```
from zipfile import ZipFile
from IPython.display import display
from PIL import Image
import pytesseract
import cv2 as cv
import numpy as np
import math

# loading the face detection classifier
face_cascade = cv.CascadeClassifier('readonly/haarcascade_frontalface_default.xml')

# extracting data - file name, text and face image
def produce_data(file_name):

    myZip = ZipFile(file_name, mode='r')

    # getting file names, original images and opencv images from zip file
    filenames = []
    original_images = []
    cv_images = []

    for zFile in myZip.namelist():
        filenames.append(zFile)
        extFile = myZip.open(zFile)

        original_image = Image.open(extFile,'r')
        original_images.append(original_image)

        cv_image = np.asarray(original_image)
        cv_images.append(cv_image)

    # getting texts from original images
    texts = []

    for image in original_images:
        img = image.convert('L')
        text = pytesseract.image_to_string(img).strip().replace('-','').replace('\n','')
        texts.append(text)

    # getting positions of faces images from opencv images
    faces = []

    for cv_image in cv_images:
        cv_img_threshold=cv.threshold(cv_image,180,255,cv.THRESH_BINARY)[1]
        gray = cv.cvtColor(cv_img_threshold, cv.COLOR_BGR2GRAY)
        face_list = face_cascade.detectMultiScale(gray,1.25, 5, cv.CASCADE_SCALE_IMAGE,(5,5))
        faces.append(face_list)

    # getting faces images from original images
    faces_image = []
```

```
for i in range(len(original_images)):
        face_image_list = []
        for x,y,w,h in faces[i]:
                face_image = original_images[i].crop((x,y,x+w,y+h))
                if face_image.width > 100 or face_image.height > 100:
                        face_image.thumbnail((100,100))
                face_image_list.append(face_image)
        faces_image.append(face_image_list)

# pack file names, texts and faces images
return list(zip(filenames, texts, faces_image))


# searching name from texts, and showing related face images
def search_person(name, mydata):

        for data in mydata:
                if name in data[1]:
                        if len(data[2]) == 0:
                                print('\n')
                                print('Results found in file {}'.format(data[0]))
                                print('But there were no faces in that file!')
                                print('\n')
                        else:
                                print('Results found in file {}'.format(data[0]))
                                first_image = data[2][0]
                                contact_sheet = Image.new(first_image.mode, (5*100,math.ceil(len(data[2])/5)*100))
                                x = 0
                                y = 0
                                for img in data[2]:
                                        contact_sheet.paste(img, (x, y) )
                                        if x + 100 == contact_sheet.width:
                                                x = 0
                                                y += 100
                                        else:
                                                x += 100

                                display(contact_sheet)


person_name = 'Christopher'
file_name = 'readonly/small_img.zip'
test_data = produce_data(file_name)
search_person(person_name, test_data)

person_name = 'Mark'
file_name = 'readonly/images.zip'
test_data = produce_data(file_name)
search_person(person_name, test_data)
```