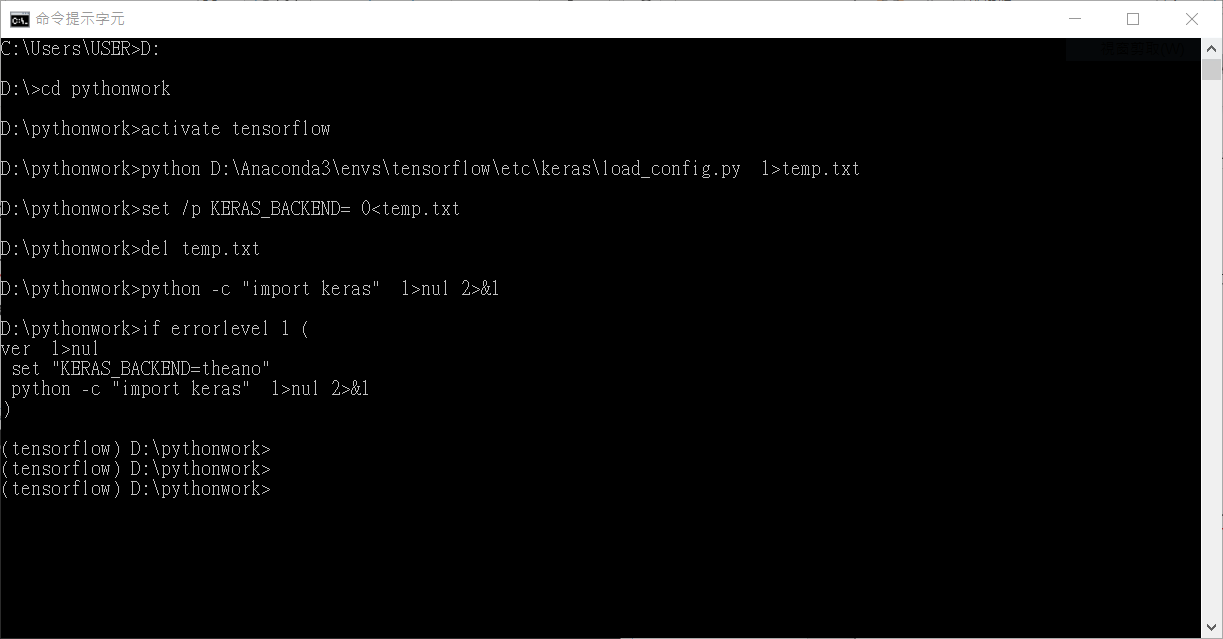
人臉辨識教學文件

1. 使用環境
2. 使用python3.5傳寫
3. 實驗步驟

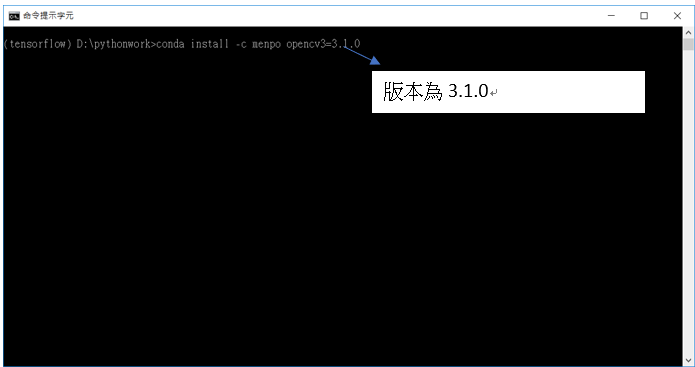
1、安裝opencv

1. 先利用CMD進入tensorflow 環境中



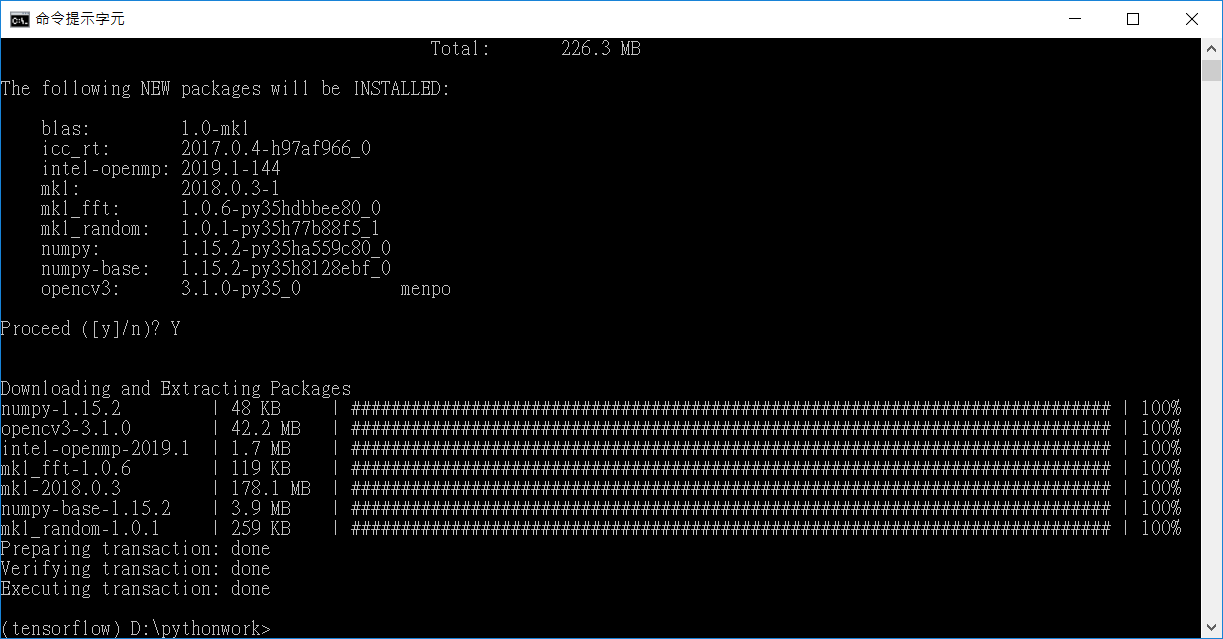
1. 複製以下指令碼執行安裝opencv

conda install -c menpo opencv3=3.1.0



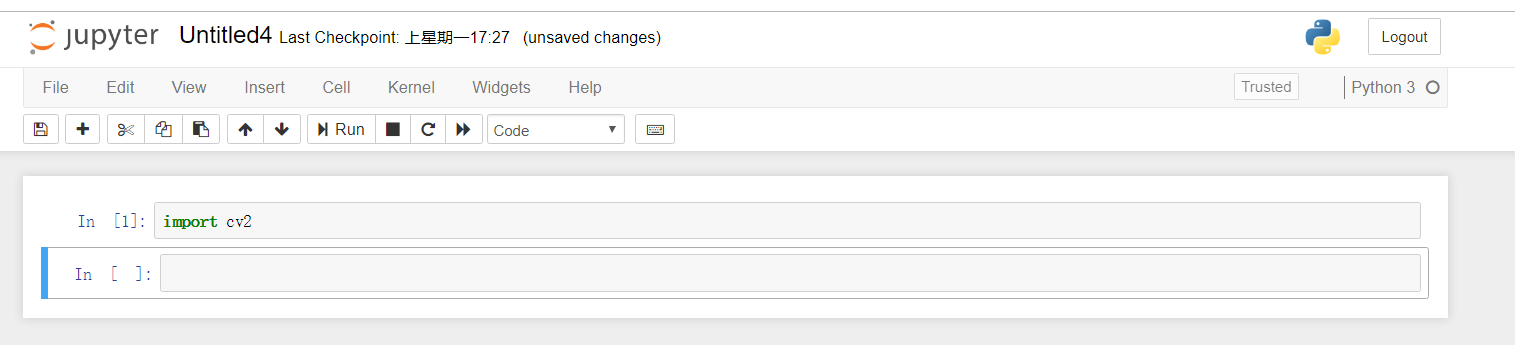


1. 完成安裝



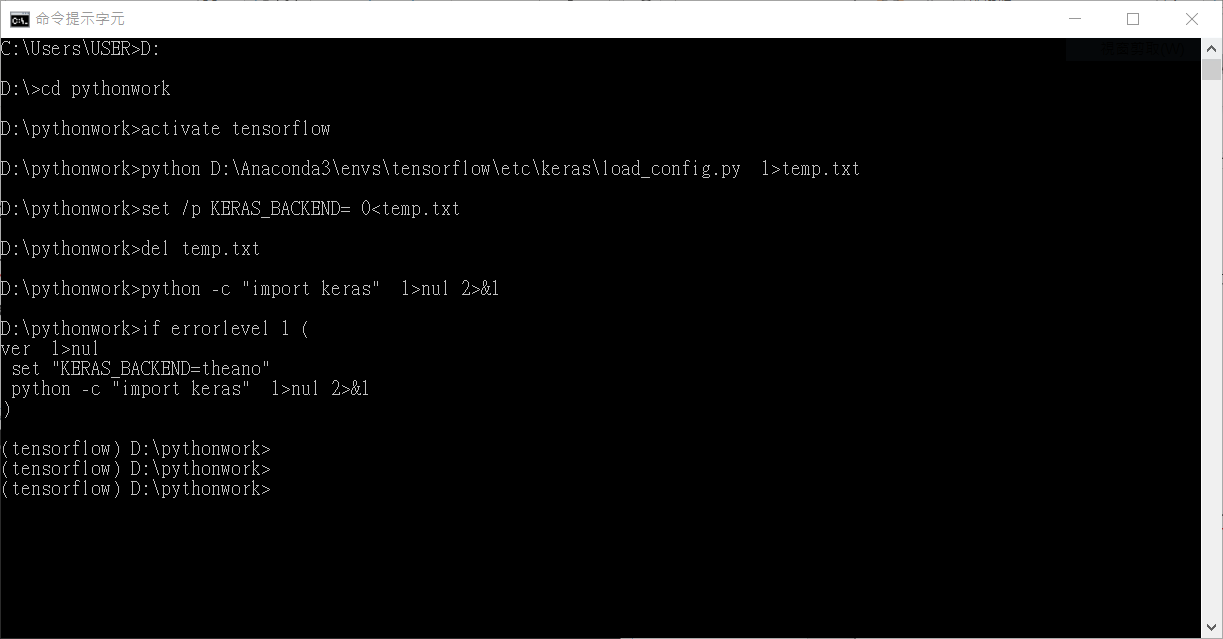
1. 利用import測試，若安裝失敗會有錯誤

import cv2



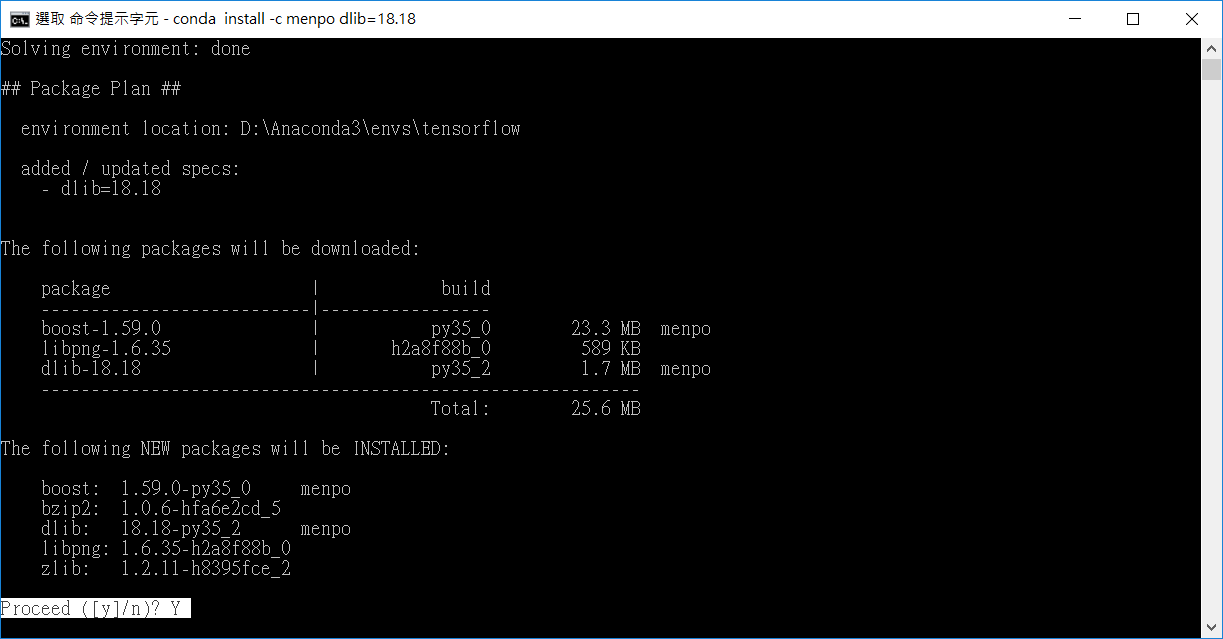
2.安裝dlib

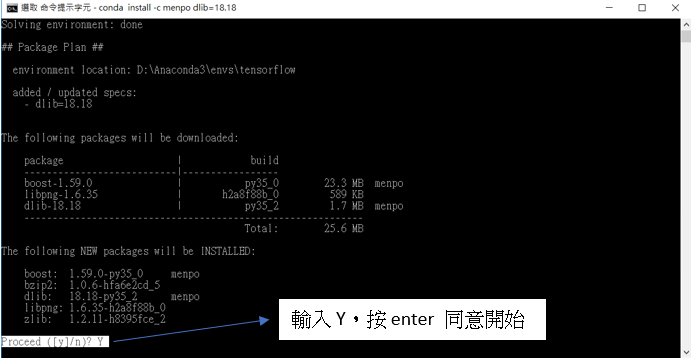
(1) 先利用CMD進入tensorflow 環境中



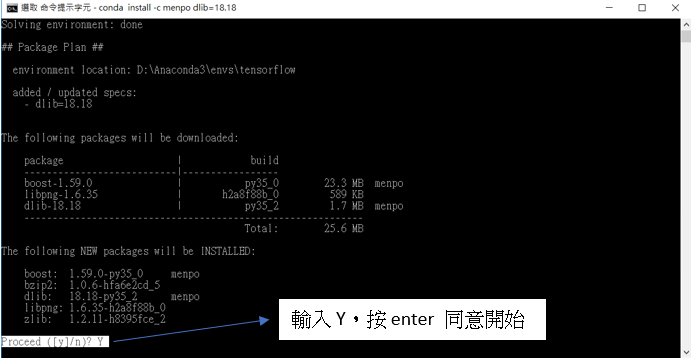
(2) 複製以下指令碼執行安裝dlib

conda install -c menpo dlib=18.18



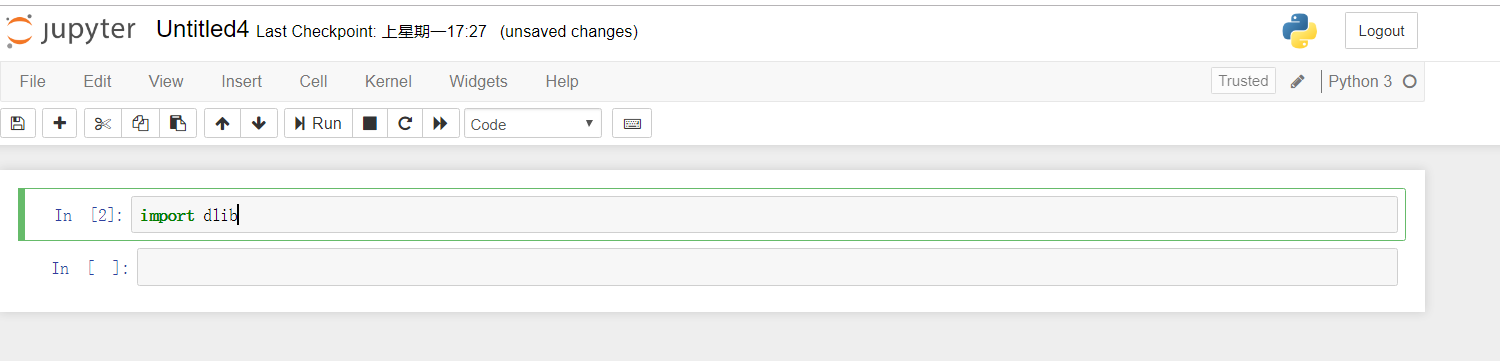


(3) 完成安裝



(4) 利用import測試，若安裝失敗會有錯誤

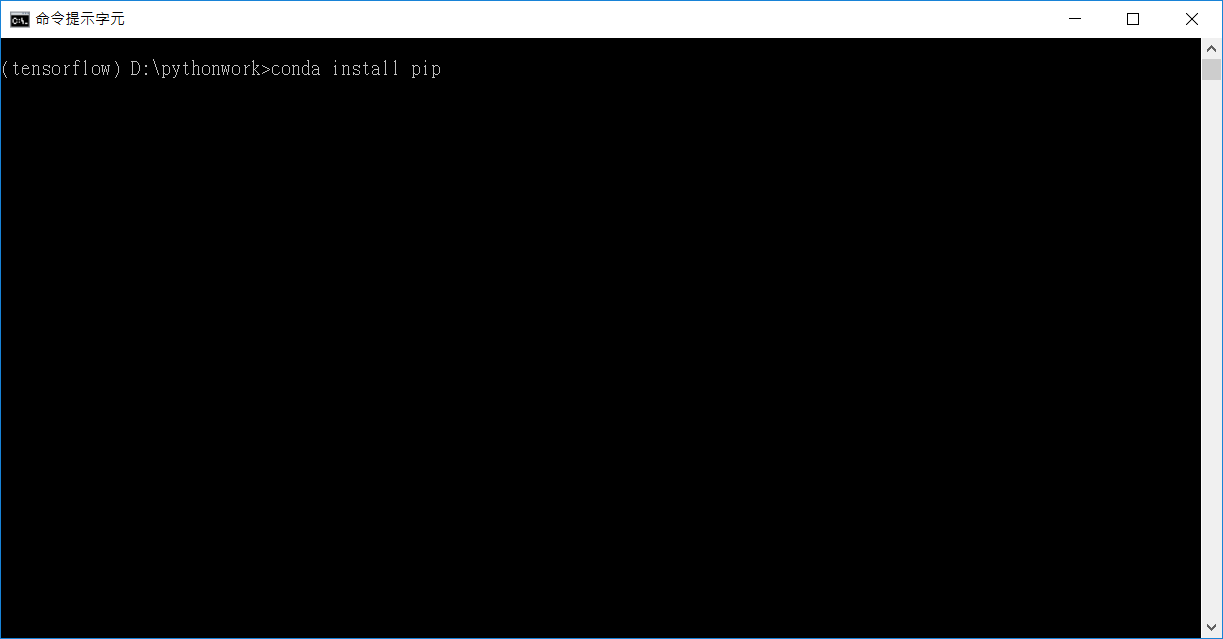
import dlib

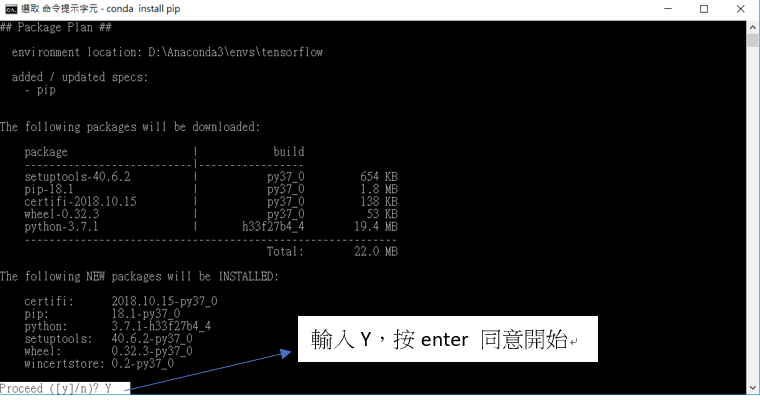


3. 安裝imutils

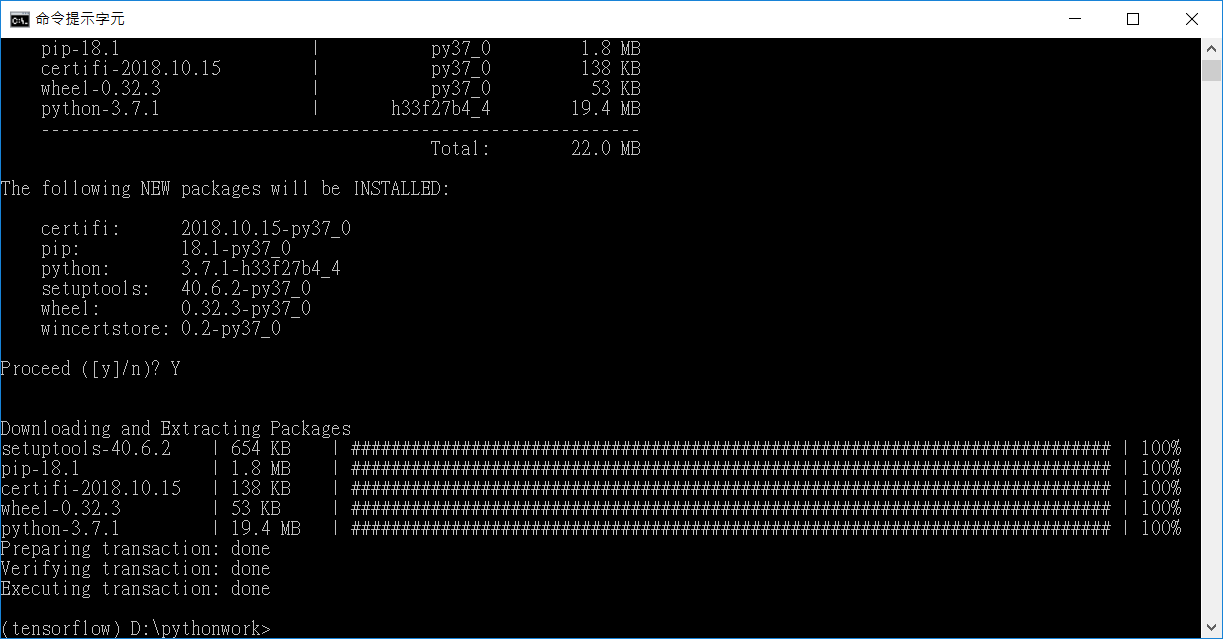
(1) 由於anaconda W64 不支援，所以imutls需由pip安裝

conda install pip



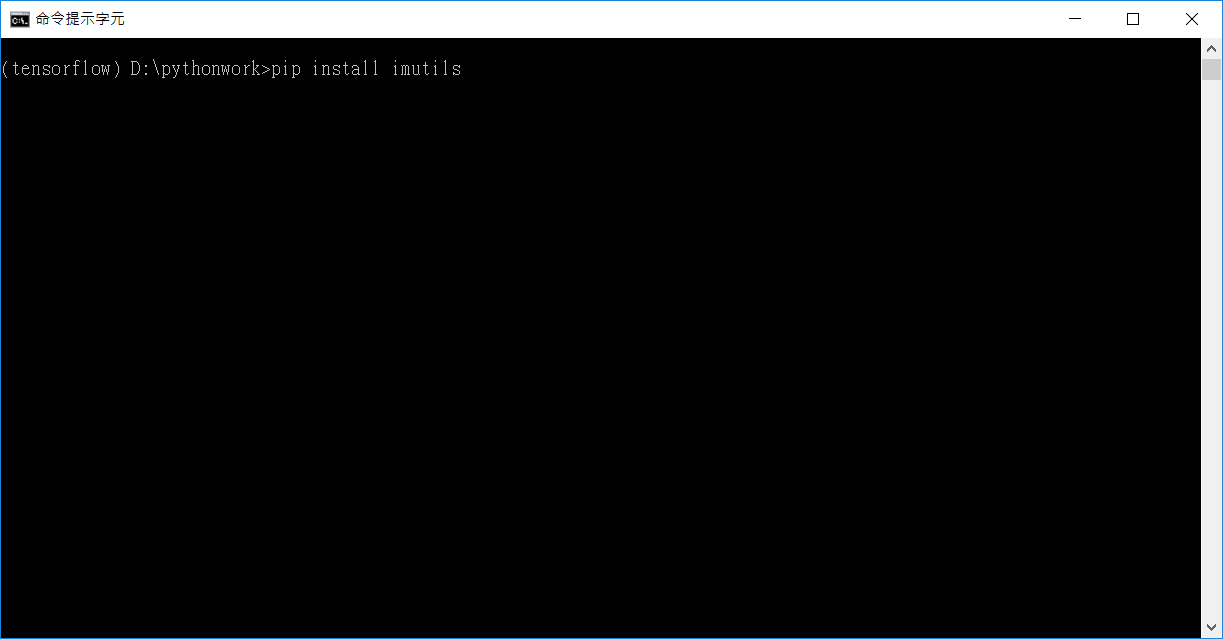


1. 安裝完成

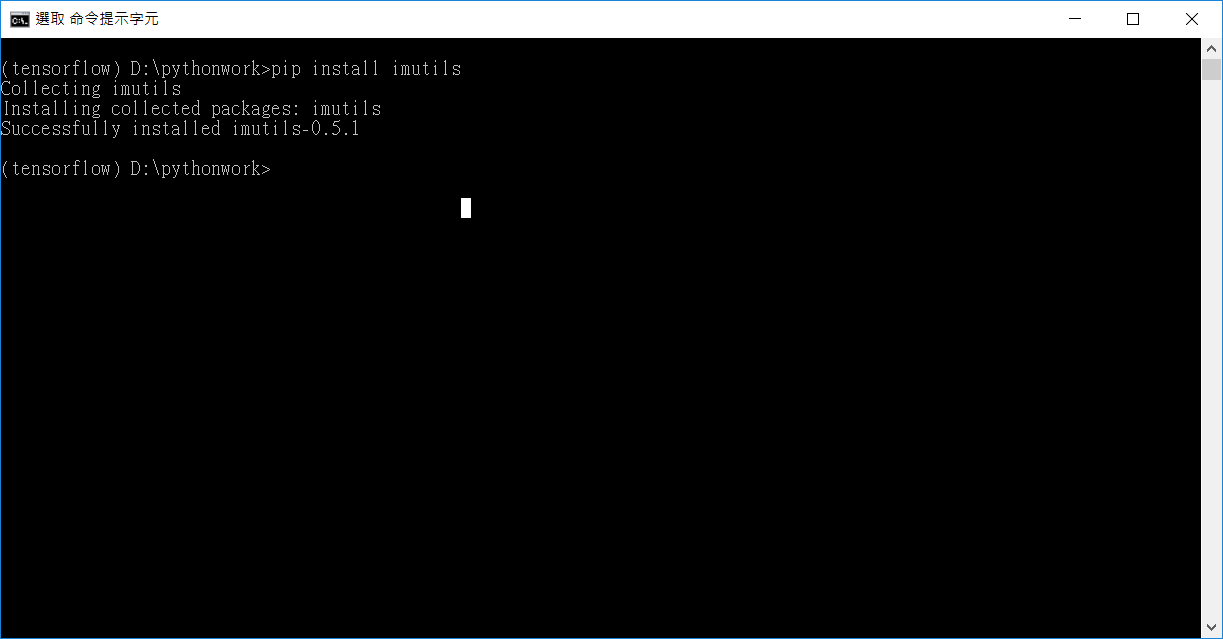


(6) 安裝imutils

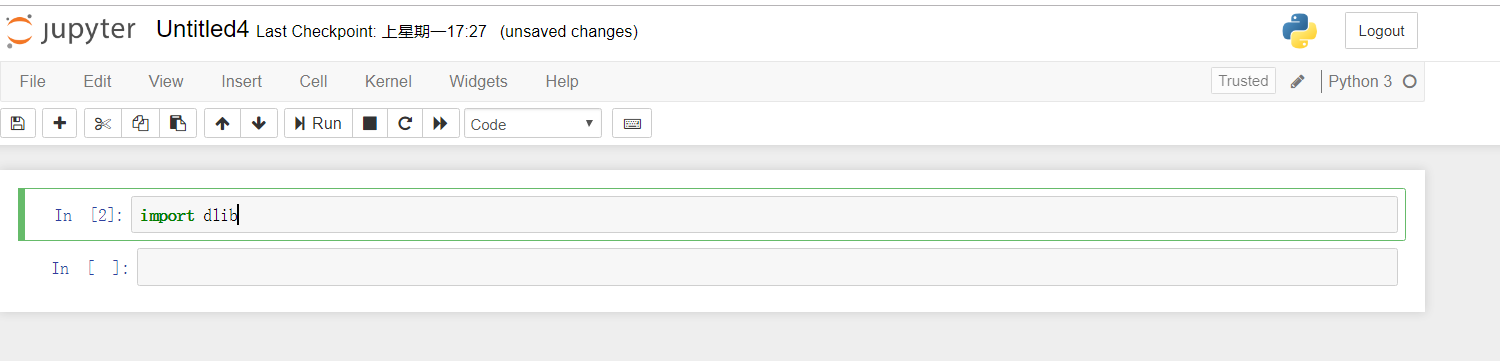
pip install imutils



(7)imutils安裝完成



(8)利用import測試，若安裝失敗會有錯誤



4. 疲勞駕駛偵測

(1) 參考來源:

<https://github.com/vitorgodeiro/DrowsinessDetection>

(2)請先複製以下程式碼

#!/usr/bin/env python

import cv2

import time

import numpy as np

from scipy.spatial import distance as dist

from imutils import face\_utils

import dlib

EYE\_AR\_THRESH = 0.3

EYE\_AR\_CONSEC\_FRAMES = 7

MOUTH\_AR\_THRESH = 0.4

SHOW\_POINTS\_FACE = False

SHOW\_CONVEX\_HULL\_FACE = False

SHOW\_INFO = False

ear = 0

mar = 0

COUNTER\_FRAMES\_EYE = 0

COUNTER\_FRAMES\_MOUTH = 0

COUNTER\_BLINK = 0

COUNTER\_MOUTH = 0

def eye\_aspect\_ratio(eye):

A = dist.euclidean(eye[1], eye[5])

B = dist.euclidean(eye[2], eye[4])

C = dist.euclidean(eye[0], eye[3])

return (A + B) / (2.0 \* C)

def mouth\_aspect\_ratio(mouth):

A = dist.euclidean(mouth[5], mouth[8])

B = dist.euclidean(mouth[1], mouth[11])

C = dist.euclidean(mouth[0], mouth[6])

return (A + B) / (2.0 \* C)

videoSteam = cv2.VideoCapture(0)

ret, frame = videoSteam.read()

size = frame.shape

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor(r"C:\Users\USER\Downloads\shape\_predictor\_68\_face\_landmarks2.dat") #要將shape\_predictor\_68\_face\_landmarks.dat路徑改為你所放路徑!!

(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"]

(rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"]

model\_points = np.array([(0.0, 0.0, 0.0),

(0.0, -330.0, -65.0),

(-225.0, 170.0, -135.0),

(225.0, 170.0, -135.0),

(-150.0, -150.0, -125.0),

(150.0, -150.0, -125.0)])

focal\_length = size[1]

center = (size[1]/2, size[0]/2)

camera\_matrix = np.array([[focal\_length, 0, center[0]],

[0, focal\_length, center[1]],

[0, 0, 1]], dtype = "double")

dist\_coeffs = np.zeros((4,1))

t\_end = time.time()

while(True):

ret, frame = videoSteam.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

rects = detector(gray, 0)

for rect in rects:

shape = predictor(gray, rect)

shape = face\_utils.shape\_to\_np(shape)

leftEye = shape[lStart:lEnd]

rightEye = shape[rStart:rEnd]

jaw = shape[48:61]

leftEAR = eye\_aspect\_ratio(leftEye)

rightEAR = eye\_aspect\_ratio(rightEye)

ear = (leftEAR + rightEAR) / 2.0

mar = mouth\_aspect\_ratio(jaw)

image\_points = np.array([

(shape[30][0], shape[30][1]),

(shape[8][0], shape[8][1]),

(shape[36][0], shape[36][1]),

(shape[45][0], shape[45][1]),

(shape[48][0], shape[48][1]),

(shape[54][0], shape[54][1])

], dtype="double")

(success, rotation\_vector, translation\_vector) = cv2.solvePnP(model\_points, image\_points, camera\_matrix, dist\_coeffs, flags=cv2.SOLVEPNP\_ITERATIVE)

(nose\_end\_point2D, jacobian) = cv2.projectPoints(np.array([(0.0, 0.0, 1000.0)]), rotation\_vector, translation\_vector, camera\_matrix, dist\_coeffs)

if SHOW\_POINTS\_FACE:

for p in image\_points:

cv2.circle(frame, (int(p[0]), int(p[1])), 3, (0,0,255), -1)

p1 = (int(image\_points[0][0]), int(image\_points[0][1]))

p2 = (int(nose\_end\_point2D[0][0][0]), int(nose\_end\_point2D[0][0][1]))

if SHOW\_CONVEX\_HULL\_FACE:

leftEyeHull = cv2.convexHull(leftEye)

rightEyeHull = cv2.convexHull(rightEye)

jawHull = cv2.convexHull(jaw)

cv2.drawContours(frame, [leftEyeHull], 0, (255, 255, 255), 1)

cv2.drawContours(frame, [rightEyeHull], 0, (255, 255, 255), 1)

cv2.drawContours(frame, [jawHull], 0, (255, 255, 255), 1)

cv2.line(frame, p1, p2, (255,255,255), 2)

if p2[1] > p1[1]\*1.5 or COUNTER\_BLINK > 25 or COUNTER\_MOUTH > 2:

cv2.putText(frame, "Send Alert!", (200, 60), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

if ear < EYE\_AR\_THRESH:

COUNTER\_FRAMES\_EYE += 1

if COUNTER\_FRAMES\_EYE >= EYE\_AR\_CONSEC\_FRAMES:

cv2.putText(frame, "Sleeping Driver!", (200, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

else:

if COUNTER\_FRAMES\_EYE > 2:

COUNTER\_BLINK += 1

COUNTER\_FRAMES\_EYE = 0

if mar >= MOUTH\_AR\_THRESH:

COUNTER\_FRAMES\_MOUTH += 1

else:

if COUNTER\_FRAMES\_MOUTH > 5:

COUNTER\_MOUTH += 1

COUNTER\_FRAMES\_MOUTH = 0

if (time.time() - t\_end) > 60:

t\_end = time.time()

COUNTER\_BLINK = 0

COUNTER\_MOUTH = 0

if SHOW\_INFO:

cv2.putText(frame, "EAR: {:.2f}".format(ear), (30, 450),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0), 2)

cv2.putText(frame, "MAR: {:.2f}".format(mar), (200, 450),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0), 2)

cv2.putText(frame, "Blinks: {}".format(COUNTER\_BLINK), (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0), 2)

cv2.putText(frame, "Mouths: {}".format(COUNTER\_MOUTH), (10, 60),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0), 2)

cv2.imshow("Output", frame)

key = cv2.waitKey(1) & 0xFF

if key == ord('q'):

break

if key == ord('p'):

SHOW\_POINTS\_FACE = not SHOW\_POINTS\_FACE

if key == ord('c'):

SHOW\_CONVEX\_HULL\_FACE = not SHOW\_CONVEX\_HULL\_FACE

if key == ord('i'):

SHOW\_INFO = not SHOW\_INFO

time.sleep(0.02)

videoSteam.release()

cv2.destroyAllWindows()

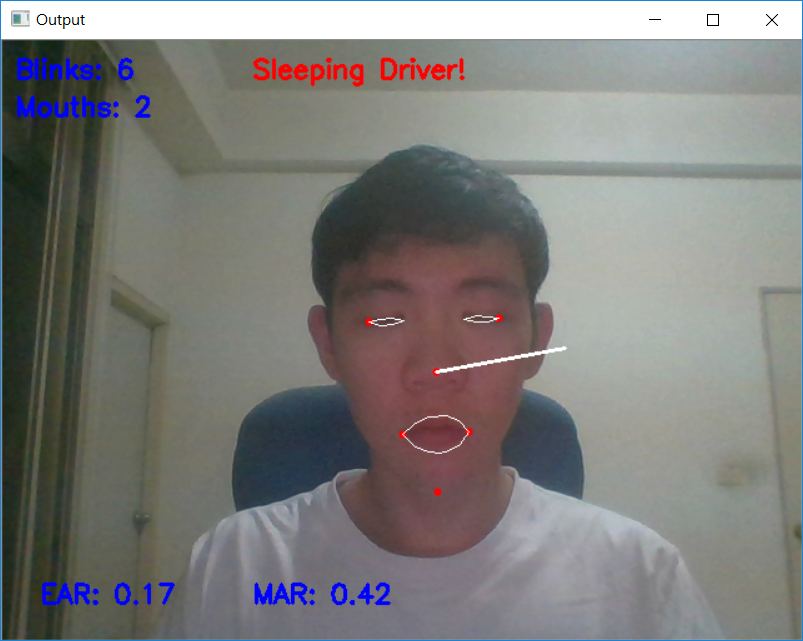
(3)測試若成功攝影機將會啟動 並且開始執行偵測打哈欠跟眨眼等動作

可以用p鍵來開啟關鍵點偵測

c顯示輪廓

i顯示偵測結果

q離開程式



5. 人臉辨識

(1) 參考來源:

<https://github.com/vitorgodeiro/DrowsinessDetection>

(2) 建立人臉資料，請先複製以下程式碼

import dlib # 人臉處理的庫 Dlib

import numpy as np # 資料處理的庫 Numpy

import cv2 # 影像處理的庫 OpenCv

import os # 讀寫文件

import shutil # 讀寫文件

# Dlib 正向人臉檢測器

detector = dlib.get\_frontal\_face\_detector()

# Dlib 68 點特徵預測器

predictor = dlib.shape\_predictor('D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_dlib/shape\_predictor\_68\_face\_landmarks.dat')

# OpenCv 調用攝像頭

cap = cv2.VideoCapture(0)

# 設置視頻參數

cap.set(3, 480)

# 人臉截圖的計數器

cnt\_ss = 0

# 存儲人臉的資料夾

current\_face\_dir = 0

# 保存的路徑

path\_make\_dir = "D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_faces\_from\_camera/"

path\_csv = "D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_csvs\_from\_camera/"

# 新建資料夾, 刪除之前存的人臉資料檔案夾

def pre\_work():

# 新建資料夾

if os.path.isdir(path\_make\_dir):

pass

else:

os.mkdir(path\_make\_dir)

if os.path.isdir(path\_csv):

pass

else:

os.mkdir(path\_csv)

# 刪除之前存的人臉資料檔案夾

folders\_rd = os.listdir(path\_make\_dir)

for i in range(len(folders\_rd)):

shutil.rmtree(path\_make\_dir+folders\_rd[i])

csv\_rd = os.listdir(path\_csv)

for i in range(len(csv\_rd)):

os.remove(path\_csv+csv\_rd[i])

# 每次程式錄入之前，刪掉之前存的人臉資料

pre\_work()

# 人臉種類數目的計數器

person\_cnt = 0

# The flag of if u can save images

save\_flag = 1

while cap.isOpened():

# 480 height \* 640 width

flag, img\_rd = cap.read()

kk = cv2.waitKey(1)

img\_gray = cv2.cvtColor(img\_rd, cv2.COLOR\_RGB2GRAY)

# 人臉數 faces

faces = detector(img\_gray, 0)

# 待會要寫的字體

font = cv2.FONT\_HERSHEY\_COMPLEX

# 按下 'n' 新建存儲人臉的資料夾

if kk == ord('n'):

person\_cnt += 1

current\_face\_dir = path\_make\_dir + "person\_" + str(person\_cnt)

print('\n')

for dirs in (os.listdir(path\_make\_dir)):

if current\_face\_dir == path\_make\_dir + dirs:

shutil.rmtree(current\_face\_dir)

print("刪除舊的資料夾:", current\_face\_dir)

os.makedirs(current\_face\_dir)

print("新建的人臉資料夾: ", current\_face\_dir)

# 將人臉計數器清零

cnt\_ss = 0

if len(faces) != 0:

# 檢測到人臉

# 矩形框

for k, d in enumerate(faces):

# 計算矩形大小

# (x,y), (寬度width, 高度height)

pos\_start = tuple([d.left(), d.top()])

pos\_end = tuple([d.right(), d.bottom()])

# 計算矩形框大小

height = (d.bottom() - d.top())

width = (d.right() - d.left())

hh = int(height/2)

ww = int(width/2)

# 設置顏色 / The color of rectangle of faces detected

color\_rectangle = (255, 255, 255)

if (d.right()+ww) > 640 or (d.bottom()+hh > 480) or (d.left()-ww < 0) or (d.top()-hh < 0):

cv2.putText(img\_rd, "OUT OF RANGE", (20, 300), font, 0.8, (0, 0, 255), 1, cv2.LINE\_AA)

color\_rectangle = (0, 0, 255)

save\_flag = 0

else:

color\_rectangle = (255, 255, 255)

save\_flag = 1

cv2.rectangle(img\_rd,

tuple([d.left() - ww, d.top() - hh]),

tuple([d.right() + ww, d.bottom() + hh]),

color\_rectangle, 2)

# 根據人臉大小生成空的圖像

im\_blank = np.zeros((int(height\*2), width\*2, 3), np.uint8)

if save\_flag:

# 按下 's' 保存攝像頭中的人臉到本地

if kk == ord('s'):

cnt\_ss += 1

for ii in range(height\*2):

for jj in range(width\*2):

im\_blank[ii][jj] = img\_rd[d.top()-hh + ii][d.left()-ww + jj]

cv2.imwrite(current\_face\_dir + "/img\_face\_" + str(cnt\_ss) + ".jpg", im\_blank)

print("寫入本地：", str(current\_face\_dir) + "/img\_face\_" + str(cnt\_ss) + ".jpg")

# 顯示人臉數

cv2.putText(img\_rd, "Faces: " + str(len(faces)), (20, 100), font, 0.8, (0, 255, 0), 1, cv2.LINE\_AA)

# 添加說明

cv2.putText(img\_rd, "Face Register", (20, 40), font, 1, (0, 0, 0), 1, cv2.LINE\_AA)

cv2.putText(img\_rd, "N: New face folder", (20, 350), font, 0.8, (0, 0, 0), 1, cv2.LINE\_AA)

cv2.putText(img\_rd, "S: Save face", (20, 400), font, 0.8, (0, 0, 0), 1, cv2.LINE\_AA)

cv2.putText(img\_rd, "Q: Quit", (20, 450), font, 0.8, (0, 0, 0), 1, cv2.LINE\_AA)

# 按下 'q' 鍵退出

if kk == ord('q'):

break

# 視窗顯示

# cv2.namedWindow("camera", 0) # 如果需要攝像頭視窗大小可調

cv2.imshow("camera", img\_rd)

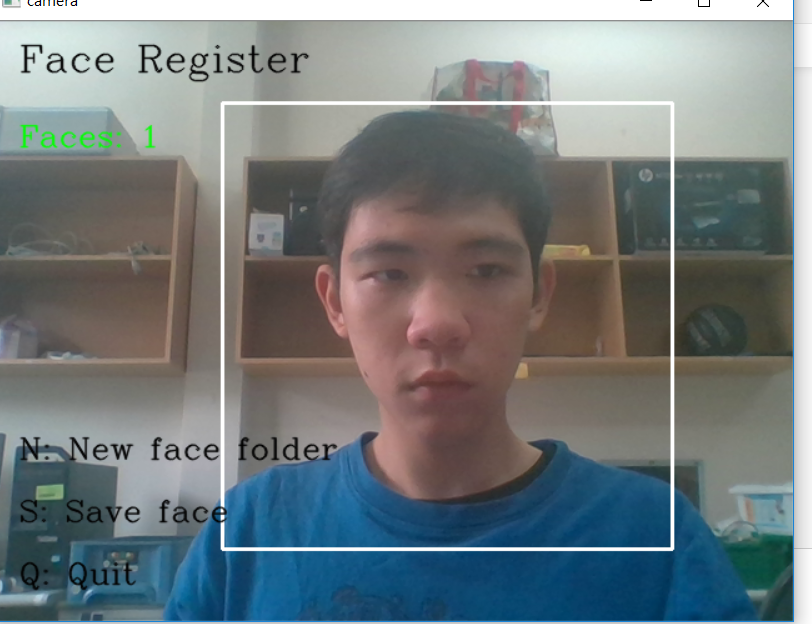
# 釋放攝像頭

cap.release()

# 刪除建立的視窗

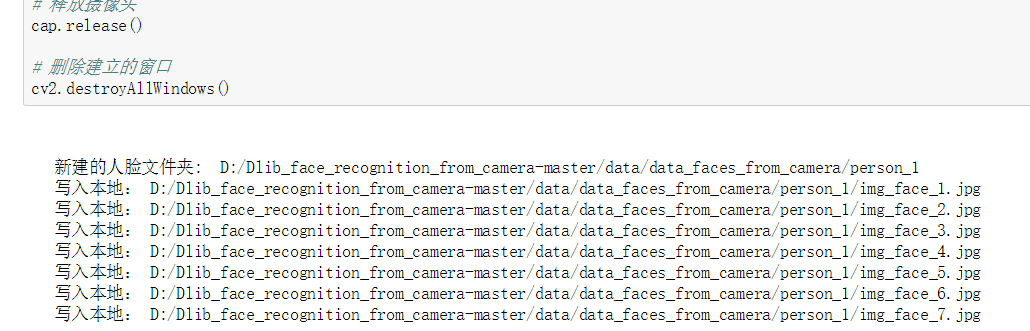
cv2.destroyAllWindows()

2.1 點選執行，會跳出相機視窗



2.2按N先建立一個新的資料夾

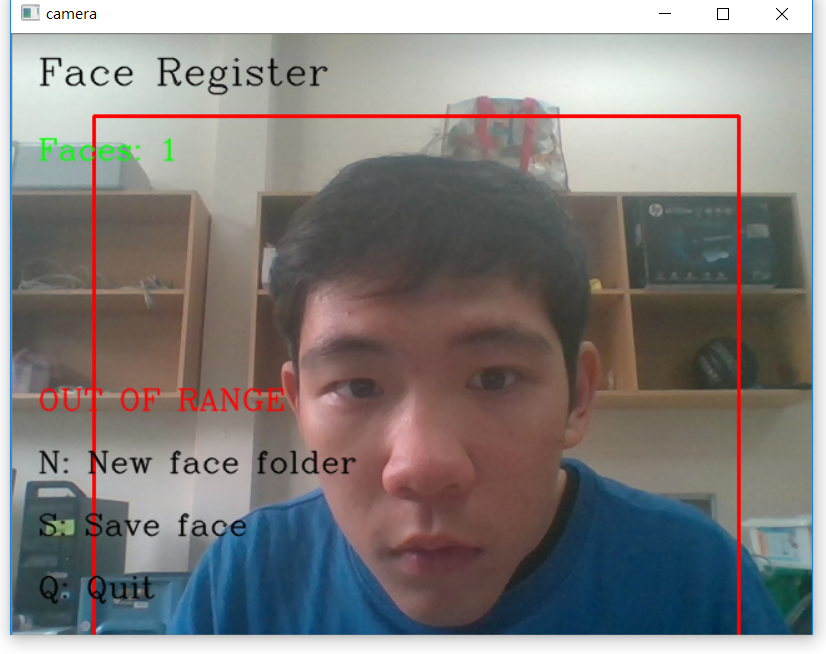


2.3按S 擷取臉部畫面

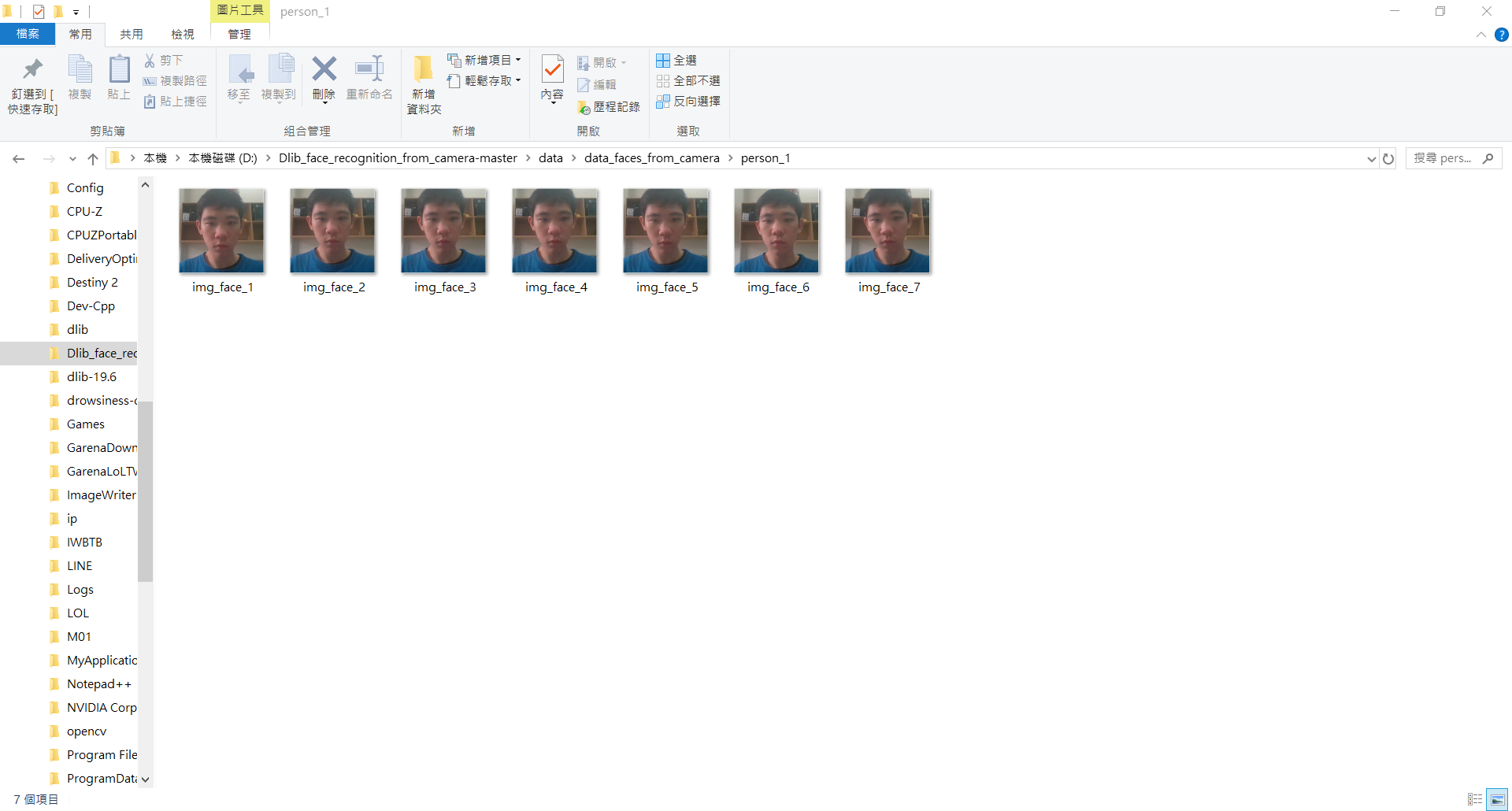
若換下一個人 重複以上步驟

結束後按Q可以結束視窗

若臉部超過範圍 會顯示out of range 則調整座的距離即可



2.4 去資料夾看是否建立成功



(3) 計算人臉特徵，請先複製以下程式碼

import cv2

import os

import dlib

from skimage import io

import csv

import numpy as np

import pandas as pd

path\_faces\_rd = "D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_faces\_from\_camera/"

path\_csv = "D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_csvs\_from\_camera/"

# Dlib 正向人臉檢測器

detector = dlib.get\_frontal\_face\_detector()

# Dlib 人臉預測器

predictor = dlib.shape\_predictor("D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_dlib/shape\_predictor\_68\_face\_landmarks.dat")

# Dlib 人臉識別模型

# Face recognition model, the object maps human faces into 128D vectors

facerec = dlib.face\_recognition\_model\_v1("D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_dlib/dlib\_face\_recognition\_resnet\_model\_v1.dat")

# 返回單張圖像的 128D 特徵

def return\_128d\_features(path\_img):

img = io.imread(path\_img)

img\_gray = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

faces = detector(img\_gray, 1)

print("檢測到人臉的圖像：", path\_img, "\n")

# 因為有可能截下來的人臉再去檢測，檢測不出來人臉了

# 所以要確保是 檢測到人臉的人臉圖像 拿去算特徵

if len(faces) != 0:

shape = predictor(img\_gray, faces[0])

face\_descriptor = facerec.compute\_face\_descriptor(img\_gray, shape)

else:

face\_descriptor = 0

print("no face")

# print(face\_descriptor)

return face\_descriptor

# 將資料夾中照片特徵提取出來，寫入 CSV

# path\_faces\_personX: 影像檔夾的路徑

# path\_csv: 要生成的 CSV 路徑

def write\_into\_csv(path\_faces\_personX, path\_csv):

dir\_pics = os.listdir(path\_faces\_personX)

with open(path\_csv, "w", newline="") as csvfile:

writer = csv.writer(csvfile)

for i in range(len(dir\_pics)):

# 調用return\_128d\_features()得到128d特徵

print("正在讀的人臉圖像：", path\_faces\_personX + "/" + dir\_pics[i])

features\_128d = return\_128d\_features(path\_faces\_personX + "/" + dir\_pics[i])

# print(features\_128d)

# 遇到沒有檢測出人臉的圖片跳過

if features\_128d == 0:

i += 1

else:

writer.writerow(features\_128d)

# 讀取某人所有的人臉圖像的資料，寫入 person\_X.csv

faces = os.listdir(path\_faces\_rd)

for person in faces:

print("##### " + person + " #####")

print(path\_csv + person + ".csv")

write\_into\_csv(path\_faces\_rd + person, path\_csv + person + ".csv")

# 從 CSV 中讀取資料，計算 128D 特徵的均值

def compute\_the\_mean(path\_csv\_rd):

column\_names = []

# 128列特徵

for feature\_num in range(128):

column\_names.append("features\_" + str(feature\_num + 1))

# 利用pandas讀取csv

rd = pd.read\_csv(path\_csv\_rd, names=column\_names)

# 存放128維特徵的均值

feature\_mean = []

for feature\_num in range(128):

tmp\_arr = rd["features\_" + str(feature\_num + 1)]

tmp\_arr = np.array(tmp\_arr)

# 計算某一個特徵的均值

tmp\_mean = np.mean(tmp\_arr)

feature\_mean.append(tmp\_mean)

return feature\_mean

# 存放所有特徵均值的 CSV 的路徑

path\_csv\_feature\_all = "D:/Dlib\_face\_recognition\_from\_camera-master/data/features\_all.csv"

# 存放人臉特徵的 CSV 的路徑

path\_csv\_rd = "D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_csvs\_from\_camera/"

with open(path\_csv\_feature\_all, "w", newline="") as csvfile:

writer = csv.writer(csvfile)

csv\_rd = os.listdir(path\_csv\_rd)

print("##### 得到的特徵均值 / The generated average values of features stored in: #####")

for i in range(len(csv\_rd)):

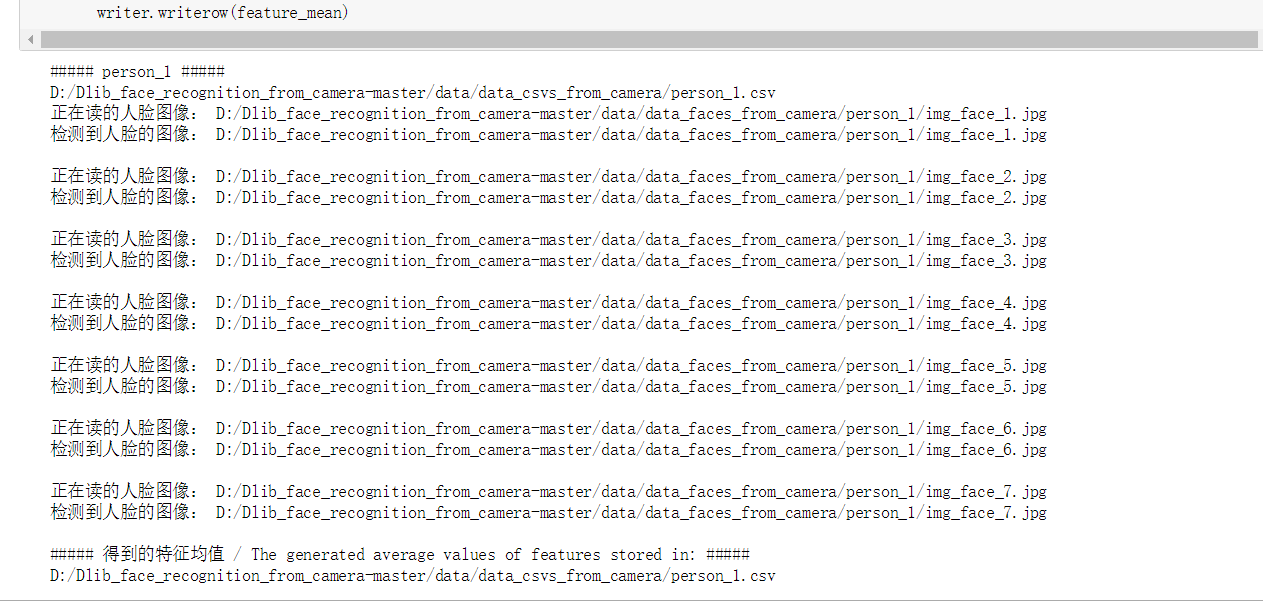
feature\_mean = compute\_the\_mean(path\_csv\_rd + csv\_rd[i])

# print(feature\_mean)

print(path\_csv\_rd + csv\_rd[i])

writer.writerow(feature\_mean)

3.2按執行便會開始計算 臉部68個特徵點



(4) 實行人臉辨識，請先複製以下程式碼，執行完成後，便會跳處攝像頭視窗 即可辨識

import dlib # 人臉處理的庫 Dlib

import numpy as np # 資料處理的庫 numpy

import cv2 # 影像處理的庫 OpenCv

import pandas as pd # 資料處理的庫 Pandas

import pygame

# 人臉識別模型，提取 128D 的特徵向量

# face recognition model, the object maps human faces into 128D vectors

facerec = dlib.face\_recognition\_model\_v1("D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_dlib/dlib\_face\_recognition\_resnet\_model\_v1.dat")

def start\_sound():

pygame.mixer.init()

pygame.mixer.music.load("z.ogg")

pygame.mixer.music.play()

# 計算兩個向量間的歐式距離

def return\_euclidean\_distance(feature\_1, feature\_2):

feature\_1 = np.array(feature\_1)

feature\_2 = np.array(feature\_2)

dist = np.sqrt(np.sum(np.square(feature\_1 - feature\_2)))

print("e\_distance: ", dist)

if dist > 0.4:

return "diff"

else:

return "same"

# 處理存放所有人臉特徵的 CSV

path\_features\_known\_csv = "D:/Dlib\_face\_recognition\_from\_camera-master/data/features\_all.csv"

csv\_rd = pd.read\_csv(path\_features\_known\_csv, header=None)

# 存儲的特徵人臉個數

# print(csv\_rd.shape[0])

# 用來存放所有錄入人臉特徵的陣列

features\_known\_arr = []

# 讀取已知人臉資料

# known faces

for i in range(csv\_rd.shape[0]):

features\_someone\_arr = []

for j in range(0, len(csv\_rd.ix[i, :])):

features\_someone\_arr.append(csv\_rd.ix[i, :][j])

# print(features\_someone\_arr)

features\_known\_arr.append(features\_someone\_arr)

print("Faces in Database：", len(features\_known\_arr))

# Dlib 檢測器和預測器

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor('D:/Dlib\_face\_recognition\_from\_camera-master/data/data\_dlib/shape\_predictor\_68\_face\_landmarks.dat')

# 創建 cv2 攝像頭對象

cap = cv2.VideoCapture(0)

# cap.set(propId, value)

# 設置視頻參數，propId 設置的視頻參數，value 設置的參數值

cap.set(3, 480)

# 返回一張圖像多張人臉的 128D 特徵

def get\_128d\_features(img\_gray):

faces = detector(img\_gray, 1)

if len(faces) != 0:

face\_des = []

for i in range(len(faces)):

shape = predictor(img\_gray, faces[i])

face\_des.append(facerec.compute\_face\_descriptor(img\_gray, shape))

else:

face\_des = []

return face\_des

# cap.isOpened() 返回 true/false 檢查初始化是否成功

while cap.isOpened():

flag, img\_rd = cap.read()

kk = cv2.waitKey(1)

# 取灰度

img\_gray = cv2.cvtColor(img\_rd, cv2.COLOR\_RGB2GRAY)

# 人臉數 faces

faces = detector(img\_gray, 0)

# 待會要寫的字體

font = cv2.FONT\_HERSHEY\_COMPLEX

cv2.putText(img\_rd, "Press 'q': Quit", (20, 450), font, 0.8, (84, 255, 159), 1, cv2.LINE\_AA)

# 存儲人臉名字和位置的兩個 list

# 存儲所有人臉的名字

pos\_namelist = []

name\_namelist = []

# 按下 q 鍵退出

if kk == ord('q'):

break

else:

# 檢測到人臉

if len(faces) != 0:

# 獲取當前捕獲到的圖像的所有人臉的特徵，存儲到 features\_cap\_arr

features\_cap\_arr = []

for i in range(len(faces)):

shape = predictor(img\_rd, faces[i])

features\_cap\_arr.append(facerec.compute\_face\_descriptor(img\_rd, shape))

# 遍歷捕獲到的圖像中所有的人臉

for k in range(len(faces)):

# 讓人名跟隨在矩形框的下方

# 確定人名的位置座標

# 先預設所有人不認識，是 unknown

name\_namelist.append("unknown")

# 每個捕獲人臉的名字座標

pos\_namelist.append(tuple([faces[k].left(), int(faces[k].bottom() + (faces[k].bottom() - faces[k].top()) / 4)]))

# 對於某張人臉，遍歷所有存儲的人臉特徵

for i in range(len(features\_known\_arr)):

print("with person\_", str(i+1), "the ", end='')

# 將某張人臉與存儲的所有人臉資料進行比對

compare = return\_euclidean\_distance(features\_cap\_arr[k], features\_known\_arr[i])

if compare == "same": # 找到了相似臉

name\_namelist[k] = "person\_" + str(i+1)

# 矩形框

for kk, d in enumerate(faces):

# 繪製矩形框

cv2.rectangle(img\_rd, tuple([d.left(), d.top()]), tuple([d.right(), d.bottom()]), (0, 255, 255), 2)

#if name\_namelist[k]=="unknown":

# pygame.mixer.init()

#track=pygame.mixer.music.load("D:/Dlib\_face\_recognition\_from\_camera-master/z.ogg") #可以播放.mp3 .wav等多種格式的音訊檔

#pygame.mixer.music.play()

# 在人臉框下面寫人臉名字

for i in range(len(faces)):

cv2.putText(img\_rd, name\_namelist[i], pos\_namelist[i], font, 0.8, (0, 255, 255), 1, cv2.LINE\_AA)

print("Name list now:", name\_namelist, "\n")

cv2.putText(img\_rd, "Face Recognition", (20, 40), font, 1, (0, 0, 0), 1, cv2.LINE\_AA)

cv2.putText(img\_rd, "Faces: " + str(len(faces)), (20, 100), font, 1, (0, 0, 255), 1, cv2.LINE\_AA)

# 視窗顯示

cv2.imshow("camera", img\_rd)

# 釋放攝像頭

cap.release()

# 刪除建立的視窗

cv2.destroyAllWindows()

執行結果

