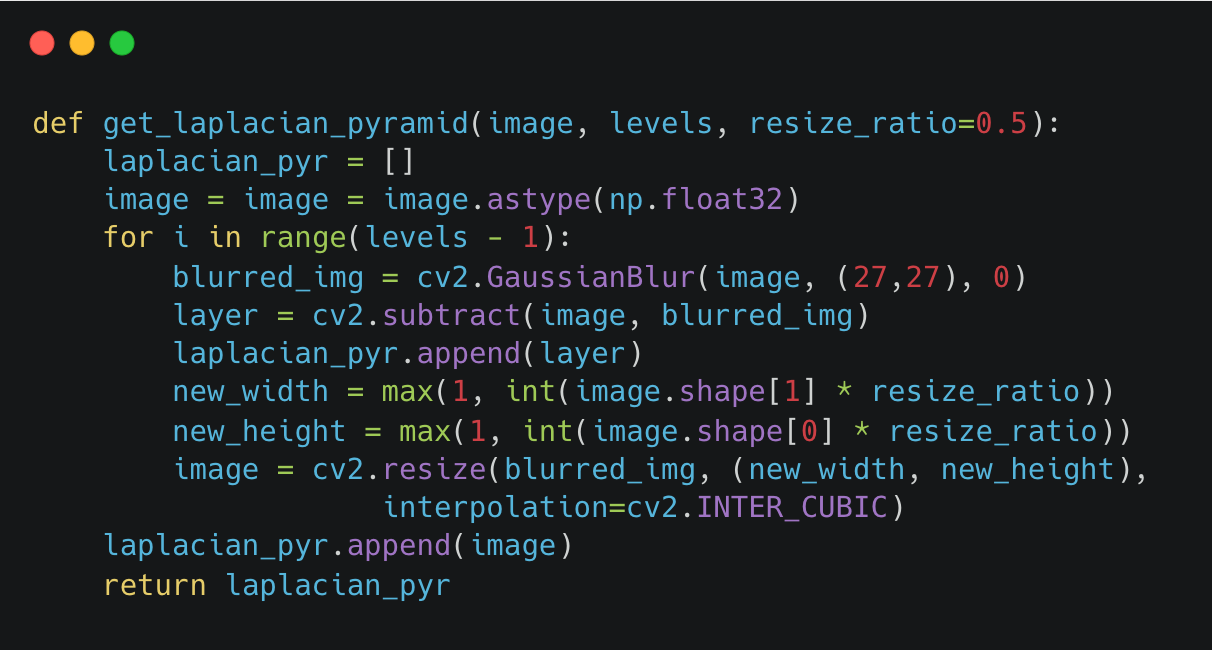
**Problem 2 - Multiband blending**

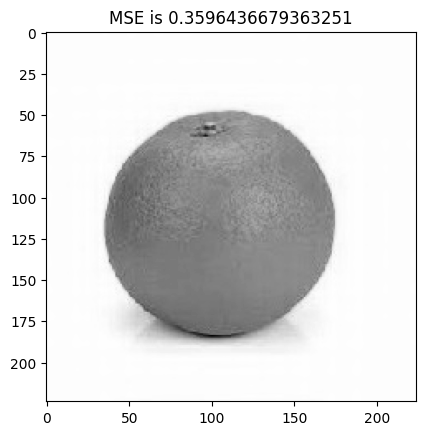
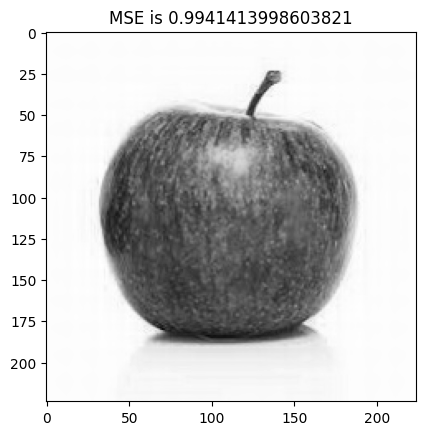
The function 'get\_laplacian\_pyramid' implemented such that it splits the image into multiple Laplacian layers and returns a list of these layers. A layer is created by subtracting a gaussian blurred version of the image from the original one. After each subtraction, the image is replaced by the blurred one and get downsampled by a given ratio. Finally, the remaining image stored as the last layer.



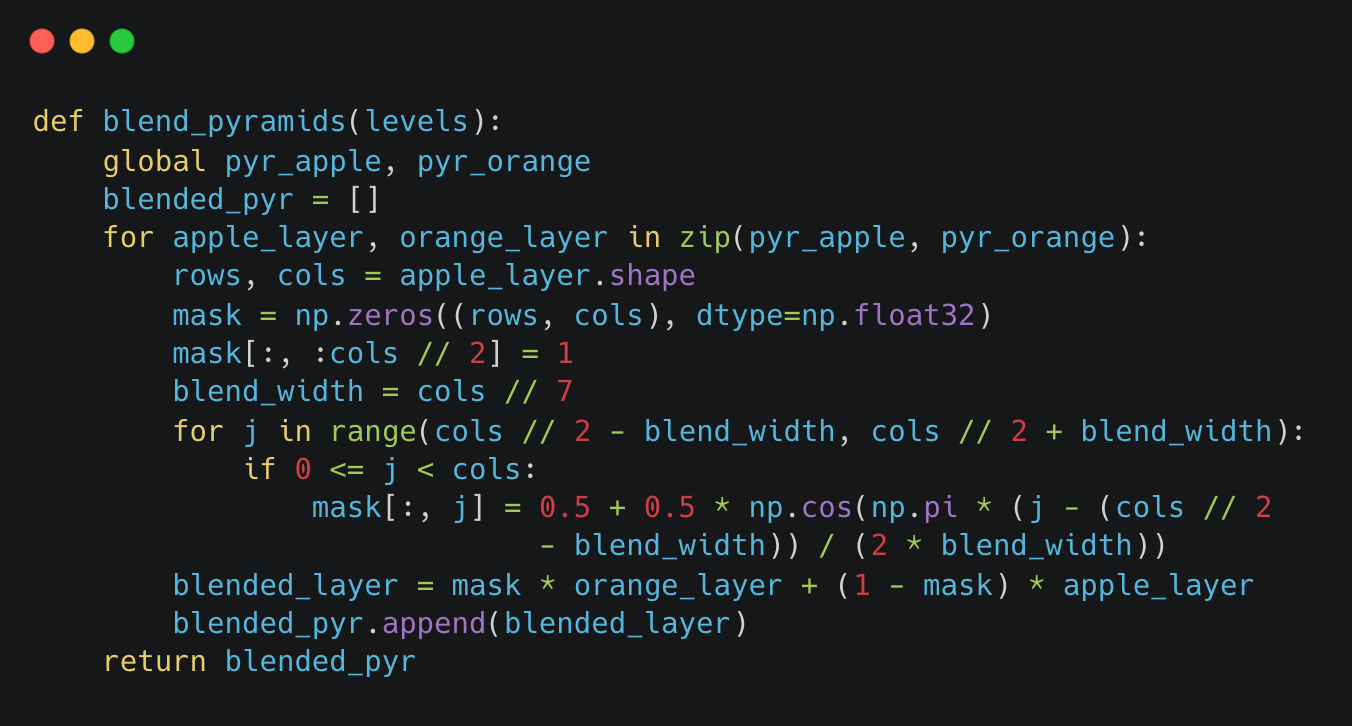
The function 'restore\_from\_pyramid' implemented such that it do the inverse operation. It gets a Laplacian pyramid and builds an image out of it. The function initialize the image as the highest layer of the pyramid (the last element of the given list) and iteratively adds the layers from the highest to the lowest, while resizing the image each time to match the next Laplacian layer. The resizing ratio (the difference in size between each two consecutive layers) is given to the function as a parameter.



Both functions applied to the two provided images, and the results were close to the original with :



The function 'blend\_pyramids' was also implemented. It aims to blend the two provided images in a smooth nice looking way, using their Laplacian pyramids. The function loops through the corresponding layers of the two pyramids, and applies a cosine function to define the gradual transition. The size of the transition window is set to 1/7 of the constructed image width. Finally the blending is performed using a cross-dissolve operation with the generated mask.



The function was tested on the two provided image using both the method mentioned in the PDF and our implemented one, and we preferred the last. In addition, we experimented with different transition window sizes, aiming to achieve the best transition in terms of smooth transition, uniform background, natural-looking shadow, and minimizing noticeable artifacts, especially around the stem of the apple. Here's the result:

****

Below are some of the less successful results received:

* The result obtained with the too small window size of 1/10:

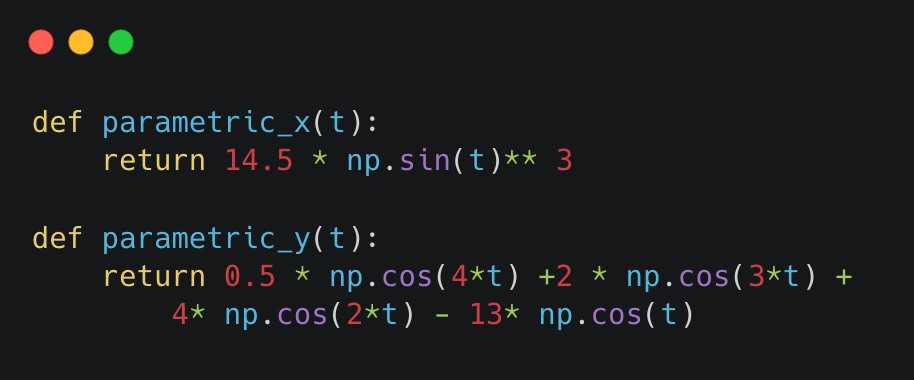


* The result obtained using the method mentioned in the PDF:

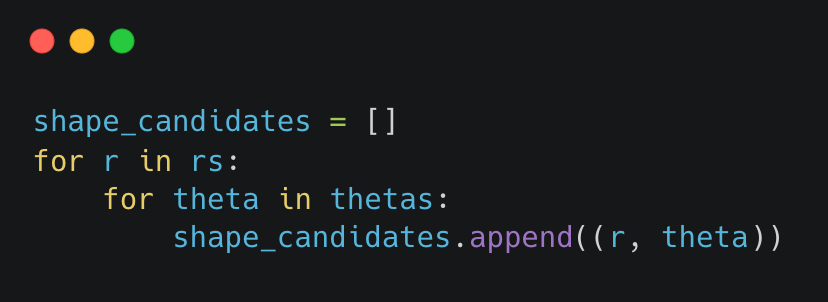
****

**Problem 3 - Hough transform**

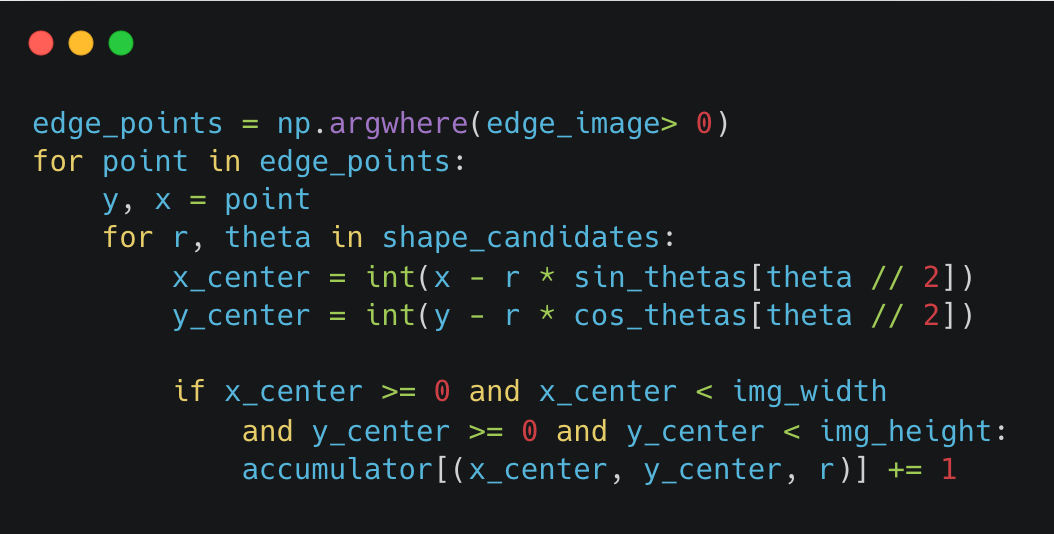
The two functions 'parametric\_x' and 'parametric\_y' were implemented based on the heart's parametric equation mentioned in the instructions document:



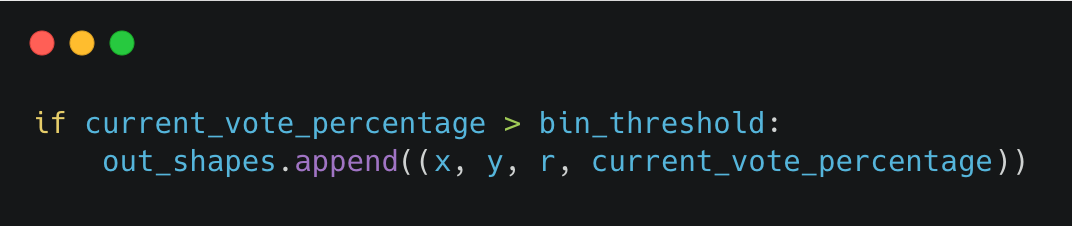
The following code was modified so it now generates all possible heart shape candidates using different values of r and θ:



The implementation of the following piece of code which iterate over all edge points and candidate hearts computing the center of each possible heart was completed. The code in addition ensures that the detected centers are within the image boundaries:



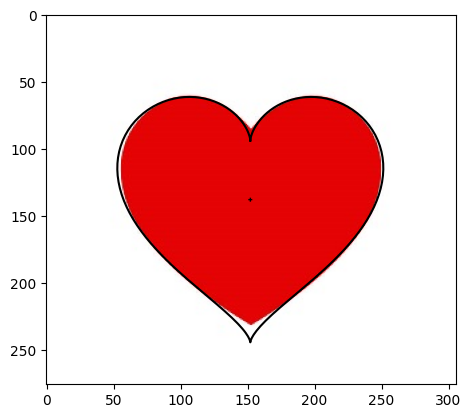
The following code filters weak detection was implemented:



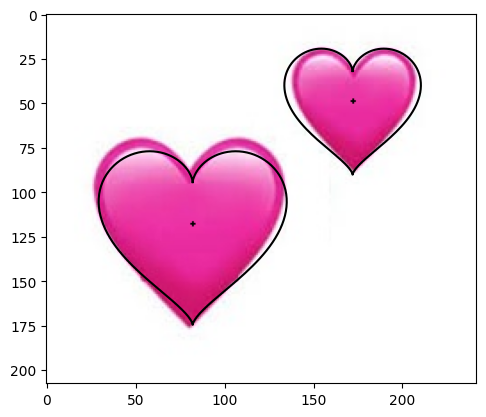
Below are the detection output for each image along with the parameter values which provided the best results:

Note: the bin threshold is given as a range of values that all produce the same result. If a single value had to be chosen, we would the midpoint of the range, which balances between avoiding false detections and succussing in detecting hearts when edges are weak.

* Simple:



* Med:



* Hard:

תמונה שמכילה טקסט, צילום מסך, לב, אדום

התיאור נוצר באופן אוטומטי