Canny Edge Detection is an edge detection algorithm that identifies the strongest edges in the image. This is implemented in a 6 step process which corresponds to different functions in my algorithm.

1. Converting to Grayscale
    a. The program reads the image and then converts it to grayscale by calculating the mean of the intensity.
2. Gaussian Blur
    a. With the given gaussian kernel, the function goes through the grayscale image and computes the sum of the kernel multiplied by each of the 9 values in the grayscale image.
3. Sobel Operators
    a. Uses kernels in the x and y direction to calculate the gradient and magnitude of each point in the image.
4. Non-maximum suppression
    a. The direction is converted to 0, 1, 2, 3 to represent 0 , 45, 90, 135 degrees respectively. This is done by calculations based on the unit circle. It then checks the magnitude of the two neighbors along that direction and only keeps the edge if the magnitude of the pixel is greater than the magnitude of its two edges.
5. Double Thresholding
    a. The strong edges are the ones where the magnitude is greater than the high threshold and the weak edges are the ones where the magnitude is greater than the low threshold, but greater than the high threshold.
6. Hysteresis
    a. The final result keeps the strong edges and only keeps the weak edges that are adjacent to at least one strong edge.

Evaluation:

Some of the biggest challenges I faced was figuring out how to use numpy more effectively. Prior to this assignment, I had treated it as any other array disregarding the fact that it has a lot more capabilities. For example, I learned that I can compare and array to a value to essentially filter out anything lower than that value.

Another challenge was trying to implement thresholding in a way that it would be applicable to every image, instead of just one. I did a canny edge before, but we changed the thresholding values for each image manually.