**Ultrasound QA code explanation sheet**

Object-Oriented Programming (OOP):

The code for this project has been written in an object-oriented style, meaning that the DICOM images themselves are treated as objects and have attributes assigned to them (as seen in DICOMimages.py). This allows this code to be reused for other projects in future.

You can read more about OOP here: <https://realpython.com/python3-object-oriented-programming/>

Parts of the code:

1. Attributes of the DICOM image are set here. These self.\_\_\_\_ lines mainly search the DICOM tags for useful information (e.g. Date, Scanner Model etc). There is also a self.region attribute which is essential for the refactoring of the curved image pixel data.
2. The analyse() method takes in a cropped pixel array and outputs the coefficient of variation(COV), skew, and low values for the image (as seen in the paper: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5676531/>). These are outputted as 7 values (as the low values are based on different locations on the image).
3. The linearDICOMimage class has special methods that crop linear DICOM images down to size. These differ from the curved DICOM image methods due to the refactoring process. The crop\_bottom() method can be found in the DICOMimage class as this is used for both linear and curved images and is inherited by both the curvedDICOMimage and linearDICOMimage classes (see OOP link for more info).
4. The curvedDICOMimage class has some key differences to the DICOMimage class. This is because a curved image needs to be linearised before it can be analysed. This is done by:
5. Finding the centre of the circle of which the curved lines on the image rotate around. self.centre is an attribute for this (see diagram) (REF:4.1.1 in code comment)
6. Finding the two points (labelled point 1 and point 2 on the diagram) (REF:4.1.2)
7. Finding the maximum and minimum radius of the circle. (REF:4.1.3)
8. Finding the maximum and minimum angle that the image sweeps between. (REF:4.1.4)
9. Finding the arc length of the curved image (REF:4.1.5)
10. Dividing the sweep angle by the arc length to find the angle increment when sweeping. (REF:4.1.6)
11. Taking the max radius – min radius and multiplying by a stretch constant to find the scan radius. (REF:4.1.7)
12. Creating a blank numpy array with a y-range of the scan radius and an x-range of the arc length. (REF:4.1.8)
13. Incrementing over the scan radius in polar coordinates and converting these to cartesian coordinates. (REF:4.1.9)
14. Interpolating these points using the nearest neighbour function. (REF:4.2.0)
15. Inputting the pixel value into the blank numpy array. (REF:4.2.1)
16. Repeat steps x-xii for each radial point and then iterate over the scan angle. (REF:4.2.2)
17. Analyse function creates a DICOMimage object, reads if the image is linear or curved, deletes the object and makes a new one with either the linearDICOMimage class or the curvedDICOMimage class depending on the image type. The image is then cropped and analysed using the image.analyse() method and outputted to a csv file which can be viewed in Excel or used in other ways.