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。

※以下作品說明書省略部分過於長的程式碼，詳細程式碼請參考：
https://github.com/JyunWei-Su/TravelPost
※作品說明影片：
https://youtu.be/PEfG1GE5CCU

貳、創作理念

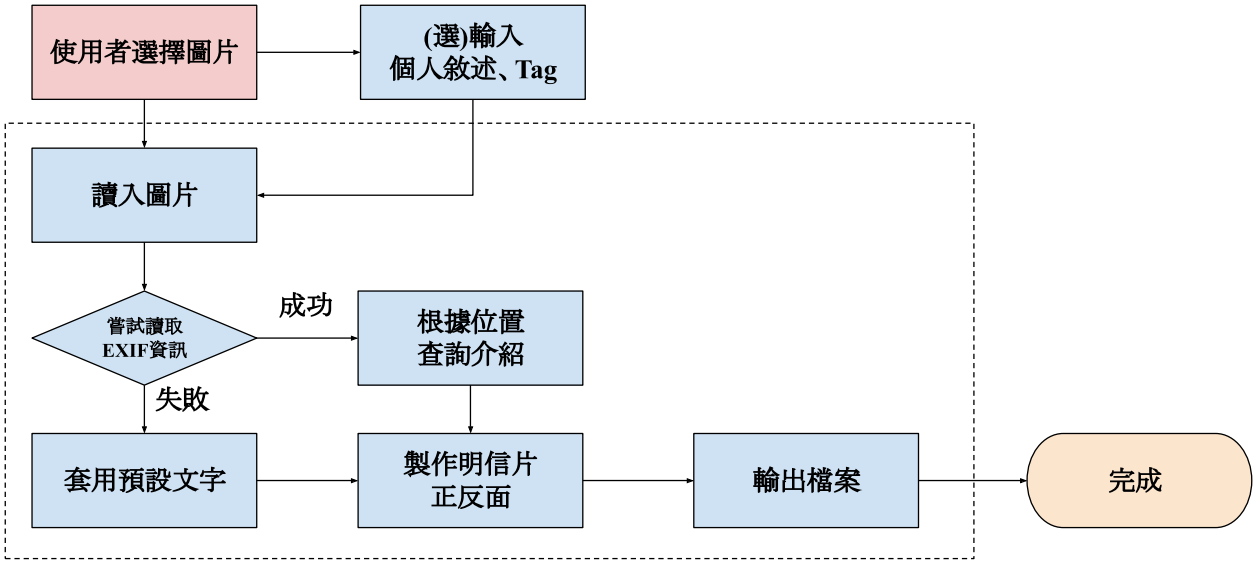
一、創作動機

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，拍完還會買些明信片紀念，因此我們做出這個**可以讓使用者輕鬆將照片轉為明信片**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

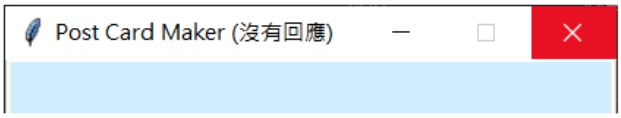
一、應用性

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

在作品主題與內容與呈現方式上，我們花了不少時間討論。此外，雖然我們順利地透過 GPS 座標取得地點名稱，但在透過維基百科 API 抓取部分國家的介紹時，無法正常抓取，後來使用 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。

還有一點，我們使用 python 中的 tkinter 製作用戶介面時，每當點擊按鈕，運行了一個比較耗時的操作，界面會卡死，後來我們也找到了使用多線程的方式解決。



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。

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2320m

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。

萬人可乘坐的邊闊草原，沿途邊闊蔚藍海景令人心曠神怡。

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。

@Marisol @Ayden

#沖繩萬座毛

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，*file_path* 是我們的來源照片的檔案路徑，會再使用者選擇相片後更動。*now* 是現在時間，會在之後使用者按下產生明信片後更動。

```
1.  import requests                                  #網路資料抓取
2.  import cv2, exifread, numpy as np                #圖片處理
3.  import wikipedia                                 #維基百科查詢
4.  import string                                    #字串處理
5.  from PIL import ImageFont, ImageDraw, Image, ImageTk #圖片處理
6.  from geopy.geocoders import Nominatim            #geopy(地理資訊獲取)
7.  from langconv import *                           #簡字轉繁字
8.  import tkinter as tk                             #用戶介面(下同)
9.  import tkinter.filedialog ,tkinter.messagebox
10. import os, sys, datetime                         #系統、系統時間
11. import threading                                 #多執行緒
12. from pathlib import Path                         #檔案系統路徑
13. inFile_path = ''                                 #來源影像檔名
14. 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.      #將括號內容清除
6.          '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
7.      #若段落過長，移除最長的句子
```

```
    while(len(text)>81):
        '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
    return text #回傳地點介紹
```

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 gorpy 函式庫來實現。函數的輸入 *Coordinate*，格式是根據函式庫的規範，我們透過經緯度去做請求後，從回傳的字串取出所在地國家及城市的名稱，並使用前述維基百科查詢函數來取得國家介紹，最後回傳結果。詳細程式說明如下註解：

```
def findLocationName(Coordinate): #依 GPS 座標取得地名與介紹(使用 geopy)
    global now
    #嘗試透過 geopy 獲取地名
    try:
        geolocator = Nominatim(user_agent='TravelPost' + now, timeout = 10)
        location = str(geolocator.reverse(Coordinate))
        '''''format:地名, 村里, 鄉鎮, 縣市, 區省, 郵遞號, 國家'''
    except:
        print('...geolocator_Err', end ='')
        return ('', '疊嶂峰上明月羞、翠光浮動萬山秋。')
    loc_lists = location.split(', ')            #將地址字串轉為 list
    if(loc_lists[-1] == 'Taiwan'):              #修正臺灣行政地理層級錯誤(台灣省)
        loc_lists[-1] = '台灣'
        if '臺灣省' in loc_lists :
            loc_lists.remove('臺灣省')
        else:
            loc_lists[-3] = loc_lists[-2]
    loc_lists = [ loc_lists[-1].split(' ')[0], loc_lists[-3] ]
    Location = loc_lists[0] + ' ' + loc_lists[1]#format:['國家', '城市']
    for char in ['縣', '市', '県']:             #行政級別刪除
        Location = Location.replace(char, '')
    Introduction = getWiki(loc_lists[0])        #呼叫 getWiki 取得"國家"介紹
    print('...findLocationName_OK')
    return (Location, Introduction) #回傳地點與介紹
```

3.經緯度座標轉換(60 進位 to 10 進位)

因為從圖片 EXIF 資訊中所獲得的 GPS 資料是 60 進位，因此我們製作這個函數來完成座標轉換。輸入 *coordinate* 會是經度或緯度，呼叫一次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)：

```
1.  def coordinateConvert(coordinate): #GPS 座標換算(60 進位 to 10 進位)
2.      '''''input format: '[nn, mm, aa/bb]' (60 進位,百分位為分數)'''
3.      coordinate = coordinate.replace(' ', '')    #移除空白
4.      coordinate = coordinate.replace('[', '')    #移除左中括號
5.      coord_2_a = (coordinate[coordinate.rfind(',') + 1:coordinate.rfind('/')])   #取得百分位數值分母
6.      coord_2_b = (coordinate[coordinate.rfind('/') + 1:coordinate.rfind(']')])   #取得百分位數值分子
7.      coordinate = coordinate.replace('/', '').replace(']', '')             #移除左斜和右括號
8.      coordinate = coordinate.split(',')          #此時 fomat 'nn,nn,aa/bb'，分割為 list
9.      coordinate[2] = int(coord_2_a) / int(coord_2_b)     #計算百分位之分數
10.     coordinate = int(coordinate[0]) + int(coordinate[1])/60 + coordinate[2]/600 #計算座標十進位值
11.     coordinate = round(coordinate, 6)                   #小數取 6 位
12.     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：

```
1.  def analyzePicture(inFile_path): #分析照片資訊，取得時間及 GPS
2.      f = open(inFile_path, 'rb')
3.      tags = exifread.process_file(f)
4.      #嘗試讀取 GPS 資訊
5.          '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
6.      #嘗試讀取時間(GPS 資料優先)
7.      try:
8.          '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
9.      except:
10.         Time = 'Fascinating.'
11.         print('時間讀取失敗')
12.     print('...analyzePicture_OK')
13.     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(圖片面)。詳細程式說明如下註解(部分文字因排版有壓縮字寬)：

```
1.  def mainPictureAddText(Location, Time): #繪製明信面反面(風景
2.      bk_img = cv2.imread(inFile_path)
3.      (x,y,z) = bk_img.shape          #取得相片的長與寬(x,y)，z 在這裡用不到
4.      text_x = int(0.05*y)            #設定文字 x 座標
5.      text_y = int(0.9*x)             #設定文字 y 座標
6.      text = Location + ' ' + Time    #設定文字(國家名稱 + 時間(日期) )
7.      backClor =(102, 102, 102)       #設定字卡底色
8.      font = ImageFont.truetype(str(Path(sys.argv[0]).parent.joinpath('font_MicrosoftJhengHei.ttf')), y//30) #設定字體與字型大小
9.      (width, heigh), (offset_x, offset_y) = font.font.getsize(text) #取得文字方框大小
10.     #繪製字卡
11.         '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
12.     #繪製文字
13.         '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
14.     #寫入並回傳長寬
15.     cv2.imwrite(str(Path(sys.argv[0]).parent.joinpath('postcard_' + now + '_back.jpg')),bk_img)
16.     print('...mainPictureAddText_OK')
17.     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)

函數中有五個輸入分別是寬、長、地點介紹、Tag 文字及使用者敘述。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、地點介紹的文字、Tag 內容及使用者敘述。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, textTag, textUsr):          #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(str(Path(sys.argv[0]).parent.joinpath('postcard_' + now + '_front.jpg')), postcard) #先寫檔
9.
10.     #繪製文字(國家介紹、個人敘述、Tag)
11.     img = cv2.imread(str(Path(sys.argv[0]).parent.joinpath('postcard_' + now + '_front.jpg'))) #讀檔
12.     img_pil = Image.fromarray(img)                          #繪版
13.     #繪製國家介紹文字
14.     font_itr = ImageFont.truetype(str(Path(sys.argv[0]).parent.joinpath('font_MicrosoftJhengHei.ttf')), y//30)      #設定需要顯示的字體與大小
15.     font_usr = ImageFont.truetype(str(Path(sys.argv[0]).parent.joinpath('font_HuakangBamboo.ttc')), y//30)
16.     #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)
17.     (width, heigh), (offset_x, offset_y) = font_itr.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。')
18.     #檢查是否超出格式範圍(版面是否會異常)
19.         '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
20.     #繪製文字_國家介紹資訊(文字多時須分多行)
21.     introduction = '    ' + introduction     #開頭空兩格全形格
22.     #將段落分行並繪製文字
23.         '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
24.     #繪製文字_個人敘述
25.     text_offset = 0
26.     textUsr = '    ' + textUsr
27.     #每 27 字繪製一行
28.         '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
29.     #繪製文字_Tag
30.         '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
31.     #寫檔
32.     img = np.array(img_pil)
33.     cv2.imwrite(str(Path(sys.argv[0]).parent.joinpath('postcard_' + now + '_front.jpg')),img)
34.     print('...makePostcard_OK')
```

7.分析相片並製作明信片

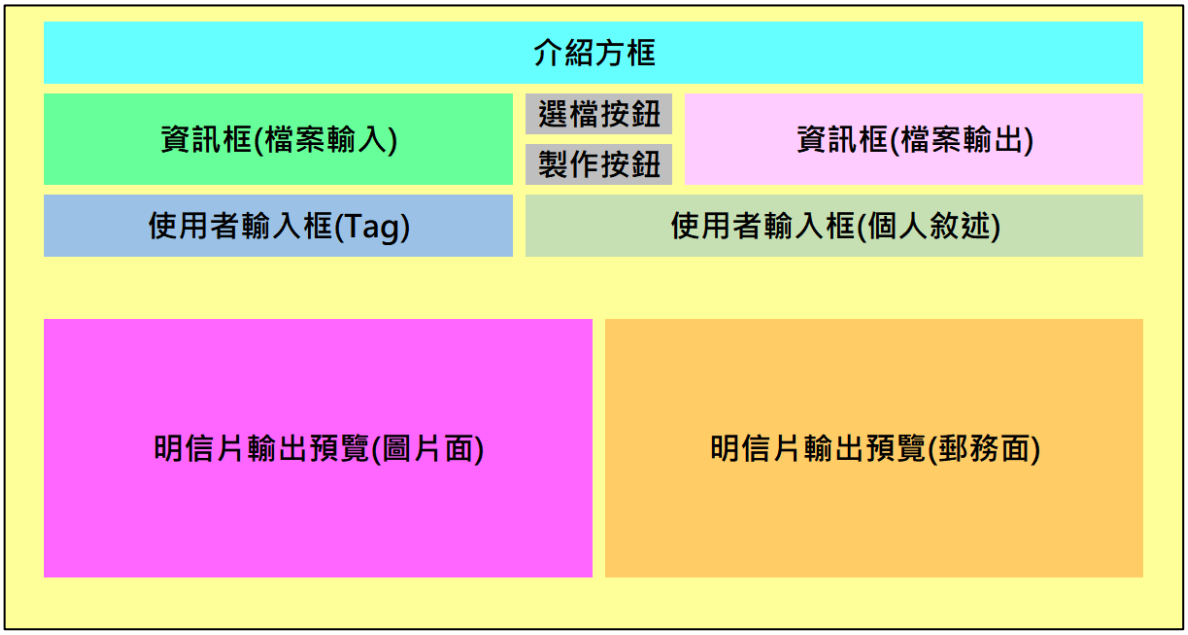
依據我們在方案建立時繪製的流程圖，並且在所有功能寫成函數後，我們將它進行組合。詳細程式說明如下註解：

```
1.  def photoProcess(path, textTag, textUsr): #照片分析並製作明信片
2.      print(path)
3.      global now, inFile_path
4.      inFile_path = path
```

```
5.      now = datetime.datetime.now().strftime('%f')              #取得現在時間(作為輸出檔名)
6.      (Time, Coordinate) = analyzePicture(inFile_path)          #分析相片資訊(GPS、時間)
7.      if (os.system('ping -n 1 -w 100 8.8.8.8 ') != 0):         #如果無法連線到網路 連通為0
8.          (Location, Introduction) = ('', '疊嶂峰上明月羞、翠光浮動萬山秋。')
9.      elif (Coordinate == 'Null'):                              #如果抓不到地理資訊與介紹
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)    #繪製明信片反面(圖案)
14.     makePostcard(weith, heigh, Introduction, textTag, textUsr)  #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。如下圖所示，我們使用 tkinter 設計介紹方框、資訊框*2，按鈕*2、輸入框*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('1200x680')
```

```
6.      window.configure(background='lemon chiffon')
7.
8.      path = ''
9.      def selectFile(): #建立選擇檔案函數，供按鈕元件呼叫
10.         filename = tk.filedialog.askopenfilename() #呼叫文件選擇器
11.         nonlocal path
12.         #檢查檔案格式
13.         if ('.jp' not in filename) and ('.JP' not in filename) :
14.             path = ''
15.             left_label.config(text = '尚未選擇檔案', bg = 'RosyBrown1')
16.             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
17.         else:
18.             path = filename
19.             left_label.config(text = '檔案來源：' + path, bg = 'aquamarine')
20.         right_label.config(text = '尚未產生明信片', bg = 'RosyBrown1')
21.     def threadFunc(func, *args): #將函數打包進線程
22.         trd_func = threading.Thread(target=func, args=args) #創建
23.         trd_func.setDaemon(True) #守護
24.         trd_func.start() #啟動
25.     def process(): #建立圖片處理函數，供按鈕元件呼叫
26.         nonlocal path
27.         mak_btn.config(text = '明信片產生中...', bg = 'salmon')
28.         textTag = (tag_text.get(1.0, 'end') + 'End').replace('\nEnd','')
29.         textUsr = (usr_text.get(1.0, 'end') + 'End').replace('\nEnd','')
30.         try:
31.             out_file = photoProcess(path, textTag, textUsr) #照片分析並製作明信片
32.             #取得輸出檔案並調整大小以適合瀏覽
33.             right_label.config(text = '輸出成功：' + out_file, bg = 'aquamarine')
34.             img_1 = Image.open(str(Path(sys.argv[0]).parent.joinpath(out_file + '_back.jpg')))
35.             img_2 = Image.open(str(Path(sys.argv[0]).parent.joinpath(out_file + '_front.jpg')))
36.             (x,y) = img_1.size
37.             if y > x :
38.                 (x,y) = (y,x)
39.                 img_1 = img_1.transpose(Image.ROTATE_270)
40.             img_1 = img_1.resize((600,y*600//x))
41.             img_2 = img_2.resize((600,y*600//x))
42.             #更新並顯示預覽圖片
43.                 '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
```

```
            #重置使用者介面
            tag_text.delete(1.0, 'end')
            usr_text.delete(1.0, 'end')
            mak_btn.config(text = '產生明信片', bg = 'SystemButtonFace')
            tkinter.messagebox.showinfo(title='輸出完成', message='輸出完成：'+ out_file)       #提示資訊對話窗
        except:
            mak_btn.config(text = '產生明信片', bg = 'SystemButtonFace')
            if(path == ''):
                tkinter.messagebox.showerror(title='錯誤', message='請先選擇檔案') #提出錯誤對話窗
            else:
                tkinter.messagebox.showerror(title='錯誤', message='發生意外錯誤，請重新操作')  #提出錯誤對話窗
            path = ''
            left_label.config(text = '尚未選擇檔案', bg = 'RosyBrown1')

    #標頭介紹文字
    header_label_frame = tk.Frame(window, width = 1200, height=60)
    header_label_frame.pack_propagate(0)
    header_label_frame.pack(side=tk.TOP)
    header_label = tk.Label(header_label_frame, 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』。\n『TAG』及『個人敘述』可選擇性輸入', bg='light sky blue')
    header_label.pack(fill=tk.BOTH, expand=True)
    '''''功能區元件配置(為了版面配置，所有元件置於框架中)'''
    #功能區框架
    func_frame = tk.Frame(window)
    func_frame.pack(side=tk.TOP)
    #文字元件 left_label、right_label
        '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
    #按鈕元件 slt_btn、mak_btn
        '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
    #使用者輸入框架
    usr_frame = tk.Frame(window)
    usr_frame.pack(side=tk.TOP)
    #文字元件及文字框元件 tag_label、tag_text、usr_label、usr_text
        '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''
    #圖片元件 img_1_label、img_2_label(未產生明信片時不會有東西)
        '''''部分程式碼省略，詳細請參照 github.com/JyunWei-Su/TravelPost'''

    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。

```
try:
    main()
except:
    print('很抱歉，發生意外錯誤。')
```

五、必要的檔案

檔案	目的
font_HuakangBamboo.ttc font_MicrosoftJhengHei.ttf	繪製文字時所需的字體
langconv.py	語言轉換(簡-繁)函式
zh_wiki.py	繁簡字符對應表

伍、未來發展空間

「信戀千里-旅遊明信片產生器」最大特色是讓人享受隨玩、隨拍、隨時製發行動明信片的樂趣。我們希望未來能將目前的電腦版轉成行動版。除此之外，未來希望結合 AI 影像辨識及語音輸入，讓圖片說明不只是地理風景。只要是能被捕捉的照片，就能搖身一變成為極具收藏價值的「信戀明信片」。例如，隨時能將動物、食物進行辨識，自動抓取維基介紹或推薦社群，以語音輸入能將當時自己想說的話「說出來」，隨處分享給好友。

陸、參考資料

1. 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
2. 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
3. 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
4. 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
5. 簡書。Tkinter 顯示 jpg 格式圖片。2017 年 11 月 7 日。取自：https://www.jianshu.com/p/f5db045e01aa
6. 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
7. SCDN 博客。python tkinter 界面卡死的解決辦法。2019 年 1 月 30 日。取自：https://blog.csdn.net/qq_41204464/article/details/86707216