人口增長推測

前言

在現今人口增長迅速、糧食不足、土地缺乏、老年化的情況下，人口數量已成為一個大問題。 故此本人選擇以人口作為今次微積分python報告的題目，希望能藉此對人口、python及數學有進一步的認識。

人口增長推測

本人參考了人口成長模型這篇文章的數學模型去編寫`Malthus`及`Logistic`這兩個class。

利用Malthus推測人口增長

Malthus數學模組是由英國經濟學家Malthus在1798年匿名發表的《人口原理》中用來描述人口成長的數學模組。

$$P(t) = P_0 \cdot e^{\lambda(t-t_0)}$$

> t 為當前所求人口的時間 $P(t)$ 為求出的人口 t_0 為任意時間點 P_0 為任意時間點的人口 λ 為一常數（需由計算求得）

而本人利用上式推導以求出λ： $\lambda = \frac{ln(\frac{P(t)}{P_0})}{t-t_0}$

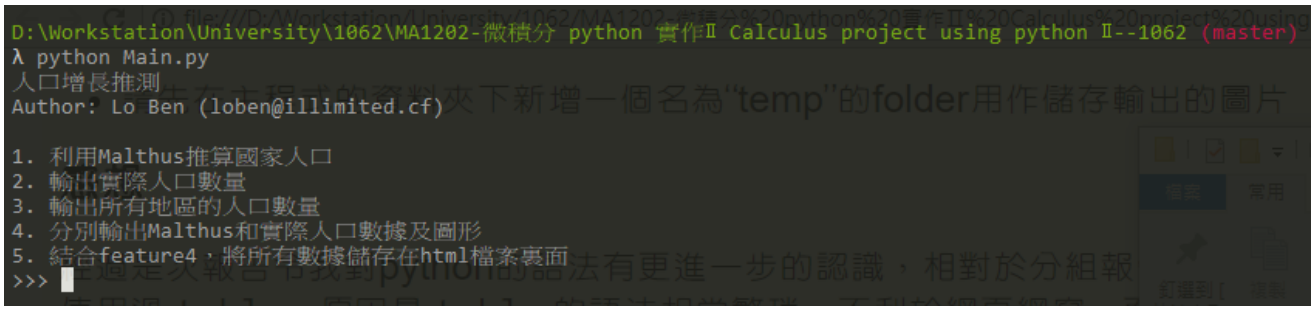
功能

- 藉由 **Malthus** 及 api.population.io 所取得的資料去推算某一（數個）地區的人口數量。
- 輸出 **Malthus** 預測及實際人口數量圖形
- 輸出結果為html file

教學

- 請先在主程式的資料夾下新增一個名為“temp”的folder用作儲存輸出的圖片

主畫面



主畫面有5個功能

1. 第一個功能是透過Malthus去求出特定時間段內特定國家的人口數量（彈出windows）
2. 藉由網上提供的`api`輸出特定時間段內的實際人口數量（彈出windows）
3. 儲存特定時間段內所有國家的實際人數及 **Malthus** 預測的人口數量（儲存為png圖片及json）
4. 顯示特定時間段內所有國家的實際人數及 **Malthus** 預測的人口數量（彈出windows)
5. 將所有結果輸出成`html`檔案

feature 1

選擇feature1

```
115: Less developed regions, excluding China                    116: Liberia
117: Libya                      118: Lithuania
119: Luxembourg                 120: Madagascar
121: Malawi                     122: Malaysia
123: Maldives                   124: Mali
125: Malta                      126: Martinique
127: Mauritania                 128: Mauritius
129: Mayotte                    130: Melanesia
131: Mexico                     132: Micronesia
133: Middle Africa                  134: Moldova
135: Mongolia                   136: Montenegro
137: More developed regions                 138: Morocco
139: Mozambique                 140: Myanmar
141: Namibia                    142: Nepal
143: The Netherlands                144: New Caledonia
145: New Zealand                    146: Nicaragua
147: Niger                      148: Nigeria
149: Northern Africa                150: NORTHERN AMERICA
151: Northern Europe                152: Norway
153: OCEANIA                    154: Oman
155: Other non-specified areas              156: Pakistan
157: Panama                     158: Papua New Guinea
159: Paraguay                   160: Peru
161: Philippines                    162: Poland
163: Polynesia                  164: Portugal
165: Puerto Rico                    166: Qatar
167: Reunion                    168: RB-de-Venezuela
169: Rep of Korea                   170: Rep of Yemen
171: Romania                    172: Russian Federation
173: Rwanda                     174: St-Lucia
175: St-Vincent and the Grenadines                  176: Samoa
177: Sao Tome and Principe                  178: Saudi Arabia
179: Senegal                    180: Serbia
181: Seychelles                 182: Sierra Leone
183: Singapore                  184: Slovak Republic
185: Slovenia                   186: Solomon Islands
187: Somalia                    188: South Africa
189: South America                  190: South Sudan
191: South-Central Asia             192: South-Eastern Asia
193: Southern Africa                194: Southern Asia
195: Southern Europe                196: Spain
197: Sri Lanka                  198: West Bank and Gaza
199: Sub-Saharan Africa             200: Sudan
201: Suriname                   202: Swaziland
203: Sweden                     204: Switzerland
205: Syrian Arab Rep                206: Tajikistan
207: Tanzania                   208: Thailand
209: Timor-Leste                    210: Togo
211: Tonga                      212: Trinidad and Tobago
213: Tunisia                    214: Turkey
215: Turkmenistan                   216: Uganda
217: Ukraine                    218: United Arab Emirates
219: United Kingdom                 220: United States
221: US Virgin Islands              222: Uruguay
223: Uzbekistan                 224: Vanuatu
225: Vietnam                    226: Western Africa
227: Western Asia                   228: Western Europe
229: Western Sahara                 230: World
231: Zambia                     232: Zimbabwe
請輸入上表中的國家: （例如： Swaziland, Thailand, Canda 或者 對應國家的數字編號如： 100, 102）
 >>>
```

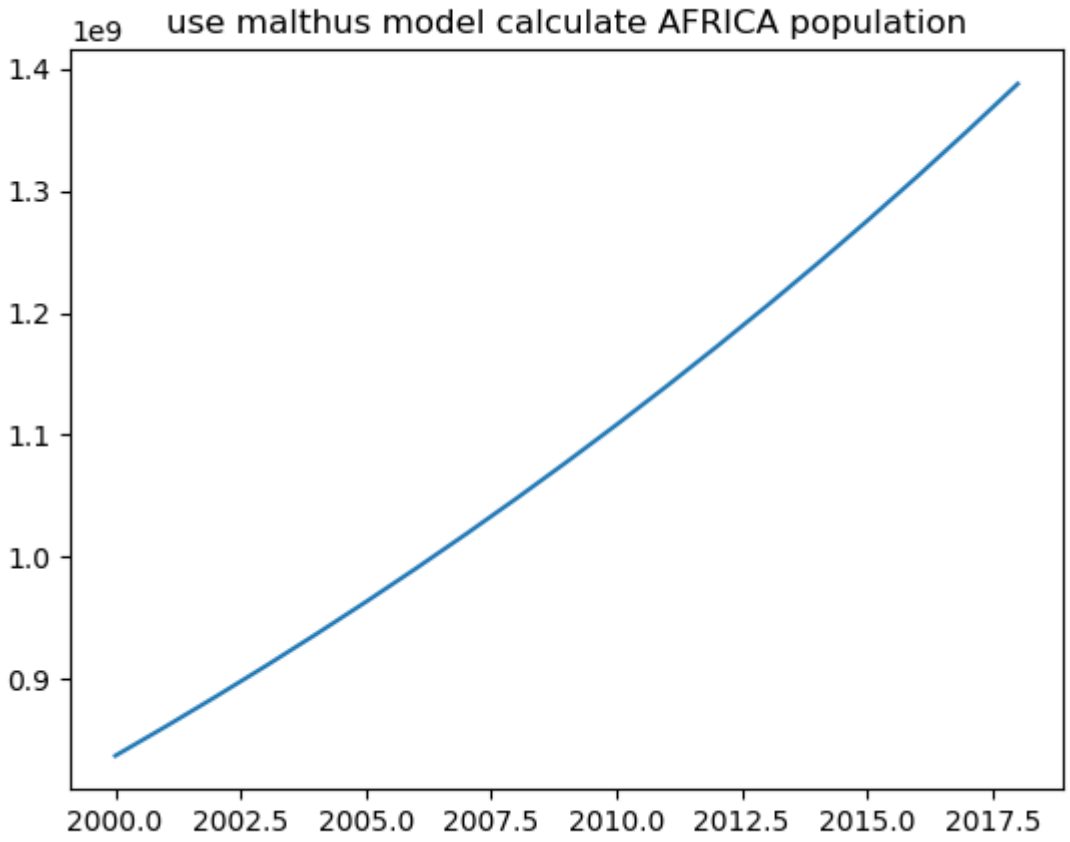
然後輸出想得知的國家編號或名稱，並以 `,` 作分隔

例如：`1, 222, 232` 或 `Serbia, South Africa` 或 `1, South Africa`

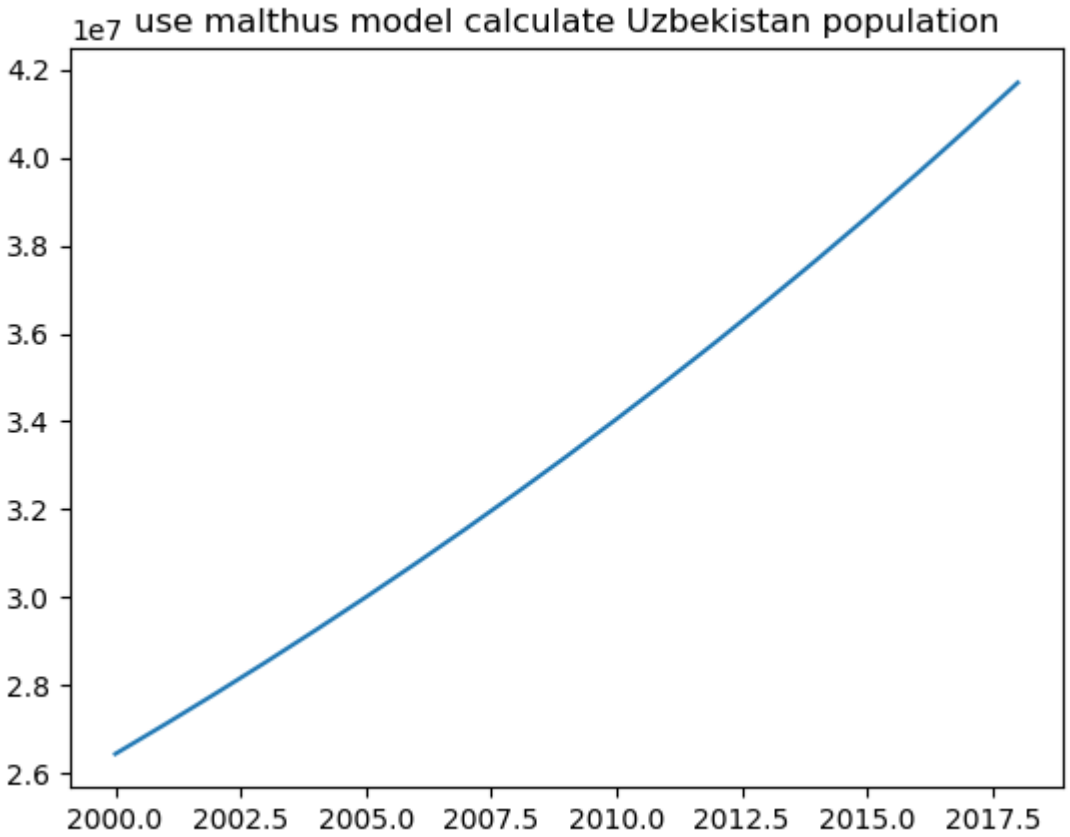
```
227: Western Asia                    228: Western Europe
229: Western Sahara                  230: World
231: Zambia                      232: Zimbabwe
請輸入上表中的國家: （例如： Swaziland, Thailand, Canda 或者 對應國家的數字編號如： 100, 102）
 >>> 1, 222
輸入開始年份及結束年份
開始年份： 2000
結束年份： 2018
```

並且輸入開始及結束年份

Output:

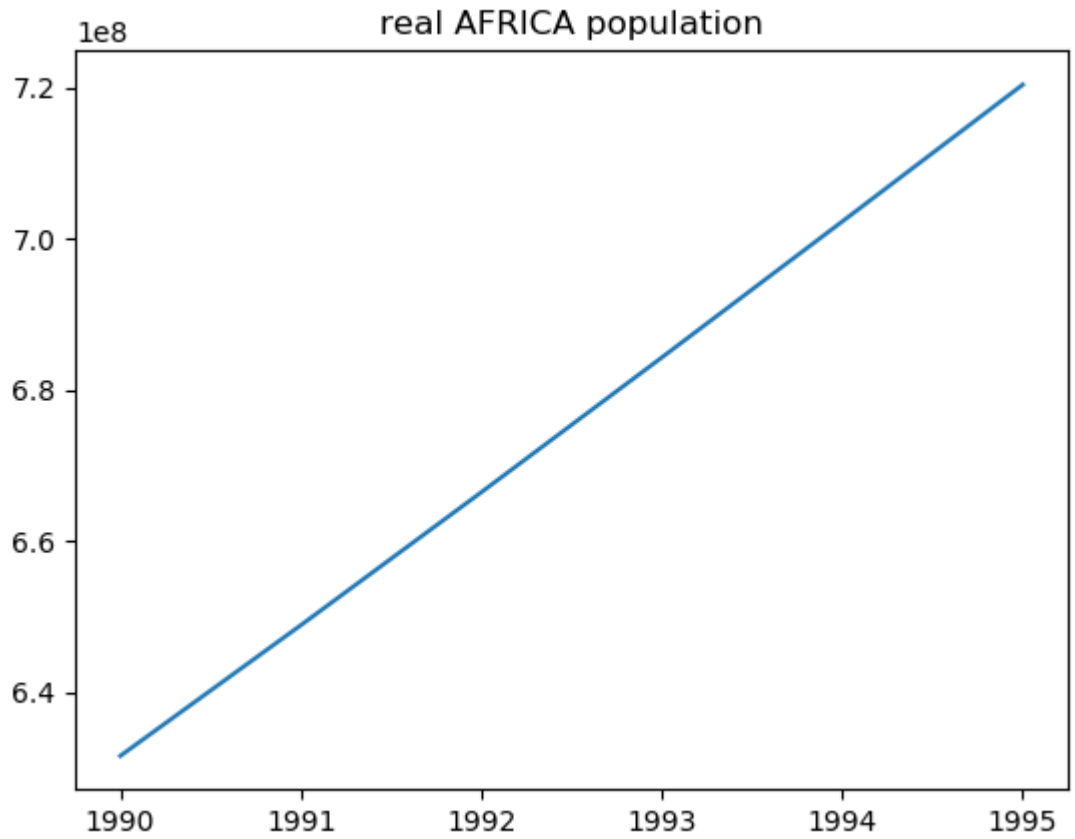



```
Country:  Uzbekistan     n:  {'females': 580, 'country': 'Uzbekistan', 'age': 98, 'males': 273, 'year': 1990, 'total': 853}
Country:  Uzbekistan     n:  {'females': 380, 'country': 'Uzbekistan', 'age': 99, 'males': 178, 'year': 1990, 'total': 558}
Country:  Uzbekistan     n:  {'females': 799, 'country': 'Uzbekistan', 'age': 100, 'males': 394, 'year': 1990, 'total': 1190}
AFRICA :
2000 :  836681522                    2001 :  860539876                    2002 :  885078563              2003 :  910316981
2004 :  936275084                    2005 :  962973394                    2006 :  990433019
2007 :  1018675667                   2008 :  1047723668                   2009 :  1077599985
2010 :  1108328239                   2011 :  1139932724                   2012 :  1172438424
2013 :  1205871039                   2014 :  1240257001                   2015 :  1275623494
2016 :  1311998479                   2017 :  1349410713                   2018 :  1387889773
```



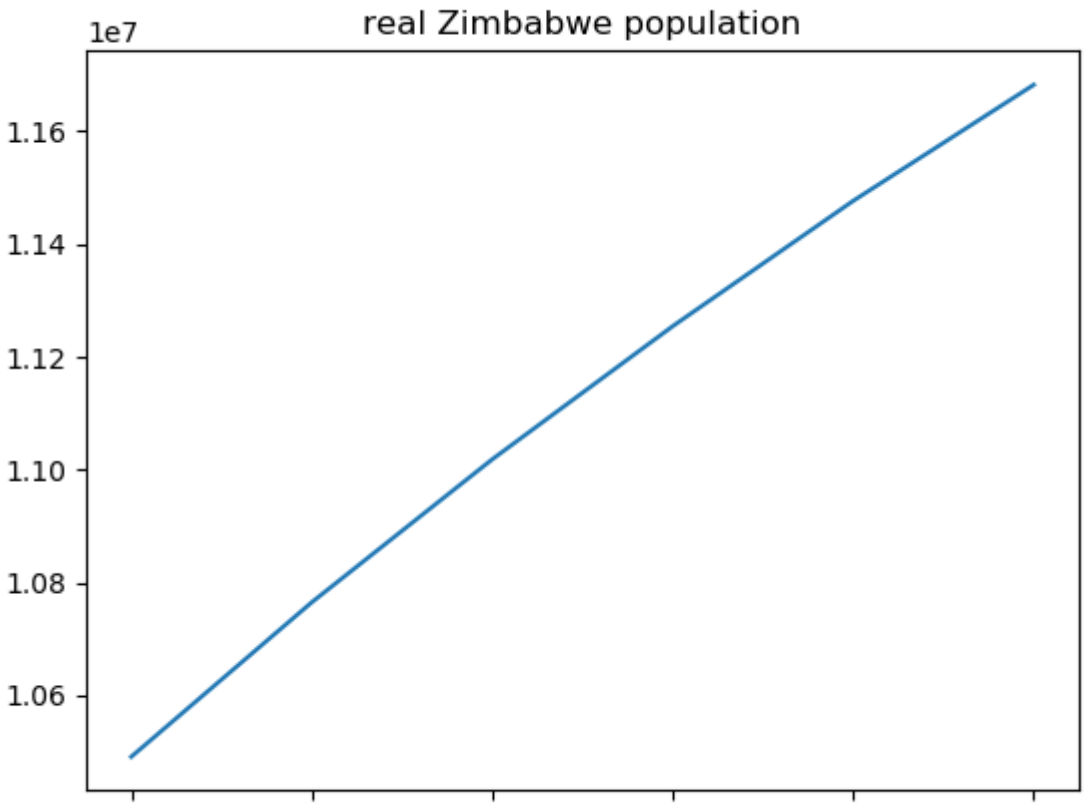

```
Uzbekistan :
2000 :  26432336                2001 :  27110620                2002 :  27806310
2003 :  28519853                2004 :  29251705                2005 :  30002338
2006 :  30772233                2007 :  31561884                2008 :  32371799
2009 :  33202497                2010 :  34054512                2011 :  34928390
2012 :  35824693                2013 :  36743997                2014 :  37686890
2015 :  38653980                2016 :  39645886                2017 :  40663246
2018 :  41706712
```

feature 2




```
AFRICA :
1990 :  631614304                1991 :  648899767                1992 :  666488668                1993 :  684318639
1994 :  702310012                1995 :  720416386
```

輸出



```
1990        1991        1992        1993        1994        1995
```

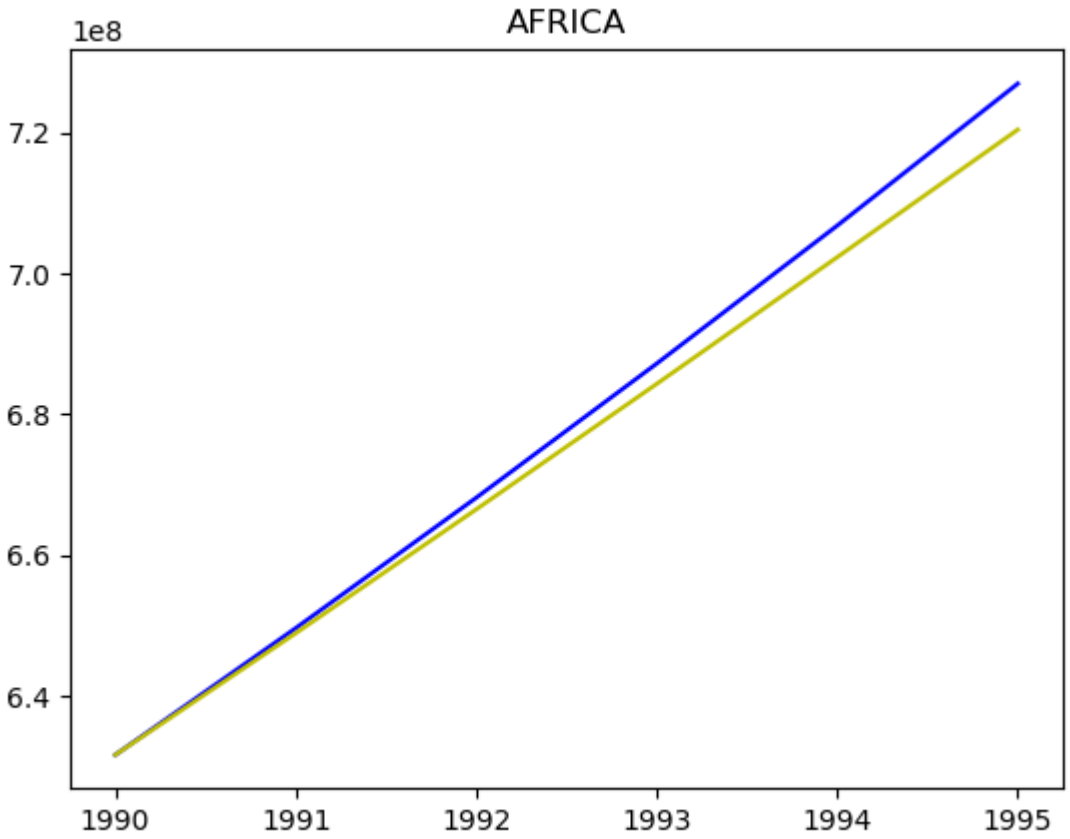
```
Zimbabwe :
1990 :  10490960
1991 :  10764938                    1992 :  11018504                    1993 :  11254580
1994 :  11476873                    1995 :  11682641
```

feature 3

```
Country:  Turkey        n:  {'females': 20500, 'country': 'Turkey', 'age': 74, 'males': 12500, 'year': 1954, 'total': 33
000}
Country:  Turkey        n:  {'females': 20100, 'country': 'Turkey', 'age': 75, 'males': 12200, 'year': 1954, 'total': 32
300}
Country:  Turkey        n:  {'females': 18800, 'country': 'Turkey', 'age': 76, 'males': 11600, 'year': 1954, 'total': 30
400}
Country:  Turkey        n:  {'females': 16200, 'country': 'Turkey', 'age': 77, 'males': 10100, 'year': 1954, 'total': 26
300}
Country:  Turkey        n:  {'females': 12800, 'country': 'Turkey', 'age': 78, 'males': 8110, 'year': 1954, 'total': 209
00}
Country:  Turkey        n:  {'females': 9770, 'country': 'Turkey', 'age': 79, 'males': 6390, 'year': 1954, 'total': 1620
0}
Country:  Turkey        n:  {'females': 4750, 'country': 'Turkey', 'age': 80, 'males': 4750, 'year': 1954, 'total': 4750
}
Country:  Turkey        n:  {'females': 3440, 'country': 'Turkey', 'age': 81, 'males': 3440, 'year': 1954, 'total': 3440
}
Country:  Turkey        n:  {'females': 2650, 'country': 'Turkey', 'age': 82, 'males': 2650, 'year': 1954, 'total': 2650
}
Country:  Turkey        n:  {'females': 2210, 'country': 'Turkey', 'age': 83, 'males': 2210, 'year': 1954, 'total': 2210
}
Country:  Turkey        n:  {'females': 1770, 'country': 'Turkey', 'age': 84, 'males': 1770, 'year': 1954, 'total': 1770
}
Country:  Turkey        n:  {'females': 1400, 'country': 'Turkey', 'age': 85, 'males': 1400, 'year': 1954, 'total': 1400
}
Country:  Turkey        n:  {'females': 1100, 'country': 'Turkey', 'age': 86, 'males': 1100, 'year': 1954, 'total': 1100
}
Country:  Turkey        n:  {'females': 818, 'country': 'Turkey', 'age': 87, 'males': 818, 'year': 1954, 'total': 818}
Country:  Turkey        n:  {'females': 577, 'country': 'Turkey', 'age': 88, 'males': 577, 'year': 1954, 'total': 577}
Country:  Turkey        n:  {'females': 414, 'country': 'Turkey', 'age': 89, 'males': 414, 'year': 1954, 'total': 414}
Country:  Turkey        n:  {'females': 320, 'country': 'Turkey', 'age': 90, 'males': 320, 'year': 1954, 'total': 320}
Country:  Turkey        n:  {'females': 244, 'country': 'Turkey', 'age': 91, 'males': 244, 'year': 1954, 'total': 244}
Country:  Turkey        n:  {'females': 167, 'country': 'Turkey', 'age': 92, 'males': 167, 'year': 1954, 'total': 167}
Country:  Turkey        n:  {'females': 95, 'country': 'Turkey', 'age': 93, 'males': 95, 'year': 1954, 'total': 95}
Country:  Turkey        n:  {'females': 62, 'country': 'Turkey', 'age': 94, 'males': 62, 'year': 1954, 'total': 62}
Country:  Turkey        n:  {'females': 45, 'country': 'Turkey', 'age': 95, 'males': 45, 'year': 1954, 'total': 45}
Country:  Turkey        n:  {'females': 31, 'country': 'Turkey', 'age': 96, 'males': 31, 'year': 1954, 'total': 31}
Country:  Turkey        n:  {'females': 19, 'country': 'Turkey', 'age': 97, 'males': 19, 'year': 1954, 'total': 19}
Country:  Turkey        n:  {'females': 8, 'country': 'Turkey', 'age': 98, 'males': 8, 'year': 1954, 'total': 8}
Country:  Turkey        n:  {'females': 3, 'country': 'Turkey', 'age': 99, 'males': 3, 'year': 1954, 'total': 3}
Country:  Turkey        n:  {'females': 2, 'country': 'Turkey', 'age': 100, 'males': 2, 'year': 1954, 'total': 2}
```

輸出結果

feature 4



此模式下 藍色為預測人口數量 而 黃色為實際人口數量

feature 5

```
λ python Main.py
人口增長推測
Author: Lo Ben (loben@illimited.cf)

1. 利用Malthus推算國家人口
2. 輸出實際人口數量
3. 輸出所有地區的人口數量
4. 分別輸出Malthus和實際人口數據及圖形
5. 結合feature4，將所有數據儲存在html檔案裏面
>>> 5
將所有數據輸出成html
1. 未執行feature3
2. 已執行feature3

>>>2
請輸入您想要的檔案名稱：1990
需要加入readme嗎？(y or n): y
```

這裏有兩個選項，對應之前是否已經輸出數據 同時可以將`README.md`輸出成`html`

人口增長推測

[toc]

前言

在現今人口增長迅速、糧食不足、土地缺乏、老年化的情況下，人口數量已成為一個大問題。 故此本人選擇以人口作為今次微積分python報告的題目，希望能藉此對人口、python及數學有進一步

人口增長推測

本人參考了人口成長模型這篇文章的數學模型去編寫Malthus及Logistic這兩個class。

利用Malthus推測人口增長

Malthus數學模組是由英國經濟學家Malthus在1798年匿名發表的《人口原理》中用來描述人口成長的數學模組。

$$P(t) = P0$$

t 為當前所求人口的時間 $P(t)$ 為求出的人口 $t0$ 為任意時間點 $P0$ 為任意時間點的人口 λ 為一常數（需由計算求得）

而本人利用上式推導以求出λ：

$$\lambda = \frac{ln($$

功能

- 藉由 **Malthus** 及 api.population.io 所取得的資料去推算某一（數個）地區的人口數量。
- 輸出 **Malthus** 預測及實際人口數量圖形
- 輸出結果為html file

教學

- 請先在主程式的資料夾下新增一個名為"temp"的folder用作儲存輸出的圖片

主畫面

```
D:\Workstation\University\1062\MA1202-微積分 python
λ python Main.py
人口增長推測
Author: Lo Ben (loben@illimited.cf)

1. 利用Malthus推算國家人口
2. 輸出實際人口數量
3. 輸出所有地區的人口數量
4. 分別輸出Malthus和實際人口數據及圖形
5. 結合feature4，將所有數據儲存在html檔案裏面
>>>
>>>
```

主畫面有5個功能

1. 第一個功能是透過Malthus去求出特定時間段內特定國家的人口數量（彈出windows）
2. 藉由網上提供的api輸出特定時間段內的實際人口數量（彈出windows）
3. 儲存特定時間段內所有國家的實際人數及 **Malthus** 預測的人口數量（儲存為png圖片及json）
4. 顯示特定時間段內所有國家的實際人數及 **Malthus** 預測的人口數量（彈出windows)
5. 將所有結果輸出成html檔案

feature 1

選擇feature1

此模式下 **藍色為預測人口數量** 而 **黃色為實際人口數量**

feature

感想

經過是次報告令我對python的語法有更進一步的認識，相對於分組報告，我更喜歡自己一個人工作，反正也是只有自己一個人工作。

而且對 `html` 有進一步的認識，本人之前從未使用過 `table`，原因是 `table` 的語法相當繁瑣，不利於網頁編寫，而且在較舊的網頁 `table` 通常用作排版之用。

而輸出的結果也令我很食驚，想不到 `Malthus` 的模型對於某些國家還適用。

改進

對於老師要求的object oriented，我只能實現一部份，原因是我沒有太多寫程式的經驗，以致無法活用object oriented的概念。

但基於上學期作業的經驗，此次的報告應該會更好一點。

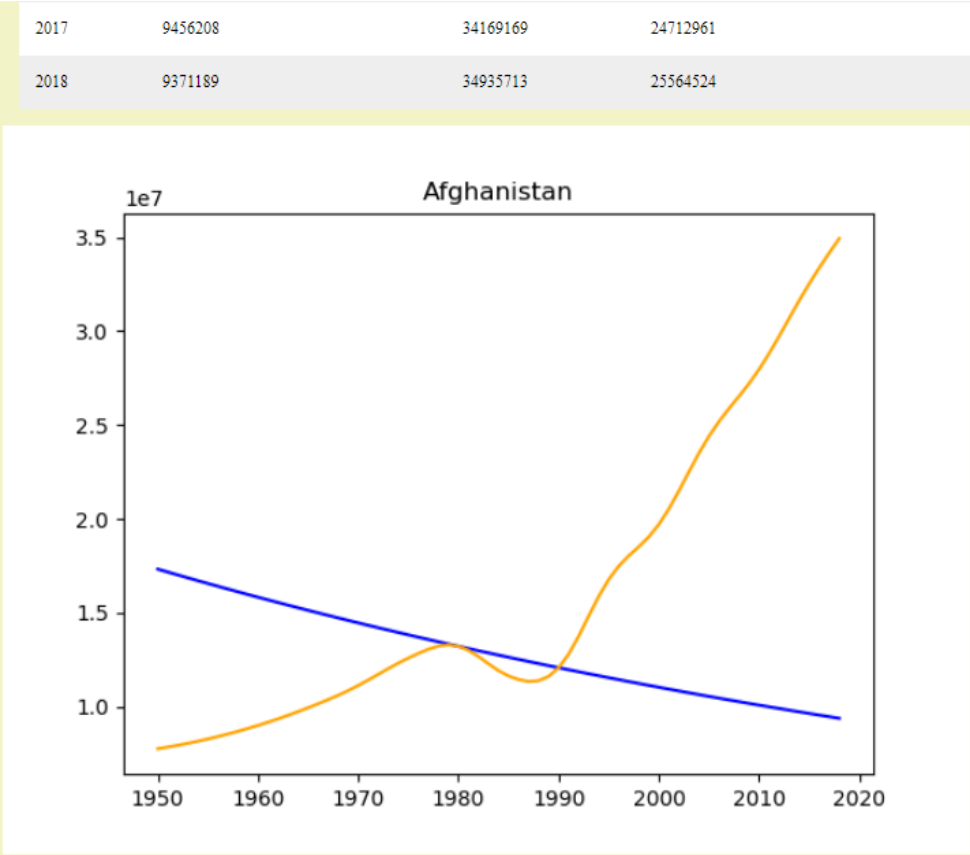
而且我認為可以在報告新增更多人口預測模型，從而比較那一模型更貼及現況。

以及輸出的網頁美化不足。

Afghanistan.png

年份	Malthus模型預測	真實人口	預測與實際人口相差
1950	17318601	7752236	-9566365
1951	17162893	7839500	-9323393
1952	17008584	7934792	-9073792
1953	16855663	8038199	-8817464
1954	16704117	8149787	-8554330
1955	16553933	8269601	-8284332
1956	16405099	8397789	-8007310
1957	16257604	8534300	-7723304
1958	16111435	8679134	-7432301
1959	15966580	8832291	-7134289
1960	15823027	8993806	-6829221
1961	15680765	9163838	-6516927
1962	15539782	9342548	-6197234
1963	15400066	9530204	-5869862

2017	9456208	34169169	24712961
2018	9371189	34935713	25564524



AFRICA.png

年份	Malthus模型預測	真實人口	預測與實際人口相差
1950	205125225	228562498	23437273
1951	210974465	233065181	22090716
1952	216990498	237819151	20828653

程式碼解說

Main.py

```
from population_api import * #使用網路提供的API
from Malthus_Country import * #以Malthus class作為基礎寫出來的class
import pylab #用作繪圖之用
import math
import HTML_write as i #輸出網頁之用
import re #使用regular expression

def main():
    '''
    人口增長推測
    Author: Lo Ben (loben@illimited.cf)
    '''


    info = "人口增長推測 \nAuthor: Lo Ben (loben@illimited.cf) \n"
```

```
    print(info)

    feature = "1. 利用Malthus推算國家人口 \n2. 輸出實際人口數量 \n3. 輸出所有地區的人口數量 \n4. 分別輸出Malthus和實際人口數據及圖形 \n5. 結合feature4，將所有數據儲存在html檔案裏面 \n>>> "
    selection = int(input(feature))

    if int(selection) == 1:
        feature1() #執行feature1
    elif int(selection) == 2:
        feature2() #執行feature2
    elif int(selection) == 3:
        feature3() #執行feature3
    elif int(selection) == 4:
        feature4() #執行feature4
    elif int(selection) == 5:
        feature5() #執行feature5

def feature5():
    info = "將所有數據輸出成html \n1. 未執行feature3 \n2. 已執行feature3\n"
    print(info)
    opition = input(">>>")
    if opition == 1:
        feature3()
        title = input("請輸入您想要的檔案名稱：")
        add_readme = input("需要加入readme嗎？(y or n): ")
        add_readme_bool = True
        if re.search("y", add_readme) == None:
            add_readme_bool = False
        i.HTML_write(title, add_readme = add_readme_bool)
    else:
        title = input("請輸入您想要的檔案名稱：")
        add_readme = input("需要加入readme嗎？(y or n): ")
        add_readme_bool = True
        if re.search("y", add_readme) == None:
            add_readme_bool = False
        i.HTML_write(title, add_readme = add_readme_bool)


def saveData2Json(data, country):
    '''
    將得出的人口數據轉存成json格式
    '''
    path = "temp/"
    with open(path+country+".json", "w") as outfile:
        json.dump(data, outfile)

def saveImg(data, country):
    '''
    將得出的數據繪圖成圖片
    '''
    #covert dict to list
    cal_xs, cal_ys = [], []

    real_xs, real_ys = [], []
```

```
    for i in data[0]:
        cal_xs.append(i)
        cal_ys.append(data[0][i])
    for j in data[1]:
        real_xs.append(j)
        real_ys.append(data[1][j])
    #save figure
    grap = pylab
    grap.clf()
    grap.title(country)
    grap.plot(cal_xs, cal_ys, "b")
    grap.plot(real_xs, real_ys, "#FFA500")
    grap.savefig("temp/"+str(country))


def standard_deviation_of_compare(data_, real_population):
    '''
    計算標準差以比較那一地區人口數量更貼近Malthus數學模型的預測
    '''
    sn = 0
    temp = 0

    data_list = []
    real_population_list = []

    #convert dict to list
    for a in data_.values():
        data_list.append(a)
    for b in real_population.values():
        real_population_list.append(b)
    try:
        for a in range(len(data_list)):
            temp += math.pow(real_population_list[a] - data_list[a], 2)
        sn = math.sqrt(temp/len(data_)-1)
    except:
        sn = None
    return sn

def feature1():
    _input = input_country()
    info = "輸入開始年份及結束年份"
    print(info)

    while(True):
        _begin = int(input("開始年份： "))
        _end = int(input("結束年份： "))
        if _begin < _end:
            break
        else:
            print("輸入錯誤，請重新輸入")

    #Malthus Calculate

    print("\n\n使用Malthus計算出的數據：")
```

```
    malthus_model = []
    for n in _input:
        malthus_model.append(Malthus_Country(n).get_interval_population(_begin, _end))
    index = 0
    counter = 0
    for i in malthus_model:
        Malthus_xs, Malthus_ys = [], []
        print(_input[index], ":")
        for j in i:
            Malthus_xs += [j]
            Malthus_ys += [i[j]]
            print(j, ": ", i[j], end = "\t\t\t")
            if counter > 2:
                print()#LF
                counter = 0
            counter += 1
        print("\n")
        pylab.title("use malthus model calculate "+_input[index]+" population")
        pylab.plot(Malthus_xs, Malthus_ys)
        pylab.show()
        index += 1

def feature2():
    _input = input_country()
    print(_input)
    info = "輸入開始年份及結束年份"
    print(info)

    while(True):
        _begin = int(input("開始年份： "))
        _end = int(input("結束年份： "))
        if _begin < _end:
            break
        else:
            print("輸入錯誤，請重新輸入")
    #Get data from website
    print("\n\n實際求出的數據：")
    population_api_result = []
    for n in _input:
        population_api_result.append(population_api(n).population_of_year_interval(_begin,
_end))
    index = 0
    counter = 0
    for i in population_api_result:
        population_xs, popylation_ys = [], []
        print(_input[index], ":")
        for j in i:
            population_xs += [j]
            popylation_ys += [i[j]]
            print(j, ": ", i[j], end = "\t\t\t")
            if counter > 2:
                print()#LF

                counter = 0
```

```
            counter += 1
        print("\n")
        pylab.title("real "+_input[index]+" population")
        pylab.plot(population_xs, popylation_ys)
        pylab.show()
        index += 1

def feature3():

    info = "這個選項可以把所有國家的人口數量及以Malthus推算的人口數量畫出，並儲存在 .\\temp 裏面 \n而圖中藍色線為真實人口數量，而燈色線則為利用Malthus推測的人口數量"
    print(info)

    #檢查開始和結束時間上時是否有logic error
    while(True):
        _begin = int(input("開始年份： "))
        _end = int(input("結束年份： "))
        if _begin < _end:
            break
        else:
            print("輸入錯誤，請重新輸入")

    all_countries = population_api(None).all_countries()
    data = {} #store country all data
    ext_data = {}
    for country in all_countries:
        #malthus_model
        malthus_model = Malthus_Country(country).get_interval_population(_begin, _end)
        #real population
        real_population = population_api(country).population_of_year_interval(_begin, _end)

        standard_deviation_of_compare(malthus_model, real_population)

        #data
        data = [malthus_model, real_population, standard_deviation_of_compare(malthus_model, real_population)]

        #ext_data
        ext_data[country] =  data

        saveImg(data, country)
        saveData2Json(data, country)

    #for test only
    #print(data)


def feature4():
    _input = input_country()
    info = "輸入開始年份及結束年份，並輸出以Malthus計算與實際人口數量的圖形"
    print(info)


    while(True):
```

```
        _begin = int(input("開始年份： "))
        _end = int(input("結束年份： "))
        if _begin < _end:
            break
        else:
            print("輸入錯誤，請重新輸入")

    #Malthus Calculate
    print("\n\n使用Malthus計算出的數據：")
    malthus_model = []
    for n in _input:
        malthus_model.append(Malthus_Country(n).get_interval_population(_begin, _end))
    index = 0
    counter = 0
    for i in malthus_model:
        Malthus_xs, Malthus_ys = [], []
        print(_input[index], ":")
        for j in i:
            Malthus_xs += [j]
            Malthus_ys += [i[j]]
            print(j, ": ", i[j], end = "\t\t\t")
            if counter > 2:
                print()#LF
                counter = 0
            counter += 1
        print("\n")
        pylab.title(_input[index])
        pylab.plot(Malthus_xs, Malthus_ys, "b")
        #pylab.show()
        index += 1


    #Get data from website
    print("\n\n實際求出的數據：")
    population_api_result = []
    for n in _input:
        population_api_result.append(population_api(n).population_of_year_interval(_begin,
_end))
    index = 0
    counter = 0
    for i in population_api_result:
        population_xs, popylation_ys = [], []
        print(_input[index], ":")
        for j in i:
            population_xs += [j]
            popylation_ys += [i[j]]
            print(j, ": ", i[j], end = "\t\t\t")
            if counter > 2:
                print()#LF
                counter = 0
            counter += 1
        print("\n")

        #pylab.title("real "+_input[index]+" population")
```

```
        pylab.plot(population_xs, popylation_ys, "y")
        pylab.show()
        index += 1

def input_country():
    '''
    接受使用者輸入
    '''
    while True:
        all_countries = list_all_coutries() #list all countries

        country = input("請輸入上表中的國家: （例如： Swaziland, Thailand, Canda 或者 對應國家的數字編號如： 100, 102） \n >>> ")
        input_list = country.split(",")

        if(is_input_valid(all_countries, input_list)):
            break
        else:
            print("輸入錯誤，請重新輸入。 你的輸入是：", input_list)

    index = 0
    for n in input_list:
        n = n.strip()
        if n.isnumeric():
            input_list[index] = all_countries[int(n)]
        index += 1
    return input_list


def list_all_coutries():
    '''
    輸出所有可用的國家
    '''
    all_countries = population_api(None).all_countries()
    index, i = 1, 0 #counter
    for n in all_countries:
        print(index, n, sep = ": ", end="\t\t\t")
        index += 1
        i += 1
        if (i > 1):
            print() #LF
            i = 0
    return all_countries

def is_input_valid(all_countries, input_list):
    '''
    判定輸入的國家是否在可求出答案的範圍內
    '''
    for n in input_list:
        n = n.lstrip() #remove left space
        a = n.strip() #remove all space
        if a.isnumeric():

            if int(a) <= len(all_countries) and int(a) > 0: #is input number between index
```

```
                pass
            else:
                return False
                break
        elif not n.isnumeric() and n not in all_countries: #is input string in the all countries
            return False
            break
    return True

if __name__ == "__main__":
    main()
```

Malthus.py

```
from math import log
from math import exp
from math import pow

class Malthus:
    '''
    Use Malthus Model to calculate population.
    '''
    def __init__(self, t0 = None, p0 = None, t1=None, p1=None, _lambda=None):
        '''
        Args:
            t0: 任意時間點
            p0: 任意時間點的人口
            t1: 特定時間點
            p1: 特定時間點的人口
            _lambda: 可以代入已知constant
        '''
        self.t0 = t0
        self.p0 = p0
        self.t1 = t1
        self.p1 = p1
        self._lambda = _lambda

        if (_lambda == None and (t0 != None and p0 != None)) and t1 != None and p1 != None:
            self.calculate_lambda()


    def calculate_lambda(self, t1 = None, p1 = None):
        '''
        Calculate constant lambda
        '''
        try:
            _lambda = lambda _t1, _p1: log(_p1/self.p0)/(_t1-self.t0)
            if t1 != None and p1 != None:
                self._lambda = _lambda(t1, p1)
                return self._lambda
            else:
                self._lambda = _lambda(self.t1, self.p1)
```

```
                return self._lambda
            except:
                print("Error model!!!")


    def calculate_population(self, t1 = None):
        '''
        Calculate population
        args:
            t1: 特定時間
        '''
        try:
            _population = lambda _t1: self.p0 * pow(exp(1), self._lambda * (_t1 - self.t0))
            if t1 != None:
                self.population = _population(t1)
                return int(self.population)
            else:
                self.population = _population(self.t1)
                return int(self.population)
        except:
            print("Error model!!!")

if __name__ == "__main__":
    a = Malthus(1966, 1300e6, 1971.1, 1500e6)
    print("calculate lambda: ", a.calculate_lambda())
    print("calculate population: ", a.calculate_population(2000))
```

Malthus_Country.py

```
from Malthus import *
from population_api import *

class Malthus_Country:
    def __init__(self, country, default_begin_year = 1980, interval = 10):
        self.country = country
        self.default_begin_year = default_begin_year
        self.interval = interval

        self.get_essential_info() #get population to build up Malthus Module
        self.math_obj() #build up Malthus Module

    def get_essential_info(self):
        '''
        藉由預設的年份可以建立Malthus模型
        '''
        self.p0 = population_api(self.country).population_of_year(self.default_begin_year)
        self.p1 = population_api(self.country).population_of_year(self.default_begin_year +
self.interval)

    def math_obj(self):

        self.Malthus_obj = Malthus(self.default_begin_year, self.p0, self.default_begin_year +
```

```
self.interval, self.p1)

    def get_population_of_year(self, year):
        try:
            return int(self.Malthus_obj.calculate_population(year))
        except:
            print("get_population_of_year error!!!")
            return 0

    def get_interval_population(self, begin_year, end_year):
        '''
        取得特定時間區間內的人口數量
        '''
        result = {}
        for n in range(end_year - begin_year + 1):
            result[begin_year + n] = self.get_population_of_year(begin_year + n)
        return result

def main():
    '''
    測試用程式碼
    '''
    a = Malthus_Country("World")
    print(a.get_interval_population(1990, 2012))

if __name__ == "__main__":
    main()
```

population_api.py

```
import requests #由於要使用到population網頁提供的api，故需用到requests
import json #解讀json

api_url = "http://api.population.io/1.0/" #api url

class population_api:
    '''
    api.population.io python api
    Author: Lo Ben (loben@illimited.cf)
    '''
    def __init__(self, country):
        self.country = country
        available_countries = self.all_countries() #find out which country have population
result
        if (country not in available_countries):
            #print("Not available country!!!")
            pass

    def all_countries(self):
        '''
        list available countries.
        '''
```

```
        json_data = json.loads(requests.get(api_url+"/countries").text)
        result = json_data["countries"]
        result.remove("Australia/New Zealand")
        result.remove("Least developed countries")
        result.remove("ASIA")
        result.remove("Less developed regions, excluding least developed countries")
        return result

    def wp_rank(self, dob = None, sex = None, country = None, require = None, date = None):
        '''
        determine world population rank.
        argv:
            dob (date string): the given date of birth, example: "1952-03-11"
            sex (string): the given sex, example: "male"
            country (string): the given country, example: "United Kingdom"
        return:
            rank (int): the calculated rank, example: 27228942
        '''

        if require == "today":
            payload = "{0}/wp-rank/{1}/{2}/{3}/{4}/".format(api_url, dob, sex, self.country,
require)
            json_result = json.loads(requests.get(payload).text)
            return json_result['rank']

    def population_of_year(self, year = None):
        '''
        return all population of year in specific country.
        argv:
            year (int): the give year, example: 1980
            country(String): the given country, example:"United Kingdom"
        '''

        payload = "{0}/population/{1}/{2}".format(api_url, int(year), self.country) #construct
url
        json_result = json.loads(requests.get(payload).text)

        sum_of_population = 0

        for n in json_result:
            #for debug
            print("Country: ", self.country, "\t n: ", n)
            try:
                sum_of_population += int(n['males']) + int(n['females'])
            except:
                print("錯誤！")
            #print(sum_of_population)

        return sum_of_population

    def population_of_year_interval(self, begin_year, end_year):
        result = {}

        for n in range(end_year - begin_year + 1):
```

```
            result[begin_year + n] = self.population_of_year(begin_year + n)
        return result


if __name__ == "__main__":
    '''
    a = population_api("Japan") #create class
    temp = a.wp_rank(dob = "2018-1-1", sex = "female", require = "today") #test wp_rank class
    #print("test wp_rank function:" + str(temp))
    print(a.population_of_year(1955))
    '''
    a = population_api("World") #create class
    a.all_countries()
    #print(a.population_of_year(1955))
```

HTML_write.py

```
import markdown2
import os
from bs4 import BeautifulSoup
import base64
import re
from pathlib import Path
import json

class HTML_write:
    def __init__(self, title, css = "custom_css.html", js = "custom_js.html", add_readme =
False):
        self.title = title
        #css
        self.css = ""
        with open(css, "r", encoding="utf-8") as i:
            self.css += i.read()
        #js
        self.js = ""
        with open(js, "r", encoding="utf-8") as i:
            self.js += i.read()

        #construct head
        self.head = self.__construct_head()

        #store result
        self.table = []

        self.filename = []

        self.sn = []

        self.sn_filename_dict = {}

        self.construct_html(self.title, add_readme)
```

```
    def __construct_head(self):
        head = '''
            <!doctype html>
            <html>
                <head>
                    <meta charset = "utf-8">
                    <meta name="author" content="Lo Ben">
                    <title>'''+self.title+'''</title>
                    '''
        head += str(self.css)+str(self.js)
        head += '''
                </head>
                <body>
            '''
        return head

    def README(self):
        readme = '''
                <div class = "README">
        ''' + str(markdown2.markdown_path("README.md")).replace("<em>", "").replace("</em>", "")
+'''
                </div>
        '''
        return readme

    def __img2base64_in_Path(self):
        '''
        covert img to base64
        '''
        img_tag_list = []

        path = "temp/"
        for filename in os.listdir(path):
            if re.search(".png", filename) != None:
                self.filename.append(filename.rstrip(".png"))
                ext_filename = re.findall('\..*$', filename)[0].strip(".")
                payload = ""
                print(filename + "已加入")

                image = open(path + filename, 'rb')
                image_read = image.read()
                image_64_encode = base64.b64encode(image_read)
                image_64_encode = payload + str(image_64_encode).lstrip("b'").rstrip("'")
                #img_tag = '<img src = "'+image_64_encode+'" alt = "'+filename+'"/>' #construct
img tag
                img_tag_list.append([image_64_encode, filename])
        self.img_tag_list = img_tag_list

    def __construct_tail(self):
        html_head = "</body></html>"
        return html_head
```

```
    def __img_tag2html(self):
        self.__img2base64_in_Path()
        img_html_code = ""

        '''
        for i in self.img_tag_list:
            img_html_code += '<div class = "figure"><h3 class = "figure_title">'+i[1]+'</h3>
<img src = "'+i[0]+'"></div>'
        '''
        for i in range(len(self.img_tag_list)):
            img_html_code += '<div class = "figure"><h3 class =
"figure_title">'+self.img_tag_list[i][1]+'</h3>'
            img_html_code += self.table[i]
            img_html_code += '<img src = "'+self.img_tag_list[i][0]+'"></div><hr>'
        return img_html_code

    def __result(self):
        code = '<div class = "result">'
        code += self.__img_tag2html()
        code += '</div>'
        return code

    def construct_html(self, filename = None, add_readme = False):
        html_code = ""
        html_code += self.__construct_head() #add head
        if add_readme:
            html_code += self.README() #add readme
        self.construct_hole_table() #construct result table
        html_code += self.__result() #add result img
        html_code += self.__construct_tail()

        output_html_file = open(filename+".html", "w", encoding="utf-8")
        output_html_file.write(html_code)


    def construct_hole_table(self):
        _temp_list = []
        path = "temp/"
        for filename in os.listdir(path):
            if re.search(".json", filename) != None:
                self.__process_a_table(self.__load_json(filename))


    def __process_a_table(self, list):
        table = "<table class = 'rwd-table'>"
        table += "<tr><th>年份</th><th>Malthus模型預測</th><th>真實人口</th><th>預測與實際人口相差
</th></tr>"
        year_list = [i for i in list[0]]
        malthus_data_list = [list[0][i] for i in list[0]]
        real_population_list = [list[1][i] for i in list[1]]
        sd = list[2]
        for i in range(len(year_list)):

            table += "<tr>"
```

```
            table += "<td>"+str(year_list[i])+"</td>"
            table += "<td>"+str(malthus_data_list[i])+"</td>"
            table += "<td>"+str(real_population_list[i])+"</td>"
            if real_population_list[i]-malthus_data_list[i] >= 0:
                table += "<td class = 'positive'>"+str(real_population_list[i]-
malthus_data_list[i])+"</td>"
            else:
                table += "<td class = 'negative'>"+str(real_population_list[i]-
malthus_data_list[i])+"</td>"
            table += "</tr>"
        table += "</table>"
        self.table += [table]
        #self.sn += ["<span class= 'sd'>"+str(sd)+"</span>"]

        #test
        return table

        #open("test_table.html", "w").write(table)
        #print(table)

    def __popluation_data_(self):
        path = "temp/"
        self.__data_list = []
        for filename in os.listdir(path):
            if re.search(".json", filename) != None:
                self.__data_list += [self.__load_json(filename)]

        print(self.__data_list)

    def __load_json(self, filename):
        path = "temp/"
        json_data_list = None
        with open(path+filename, "r", encoding="utf-8") as i:
            json_data_list = json.loads(i.read())
        return json_data_list

def main():
    a = HTML_write("test")
    a.construct_html("html_test.html", True)
    a.construct_hole_table()
    #test
    '''
    with open("temp/Afghanistan.json", "r", encoding="utf-8") as i:
            json_data_list = json.loads(i.read())
    print(json_data_list, type(json_data_list))
    print("-"*99)
    a.process_a_table(json_data_list)
    '''


if __name__ == "__main__":
    main()
```

- `Malthus_Country.py` 建立的目的是為了更方便去查出特定期間內人口的數量

感想

經過是次報告令我對python的語法有更進一步的認識，相對於分組報告，我更喜歡自己一個人工作，反正也是只有自己一個人工作。

而且對 `html` 有進一步的認識，本人之前從未使用過 `table`，原因是 `table` 的語法相當繁瑣，不利於網頁網寫，而且在較舊的網頁 `table` 通常用作排版之用。

而輸出的結果也令我很食驚，想不到 `Malthus` 的模型對於某些國家還適用。

改進

對於老師要求的object oriented，我只能實現一部份，原因是我沒有太多寫程式的經驗，以致無法活用object oriented的概念。

但基於上學期作業的經驗，此次的報告應該會更好一點。

而且我認為可以在報告新增更多人口預測模型，從而比較那一模型更貼及現況。

以及輸出的網頁美化不足。

完整程式碼

請到github上下載