

## 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:

 Christopher Search

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!):

 Mark Search

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

```
In [1]: import zipfile
import PIL
from PIL import Image
from PIL import ImageDraw
import pytesseract
import cv2 as cv
import numpy as np

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

# the rest is up to you!

def process_images(file_name,search_txt):
    search_txt = search_txt.lower()
    imgz = zipfile.ZipFile(file_name)
    imgzinf = imgz.infolist()
    contact_sheet = {}
    number_of_images = 0
    for imgzin in imgzinf:
        print(imgzin.filename)
        img_orig = Image.open(imgz.open(imgzin))
        imgtxt = pytesseract.image_to_string(img_orig).lower()
        img = cv.cvtColor(np.array(img_orig), cv.COLOR_BGR2GRAY)
        imgfaces = face_cascade.detectMultiScale(img,minNeighbors=10)
        if search_txt in imgtxt:
            contact_sheet[imgzin.filename]=[]
            for (x, y, w, h) in imgfaces:
                #print("{0} {1} {2} {3} {4}".format(imgzin.filename,x,y,w,h))
                contact_sheet[imgzin.filename].append(img_orig.crop((x,y,x+w,y+h)).resize((128,128)))
            number_of_images += len(contact_sheet[imgzin.filename])
    contact_sheet_full_height = (number_of_images*128)+(len(contact_sheet)*100)
    print('Building contact sheet image with height {0}'.format(contact_sheet_full_height))
    contact_sheet_image = Image.new("RGB", (200,contact_sheet_full_height))
    contact_sheet_image_draw = ImageDraw.Draw(contact_sheet_image)
    print("{0} result with number of images {1}".format(len(contact_sheet),number_of_images))
    y = 0
    for x in contact_sheet.items():
        print("File {0} has {1} matches".format(x[0],len(x[1])))
        contact_sheet_image_draw.text((0,y),'Results found in file {0}'.format(x[0]))
        y+=100
        if(len(x[1])==0):
            contact_sheet_image_draw.text((0,y),'But there were no faces in that file!')
            y+=128
        else:
            for img in x[1]:
                contact_sheet_image.paste(img, (0,y))
                y+=128
    return contact_sheet_image
```

```
In [2]: contact_sheet_image = process_images('readonly/small_img.zip','Chris')
```

```
a-0.png
a-1.png
a-2.png
a-2.png
a-3.png
Building contact sheet image with height 1480
2 result with number of images 10
File a-0.png has 8 matches
File a-3.png has 2 matches
```

```
In [3]: display(contact_sheet_image)
```



```
In [4]: contact_sheet_image = process_images('readonly/images.zip', 'Mark')
```

```
a-0.png  
a-1.png  
a-2.png  
a-3.png  
a-4.png  
a-5.png  
a-6.png  
a-7.png  
a-8.png  
a-9.png  
a-10.png  
a-11.png  
a-12.png  
a-13.png  
Building contact sheet image with height 4740  
9 result with number of images 30  
File a-0.png has 8 matches  
File a-1.png has 5 matches  
File a-2.png has 2 matches  
File a-3.png has 2 matches  
File a-5.png has 8 matches  
File a-8.png has 0 matches  
File a-9.png has 1 matches  
File a-10.png has 1 matches  
File a-13.png has 3 matches
```

```
In [5]: display(contact_sheet_image)
```

Results found in file a-0.png



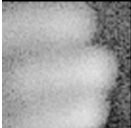
Results found in file a-1.png



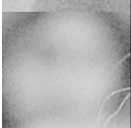
Results found in file a-2.png



Results found in file a-3.png



Results found in file a-5.png



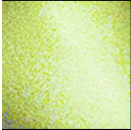
Results found in file a-8.png

But there were no faces in that f

Results found in file a-9.png



Results found in file a-10.png



Results found in file a-13.png

