信 戀 千 里

旅遊明信片產生器

壹、團隊介紹

隊名：Think Spark

蘇浚緯

台南人。一個喜歡從生活中找問題解決，並獲得成就感的人。目前就讀電機工程學系，高中時曾參加科研社、國樂社。個人平時喜歡動手實作、機械組裝、是個樂於嘗試新事物，有創意跟想法的人。除此之外，我熱愛音樂，平常透過音樂充實自我。

高韡哲

台南人。一個對事情充滿好奇的人，對於有不懂的事情都會像盡辦法去了解，目前就讀電機工程學系，第一次接觸程式，雖然沒有經驗，但我勇於嘗試，想利用所學的知識解決生活上的問題，平時樂於幫助他人，只要看見有人有困難都會盡自己能力去幫助。

吳信賢

屏東人，目前就讀電機工程系，樂於助人，高中時參加雨豆傳愛社，平常在為弱勢團體募捐發票，幫助老人、小孩等社福機構掃地。到了大學才第一次接觸程式，雖然沒有經驗，但是會盡量去嘗試。平時喜歡接觸日文方面的事物，當今流行的日文歌，及部分日本名人的動態。

貳、創作理念

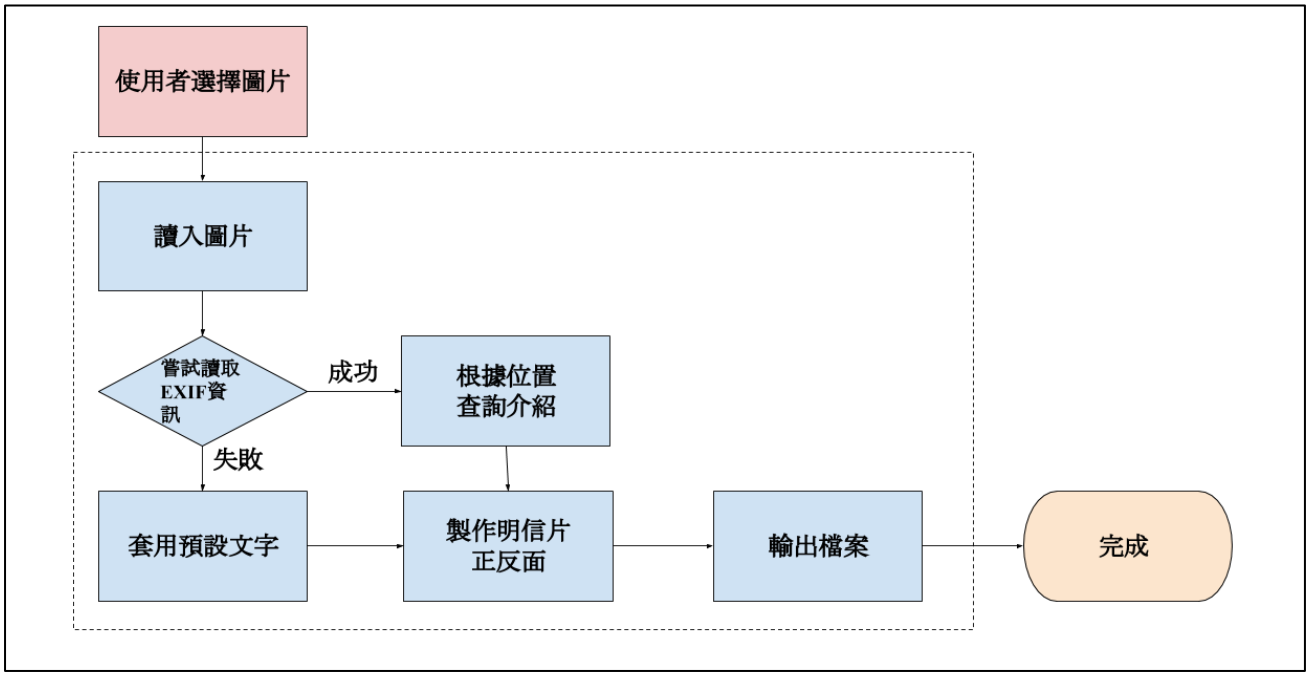
一、創作動機

信戀千里，我們做的是一個透過目光所及之景色製作為明信片，加強人與人之間情感維繫的程式。

現今的人們看見漂亮的事物，第一個反應總是怎麼將它保存下來，到各地旅遊之時，總是拿個相機使勁地拍，拍完還總會買些明信片紀念，因此我們做出了可以將自己拍的照片轉為明信片的程式，同一幅風景在每個人的眼中都不同，透過這個程式，可以將每個人眼中的樣貌呈現出來，變成實體的明信片，使人人都能輕鬆地擁有專屬於自己的一片風景。

二、方案建立

我們根據我們的構想繪製出以下流程圖：



參、成果說明

一、應用性

人們到了各個國家去旅遊，一定會拍下許多當地的名勝古蹟、自然美景，倘若還想留下更多的紀念，抑或是想要分享這個喜悅，不少人會購買明信片寄回家鄉或寄給好友，但明信片款式再多都不是全部，這時『明信片產生器』就有了他實用之處，直接將所拍的照片轉變為明信片，應用甚廣。

二、挑戰性

這次作品我們花費相當多時間討論，包括主題與內容與呈現方式，最後決定要做這次的明信片產生器，但在過程中出現了一些問題。首先雖然我們順利地透過 GPS 座標取得地點名稱，但在使用維基百科的 API 抓取資料當作明信片內容時，發現有些國家會無法正常抓取，後來使用到 wkikpedia 函式庫後，問題終於解決。另外，因為爬下來的資料容易超出明信片可容許的最長文字，我們也花相當多的時間找到方法去取篩選需要的文字、把多餘的字刪除。

三、創意性

目前在網路上，幾乎沒有看過類似功能的 app。在製作出明信片之時，圖片背面除了國家之外，還會有國家的簡短介紹，其面相包括了自然、人文、經濟等要素，製作明信片之後，將選擇我們認為此國家較為突出的部分，可以使不了解這個國家的人大致上知道它的特色與風情。

四、完成性

在完成性的部分，我們不僅已將我們的主要功能完全實現，並且，我們也使用 Python 中的 tkinter 打造了一個操作上直覺且簡單的用戶介面(如下圖所示)。

五、實際執行結果

●匯入圖片後自動產生的明信片

~~台灣不多的原始林，沿途都可以看到，帝雉、山羌、猴子，松鼠 值得一走放鬆的地方~~

2019.12.21

@Jack Wang @Peter Cheng

#大雪山風景 #海拔 2320 m

臺灣是位於東亞、太平洋西北側的島嶼，地處琉球群島菲律賓群島之間，西隔臺灣海峽與中國大陸相望，為東亞島中一島。

萬人可乘坐的遼闊草原，

沿途遼闊蔚藍海景令人心曠神怡。

@Marisol @Ayden

#沖繩萬座毛

日本國，通稱日本，是位於東亞的島嶼國家，由日本列
、琉球群島和伊豆－小笠原群島等6,852座島嶼組成，
積約37.8萬平方公里。

肆、程式說明

一、函式庫匯入及資料宣告

我們導入了一些函式庫如下所示，主要用來處理圖片及爬蟲、連接 API、用戶介面。變數部分，*api_key* 則是在連接 OpenWeatherMap API 時需要的金鑰，下方程式碼的金鑰部分有做模糊化。*file_name* 是我們的來源照片的檔案路徑，會再使用者選擇相片後更改。*now* 是現在時間，會在之後使用者按下產生明信片後更改。

```
1. import cv2, exifread                                  #圖片處理
2. import json, requests                                 #爬蟲與資料處理
3. from PIL import ImageFont, ImageDraw, Image, ImageTk #圖片處理
4. import numpy as np                                    #圖片處理
5. import wikipedia                                      #維基百科查詢
6. from langconv import *                                #簡字轉繁字
7. import string                                         #字串處理
8. import datetime                                       #時間
9. import tkinter as tk                                  #用戶介面(下同)
10. import tkinter.filedialog ,tkinter.messagebox
11. import os                                            #系統
12.
13. api_key = 'b6d7**********************4036'#OpenWeather API key
14. file_name = ''                                       #來源影像檔名
15. now = ''                                             #現在時間(用來記錄用戶按下按鈕的時間
```

二、定義函數

1.維基百科查詢並整理

我們透過 wikipedia 函式庫來讀取維基百科上的資料，並將回傳結果強制轉換為繁體字，另外，因為在明信片中能置入的文字數量有所限制，因此，如果介紹文字過長會進行縮減。詳細程式說明如下註解：

```
1.  def getWiki(term): #查詢維基百科介紹
2.      wikipedia.set_lang('zh')                        #設定為中文
3.      text = wikipedia.summary(term, sentences=1)     #抓取一句介紹
4.      text = Converter('zh-hant').convert(text)       #將簡字轉為繁字
5.      for char in ['(', ')', '[', ']', '（', '）']:   #將括號內容清除
6.          text = text.replace(char, '|')
7.      article_list = text.split('|')
8.      n = 0
9.      text = ''
10.     for item in article_list:
11.         if((n % 2) == 0):
12.             text += item
13.         n +=1
14.     while(len(text)>81):                            #若段落過長，移除最長的句子
15.         sentence_list = text.split('，')
16.         len_sentence = []
17.         for i in range(len(sentence_list)):
18.             len_sentence.append(len(sentence_list[i]))
19.         len_max = max(len_sentence)
20.         for i in range(len(sentence_list)):
21.             if(len(sentence_list[i]) == len_max):
22.                 sentence_list.remove(sentence_list[i])
23.                 break
24.         text = '，'.join(sentence_list)
25.
26.     return text
```

2.GPS 座標取地名與介紹

針對我們的需求，我們需要透過 GPS 座標來知道我們的地名，因此我們使用了 OpenWeatherMap 的 API 來實現(未來將進一步使用 Google Map API)。但因為此 API 無法獲得非常精準的地名，我們決定只定位國家，避免產生不必要的錯誤。

函數的輸入 *Coordinate*，格式是根據 API 的規範，我們透過經緯度去做請求，將回傳的 JSON 格式資料做進一步分析。查詢到國家代碼後，我們透過開檔查詢的方式找出所在地國家的中文名稱，並使用前述取得維基百科資訊的函數來取的地名介紹，最後回傳結果。詳細程式說明如下註解：

```
1.  def findLocationName(Coordinate): #依 GPS 座標取得國家名稱及介紹(使用天氣 API)
2.      api_url = 'http://api.openweathermap.org/data/2.5/weather?' #OpenWeatherMap API
3.      api_url += Coordinate                                       #Using GPS to get
4.      api_url = api_url + '&APPID=' + api_key + '&lang=ZH_TW'
5.      weather_data = requests.get(api_url).json()                 #convert result to JSON
6.      Location_code = weather_data['sys']['country']              #get country(國家代碼)
7.      #Location = weather_data['name'] + ', ' + weather_data['sys']['country']
8.      #抓 name 會有大誤差，故暫時註解 (name = 鄉鎮市區)
9.      json_file = open('CountryCode_ZH_TW.json', 'r+', encoding='utf-8')
10.     json_array = json.load(json_file)                           #開檔並讀取國家名稱及介紹
11.     for item in json_array:
12.         if (item['countryCode'] == Location_code):
13.             Location = item['countryName']
14.     Introduction = getWiki(Location)                            #呼叫 getWiki 取得國家介紹
15.     return (Location, Introduction)
```

3.經緯度座標轉換(60 進位 to 10 進位)

因為我們從圖片 EXIF 資訊中所獲得的 GPS 資料是不易讓程式計算的 60 進位，因此我們透過這個函數來完成座標轉換。這邊的輸入 *coordinate* 會是經度或緯度，並且呼叫一次函數只能轉換其中之一的資料。詳細程式說明如下註解(部分文字因排版有壓縮字寬)：

```
def coordinateConvert(coordinate):  #GPS 座標換算(60 進位 to 10 進位)
    #input format: '[nn, mm, aa/bb]' (60 進位,百分位為分數)
    coordinate = coordinate.replace(' ', '')    #移除空白
    coordinate = coordinate.replace('[', '')    #移除左中括號
    coord_2_a = (coordinate[coordinate.rfind(',') + 1:coordinate.rfind('/')])    #取得百分位數值分母
    coord_2_b = (coordinate[coordinate.rfind('/') + 1:coordinate.rfind(']')])    #取得百分位數值分子
    coordinate = coordinate.replace('/', '').replace(']', '')        #移除左斜和右括號
    coordinate = coordinate.split(',')          #此時 fomat 'nn,nn,aa/bb'，分割為 list
    coordinate[2] = int(coord_2_a) / int(coord_2_b)      #計算百分位之分數
    coordinate = int(coordinate[0]) + int(coordinate[1])/60 + coordinate[2]/600 #計算座標十進位值
    coordinate = round(coordinate, 6)                    #小數取 6 位
    return coordinate
```

4.分析照片 EXIF 資訊

在這個函數，我們透過讀取照片的 EXIF 資訊來取得照片的時間及 GPS 資訊。詳細程式說明如下註解：

```
def analyzePicture(file_name):    #分析照片資訊，取得時間及 GPS
    f = open(file_name, 'rb')
    tags = exifread.process_file(f)
    #print all tags
    '''for tag in tags.keys(): #輸出所有 Tags
        if tag not in ('JPEGThumbnail', 'TIFFThumbnail', 'Filename', 'EXIF MakerNote'):
            print ("Key: %s, value %s" % (tag, tags[tag]))'''
    try:            #嘗試讀取 GPS 資訊
        Latitude = str(tags['GPS GPSLatitude'])
        Latitude = coordinateConvert(Latitude)
        Longitude = str(tags['GPS GPSLongitude'])
        Longitude = coordinateConvert(Longitude)
        Coordinate = 'lat=' + str(Latitude) + '&lon=' + str(Longitude) #API_url 之格式
    except:
        Coordinate = 'Null'
        print('GPS 讀取失敗')
    try:            #嘗試讀取時間(GPS 資料優先)
        try:
            Time = str(tags['GPS GPSDate'])
            Time = Time.replace(':', '.')
```

```
        except:
            Time = str(tags['Image DateTime'])
            Time = Time.replace(':', '.')
            Time = Time.split(' ')[0]
    except:
        Time = 'Fascinating.'
        print('時間讀取失敗')
    return (Time, Coordinate)
```

5.明信片製作(圖片面)

在這個函數，我們讀入影像檔案後，首先取的照片的長寬，並根據傳入的地點與時間繪製字卡。最後，回傳圖片的長寬以方便製作明信片背面。詳細程式說明如下註解(部分文字因排版有壓縮字寬)：

```
def mainPictureAddText(Location, Time):      #繪製明信面背面(風景
    bk_img = cv2.imread(file_name)
    (x,y,z) = bk_img.shape           #取得相片的長與寬(x,y)，z 在這裡用不到
    text_x = int(0.05*y)             #設定文字 x 座標
    text_y = int(0.9*x)              #設定文字 y 座標
    text = Location + ' ' + Time     #設定文字(國家名稱 + 時間)
    backClor =(102, 102, 102)        #設定字卡底色
    font = ImageFont.truetype("bb.ttc", y//30) #設定字體與字型大小
    (width, heigh), (offset_x, offset_y) = font.font.getsize(text)#取得文字方框大小
    #繪製字卡
    r = int(heigh / 2)
    bk_img = cv2.rectangle(bk_img, (text_x, text_y), (text_x + width, text_y + heigh), backClor, -1) #繪製方框
    bk_img = cv2.circle(bk_img, (text_x, text_y + r), r, backClor, -1)           #繪製左圓
    bk_img = cv2.circle(bk_img, (text_x + width, text_y + r), r, backClor, -1)  #繪製右圓
    #繪製文字
    img_pil = Image.fromarray(bk_img)
    ImageDraw.Draw(img_pil).text((text_x , text_y), text , font = font, fill = (255, 255, 255) )   #繪製文字
    bk_img = np.array(img_pil)
    #寫入並回傳長寬
    cv2.imwrite('postcard_' + now + '_front.jpg',bk_img)
    return(max(x,y), min(x,y)) #回傳長與寬(這裡設定寬>=長
```

6.明信片製作(郵務面)

函數中有三個輸入分別是寬、長和地點介紹。首先開一個新的畫布，並依序填入底色、繪製中間隔線、郵票方框、地址欄及地點介紹的文字。詳細程式說明如下註解(部分文字因排版有壓縮字寬)：

```
1.  def makePostcard(x, y, introduction):        #繪製明信片正面(郵務
2.      postcard = np.zeros((y, x, 3), dtype="uint8") #開新畫布
3.      cv2.rectangle(postcard, (0, 0), (x, y), (255, 255, 255), -1) #填滿背景色(白色)
4.      cv2.rectangle(postcard, (int(x*0.6), int(y*0.05)), (int(x*0.6), int(y*0.95)), (68, 68, 68), y//100) #繪製中間隔線
5.      for n in [0.7, 0.8, 0.9]:                #繪製地址欄格線 3 條
6.          cv2.rectangle(postcard, (int(x*0.65), int(y*n)), (int(x*0.95), int(y*n)), (68, 68, 68), 1+y//300)
7.      cv2.rectangle(postcard, (int(x*0.95), int(y*0.05)), (int(x*0.95 - y*0.2), int(y*0.35)), (68, 68, 68), 1+y//300) #繪製郵票框
8.      cv2.imwrite('postcard_' + now + '_back.jpg', postcard)     #先寫檔
9.      #繪製國家介紹文字
10.     img = cv2.imread('postcard_' + now + '_back.jpg')
11.     font = ImageFont.truetype("bb.ttc", y//30) #設定需要顯示的字體與大小
12.     #取得 27 個字文字框大小並檢查是否會超出版面(這裡設定 27 個字的版面最漂亮)
13.     (width, heigh), (offset_x, offset_y) = font.font.getsize('這裡會有二七個字。這裡會有二七個字。這裡會有二七個字。')#取得 27 個字文字框大小
14.     if (width > x*0.6):      #檢查是否超出格式範圍(版面是否會異常)
15.         font = ImageFont.truetype("bb.ttc", y//40) #重新設定字體與大小
16.     #寫入介紹資訊(文字多時須分多行)
17.     img_pil = Image.fromarray(img)
18.     introduction = '    ' + introduction     #開頭空兩格全形格
19.     if(len(introduction) <= 27):             #將段落分行
20.         introduction = [introduction]
21.         text_offset = 0.1
22.     elif(len(introduction) <=54):
23.         introduction = [introduction[0:26], introduction[27:]]
24.         text_offset = 0.05
25.     else:
26.         introduction = [introduction[0:26], introduction[27:53], introduction[54:]]
27.         text_offset = 0
28.     for sentence in introduction:            #繪製文字
29.         ImageDraw.Draw(img_pil).text((x*0.05 , y*(0.75 + text_offset)), sentence , font = font, fill = (0, 0, 0) )#繪製文字
30.         text_offset += 0.05
31.     img = np.array(img_pil)
32.     cv2.imwrite('postcard_' + now + '_back.jpg',img)      #寫檔
```

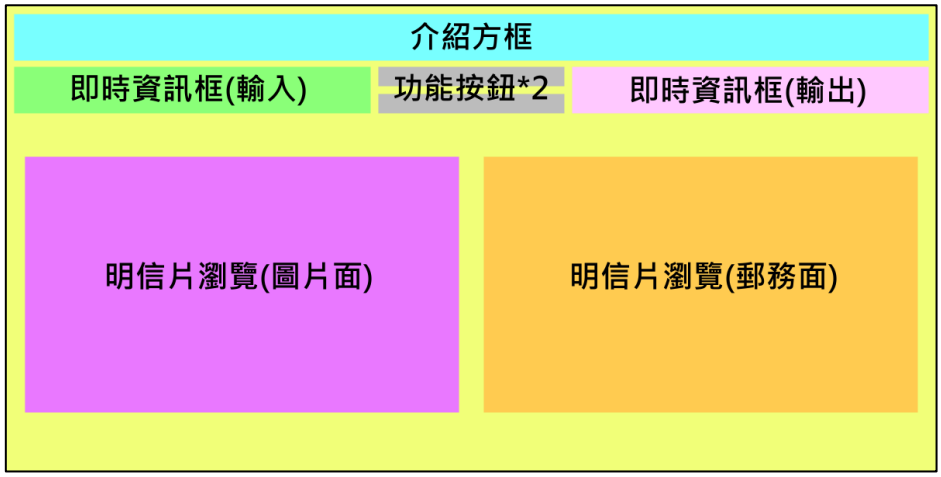
7.分析相片並製作明信片

依據我們在方案建立時繪製的流程圖，並且在所有功能寫成函數後，我們將它進行組合。詳細程式說明如下註解：

```
1.  def photoProcess(path): #照片分析並製作明信片
2.      print(path)
3.      global now, file_name
4.      file_name = path
5.      now = datetime.datetime.now().strftime('%f')     #取得現在時間(作為輸出檔名)
6.      (Time, Coordinate) = analyzePicture(file_name)  #分析相片資訊(GPS、時間)
7.      if (Coordinate == 'Null'):                       #如果抓不到地理資訊與介紹
8.          (Location, Introduction) = ('', '一片自然風景是一個心靈的境界。 — 阿米爾')
9.      elif (os.system('ping 8.8.8.8 -c 2') != 1):      #如果無法連線到網路
10.         (Location, Introduction) = ('', '一片自然風景是一個心靈的境界。 — 阿米爾')
11.     else:
12.         (Location, Introduction) = findLocationName(Coordinate) #取得地點名稱與介紹
13.     (weith, heigh) = mainPictureAddText(str(Location), Time)    #繪製明信片背面(圖案)
14.     makePostcard(weith, heigh, Introduction)                    #繪製明信片正面(郵務)
15.     print('Done.')
16.     return 'postcard_' + now #回傳輸出檔名
```

三、主程式

在執行主要功能的函數(分析相片並製作明信片)完成後，最後就是用戶介面的製作。主要有介紹方框、即時資訊框*2，按鈕*2 及圖片瀏覽框*2。詳細程式說明如下註解(部分文字因排版有壓縮字寬)：



```
1.  def main(): #主函數(用戶介面)
2.      #產生視窗並設定標題、解析度、背景色
3.      window = tk.Tk()
4.      window.title('Post Card Maker')
5.      window.geometry('1200x600')
6.      window.configure(background='lemon chiffon')
7.      #標頭介紹文字
8.      header_label = tk.Label(window, text='你好！歡迎使用明信片產生器，請『選擇照片』並點選『產生明信片』。', bg='light sky blue', width=170, height=2)
9.      header_label.pack()
10.     path = ''
11.
12.     def selectFile(): #建立選擇檔案函數，供按鈕元件呼叫
13.         filename = tk.filedialog.askopenfilename() #呼叫文件選擇器
14.         global path
15.         #檢查檔案格式
16.         if '.jpg' not in filename:
17.             path = ''
18.             left_label.config(text = '尚未選擇檔案', bg = 'RosyBrown1')
19.             tkinter.messagebox.showerror(title='錯誤', message='未選取檔案或格式不支援。')  # 提出錯誤對話窗
20.         else:
21.             path = filename
22.             left_label.config(text = path, bg = 'aquamarine')
23.         right_label.config(text = '尚未產生明信片', bg = 'RosyBrown1')
24.
25.     def process(): #建立圖片處理函數，供按鈕元件呼叫
26.         global path
27.         out_file = photoProcess(path) #照片分析並製作明信片
28.         #取得輸出檔案並調整大小以適合瀏覽
29.         right_label.config(text = '輸出成功：' + out_file, bg = 'aquamarine')
30.         img_1 = Image.open(out_file + '_front.jpg')
31.         img_2 = Image.open(out_file + '_back.jpg')
32.         (x,y) = img_1.size
33.         img_1 = img_1.resize((600,y*600//x))
34.         img_2 = img_2.resize((600,y*600//x))
35.         #更新並顯示預覽圖片
36.         photo_1 = ImageTk.PhotoImage(img_1)
37.         img_1_label.config(image = photo_1)
```

```
38.         img_1_label.image = photo_1
39.         photo_2 = ImageTk.PhotoImage(img_2)
40.         img_2_label.config(image = photo_2)
41.         img_2_label.image = photo_2
42.
43.         tkinter.messagebox.showinfo(title='輸出完成', message='輸出完成：'+ out_file) # 提示資訊對話窗
44.     #功能區元件配置
45.     height_frame = tk.Frame(window) #主框架
46.     height_frame.pack(side=tk.TOP)
47.     #文字元件
48.     left_label = tk.Label(height_frame, text='尚未選擇檔案', bg = 'RosyBrown1', width = 74, height=3) #左
49.     left_label.pack(side=tk.LEFT)
50.     right_label = tk.Label(height_frame, text='尚未產生明信片', bg = 'RosyBrown1', width = 74, height=3) #右
51.     right_label.pack(side=tk.RIGHT)
52.     #按鈕元件
53.     btn = tk.Button(height_frame, text='選擇圖片', width = 20, command = selectFile)
54.     btn.pack()
55.     btn = tk.Button(height_frame, text='產生明信片', width = 20, command = process)
56.     btn.pack()
57.     #圖片元件(未產生明信片時不會有東西)
58.     img_1_label = tk.Label(window)
59.     img_1_label.pack(side=tk.LEFT)
60.     img_2_label = tk.Label(window)
61.     img_2_label.pack(side=tk.RIGHT)
62.
63.     window.mainloop() #呼叫視窗運作
```

四、呼叫

最後，我們呼叫主程式 main()，考量到程式可能有未盡之處，因此我們這邊使用到例外處理。

```
1.  try:
2.      main()
3.  except:
4.      print('很抱歉，發生意外錯誤。')
```

五、必要的檔案

檔案	目的
bb.ttc	繪製文字時所需的字體
CountryCode_ZH_TW.json	國家代碼對應國家名稱 JSON 檔 OpenWeatherMap API 提供
langconv.py	語言轉換(簡-繁)函式
zh_wiki.py	繁簡字符對應表

伍、未來改進空間

一、功能改進

雖然功能上頗有創新，但因為我們並沒有使用 Google Map API，而是使用 OpenWeatherMap API 來取得地點資訊,我們希望近期能進一步改為使用 Google Map API 取的可以為此作品帶來更好的效果，就可以更精準地取得地點資訊。

二、功能發展

另外，雖然目前我們的目的是將風景轉換為明信片，但若拍一些各個國家特有的動物、食物之類的東西感覺也非常地合適，希望未來能結合 AI 影像辨識，讓圖片說明不再只是國家介紹。只要是能被相機保存下來的照片，就能搖身一變，變成極具收藏價值的明信片，同時也很適合寫給親朋好友，分享自己拍的照片的同時，也能將當時自己想說的話寫下來。

陸、參考資料

1. 大專欄。[Python]如何剖析 JSON 數據,如何剖析 JSON Array。2019 年 9 月 2 日。取自：https://www.dazhuanlan.com/2019/09/02/b5832d99023f
2. SCDN 博客。用 python 進行 OpenCV 實戰之畫圖。2017 年 8 月 20 日。取自：https://blog.csdn.net/u014265347/article/details/77430257
3. SCDN 博客。Python 在圖片上添加文字。2019 年 3 月 2 日。取自：https://blog.csdn.net/sinat_29957455/article/details/88071078
4. RETURN TO LAUGHTER。python Exifread, PIL 練習抽出圖片元數據。取自：https://self.jxtsai.info/2016/09/python-exifread-pil.html
5. SCDN 博客。Python 文件選擇對話框。2017 年 9 月 11 日。取自：https://blog.csdn.net/Abit_Go/article/details/77938938
6. 簡書。Tkinter 顯示 jpg 格式圖片。2017 年 11 月 7 日。取自：https://www.jianshu.com/p/f5db045e01aa
7. ITREAD01。Python GUI 之 tkinter 視窗教程大集合。2019 年 1 月 17 日。取自：https://www.itread01.com/content/1547705544.html