Hint 1

To access the newspapers in the zipfile, you must first use the Zipfile library to open the zipfile then iterate through the objects (newspapers) in the zipfile using .infolist(). Try and write a simple routine to just go through the zipfile, printing out the name of the file as well as using display(). Remember that the PIL.Image library can .open() files, and that items in .infolist() in the zipfile each appear to Python just as if they were a file (these are called "file-like" objects).

Hint 2

You can spend a lot of time converting between PIL.Image files and byte arrays, but you don't have to. Why not just store the PIL.Image objects in a global data structure, maybe a list or a dictionary indexed by name? Then you can further process this data structure, by adding in information such as the text detected on the pages or the bounding boxes behind faces. Come to think of it, a list of dictionary objects, where each entry in the list would have the PIL image, the bounding boxes, and the text discovered on the page, would be a handy way to store this data.

Hint 3

A quick reminder - in Python all strings are just like lists of characters. Kind of (remember they are immutable lists - more like tuples!). But this means you can use the in keyword to find substrings really easily. So the following statement will return True if the substring is matched: if "Christopher" in my\_text

Hint 4

Creating the contact sheet can be a bit of a pain. But you can resize images without having to worry about the aspect ratio if you use the PIL.Image.thumbnail function. I used it when creating out the output images, maybe you should too! And check out the lecture on the contact sheet, you want to be careful that you don't "walk off" the end of the images when creating a row (or column).

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

In this assignment you will have practiced the use of the libraries behind pillow, tesseract, and opencv in python to build a simple search application. In addition, you will have learned how to manipulate zip files with a new python library.

**Review criteria**

The assignment grading is very simple, if you create something that looks like the two sample outputs (searching for "Chris" in the small images and "Mark" in the large images) you get full points! If you fail to handle the case where some pages have the text but no faces (with the Mark search) you can still pass the assignment, but I know you can do better!

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

Haar Feature-based Cascade Classifier for Object Detection

## CascadeClassifier::detectMultiScale

1. **scaleFactor**: Parameter specifying how much the image size is reduced at each image scale.

  
Picture source: [Viola-Jones Face Detection](https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Object_Detection_Face_Detection_Haar_Cascade_Classifiers.php)  
This scale factor is used to create scale pyramid as shown in the picture. Suppose, the scale factor is 1.03, it means we're using a small step for resizing, i.e. reduce size by 3 %, we increase the chance of a matching size with the model for detection is found, while it's expensive.

1. **minNeighbors**: Parameter specifying how many neighbors each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces: higher value results in less detections but with higher quality. We're using 5 in the code.

<https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Object_Detection_Face_Detection_Haar_Cascade_Classifiers.php>