การเขียนโปรแกรมวิเคราะห์ใบหน้า ด้วย OpenCV

Face Recognition

ระบบการรู้จำใบหน้าหรือระบบการจดจำใบหน้า คือ ระบบการตรวจหาใบหน้าของมนุษย์และปรับภาพใบหน้าโดยอัตโนมัติ กรอบจะปรากฏขึ้นบนใบหน้าที่ถูกตรวจจับ และโฟกัส สี และค่าการวัดแสงจะถูกปรับโดยอัตโนมัติ นอกจากนั้นเมื่อบันทึกด้วยคุณภาพแบบ HD เทคโนโลยีการบีบอัดจะจัดสรรความจุของข้อมูลให้ลดลง แต่ได้ข้อมูลที่เป็นประโยชน์มากขึ้นเพื่อปรับคุณภาพของภาพ ข้อมูลที่ได้จะถูกนำไปเปรียบเทียบกับข้อมูลตัวอย่างที่เก็บบันทึกไว้ อาจจะทั้งใบหน้าหรือเพียงบางส่วน ขึ้นอยู่กับชนิดของวิธีแยกเอกลักษณ์ใบหน้า ระบบการรู้จำใบหน้าเป็นส่วนหนึ่งของเทคโนโลยีปัญญาประดิษฐ์ ในส่วนเนื้อหาของเรื่อง การรับรู้ของเครื่อง (Machine perception)

ระบบวิเคราะห์ใบหน้าถือว่าเป็นหนึ่งในระบบที่ใช้ในการพิสูจน์ยืนยันตัวตนบุคคล โดยใช้คุณลักษณะจำเพาะทางสรีระ (Biometric) โดยระบบรู้จำใบหน้าจะทำงานโดยการเปรียบเทียบใบหน้าจากภาพถ่ายดิจิตอล หรือภาพจากกล้องวิดีโอของบุคคลที่เราสนใจกับฐานข้อมูลใบหน้าที่มีอยู่ และเมื่อเปรียบเทียบเสร็จก็จะแสดงผลใบหน้าที่อยู่ในฐานข้อมูล ที่มีใบหน้าเหมือนกับภาพที่นำมาเปรียบเทียบออกมา ระบบรู้จำใบหน้านั้นได้ถูกพัฒนามาอย่างต่อเนื่องเป็นเวลามากกว่าสิบปีมาแล้ว

เทคโนโลยีการเรียนรู้จดจำใบหน้า (Face Recognition) คือ เทคโนโลยีที่ถูกสร้างขึ้นมาเพื่อเรียนรู้และจดจำโครงสร้างใบหน้าของมนุษย์ แล้วนำข้อมูลใบหน้าที่จดจำหรือตรวจจับได้ส่งไปให้ระบบ เพื่อนำไปใช้วิเคราะห์หรือประมวลผลในการทำงานในส่วนขั้นตอนอื่น ๆ อีกต่อไป ซึ่งเทคโนโลยีที่นำระบบการเรียนรู้จดจำใบหน้า ไปใช้งานมากที่สุดคือ เทคโนโลยีที่เกี่ยวข้องกับระบบความปลอดภัย ไม่ว่าจะเป็นระบบ Access Control ระบบกล้องวงจรปิด หรือ ระบบรักษาความปลอดภัยในมือถือ เป็นระบบที่ช่วยเพิ่มประสิทธิภาพการระบุและยืนยันอัตลักษณ์บุคคลด้วยความรวดเร็ว แม่นยำ และสามารถเพิ่มโอกาสในการต่อยอดความสำเร็จให้หลากหลายแวดวงธุรกิจได้ด้วย

หลักการทำงานของ Face Recognition คือ การสร้างโมเดลการอ้างอิง ที่เรียกว่า “faceprint” ขึ้นมา โดยระบบจะวิเคราะห์จากลักษณะเฉพาะต่าง ๆ บนใบหน้า เช่น โครงหน้า ความกว้างของจมูก ระยะห่างระหว่างตาทั้งสองข้าง ขนาดของโหนกแก้ม ความลึกของเบ้าตา รวมถึงพื้นผิวบนใบหน้า (facial texture) เป็นต้น จากนั้น ระบบจะทำการสร้างจุดเชื่อมโยงบนใบหน้า (nodal points) เพื่อเปรียบเทียบกับรูปภาพที่ถูกเก็บไว้ในฐานข้อมูล (data base) ทั้งในลักษณะภาพนิ่งและภาพเคลื่อนไหว เพื่อความแม่นยำในการระบุตัวตนของผู้ที่ต้องเข้าสู่กระบวนการตรวจสอบ

ดังนั้น หลักการทำงานของระบบรู้จำใบหน้า ที่ถูกออกแบบมาให้ทำการเปรียบเทียบใบหน้าบุคคลที่เราต้องการจะตรวจสอบ กับฐานข้อมูลใบหน้าที่มีอยู่ โดยอัลกอริทึมที่ใช้ในขั้นตอนการสร้างแม่แบบและขั้นตอนการเปรียบเทียบอาจมีความแตกต่างกันไป แล้วแต่วิธีการออกแบบระบบของแต่ละระบบ แต่ไม่ว่าจะมีอัลกอริทึมในการทำงานในขั้นตอนการสร้างแม่แบบและขั้นตอนการเปรียบเทียบยังไง แต่ขั้นตอนการทำงานโดยรวมของระบบก็ยังคงเหมือนกันอยู่

ขั้นตอนในการทำ Face Recognition

1. การตรวจจับใบหน้า (Face Detection) คือ กระบวนการค้นหาใบหน้าของบุคคลจากภาพหรือวิดีโอ หลังจากนั้นก็จะทำการประมวลผลภาพใบหน้าที่ได้สำหรับขั้นตอนถัดไป เพื่อให้ภาพใบหน้าที่ตรวจจับได้ง่ายต่อการจำแนก และ อัลกอริทึมที่ใช้ในการตรวจจับใบหน้าในปัจจุบันก็มีอยู่ด้วยกันหลายวิธี ซึ่งอัลกอริทึมในการตรวจจับใบหน้าที่ดีนั้น มีส่วนช่วยในการจำแนกใบหน้าได้แม่นยำและรวดเร็วขึ้นเป็นอย่างมาก

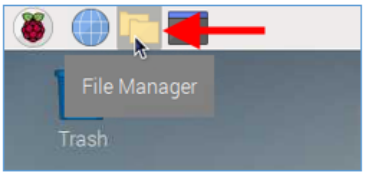
2. การรู้จดจำใบหน้า (Face Recognition) คือ กระบวนการที่ได้นำภาพใบหน้าที่ตรวจจับได้ และประมวลผลแล้วจากขั้นตอนการตรวจจับใบหน้า มาเปรียบเทียบกับฐานข้อมูลของใบหน้าเพื่อระบุว่าใบหน้าที่ตรวจจับได้ตรงกับบุคคลใด แล้วจึงนำผลลัพธ์ที่ได้ ส่งไปให้ระบบหรือโปรแกรมเพื่อประมวลผลอื่นๆ ต่อไป

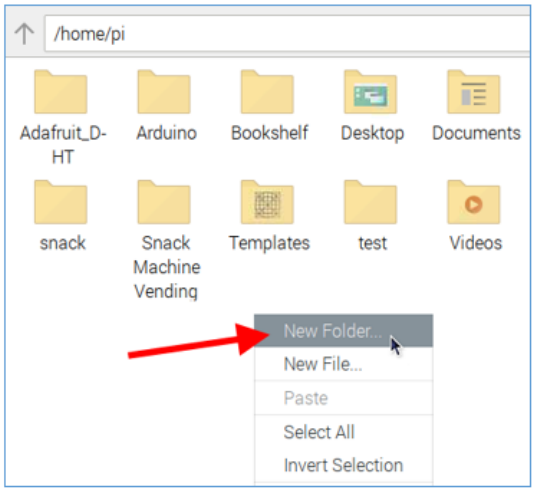
OpenCV (Open source Computer Vision) คือ ไลบรารีฟังก์ชันการเขียนโปรแกรม (Library of Programming Functions) โดยส่วนใหญ่จะมุ่งเป้าไปที่การแสดงผลด้วยคอมพิวเตอร์แบบเรียลไทม์ (Real-Time Computer Vision) เดิมทีแล้วถูกพัฒนาโดย Intel แต่ภายหลังได้รับการสนับสนุนโดย Willow Garage ตามมาด้วย Itseez (ซึ่งต่อมาถูกเข้าซื้อโดย Intel) OpenCV เป็นไลบรารีแบบข้ามแพลตฟอร์ม (Cross-Platform) และใช้งานได้ฟรีภายใต้ลิขสิทธิ์ของ BSD แบบโอเพ่นซอร์ส (Open Source BSD License) OpenCV ยังสนับสนุนเฟรมเวิร์กการเรียนรู้เชิงลึก (Deep Learning Frameworks) ได้แก่ TensorFlow, Torch/PyTorch และ Caffe

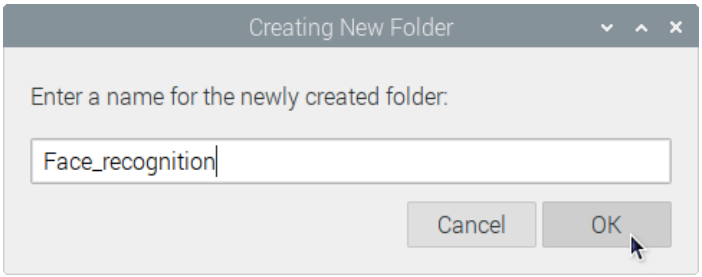
Haar Cascades เป็น Object Detection Algorithm ใช้ในการตรวจจับวัตถุแบบ real-time อ้างอิงจากงานวิจัยของ Paul Viola และ Michael Jones ชื่อ “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001
โดยหลักการทำงานของ Haar Cascades จะอ่านข้อมูล pixel ของรูป และนำมาเปรียบเทียบกับ feature ที่มีอยู่บนใบหน้าซึ่ง Haar Cascades สามารถปรับแต่งเพื่อตรวจจับวัตถุได้หลายอย่าง และเก็บข้อมูลในการตรวจ และโปรแกรมที่ถูกปรับแต่งเพื่อค้นหาวัตถุต่าง ๆ เก็บไว้ในรูปของไฟล์ xml ซึ่งหากเราต้องการตรวจจับใบหน้า เราต้องนำไฟล์ cascades ที่ปรับแต่งสำหรับการจับใบหน้านั่นเอง

1. ดาวน์โหลดไฟล์ haarcascade_frontalface_default.xml

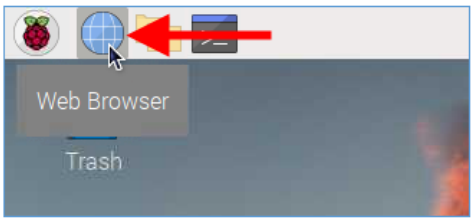
1.1 สร้างโฟลเดอร์ Face_recognition ไว้ในโฟลเดอร์ /home/pi โดยคลิก File Manager แล้วคลิกขวาตรงพื้นที่ว่างเลือก New Folder… ตั้งชื่อโฟลเดอร์ว่า “Face_recognition”

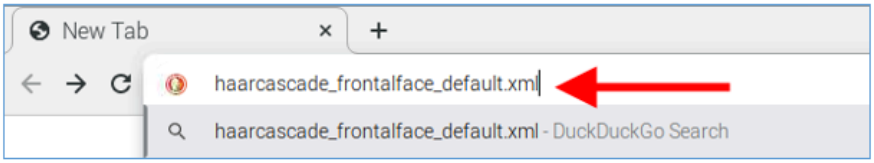




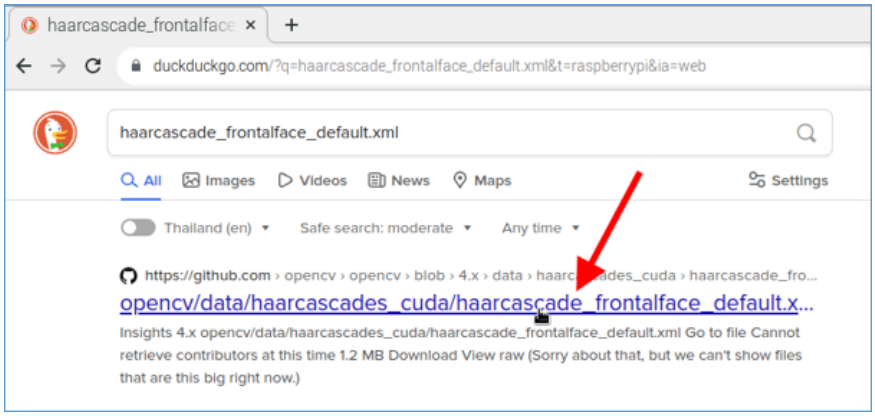


1.2 คลิกเปิด Web Browser พิมพ์ข้อความค้นหาว่า “haarcascade_frontalface_default.xml” จากนั้นกด Enter

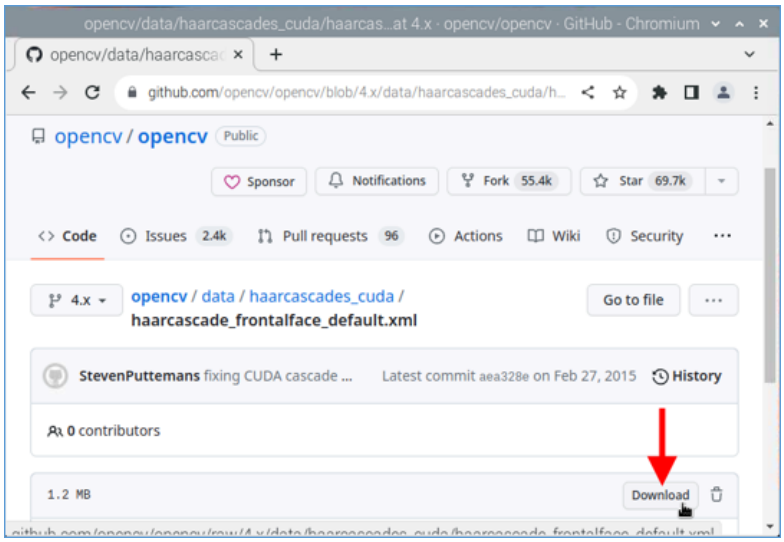




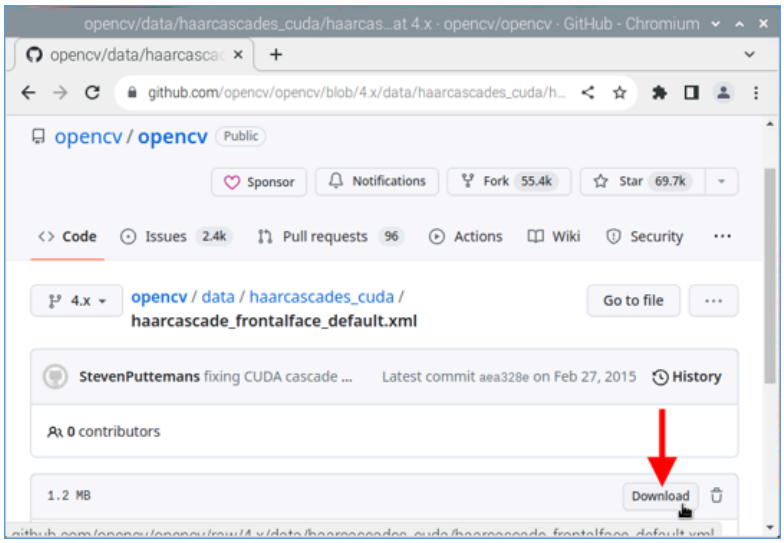
1.3 คลิกเปิดเว็บไซต์ github.com ดังรูป



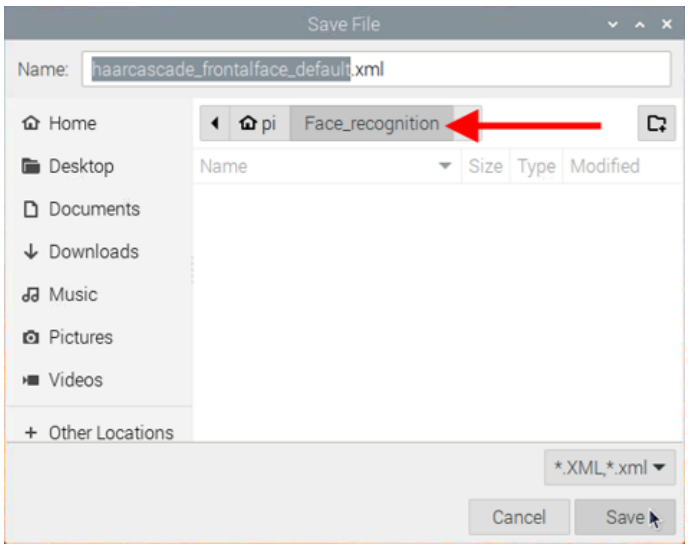
1.4 คลิกปุ่ม Download



1.5 คลิกจุด 3 จุดด้านบนขวาของเพจ จากนั้นเลือกเมนู More Tools --> Save page as …



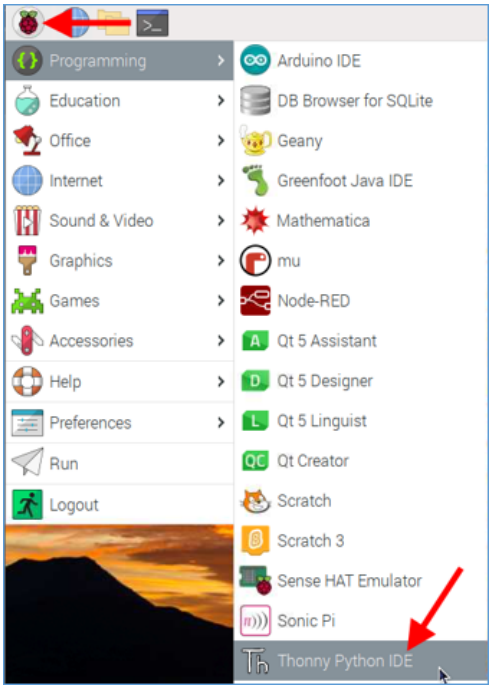
1.6 บันทึกไฟล์ไว้ในโฟลเดอร์ Face_recognition จากนั้นคลิกปุ่ม Save



2. เขียนโปรแกรมสำหรับบันทึกใบหน้า

2.1 เสียบสายกล้องเว็บแคมเข้ากับพอร์ต USB ของบอร์ด Raspberry Pi

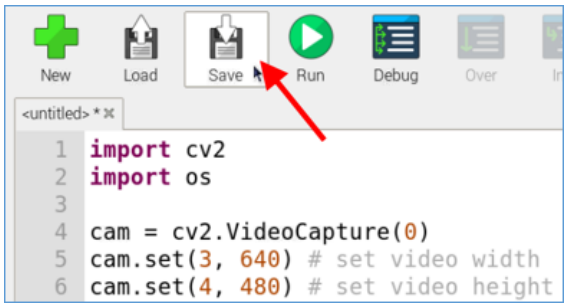
2.2 เปิดโปรแกรม Thonny Python IDE ขึ้นมาเพื่อเขียนโค้ดคำสั่งด้วยภาษา Python โดยคลิกที่สัญลักษณ์  เลือกเมนู Programming แล้วคลิกเลือกโปรแกรม Thonny Python IDE

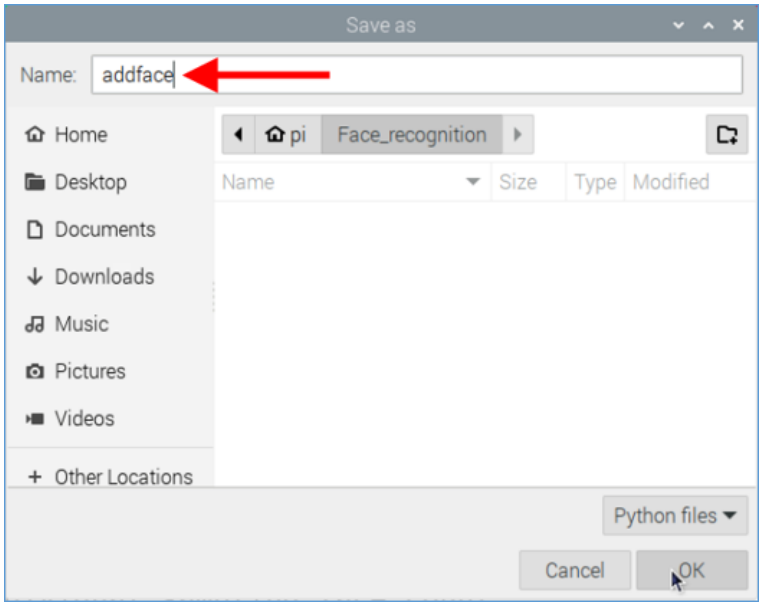


2.3 เขียนโค้ดคำสั่ง ดังนี้

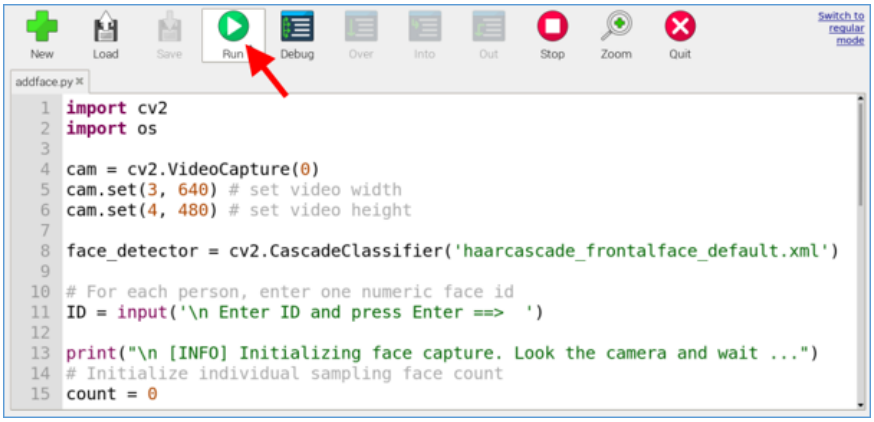
```python
import cv2 #นำเข้าไลบรารี cv2
import os  #นำเข้าโมดูล OS เพื่อทำงานกับระบบปฏิบัติการ

cam = cv2.VideoCapture(0) #object ที่ประกาศขึ้นมาเพื่อใช้ในการ Capture Video
cam.set(3, 640) #กำหนดความกว้างของหน้าจอกล้อง
cam.set(4, 480) #กำหนดความสูงของหน้าจอกล้อง

#เรียกใช้ไฟล์ cascade
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

#รับค่าตัวเลขทางคีย์บอร์ดใส่ในตัวแปร face_id
face_id = input('\n enter user id end press <return> ==>  ')

#แสดงข้อความให้มองกล้อง
print("\n [INFO] Initializing face capture. Look the camera and wait ...")

count = 0 #กำหนดตัวแปร count = 0 เพื่อใช้ในการวนลูป

while(True): #วนลูป
    ret, img = cam.read()  #เปิดภาพวิดีโอ
    img = cv2.flip(img, 1) #พลิกภาพวิดีโอ
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #ปรับภาพวิดีโอเป็นโหมดขาวดำ
    faces = face_detector.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
        #สร้างกรอบสี่เหลี่ยมล้อมรอบใบหน้า
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1 #ปรับค่าตัวแปร count เพิ่มขึ้นทีละ 1

        #บันทึกรูปภาพใบหน้าไว้ในโฟลเดอร์ dataset
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count)
                    + ".jpg", gray[y:y+h,x:x+w])

        cv2.imshow('image', img) #เปิดโชว์ภาพโดยมีการกำหนดหัวเรื่องและแสดงรูป

    k = cv2.waitKey(100) & 0xff #หากกด Esc หรือวนครบ 30 รอบให้ปิดกล้อง
    if k == 27:
        break
    elif count >= 30:
         break

print("\n [INFO] Exiting Program and cleanup stuff") #แสดงข้อความออกจากโปรแกรม
cam.release()
cv2.destroyAllWindows() #ปิดหน้าต่าง
```

2.4 บันทึกไฟล์ โดยคลิกที่เมนู File จากนั้นเลือกโฟลเดอร์ Face_recognition พิมพ์ชื่อไฟล์ว่า “addface” ในช่อง Name แล้วคลิกปุ่ม OK



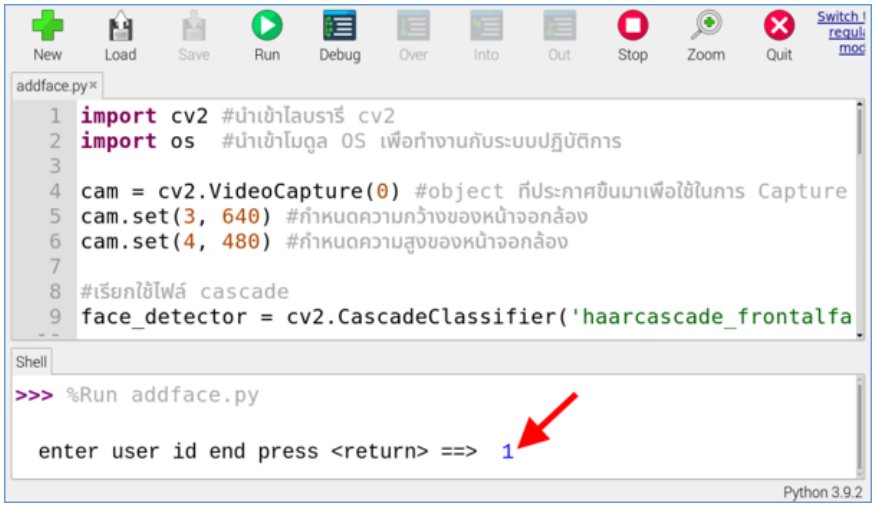


2.5 คลิก Run เพื่อรันโปรแกรม

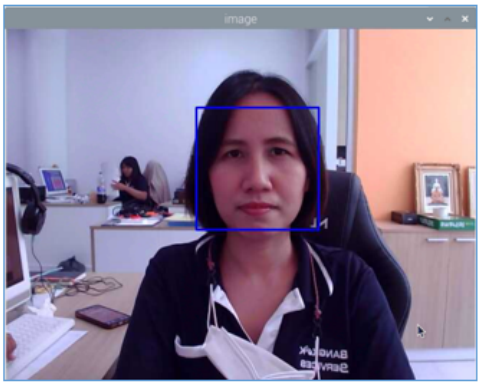


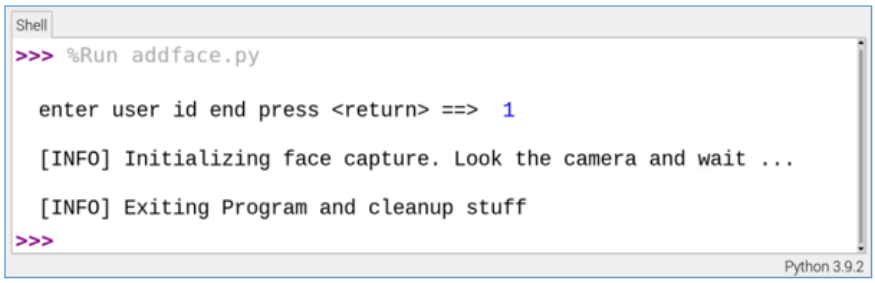
ผลลัพธ์ที่ได้

1. หน้าต่าง Shell จะปรากฏข้อความ “Enter ID and press Enter ==>” เพื่อรับชื่อจากคีย์บอร์ด

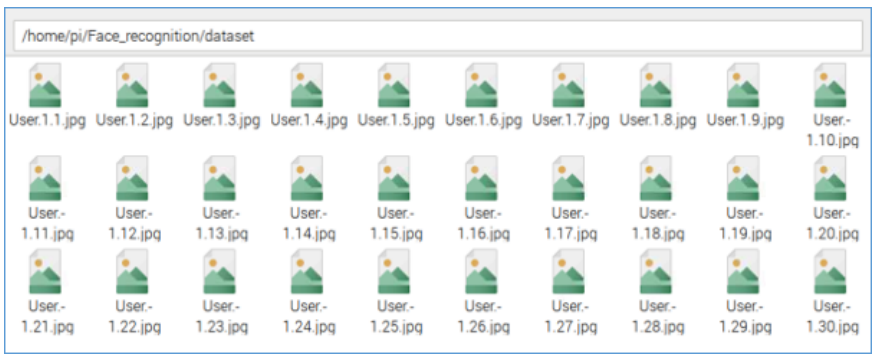


2. เมื่อพิมพ์ตัวเลข ID ลงไป กล้องเว็บแคมจะเปิดขึ้นมา โดยจะปรากฏกรอบสี่เหลี่ยมสีน้ำเงินรอบใบหน้า และจะทำการถ่ายรูปจำนวน 30 รูป





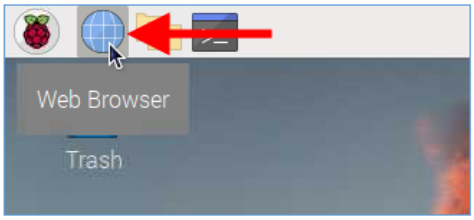
3. รูปภาพ 30 รูปจะถูกบันทึกไว้ในโฟลเดอร์ dataset โดยมีรูปแบบชื่อไฟล์รูปภาพ ดังรูป

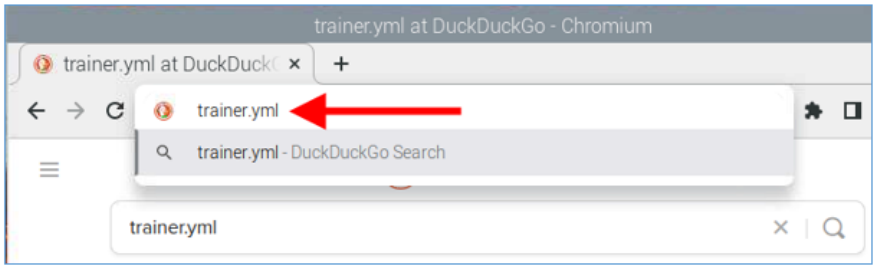


ขั้นตอนที่ 2 เรียนรู้ใบหน้า

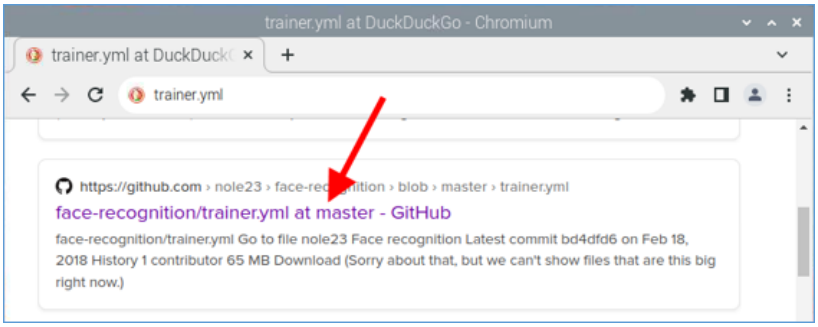
2.1 ดาวน์โหลดไฟล์ trainer.yml

1. คลิกเปิด Web Browser พิมพ์ข้อความค้นหาว่า “trainer.yml” จากนั้นกด Enter

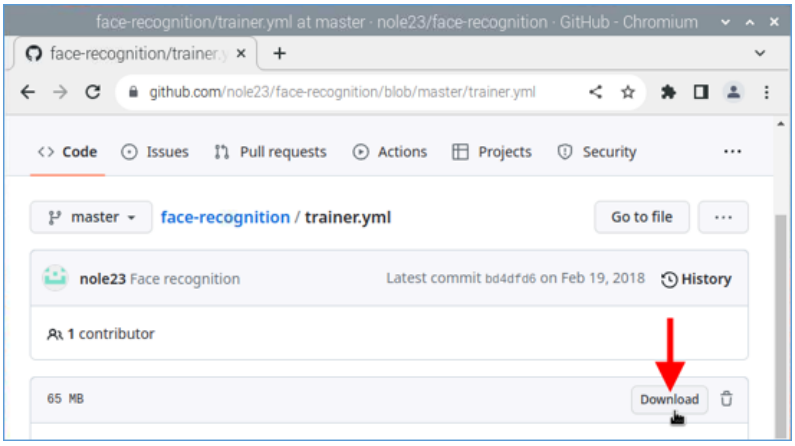




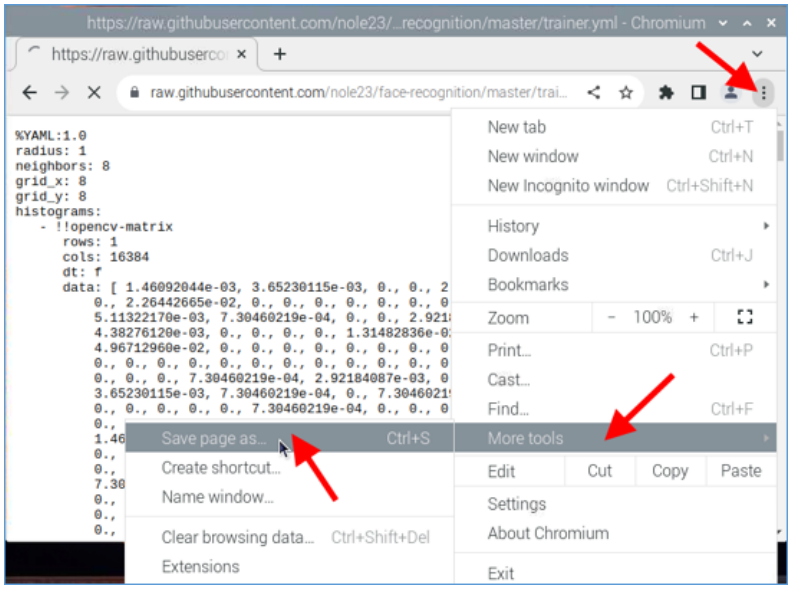
2. คลิกลิงก์เว็บไซต์ github.com ดังรูป



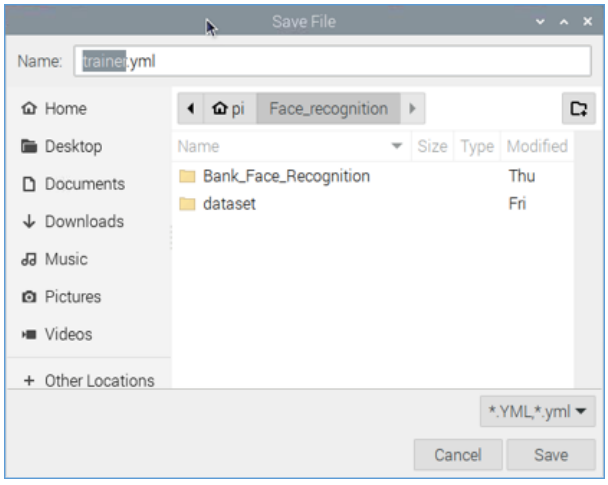
3. คลิกปุ่ม Download



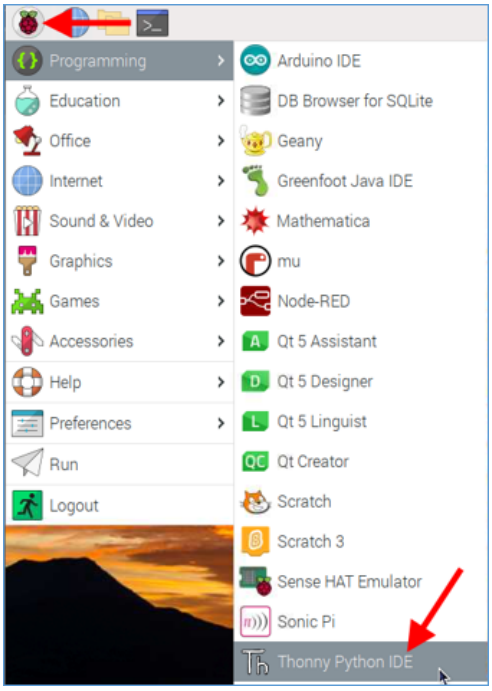
4. คลิกสัญลักษณ์ จุด 3 จุด จากนั้นคลิกคำสั่ง More tools --> Save page as…



5. บันทึกไฟล์ไว้ในโฟลเดอร์ Face_recognition จากนั้นคลิกปุ่ม Save



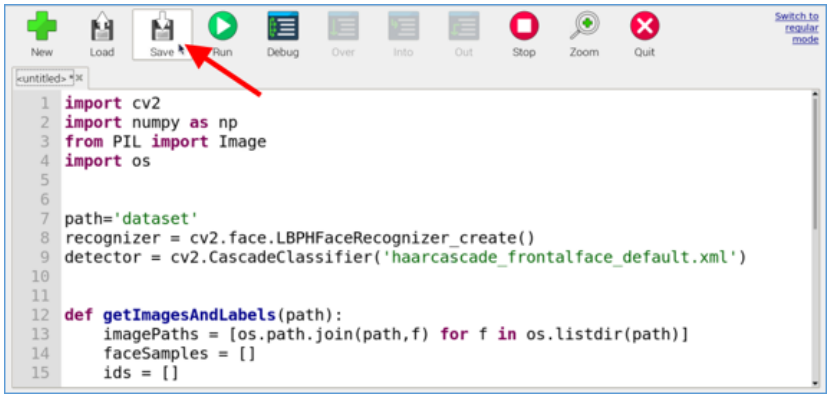
6. เปิดโปรแกรม Thonny Python IDE ขึ้นมาเพื่อเขียนโค้ดคำสั่งด้วยภาษา Python โดยคลิกที่สัญลักษณ์  เลือกเมนู Programming แล้วคลิกเลือกโปรแกรม Thonny Python IDE

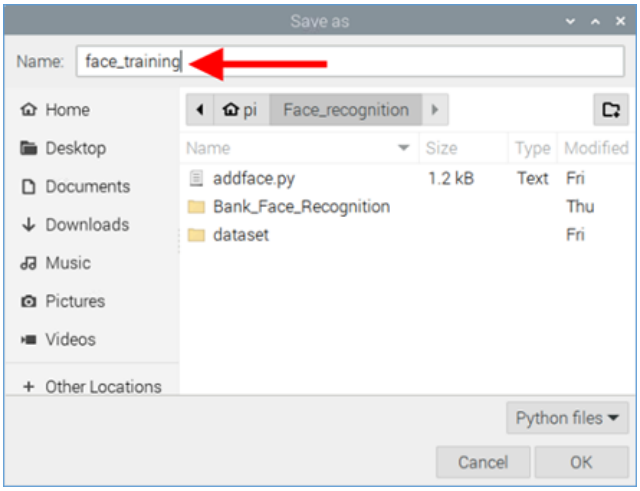


7. เขียนโค้ดคำสั่ง ดังนี้

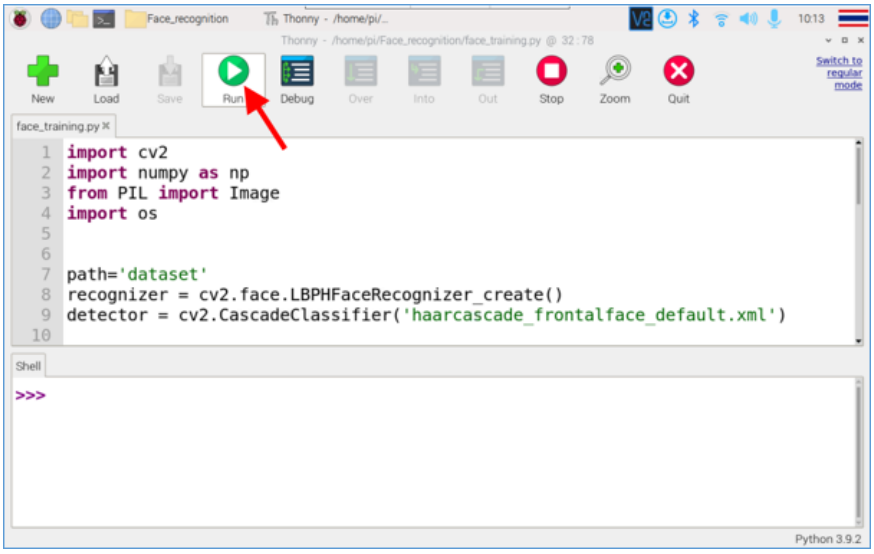
```python
import cv2
import numpy as np
from PIL import Image
import os


path='dataset'
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples = []
    ids = []

    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L')
        img_numpy = np.array(PIL_img,'uint8')
        id = int(os.path.split(imagePath)[-1].split('.')[1])
        faces = detector.detectMultiScale(img_numpy)

        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
    return faceSamples, ids

print('\n [INFO] Training faces It will take a few seconds, wait....')
faces,ids = getImagesAndLabels(path)
recognizer.train(faces,np.array(ids))
recognizer.save('trainer.yml')
print("\n [INFO] {0} Face trained. Exit Program".format(len(np.unique(ids))))
```

8. บันทึกไฟล์ โดยคลิกที่เมนู File จากนั้นเลือกโฟลเดอร์ Face_recognition พิมพ์ชื่อไฟล์ว่า “face_training” ในช่อง Name แล้วคลิกปุ่ม OK





9. คลิก Run เพื่อรันโปรแกรม

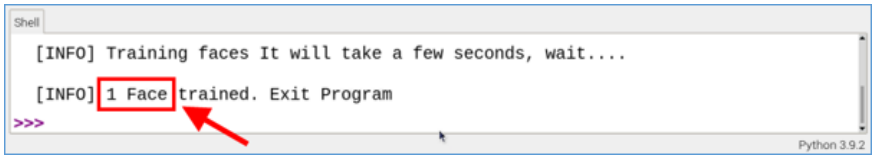


ผลลัพธ์ที่ได้

1. หน้าต่าง Shell จะปรากฏข้อความ “[INFO] Training faces It will take a few seconds, wait....” เพื่อแจ้งว่า กำลังเรียนรู้รูปภาพ

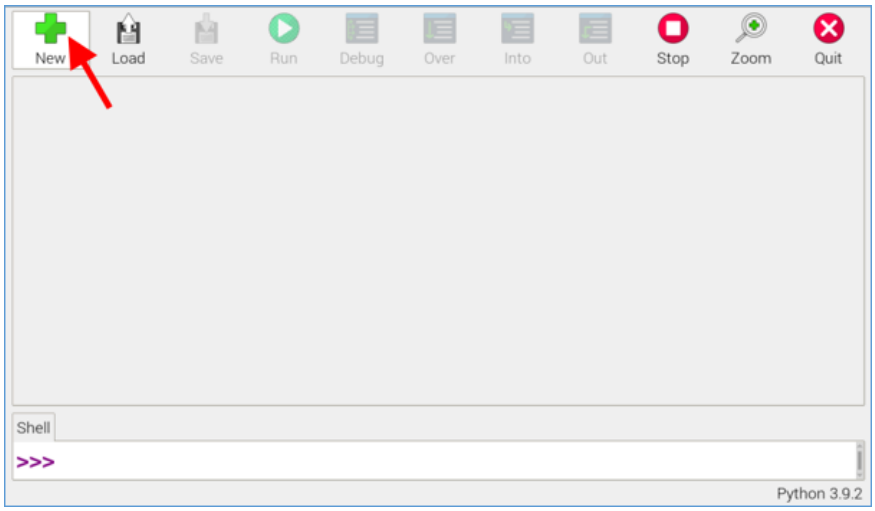
ใบหน้าอยู่ โปรดรอสักครู่

2. หน้าต่าง Shell จะปรากฏข้อความ “[INFO] [INFO] ตัวเลข Face trained. Exit Program” เพื่อบอกจำนวนรูปภาพใบหน้าทั้งหมดในโฟลเดอร์ dataset ที่ทำการเรียนรู้ (Train)



ขั้นตอนที่ 3 วิเคราะห์ใบหน้า

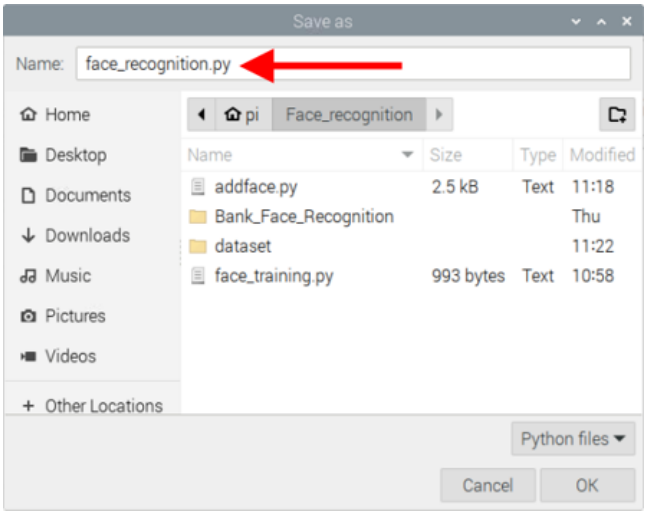
1. คลิก New เพื่อสร้างไฟล์ใหม่



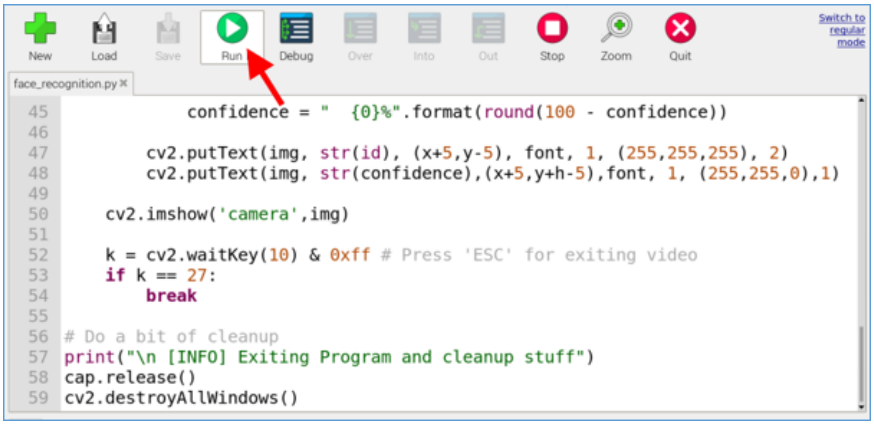
2. พิมพ์โค้ดคำสั่ง ดังนี้

```python
import cv2 #import library
import numpy as np
import os
from PIL import Image

if not os.path.isfile("trainer.yml"):
  print("Please train the data first")
  exit(0)

#iniciate id counter
id = 0
font = cv2.FONT_HERSHEY_SIMPLEX

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
cap.set(3, 640) # set video width
cap.set(4, 480) # set video height

# Define min window size to be recognized as a face
minW = 0.1*cap.get(3)
minH = 0.1*cap.get(4)

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("trainer.yml")

#กำหนดชื่อให้ตรงกับ face ID ของใบหน้า
names = ['None', 'vivy', 'NAME2', 'NAME3', 'NAME4', 'NAME5', 'NAME6']

while True:
    ret, img =cap.read()
    img = cv2.flip(img, 1) # Flip vertically
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
        id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

        # Check if confidence is less them 100 ==> "0" is perfect match
        if (confidence > 65):
            id = names[id]
            confidence = "  {0}%".format(round(100 - confidence))
        else:
            id = "unknown"
            confidence = "  {0}%".format(round(100 - confidence))

        cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
        cv2.putText(img, str(confidence),(x+5,y+h-5),font, 1, (255,255,0),1)

    cv2.imshow('camera',img)

    k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
    if k == 27:
        break

# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
cap.release()
cv2.destroyAllWindows()
```

3. บันทึกไฟล์ โดยคลิกที่เมนู File จากนั้นเลือกโฟลเดอร์ Face_recognition พิมพ์ชื่อไฟล์ว่า “face_recognition” ในช่อง Name แล้วคลิกปุ่ม OK

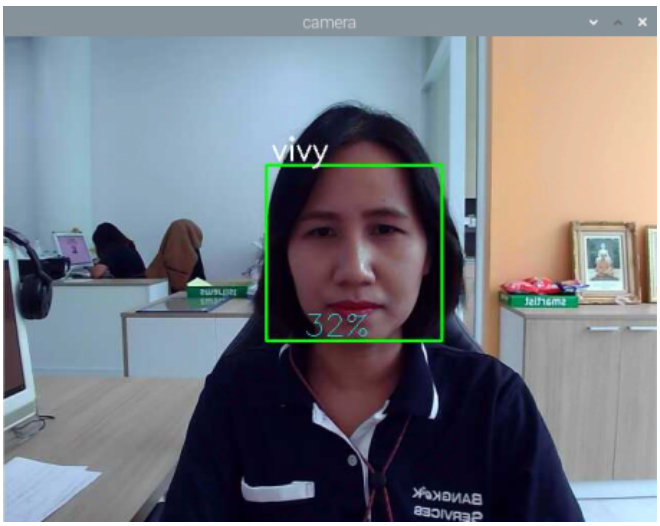


4. คลิก Run เพื่อรันโปรแกรม



ผลลัพธ์ที่ได้

จะปรากฏหน้าต่างกล้องขึ้นมา จากนั้นจะจับใบหน้าเพื่อวิเคราะห์ว่าใบหน้าตรงกับรูปภาพในโฟลเดอร์ dataset หรือไม่ โดยจะมีกรอบสี่เหลี่ยมสีเขียวขึ้นมาคลุมรอบใบหน้า หากวิเคราะห์ออกมาว่าใบหน้าตรงกับรูปภาพในโฟลเดอร์ dataset ก็จะปรากฏชื่อขึ้นมา



หากวิเคราะห์ออกมาว่าใบหน้าไม่ตรงกับรูปภาพในโฟลเดอร์ dataset จะปรากฏชื่อว่า unknownขึ้นมาตรงกรอบสี่เหลี่ยมสีเขียว