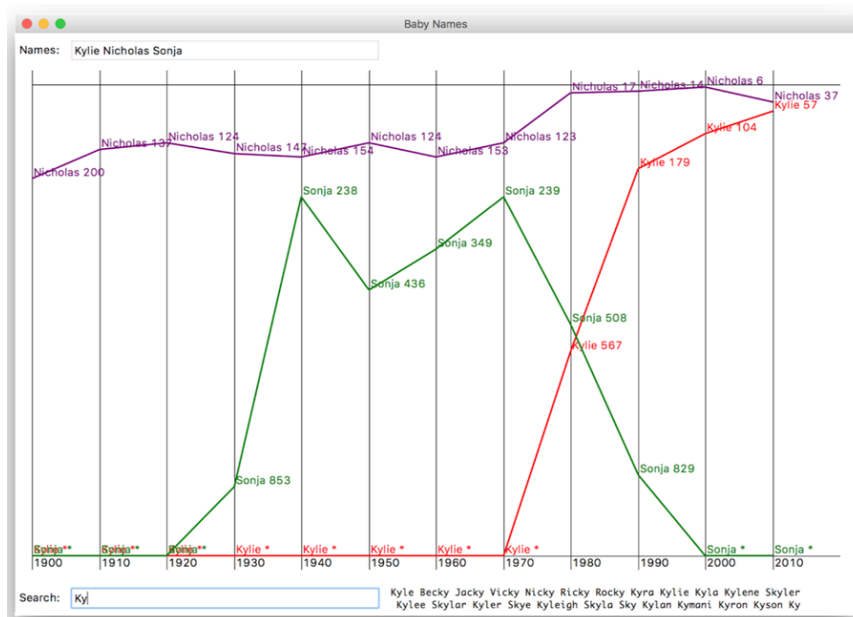


## Assignment 4

This assignment is based on the Assignment 4 of CS106P at Stanford University



點此下載作業檔案

歡迎來到大數據處理世界！不知道同學有沒有發現，我們從第一次接觸 Python 到現在，已經變得有能力來完成一些軟體工程師的工作了！我們相信您這些為作業苦思、熬夜、備感壓力的時光，也絕對會得到相對應 coding 能力提升的回饋！

作業預計時間：15 小時

如果過程卡關歡迎各位向助教詢問！也非常鼓勵同學們互相討論作業之概念，但請勿直接把 code 分享給同學看，這很可能會剝奪他獨立思考的機會，並讓他的程式碼與你的極度相似，使防抄襲軟體認定有抄襲嫌疑。

## Baby Names 介紹

「Baby Names」是一個檢索在 1900 - 2010 年間，全美新生兒英文名字受歡迎程度排名的程式。這份作業內容與一位資料科學家的日常非常相似。請同學務必安排好時間，一步步完成本次作業的 milestones，並反覆複習上課學到的資料結構觀念。

每一年美國社會安全局都會在網站 (<http://www.ssa.gov/OACT/babynames>) 上公佈最「夯」的 Top 1000 名新生兒名字。網站上的資料呈現如下圖1.所示：

Rank	Male name	Female name
1	Jacob	Emily
2	Michael	Hannah
3	Matthew	Madison
4	Joshua	Ashley
5	Christopher	Sarah
...		

圖1. 西元 2000 年新生兒名字最夯程度排名（由左至右依序為：排名 / 男生名 / 女生名）

Rank 1 代表當年度最夯的名字；Rank 1000 以後的名字就沒有記錄了。我們已經將網站上的資料整理成如下圖2.所示的文字檔，同學們可以在「data/full」資料夾中找到在西元 1990 ~ 2010 年間的所有資料。

<b>baby-1980.txt</b>	<b>baby-2000.txt</b>
1980 1,Michael, Jennifer 2,Christopher,Amanda 3, Jason,Jessica 4,David,Melissa 5,James, Sarah . . 780,Jessica,Juliana 781, Mikel, Charissa 782,Osvaldo,Fatima 783,Edwardo,Shara 784, Raymundo, Corrie . . .	2000 1,Jacob, Emily 2, Michael, Hannah 3, Matthew,Madison 4, Joshua, Ashley 5,Christopher,Sarah . . . 240, Marcos,Gianna 241,Cooper, Juliana 242, Elias,Fatima 243,Brenden,Allyson 244,Israel, Gracie . . .

圖 2. 西元 1980 年 (左) 與西元 2000 年 (右) 新生兒名字資料示意圖

由圖2.可以看到一個滿有趣的現象：1980 年時，「Jennifer」是女性名字裡最有名的，卻在 2000 年時掉到前五名之後；而在 1980 年代鮮少出現的「Juliana」(#780)，到了 2000 年竟大躍進了 300 多名！(#241)

如何將複雜的資料視覺化，並做出讓使用者能用簡單的方式查詢「新生兒名字的名氣，隨時間不同的趨勢」，是這份作業要請同學完成的工作。這份複雜的工作會被拆解成了六個 milestones，下面將一一解說。



## Milestone 1 - milestone1.py

第一部分要請同學打開「`milestone1.py`」(不是 `babyname.py`)，編輯並完成裡面的 `add_data_for_name(name_data, year, rank, name)`。

- Parameters 介紹

1. `year(str)`: 代表某一個年代
2. `rank(str)`: 代表該新生兒名字在當年的排名
3. `name(str)`: 新生兒名字
4. `name_data(dict)`:  
這是一個「value 裝著 dictionary 的 dictionary」。

key: `name(str)`, 新生兒的名字

value: (dict),

key: `year(str)`, 年份

value: `rank(str)`, 該年份的排名

- 此 function 沒有 return 任何東西

以下用三個例子說明這個 function 應該要有的功能：

### 1. 加入「未存過」的名字？

Before -

假設 name\_data 原本為：

```
{ "Kylie": {"2010": "57"}, "Nick": {"2010": "37"} }
```

After -

呼叫 add\_data\_for\_name(name\_data, "2010", "208", "Kate") 後，於 name\_data 新增一個名字 "Kate"(key)，及它的 {year: rank}(value)：

```
{ "Kylie": {"2010": "57"}, "Nick": {"2010": "37"}, "Kate": {"2010": "208"} }
```

### 2. 加入「已存在」的名字？

Before -

假設 name\_data 原本為：

```
{ "Kylie": {"2010": "57"}, "Nick": {"2010": "37"} }
```

After -

呼叫 add\_data\_for\_name(name\_data, "2000", "104", "Kylie") 後，由於 name\_data 中已經有 "Kylie" 這個名字，因此將新的 year(key) 及 rank(value) 加入 "Kylie" 底下：

```
{ "Kylie": {"2010": "57", "2000": "104"}, "Nick": {"2010": "37"} }
```

### 3. 男生 & 女生名字一樣？

有些中性名字，如「Sammy」，可能在某一年同時出現在男生 & 女生的排行榜。因此，請同學在加入一個新名字進入 name\_data 前，先檢查 name & year 是否已存在於 name\_data。若已經存在，請儲存該名字在同一年中，「比較高的排名」。舉例來說，若 "Sammy" 在 "1990" 有排名 "90" 以及 "200"，請留下比較高的排名："90"。

### Before

假設 name\_data 原本為：

```
{ "Kylie": {"2010": "57"}, "Sammy": {"1980": "451", "1990": "200"} }
```

### After

呼叫 add\_data\_for\_name(name\_data, "1990", "90", "Sammy") 後，由於 "90" 的排名比較高，因此保留 "90"：

```
{ "Kylie": {"2010": "57"}, "Sammy": {"1980": "451", "1990": "90"} }
```

## 完成之後，來測試 Milestone 1 吧！

### • 測試一

這個測試會看您的程式是否可以加入「未存過」的新名字。

macOS 請在 terminal 輸入 `python3 milestone1.py test1`  
Windows 請在 terminal 輸入 `py milestone1.py test1`

若您程式正確，會看到下列文字：

```
-----test1-----  
{'Kylie': {'2010': '57'}, 'Nick': {'2010': '37'}, 'Kate': {'2010': '208'}}  
-----
```

### • 測試二

這個測試會看您的程式是否可以處理「已存在」名字的情況。

macOS 請在 terminal 輸入 `python3 milestone1.py test2`  
Windows 請在 terminal 輸入 `py milestone1.py test2`

若程式正確，會看到下列文字：

```
-----test2-----  
{'Kylie': {'2010': '57', '2000': '104'}, 'Nick': {'2010': '37'}}  
-----
```

- 測試三 & 測試四

這兩個測試會看您的程式是否可以處理「男生 Sammy 與女生 Sammy」的情況。

macOS 請在 terminal 輸入 `python3 milestone1.py test3`  
Windows 請在 terminal 輸入 `py milestone1.py test3`

若程式正確，會看到下列文字：

```
-----test3-----  
{'Kylie': {'2010': '57'}, 'Sammy': {'1980': '451', '1990': '200'}, 'Kate': {'2000': '20'}}  
-----
```

macOS 請在 terminal 輸入 `python3 milestone1.py test4`  
Windows 請在 terminal 輸入 `py milestone1.py test4`

若您程式正確，會看到下列文字：

```
-----test4-----  
{'Kylie': {'2010': '57', '2000': '104'}, 'Nick': {'2010': '37'},  
'Kate': {'2010': '208', '2000': '108'}, 'Sammy': {'1990': '90'}}  
-----
```

## Milestone 2 - babyname.py

通過 milestone 1 的測試後，同學可以把 `add_data_for_name` 的程式碼從 `milestone1.py` 複製到 `babyname.py` 的 `add_data_for_name` 裡面。

再來要請同學編輯名為 `add_file(name_data, filename)` 的 function。 `add_file` 會取出檔名為「`filename`」的文字檔中的所有資料，並加到 `name_data` 內。請同學利用在 milestone 1 已經完成的 `add_data_for_name` 來編輯 `add_file`。

- Parameters 介紹

1. `name_data(dict)`：key 是新生兒的名字；value 是另一個 dictionary (裝著對應到的名字在某一年的排名)
2. `filename(str)`：存著某一年新生兒排名資料的文字檔檔名

- 此 function 沒有 return 任何東西

所有文字檔內的資料儲存格式大致都與第二頁出現過的示意圖相同：

baby-1980.txt	baby-2000.txt
<pre>1980 1,Michael, Jennifer 2,Christopher,Amanda 3, Jason,Jessica 4,David,Melissa 5,James, Sarah . . . 780,Jessica,Juliana 781, Mikel, Charissa 782,Osvaldo,Fatima 783,Edwardo,Shara 784, Raymundo, Corrie . . .</pre>	<pre>2000 1,Jacob, Emily 2, Michael, Hannah 3, Matthew,Madison 4, Joshua, Ashley 5,Christopher,Sarah . . . 240, Marcos,Gianna 241,Cooper, Juliana 242, Elias,Fatima 243,Brenden,Allyson 244,Israel, Gracie . . .</pre>

檔案的第一行為 year，第二行開始依序是排名 (rank) 第 1~1000 名的男生姓名 (name1) 與女生姓名 (name2)。

請注意：

- (1) 每一行的 rank、name1 及 name2 間都一定有逗號
- (2) 每一行都可能包含不固定的「空白(space)」及「換行符號」。我們在將資料存進來 name\_data 之前，要先將它們去除，讓資料以比較「乾淨」，也更容易利用的形式儲存。「strip()」為 Python String 的一個內建 method，可以去除一個文字 前後 的所有空白及換行符號。舉例來說，name1 = ' Jerry \n'；經過 name1 = name1.strip() 後，name1 就會變成 'Jerry'。

請同學在將 rank、name1 及 name2 輸入 add\_data\_for\_name 之前，先進行以下處理：

- rank = rank.strip()
- name1 = name1.strip()
- name2 = name2.strip()

- (3) Milestone 2 將與 Milestone 3 一起做測試。

## Milestone 3 - babynames.py

請繼續往下編輯並完成下方兩個 function。

### read\_files(filenamees)

- Parameters 介紹
  1. filenamees(list): 裝著所有「檔名」的 Python List
- return 一個 Python Dictionary (就是 name\_data)

「filenamees」是一個 Python List，裡面的每一個元素都是一個檔名 (filename)。請使用 Milestone 2 建造的 add\_file(name\_data, filename)，並將 filename 內的資料以正確格式加入 name\_data。「filenamees」內所有檔案的資料都被加入 name\_data 後，請「return name\_data」。

### search\_names(name\_data, target)

- Parameters 介紹
  1. name\_data(dict): 裝著新生兒資訊的 Python Dictionary
  2. target(str): 文字片段
- return 一個 Python List 裝著所有包含 target 的名字

「target」為某個名字的「一部分文字片段」。我們要搜尋在 name\_data 裡全部有包含 target 的名字，然後將它們存在一個名叫「names」的 Python List，並 return 出去。這邊要請同學們特別注意的是，search\_names 不應該受大小寫的影響，為 case-insensitive。舉例來說，若 target == "aa"，我們的 names 裡面應該要包含 "Aaron" 這個名字。

## 完成之後，來測試 Milestone 2 & 3 吧！

### • 測試一

macOS 請在 terminal 輸入：

```
python3 babynames.py data/small/small-2000.txt data/small/small-2010.txt
```

Windows 請在 terminal 輸入：

```
py babynames.py data/small/small-2000.txt data/small/small-2010.txt
```



這個指令將測試您 `add_file(...)` 與 `read_files(...)` 的程式是否通過基本要求。  
如果您的程式正確，應該會看到下方文字：

```
A [ ('2000', '1'), ('2010', '2') ]  
B [ ('2000', '1') ]  
C [ ('2000', '2'), ('2010', '1') ]  
D [ ('2010', '1') ]  
E [ ('2010', '2') ]
```

- 測試二

Mac 請在 terminal 輸入

```
python3 babynames.py -search aa data/full/baby-2000.txt data/full/baby-2010.txt
```

Windows 請在 terminal 輸入

```
py babynames.py -search aa data/full/baby-2000.txt data/full/baby-2010.txt
```

這個指令將測試您 `search_names` 的程式是否通過基本要求  
如果您的程式正確，應該會看到下方文字：

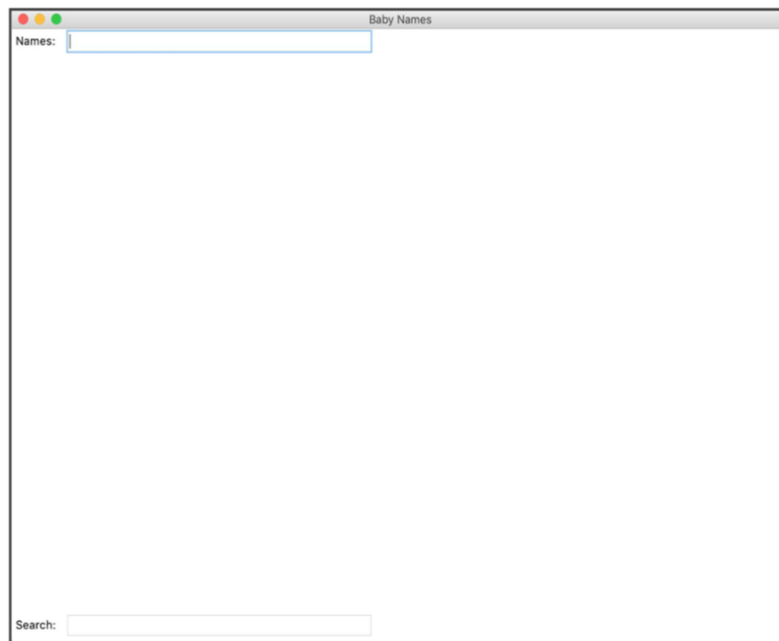
```
Aaron  
Isaac  
Aaliyah  
Isaak  
Aaden  
Aarav  
Ayaan  
Sanaa  
Ishaan  
Aarush
```

## Milestone 4 - babygraphics.py

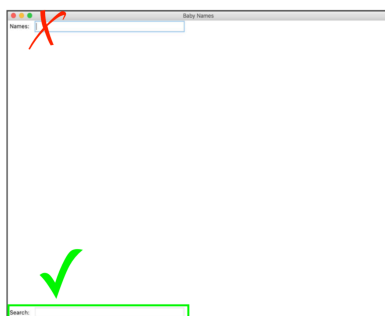
恭喜同學完成這次作業的一半囉！Milestone 1 ~ 3 就是後端工程師的工作內容。接下來的 Milestone 4 ~ 6，同學將轉換成一位前端工程師，將剛剛的資料視覺化呈現！首先，

macOS 請在 terminal 輸入 `python3 babygraphics.py`  
Windows 請在 terminal 輸入 `py babygraphics.py`

執行後應該會跳出一個視窗，這是我們先寫出來的視窗基底，接下來交給各位了！



請在視窗左下方「Search:」的欄位裡輸入「aa」，按下 Enter / return 鍵後，看看是否出現與下圖一樣的畫面（出現所有包含 target 的名字）：



請注意：

左上角有另外一個搜尋欄，小心不要填錯位置囉~

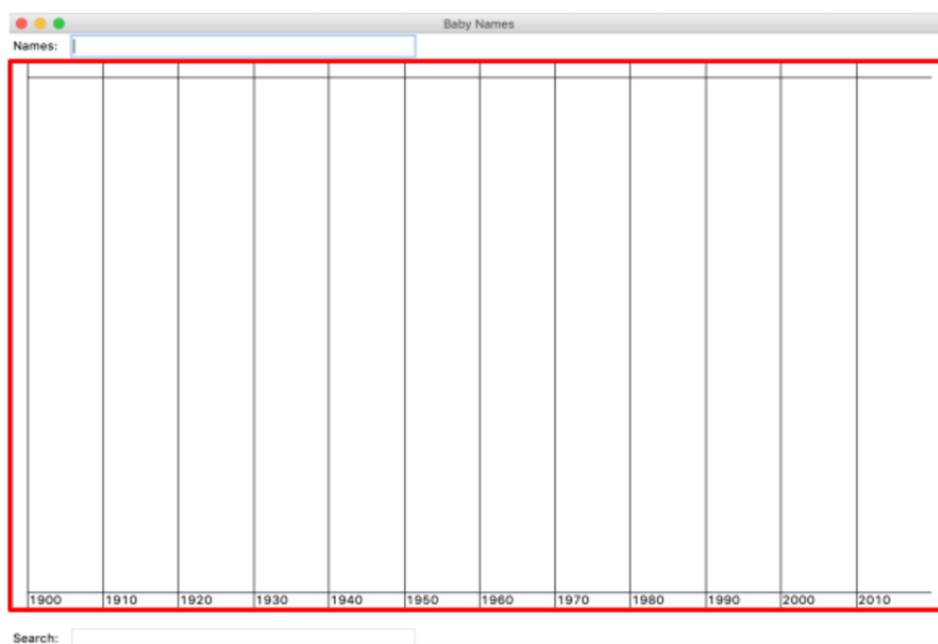
如果沒什麼問題，Milestone 4 就完成了！

## Milestone 5 - babygraphics.py

### draw\_fixed\_lines(canvas)

- Parameters 介紹
  1. canvas : tkinter.Canvas 視窗，可使用 canvas.create\_line( ... ) 以及 canvas.create\_text( ... ) 來加上不同畫面元素
- 不會 return 任何東西

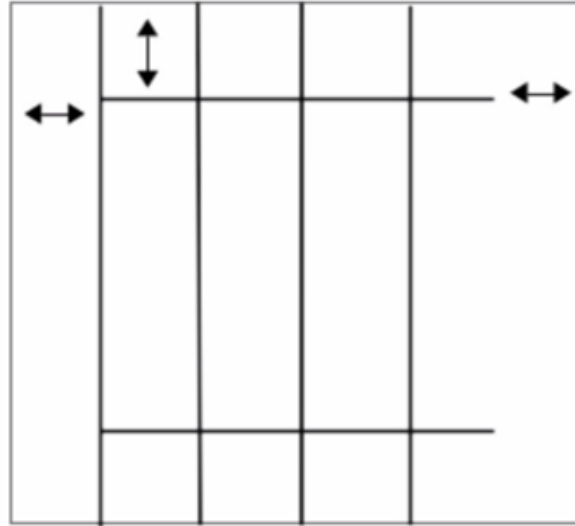
下圖用紅線匡起來的區間，我們稱之為 canvas。



請同學先在您空白的 canvas 上加兩條橫線：

1. 一條與底端距離 GRAPH\_MARGIN\_SIZE
2. 一條與頂端距離 GRAPH\_MARGIN\_SIZE

再來，請同學將這兩條線的左、右兩邊留白，同樣距離 GRAPH\_MARGIN\_SIZE (如下圖所示)。

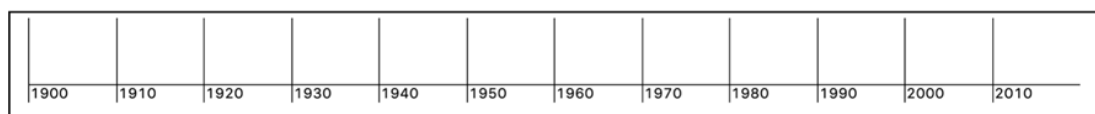


## Constants 介紹

- FILENAMES 為所有檔名資料 (您不會使用到此常數)
- CANVAS\_WIDTH 為視窗的寬
- CANVAS\_HEIGHT 為視窗的高
- YEARS 為一個 Python List，裡面的每一個元素為我們想要分析的年份  
(不一定只有 12 個年份)
- GRAPH\_MARGIN\_SIZE 為線與視窗的保留距離
- COLORS 為不同名字的折線使用的顏色，依照順序使用
- TEXT\_DX 為文字「左邊」與直線之間的距離
- LINE\_WIDTH 為每一條線的寬度
- MAX\_RANK 為每年的資料有保留到的最低的名次

接下來，我們要將 canvas

在  $x = \text{GRAPH\_MARGIN\_SIZE}$  到  $x = \text{width} - \text{GRAPH\_MARGIN\_SIZE}$  的範圍，等距畫出相同於 YEARS 裡的年份數量的垂直線。每一條垂直線由上到下的長度皆為 height，但每條線的 x 座標會隨著資料在 YEARS 中的位置不同而改變，請參考下方圖示。



由於 x 座標是最主要的變化因子，我們先將 x 座標的計算定義成另一個方便使用的 function：get\_x\_coordinate。

get\_x\_coordinate(width, year\_index)

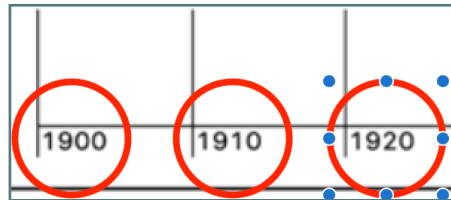
- Parameters
  - width(int): 就是 CANVAS\_WIDTH
  - year\_index(int): 在 YEARS 這個 list 裡面所屬的「index」
- return 一個整數(int)，等同於 year\_index 在 canvas 的 x 座標

舉例來說，如果

```
YEARS = ['1900', '1910', '1920', '1930', '1940', '1950', '1960', '1970', '1980',  
'1990', '2000', '2010']
```

get\_x\_coordinate(1000, 2) 應該 return 「180」，代表 index 為 2 的 1920 年線的 x 座標。

請使用 get\_x\_coordinate 在 canvas 加上許多垂直線。每一條直線與 canvas 底部橫線都有一個交叉點，請在此交叉點的右邊 TEXT\_DX 距離的地方，加上此直線所代表的「年份」（如下圖之 1900、1910、1920 …）。



💡 小提示：

canvas.create\_text(...) 中，加上 anchor=tkinter.NW 會使文字將西北方當成基準點

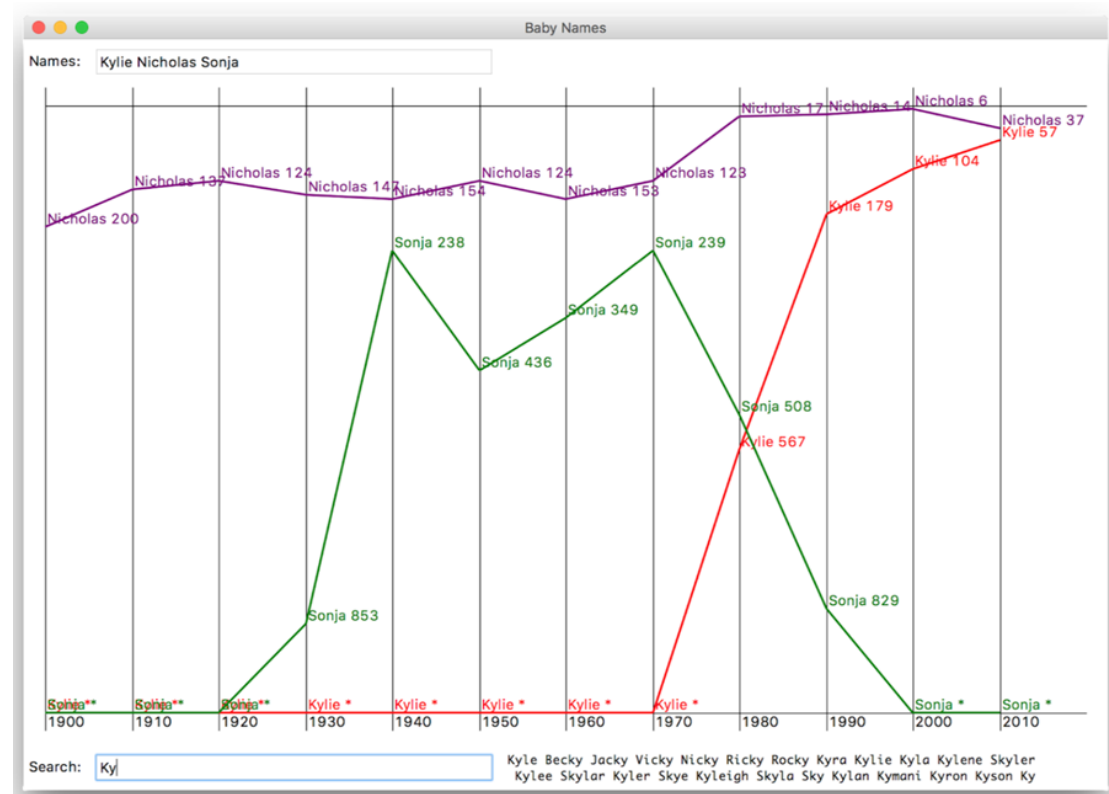


## Milestone 6 - babygraphics.py

`draw_names(canvas, name_data, lookup_names)`

- Parameters 介紹
  - `canvas` : `tkinter.Canvas` 視窗，可使用 `canvas.create_line( ... )` 以及 `canvas.create_text( ... )` 來加上不同畫面元素
  - `name_data(dict)` : 裝著所有新生兒資訊的 Python Dictionary
  - `lookup_names(list)` : 裝著所有使用者要查看的名字的 Python List
- 不會 return 任何東西

本次作業的最後一部分要請同學將 `lookup_names` 裡面的所有名字的資料畫在 `canvas` 上，像下圖這樣：



若使用者在 `canvas` 左上方「Names:」的欄位中輸入「Kylie Nicholas Sonja」，則 `lookup_names = ["Kylie", "Nicholas", "Sonja"]`（這個功能我們已經寫好了）。

接著，當使用者按下 `ENTER` / `return`，`draw_names( ... )` 就會被呼叫（連接鍵盤 `ENTER` / `return` 的功能我們也已經寫好了）

每一個資料點都會有「名字 排名」顯示在旁邊，如「Nicholas 200」。  
這行文字與資料點的水平間距應為 TEXT\_DX，並且文字應顯示在資料點的右上方（anchor=...?）。

如上圖所示，在 1970 前，Kylie 這個名字的排名在 1000 名之外，原本名字旁邊顯示排名的地方就會換成 " \* "。

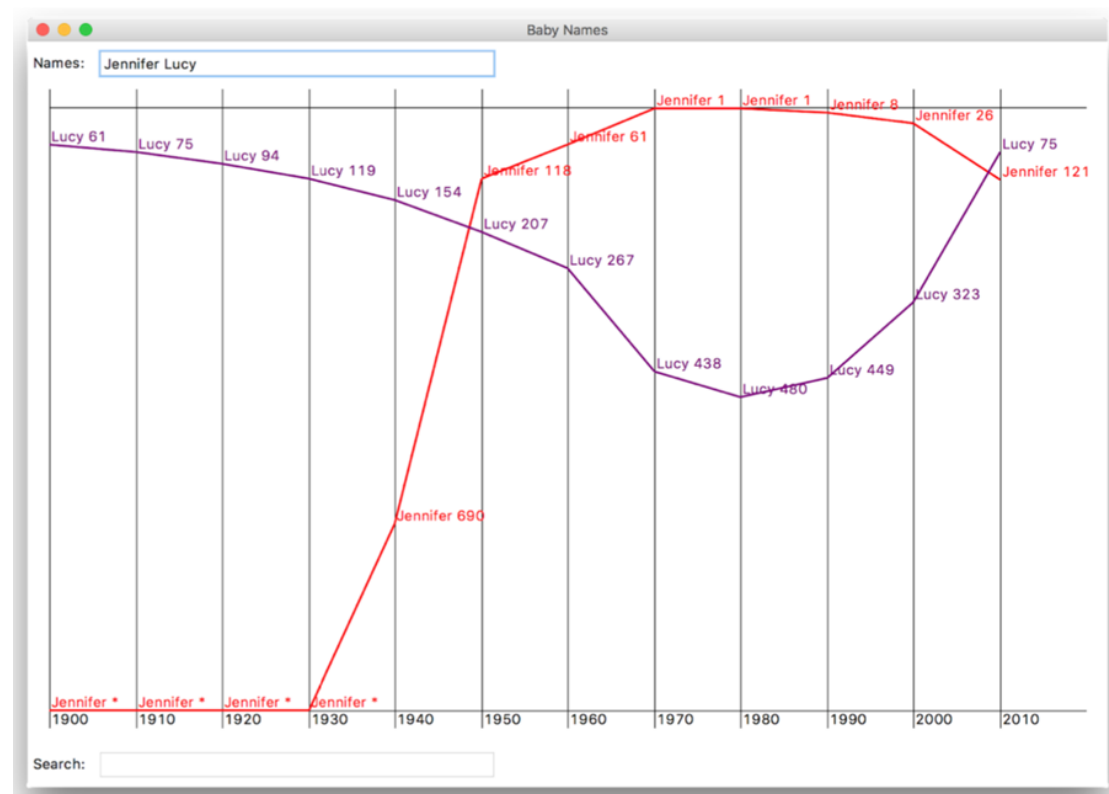
若某名字排名在 1000 名之外（name\_data 該年無資料），請將排名以 " \* " 代替。

曲線的顏色順序已經定義在名為 COLORS 的 Python List 裡。若繪製的數目大於 COLORS 的長度，請讓程式可以自動回到 COLORS[0] 從頭開始循環利用。若要改變線條的顏色（以藍色為例），只要在 canvas.create\_line (...) 的括弧裡加上 fill = "blue" 即可。

曲線的寬度請定義為 LINE\_WIDTH。要改變線條粗細為 LINE\_WIDTH，請在 canvas.create\_line (...) 括弧裡面加上 width = LINE\_WIDTH 即可。

## 最後的測試！

請在視窗上方輸入「Jennifer」和「Lucy」這兩個名字，若與下圖結果一致，恭喜！您已經是一位將前、後端一手包辦的全端工程師了（強！）

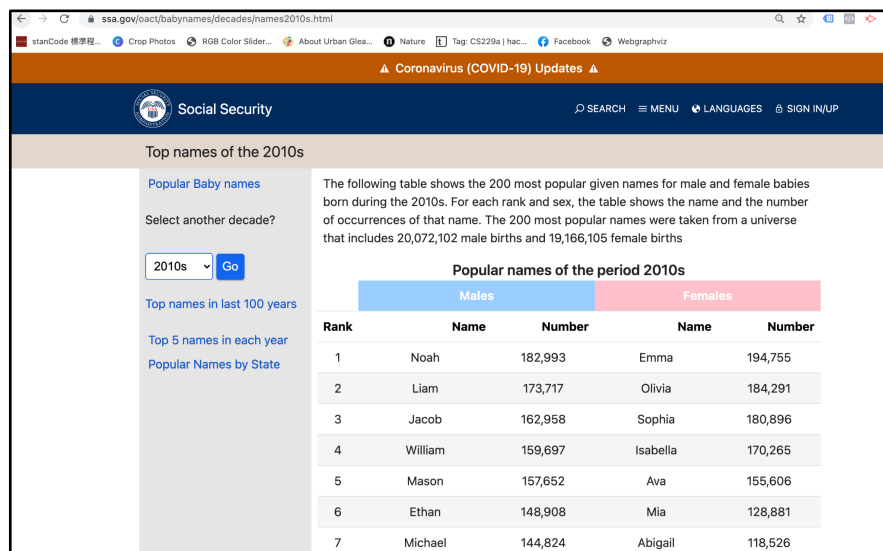


## Extension - extension.py

本題為選擇性作答，如果您想更了解爬蟲實作，並且行有餘力，歡迎繼續往下練習

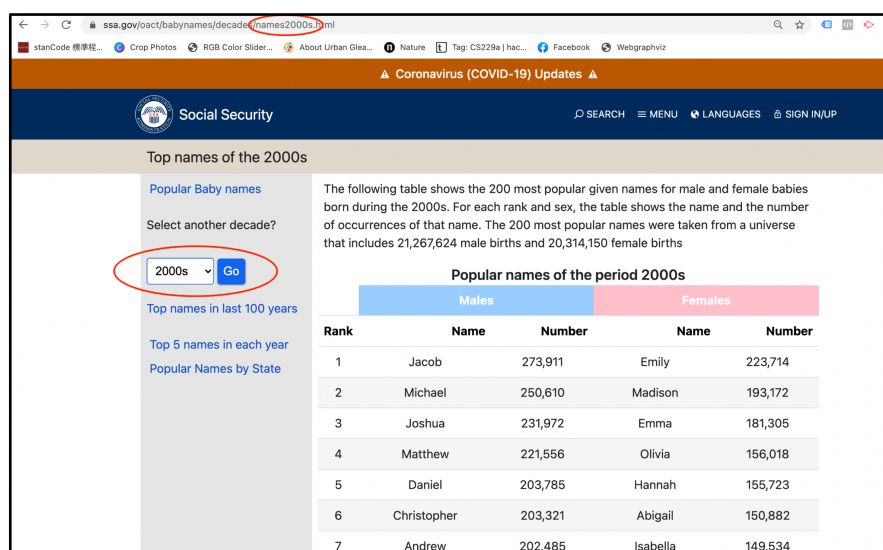
我們在上課學會了如何使用 requests, bs4 的套件去抓取 imdb 電影數據。因此，這個部分要同學們自己嘗試從 US Social Security 網站抓取 babynames 資料！

下圖是 US Social Security 網站在 2010s 的資料（該網站目前僅提供前 200 名的男生名字與女生名字）



Popular names of the period 2010s				
Males			Females	
Rank	Name	Number	Name	Number
1	Noah	182,993	Emma	194,755
2	Liam	173,717	Olivia	184,291
3	Jacob	162,958	Sophia	180,896
4	William	159,697	Isabella	170,265
5	Mason	157,652	Ava	155,606
6	Ethan	148,908	Mia	128,881
7	Michael	144,824	Abigail	118,526

若同學點選左邊下拉欄位 2000s 並按下「Go」，您會發現 2000s 資料所在之網址跟 2010s 資料的網址幾乎一模一樣！只是結尾從 2010s.html 變成了 2000s.html(如下圖紅色圈圈所示)



Popular names of the period 2000s				
Males			Females	
Rank	Name	Number	Name	Number
1	Jacob	273,911	Emily	223,714
2	Michael	250,610	Madison	193,172
3	Joshua	231,972	Emma	181,305
4	Matthew	221,556	Olivia	156,018
5	Daniel	203,785	Hannah	155,723
6	Christopher	203,321	Abigail	150,882
7	Andrew	202,485	Isabella	149,534



這個部分要同學嘗試將 2010s, 2000s, 與 1990s 的男、女新生兒前 200 名的人數總和計算出來（也就是表格裡 Number 欄位的數字總和）。

若您查看該網站的 HTML 會發現我們要的資料在 `<table class="t-stripe">` 裡面的 `<tbody>`：

The screenshot shows a web browser displaying a table of baby names and their counts for the 2000s. The table is titled "Names of the period 2000s" and is divided into two sections: "Males" and "Females". The "Males" section lists the top 17 names, and the "Females" section lists the top 17 names. The table has columns for Rank, Name, and Number. The Chrome DevTools Elements panel is open, showing the HTML structure of the table. The selected element is `<table class="t-stripe">`, and the DOM tree shows the `<tbody>` element. The `<tbody>` element contains a `<tr>` element, which contains the first row of data. The `<tr>` element has five `<td>` elements, which contain the rank, name, and number for the first male and female names.

提示：

`<tbody>` 底下有許許多多的 `<tr>`，而 `<tr>` 裡面又有許許多多的 `<td>`  
若要得到所有 `<td>` 與 `</td>` 之間的文字，可以直接對 soup 裡面的 `<tbody>` 物件取出文字

請編輯 `extension.py` 檔案，並完成上述工作！若您所撰寫的程式正確，應該會在執行 `extension.py` 時看到 console 上的輸出與下圖一致 (執行時間可能約需 1 分鐘)：

```
-----
2010s
Male Number: 10890537
Female Number: 7939153
-----
2000s
Male Number: 12975692
Female Number: 9207577
-----
1990s
Male Number: 14145431
Female Number: 10644002

Process finished with exit code 0
```

## 評分標準

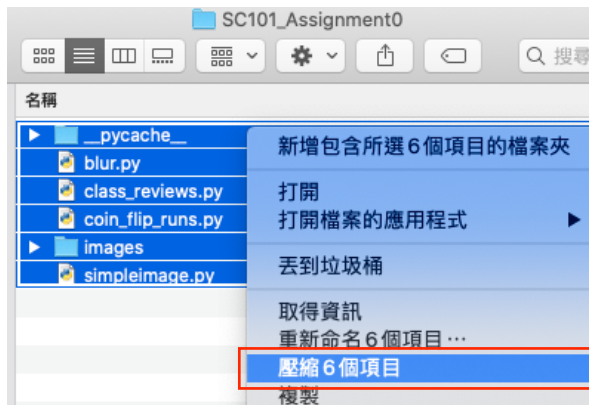
**Functionality** - 程式是否有通過我們的基本要求？程式必須沒有 bug 、能順利完成指定的任務、並確保程式沒有卡在任何的無限環圈（Infinite loop）之中。

**Style** - 好的程式要有好的使用說明，也要讓人一目瞭然，這樣全世界的人才能使用各位的 code 去建造更多更巨大更有趣的程式。因此請大家寫精簡扼要的使用說明、function敘述、單行註解。

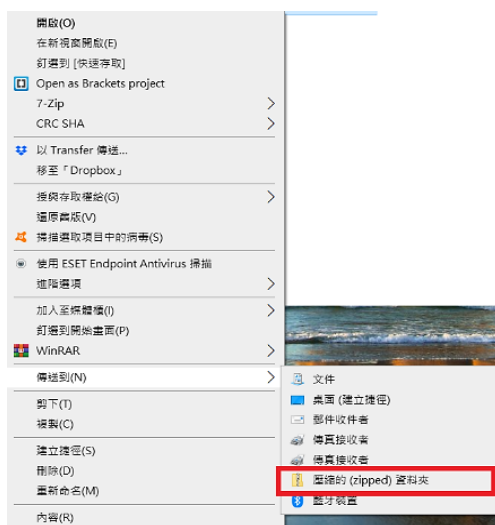
## 作業繳交

1. 以滑鼠「全選」作業資料夾內的所有檔案，並壓縮檔案。請見下圖說明。

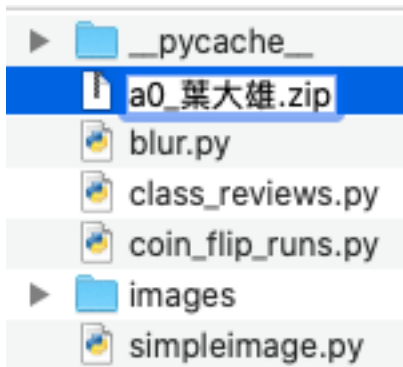
macOS：按右鍵選擇「壓縮n個項目」



Windows：按右鍵選擇「傳送到」→「壓縮的(zipped)資料夾」



2. 將壓縮檔(.zip)重新命名為「a(n)\_中文姓名」。如：  
assignment 0命名為a0\_中文姓名；  
assignment 1命名為a1\_中文姓名；…



3. 將命名好的壓縮檔(.zip)上傳至Google Drive（或任何雲端空間）
- 1) 搜尋「google drive」
  - 2) 登入後，點選左上角「新增」→「檔案上傳」→選擇作業壓縮檔(.zip)
4. 開啟連結共用設定，並複製下載連結
- 1) 對檔案按右鍵，點選「共用」
  - 2) 點擊「變更任何知道這個連結的使用者權限」後，權限會變為「可檢視」
  - 3) 點選「複製連結」



5. 將連結上傳至臉書社團的作業貼文提供的「作業提交表單」