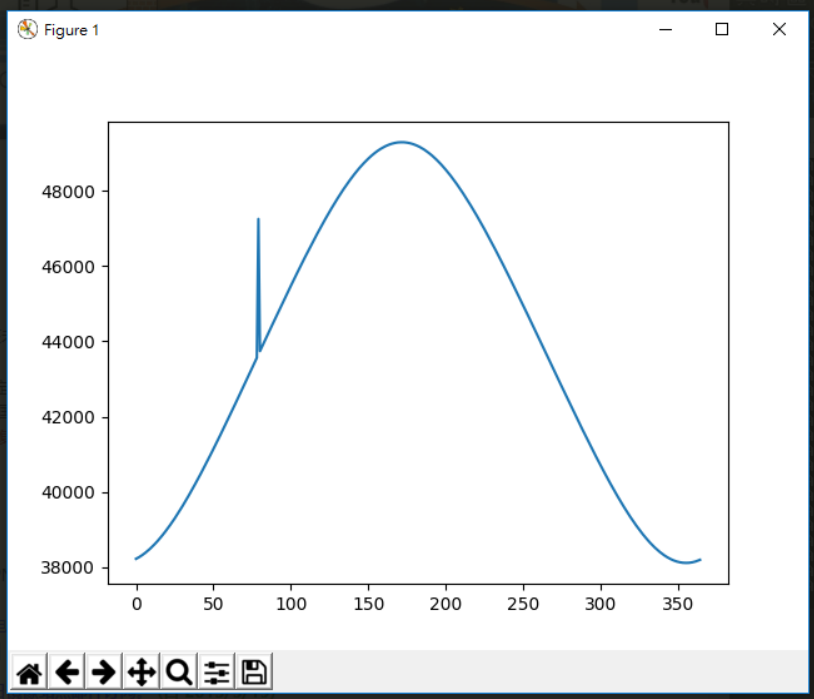


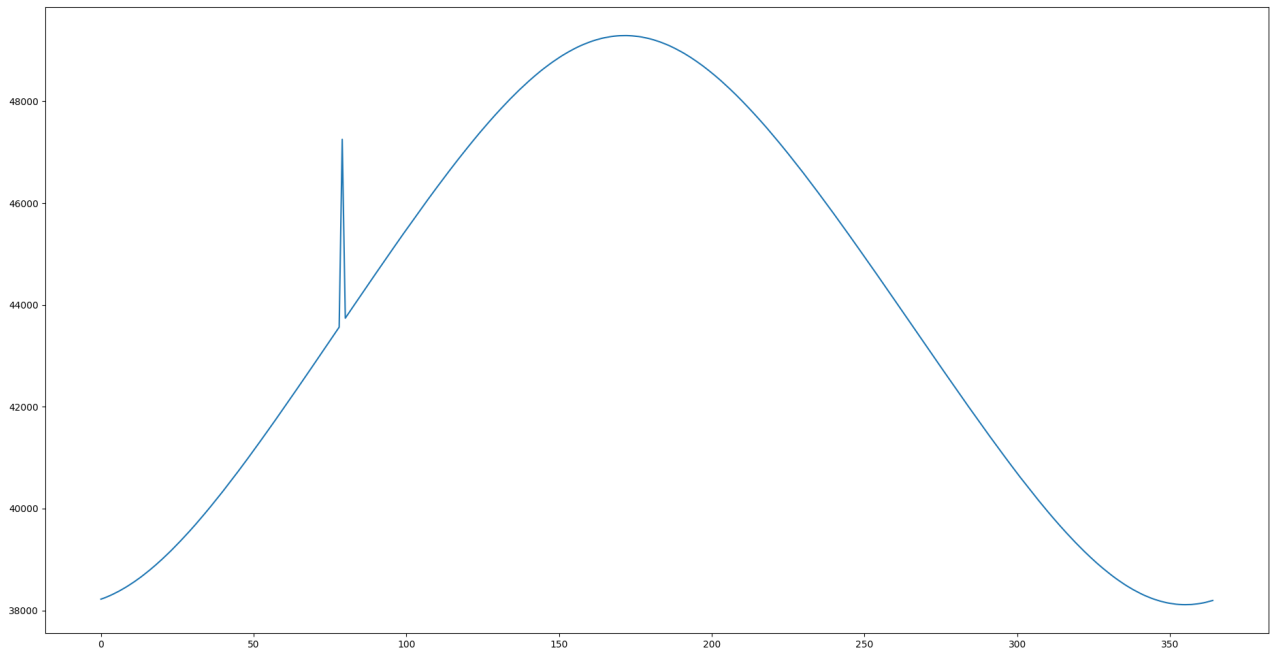
實際測試

- 繪畫一年中日照長度圖形

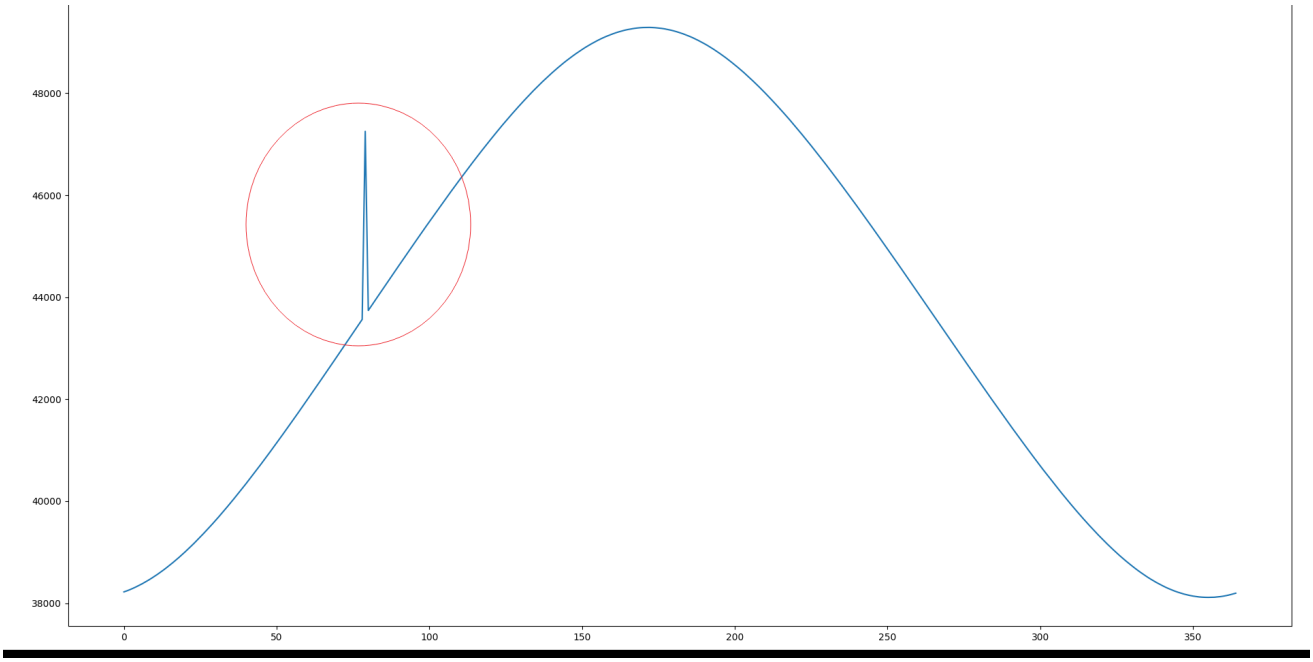
```
這個程式可以用來計算每天的日出日落時間以及每天的白天時間(daytime)
並將daytime在一年的秒數繪畫出來

請輸入您想查詢的地點（ 如想查詢特定經緯度只需直接輸入）：中央大學
功能:
1. 經緯度相關 (列出輸入地點的經緯度)
2. 時區相關 (列出與時區相關的資訊)
3. 日出日落相關 (使用Google 提供的 API)
4. 日出日落相關 (使用參考的公式計算)
5. 計算daytime (輸出白天長度)
6. 輸出特定時間內的 daytime 秒數之間關係的圖形

請輸入你的選擇 (1-4) : 6

輸入開始時間
輸入年份( 西元) : 2018
輸入月份: 1
輸入日期: 1

輸入結束時間
輸入年份( 西元) : 2019
輸入月份: 1
輸入日期: 1
date:  2018-01-01 00:00:00  result:  38222
date:  2018-01-02 00:00:00  result:  38243
date:  2018-01-03 00:00:00  result:  38267
date:  2018-01-04 00:00:00  result:  38293
date:  2018-01-05 00:00:00  result:  38321
```

```
date:  2018-11-04 00:00:00  result:  40159
date:  2018-11-05 00:00:00  result:  40086
date:  2018-11-06 00:00:00  result:  40013
date:  2018-11-07 00:00:00  result:  39941
date:  2018-11-08 00:00:00  result:  39870
date:  2018-11-09 00:00:00  result:  39800
date:  2018-11-10 00:00:00  result:  39731
date:  2018-11-11 00:00:00  result:  39663
date:  2018-11-12 00:00:00  result:  39596
date:  2018-11-13 00:00:00  result:  39530
date:  2018-11-14 00:00:00  result:  39465
date:  2018-11-15 00:00:00  result:  39402
date:  2018-11-16 00:00:00  result:  39338
date:  2018-11-17 00:00:00  result:  39277
date:  2018-11-18 00:00:00  result:  39217
date:  2018-11-19 00:00:00  result:  39157
date:  2018-11-20 00:00:00  result:  39099
date:  2018-11-21 00:00:00  result:  39043
date:  2018-11-22 00:00:00  result:  38987
date:  2018-11-23 00:00:00  result:  38933
date:  2018-11-24 00:00:00  result:  38881
date:  2018-11-25 00:00:00  result:  38830
date:  2018-11-26 00:00:00  result:  38781
date:  2018-11-27 00:00:00  result:  38733
date:  2018-11-28 00:00:00  result:  38686
date:  2018-11-29 00:00:00  result:  38641
date:  2018-11-30 00:00:00  result:  38598
date:  2018-12-01 00:00:00  result:  38557
date:  2018-12-02 00:00:00  result:  38516
date:  2018-12-03 00:00:00  result:  38478
date:  2018-12-04 00:00:00  result:  38442
date:  2018-12-05 00:00:00  result:  38407
date:  2018-12-06 00:00:00  result:  38375
date:  2018-12-07 00:00:00  result:  38344
date:  2018-12-08 00:00:00  result:  38314
date:  2018-12-09 00:00:00  result:  38287
date:  2018-12-10 00:00:00  result:  38262
date:  2018-12-11 00:00:00  result:  38238
date:  2018-12-12 00:00:00  result:  38217
date:  2018-12-13 00:00:00  result:  38198
date:  2018-12-14 00:00:00  result:  38180
date:  2018-12-15 00:00:00  result:  38165
date:  2018-12-16 00:00:00  result:  38151
date:  2018-12-17 00:00:00  result:  38140
date:  2018-12-18 00:00:00  result:  38131
date:  2018-12-19 00:00:00  result:  38123
date:  2018-12-20 00:00:00  result:  38118
date:  2018-12-21 00:00:00  result:  38115
date:  2018-12-22 00:00:00  result:  38114
date:  2018-12-23 00:00:00  result:  38115
date:  2018-12-24 00:00:00  result:  38117
date:  2018-12-25 00:00:00  result:  38123
date:  2018-12-26 00:00:00  result:  38130
date:  2018-12-27 00:00:00  result:  38139
date:  2018-12-28 00:00:00  result:  38150
date:  2018-12-29 00:00:00  result:  38164
date:  2018-12-30 00:00:00  result:  38180
date:  2018-12-31 00:00:00  result:  38196
daytime 最大時間:  2018-06-22 00:00:00  秒數為:  49290
daytime 最小時間 2018-12-22 00:00:00  秒數為:  38114
```





討論



在60日附近，可見一特殊的部分(如上圖所圓出的部分)，本人認為這是由於潤年的關係故使程式的計算有些微誤差，而該部分實際應為平滑曲線

應用

- 因為不是每台電腦有完整的開發環境，故利用`Microsoft Azure Notebooks`作為程式測試環境，以方便組員測試程式碼
- 利用`Markdown`以編寫報告

心得

吳詠碩

一年內日間長度之變化 一 緣起: 小時候，爺爺常常與年幼的我們訴說他小時候務農的事。而這些故事大致上是在說在那個年代，人們總是從太陽升起時開始工作，而在天黑時回家休息。但是在一年之中，每一天太陽升起至落下中間日間時間都不太一樣。所以我就想知道在一年內的日間長度變化，已更加了解祖父輩們的生活作息。 二 實際應用: 雖然計算出一年內日間長度的變化看似是一件徒勞無功的研究。但實際上，這卻是一件很重要的數學公式。以下是一年內日間長度之變化的應用。 1.農業 每一種作物都有者各自的最佳日照時間。倘若農夫沒有管控好，那他們便會蒙受相當大的損失。為了使農夫能使作物長的好，了解一年內日間長度之變化是一件極為重要的事。 2.太陽能 最近因為全球暖化以及環保意識抬頭，因此人們逐漸以綠色能源取代石油、天然氣以及煤炭等會大量製造溫室氣體的能源。而在取代這些能源的綠色能源中，太陽能，是最受世人所重視的一項能源。而對於太陽能這個能源而言，日照時間是一個影響能源產量的一大重要因素。因此，掌握日照時間事對於太陽能源產量事一件極為重要的事。

羅展釗

經過是次報告我對python的語法熟練度大大提高。

以及日照作為地球生物的能量來源，沒有日照，地球上的生命將會慢慢消逝。

現在全球人類都面臨著能源危機，不可再生的石化能源使用更會導致空氣、土壤及水質的污染。利用核分裂原理發電的核電站有造成輻射污染的問題。而故為可行的綠色能源的候選人則是太陽能。

而太陽能的發電量則取決於當天的日照時間長度，故藉由是次報告則可見在六月左右太陽能的發電量應為最多。

組員

- 吳詠碩
- 羅展釗
- 思維
- 陳郁傑
- 張耕培