Challenge

In this challenge, you will download studies of DICOM MRI images and contour files to prepare them for training a convolutional neural network. The inputs to the network will be dicom images and the targets will be the contours encoded as boolean masks (foreground pixels = True; background pixels = False). Note that you are not required to train the network in this challenge.

We'll give you some starter code that we've prototyped, ask that you verify that it's working correctly, and then modify it as if it were going to be integrated into our production codebase (write your code in a manner that can be used in a production codebase by others). All final code used to prepare the dicoms and contour files will be submitted to us for review. Please compose your solution in Python using whatever libraries/modules that you are most comfortable with. Please also submit any output of your code (images, logs, etc.) that will make it easier for us to understand your results.

We're looking for a few specific things:

- Well-organized, easily-understandable, documented (and self-documenting) code
- Appropriate comments where the code may be difficult to understand
- Object oriented programming where appropriate
- Unit tests if appropriate
- Use of standard library functions and open source code as much as possible. If some functionality is already available in open source code, you should use it and not waste your time duplicating effort. But if you **are** using open source code, make sure you understand it and can defend the design decisions.

You can download the DICOM images and contour files directly from here. After unzipping the file, you should see two directories named *dicoms* and *contourfiles* and one file named *link.csv* (this can be used to link up the appropriate *dicoms* and *contourfiles*). Within the *contourfiles* directory, you will see directories of *i-contours* and *o-contours* (inner contours and outer contours, which are defined below). Note in this phase of the challenge you will only be using the ***i-contours.*** Furthermore, note that this is real-world data! This means that every DICOM may not have a contour (and vice-versa), and that some data may otherwise appear missing and/or incorrect.

You can use this code to get you started with parsing a single dicom image, single contour file, and translating a parsed contour file into a boolean mask. We've prototyped these functions on a couple of studies, and as part of this challenge we want you to verify them on the given studies and **make any necessary adjustments to port the code into our production codebase**.

Definitions

- *i-contour*: inner contour that separates the left ventricular blood pool from the heart muscle (myocardium)
- *o-contour*: outer contour that defines the outer border of the left ventricular heart muscle

Part 1: Parse the DICOM images and Contour Files

Using the functions given above, build a pipeline that will parse the DICOM images and ***i-contour*** contour files, making sure to pair the correct DICOM image(s) with the correct contour file. After parsing each ***i-contour*** file, make sure to translate the parsed contour to a boolean mask.

After building the pipeline, please answer the following questions:

- How did you verify that you are parsing the contours correctly?
- What changes did you make to the code, if any, in order to integrate it into our production code base?

Note: The function that translates the parsed contours into boolean masks relies on the Pillow library. If you run into any Pillow related issues while using it, please feel free to simply leave the function call in your code, but commented out. We don’t want you to spend too much time (if any) on debugging Pillow issues (as they might be related to environment/install issues).

Part 2: Model training pipeline

Using the saved information from the DICOM images and contour files, add an additional step to the pipeline that will load batches of data for input into a 2D deep learning model. This pipeline should meet the following criteria:

1. Cycles over the entire dataset, loading a single batch (e.g. 8 observations) of inputs (DICOM image data) and targets (boolean masks) at each training step.
2. A single batch of data consists of one numpy array for images and one numpy array for targets.
3. Within each epoch (e.g. iteration over all studies once), samples from a batch should be loaded randomly from the entire dataset.

After building the pipeline, please answer the following questions:

- Did you change anything from the pipelines built in Parts 1 to better streamline the pipeline built in Part 2? If so, what? If not, is there anything that you can imagine changing in the future?
- How do you/did you verify that the pipeline was working correctly?
- Given the pipeline you have built, can you see any deficiencies that you would change if you had more time? If not, can you think of any improvements/enhancements to the pipeline that you could build in?