In this case, you are applying Canny to the image `gray` and your output will be another image called `edges`. `low_threshold` and `high_threshold` are your thresholds for edge detection.

The algorithm will first detect strong edge (strong gradient) pixels above the `high_threshold`, and reject pixels below the `low_threshold`. Next, pixels with values between the `low_threshold` and `high_threshold` will be included as long as they are connected to strong edges. The output `edges` is a binary image with white pixels tracing out the detected edges and black everywhere else. See the OpenCV Canny Docs for more details.

What would make sense as a reasonable range for these parameters? In our case, converting to grayscale has left us with an 8-bit image, so each pixel can take 2^8 = 256 possible values. Hence, the pixel values range from 0 to 255.

This range implies that derivatives (essentially, the value differences from pixel to pixel) will be on the scale of tens or hundreds. So, **a reasonable range for your threshold parameters would also be in the tens to hundreds**.

As far as a ratio of `low_threshold` to `high_threshold`, John Canny himself recommended a low to high ratio of 1:2 or 1:3.

We'll also include Gaussian smoothing, before running Canny, which is essentially a way of suppressing noise and spurious gradients by averaging (check out the OpenCV docs for GaussianBlur). `cv2.Canny()` actually applies Gaussian smoothing internally, but we include it here because you can get a different result by applying further smoothing (and it's not a changeable parameter within `cv2.Canny()`!).

You can choose the `kernel_size` for Gaussian smoothing to be any odd number. A larger `kernel_size` implies averaging, or smoothing, over a larger area. The example in the previous lesson was `kernel_size = 3`.