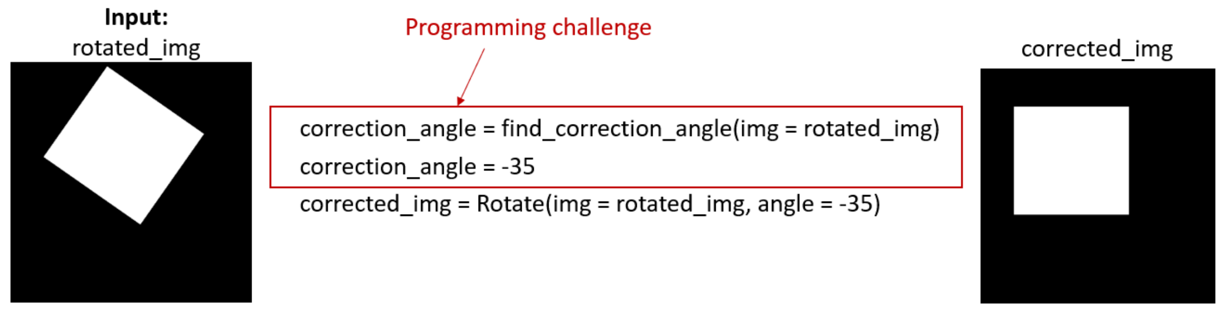
Challenge overview

A good portion of this internship will involve image processing. We often acquire images of samples and apply image processing to extract something we care about. For example, we may want to know whether a defect is present. This may be considered a classification task, as we're deciding whether the image is of a defective or non-defective sample. In other cases, we're performing regression, where we want to know how many defects there are, how bright a sample is, sample dimensions, etc. The challenge outlined here is an example of a routine image processing task that may be necessary for subsequent analyses or simply putting together images neatly in a PowerPoint slide for communicating results.

Context:

In certain cases, a technician will manually place a rectangular sample onto an imaging system (e.g. microscope) and take a picture. The samples are placed at random orientations. A common first step in image processing is to identify what pixels belong to the sample, and what pixels belong to the background. There are many ways of doing this, for example, if you had an image of a moon on a dark sky, you could "threshold" the image, converting all light pixels with intensities > 0.5 to 1, and all dark pixels with intensities less than 0.5 to 0. This produces a binary image, where pixels intensities are one of two possible values. In this challenge, we have assumed that we have a method for doing this already in place, and we've provided you with a binary image of a rectangular sample.



```
correction_angle = find_correction_angle(img = rotated_img)
correction_angle = -35
corrected_img = Rotate(img = rotated_img, angle = -35)
```

Requirements:

Deliverable: A function that accepts a binary image of a rectangular sample and <u>returns a correction angle</u>, that if applied to the original image, would returns a 'righted' image, as shown on the right in the above figure. The function you write should apply to rectangular samples of an arbitrary size and orientation, not just for the example provided.

<u>You don't need to perform the rotation correction or write a function that does so</u>, just create a function that returns the angle. In the above example, the correction angle is -35 degrees. Please assume a positive angle represents a clockwise adjustment, and a negative angle represents a counter-clockwise adjustment. Also, we have no preference on which direction the image is rotated to achieve a correct image (for example, -35 is equally as good as 55 degrees in this case).

Language: Python or R.

Packages: Basic data science libraries like Pandas and Numpy are OK to use, but not necessary or required. Packages specific to image analysis (e.g. OpenCV, EBImage, etc) are not necessary and discouraged for this challenge.

Data: A .tif and .csv file of the rotated sample is provided. Each contains the same data, so please work with whatever format you find easiest to load into your environment.

Example code: In the end, we'd like a function that's structured something like this

Python:

```
def find_correction_angle(binary_image):

        # Your code goes here


        # your code ends here

        return correction_angle
```

R:

```
find_correction_angle = function(binary_image){

        # your code goes here


        # your code ends here

        return(correction_angle)

}
```