

# The Project

1. This is a project with minimal scaffolding. Expect to use the the discussion forums to gain insights! It's not cheating to ask others for opinions or perspectives!
2. Be inquisitive, try out new things.
3. Use the previous modules for insights into how to complete the functions! You'll have to combine Pillow, OpenCV, and Pytesseract
4. There are hints provided in Coursera, feel free to explore the hints if needed. Each hint provide progressively more details on how to solve the issue. This project is intended to be comprehensive and difficult if you do it without the hints.

## The Assignment

Take a [ZIP file \(https://en.wikipedia.org/wiki/Zip\\_\(file\\_format\)\)](https://en.wikipedia.org/wiki/Zip_(file_format)) of images and process them, using a [library built into python \(https://docs.python.org/3/library/zipfile.html\)](https://docs.python.org/3/library/zipfile.html) that you need to learn how to use. A ZIP file takes several different files and compresses them, thus saving space, into one single file. The files in the ZIP file we provide are newspaper images (like you saw in week 3). Your task is to write python code which allows one to search through the images looking for the occurrences of keywords and faces. E.g. if you search for "pizza" it will return a contact sheet of all of the faces which were located on the newspaper page which mentions "pizza". This will test your ability to learn a new ([library \(https://docs.python.org/3/library/zipfile.html\)](https://docs.python.org/3/library/zipfile.html)), your ability to use OpenCV to detect faces, your ability to use tesseract to do optical character recognition, and your ability to use PIL to composite images together into contact sheets.

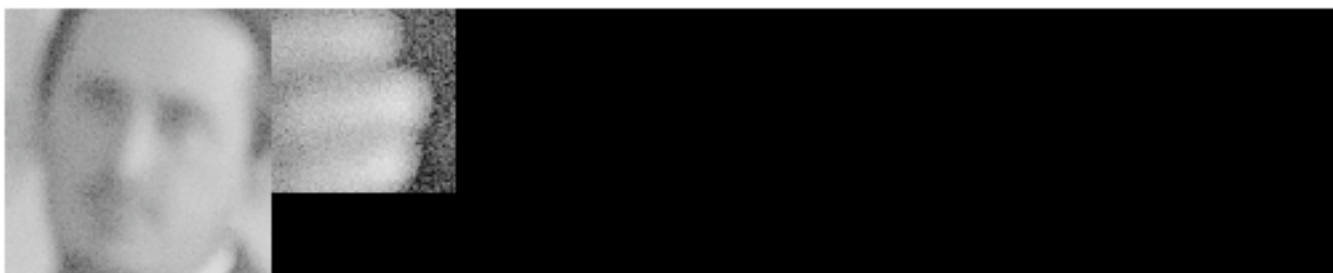
Each page of the newspapers is saved as a single PNG image in a file called [images.zip \(./readonly/images.zip\)](#). These newspapers are in english, and contain a variety of stories, advertisements and images. Note: This file is fairly large (~200 MB) and may take some time to work with, I would encourage you to use [small\\_img.zip \(./readonly/small\\_img.zip\)](#) for testing.

Here's an example of the output expected. Using the [small\\_img.zip \(./readonly/small\\_img.zip\)](#) file, if I search for the string "Christopher" I should see the following image:

Results found in file a-0.png



Results found in file a-3.png



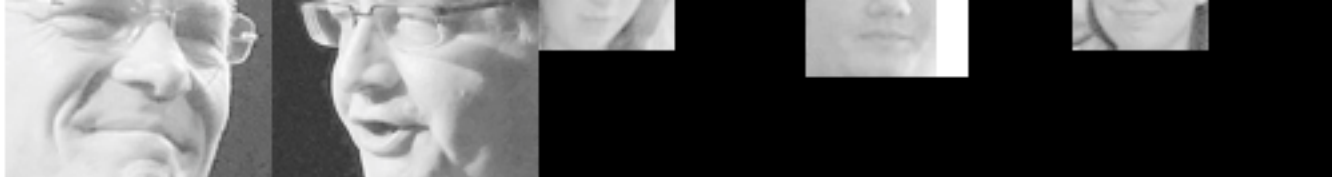
If I were to use the [images.zip](#) ([./readonly/images.zip](#)) file and search for "Mark" I should see the following image (note that there are times when there are no faces on a page, but a word is found!):

Results found in file a-0.png

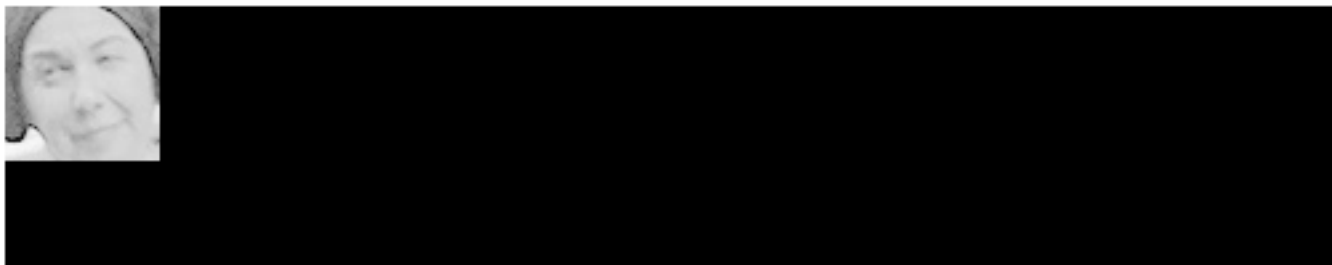


Results found in file a-1.png





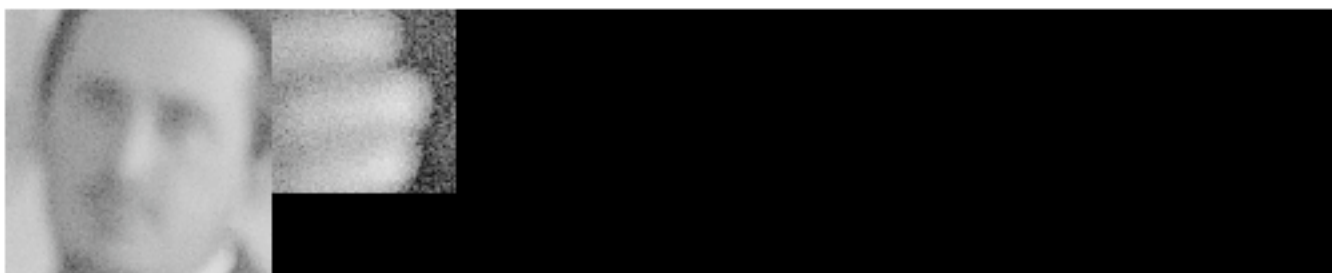
Results found in file a-10.png  
But there were no faces in that file!  
Results found in file a-13.png



Results found in file a-2.png



Results found in file a-3.png



Results found in file a-8.png  
But there were no faces in that file!

Note: That big file can take some time to process - for me it took nearly ten minutes! Use the small one for testing.

In [9]:

```
import zipfile  
  
from PIL import Image, ImageDraw  
import pytesseract  
import cv2 as cv
```

```

import cv2 as cv
import numpy as np

# loading the face detection classifier
face_cascade = cv.CascadeClassifier('readonly/haarcascade_frontalface

# This stores each page as a png image in a dictionary with key=nam

file_name = 'readonly/images.zip'
def page_images(file):
    img_dict = {}

    with zipfile.ZipFile(file) as zips:
        namelist = zips.namelist()
        file = zipfile.ZipFile.extractall(zips)
        for name in namelist:
            image = Image.open(name)
            img_dict[name] = image

    return(img_dict)

print(page_images(file_name))

#for key in page_images(file_name):
#    display(page_images(file_name)[key])

```

```

{'a-0.png': <PIL.PngImagePlugin.PngImageFile image mode
e=RGB size=3600x6300 at 0x7FE1700D54A8>, 'a-1.png': <P
IL.PngImagePlugin.PngImageFile image mode=RGB size=360
0x6300 at 0x7FE1700D5D68>, 'a-10.png': <PIL.PngImagePl
ugin.PngImageFile image mode=RGB size=6300x3600 at 0x7
FE1700D5EF0>, 'a-11.png': <PIL.PngImagePlugin.PngImage
File image mode=RGB size=3150x3600 at 0x7FE168ECD518>,
'a-12.png': <PIL.PngImagePlugin.PngImageFile image mod
e=RGB size=3150x3600 at 0x7FE171CD76D8>, 'a-13.png': <
PIL.PngImagePlugin.PngImageFile image mode=RGB size=31
50x3600 at 0x7FE171CD7668>, 'a-2.png': <PIL.PngImagePl
ugin.PngImageFile image mode=RGB size=3600x6300 at 0x7
FE171CD7C18>, 'a-3.png': <PIL.PngImagePlugin.PngImageF
ile image mode=RGB size=7200x6300 at 0x7FE171CD7630>,
'a-4.png': <PIL.PngImagePlugin.PngImageFile image mode
=RGB size=3600x6300 at 0x7FE171CD7F60>, 'a-5.png': <PI
L.PngImagePlugin.PngImageFile image mode=RGB size=3600

```

```
x6300 at 0x7FE171CD7470>, 'a-6.png': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=3600x6300 at 0x7FE171CD7518>, 'a-7.png': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=3150x3600 at 0x7FE171CD7940>, 'a-8.png': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=3150x3600 at 0x7FE171CD7588>, 'a-9.png': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=3150x3600 at 0x7FE171CD7160>}
```

In [\*]:

```
# Convert all images to strings for a search function to work so th

keyword = 'Mark' #input("Please enter search term: ")
print(keyword)

for key in page_images(file_name):
    text = pytesseract.image_to_string(page_images(file_name)[key])

    images = []

# New dict with images appended to list from selection
    if keyword in text:
        image_d = page_images(file_name)
        image_1 = image_d[key]
        img = np.asarray(image_1)

        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

        faces = face_cascade.detectMultiScale(img, 1.20, minNeighbor

        accum = 0
        for r in faces:
            rec = faces.tolist()[accum]
            accum += 1
            crop = image_1.crop((rec[0], rec[1], rec[0]+rec[2], rec[1]
            images.append(crop)

        if len(images) > 0:
            first_image=images[0]
            thumbs = []
            size = 128, 128

            for i in images:
                thumbs.append(i.thumbnail(size))
```

```

if len(images) % 5 == 0:
    rows = len(images)//5
else:
    rows = len(images)//5 + 1

contact_sheet=Image.new(first_image.mode, (128 * 5, rows

width = 128
height = 128
current_location_w = 0
current_location_h = 0
count = 0
for img in images:
    # Paste the current image into the contact sheet
    contact_sheet.paste(img, (current_location_w, curre
    # Update the current_location counter
    if count <= 4:
        current_location_w = current_location_w + width

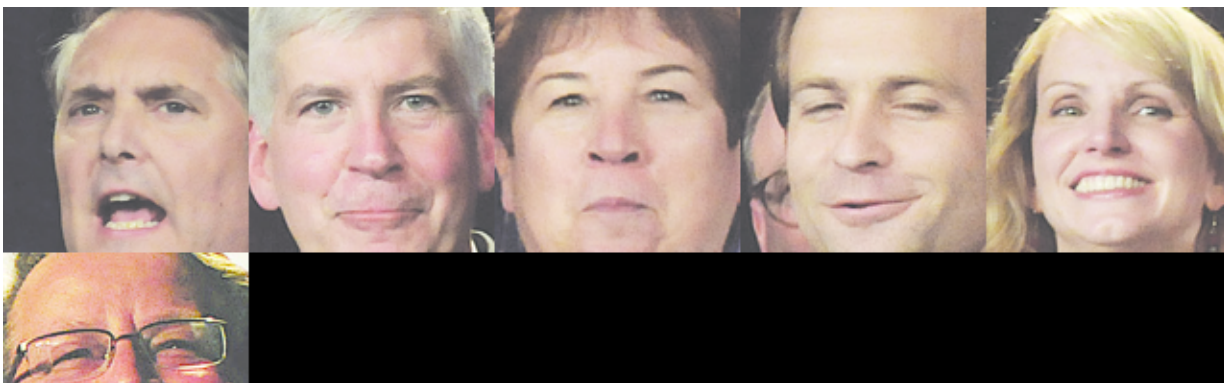
        count += 1
    else:
        count = 0
        current_location_w = 0
        current_location_h = current_location_h + heigh
print(f"Results found in {key}")
display(contact_sheet)

elif len(images) == 0:
    print(f"{keyword} was found but no images in file {key}

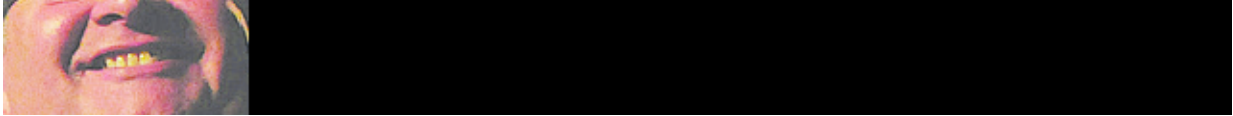
print('proceed')

```

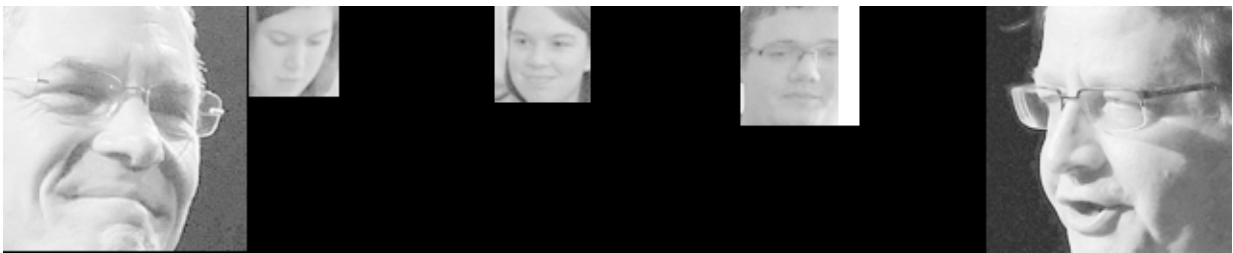
Mark  
Results found in a-0.png



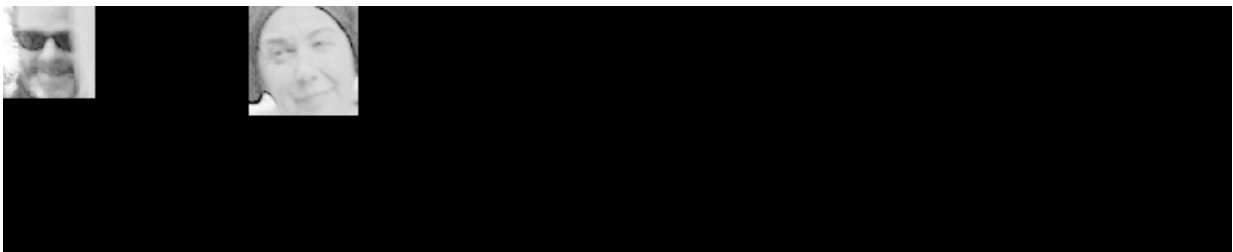




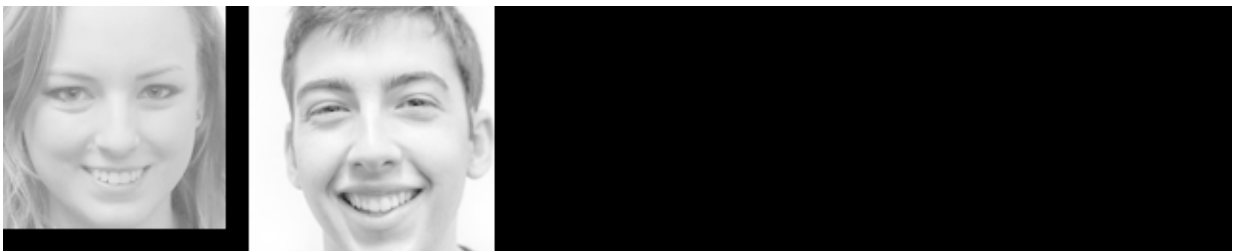
Results found in a-1.png



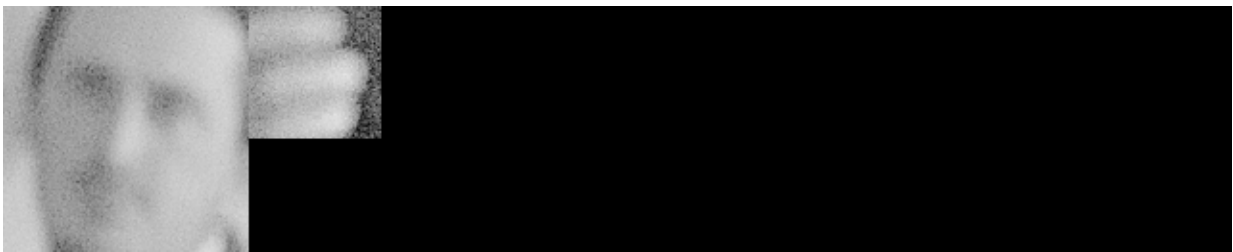
Mark was found but no images in file a-10.png  
Results found in a-13.png



Results found in a-2.png



Results found in a-3.png



In [ ]:

In [ ]:

In [ ]: