I chose to implement the “Finding a Palindrome” solution by checking elements from the outside-in. Assigning indices at the beginning and end of the tuple allowed a symmetrical comparison for equality of left and right elements. I thought it would be best to remove an element I knew to be unequal rather than removing an element iteratively and checking the tuple for a palindrome. The latter solution is what I believe the zyBooks suggested as the “straightforward O(n^2)” implementation.

This was my first time using Python and I spent a significant portion of my time experimenting with the best ways to implement my pseudocode. I would have liked to simplify my find\_palindrome function into more subroutines. It is longer than I’d like, and does more than one thing. However, I struggled with Python’s variable assignment scope and “pass by assignment” behavior. After a few iterations of failure, I settled for only a couple subroutines that were easier to implement by passing arguments.

I considered a recursive solution but ultimately stuck with the outside-in iterative approach that was submitted. There were too many base cases steering me away from recursion. The only thing I would have liked to further consider that I didn’t prioritize were the suggestions for reversal in the zyBooks hints.

The big-O complexity of this algorithm is O(n) because I only iterate through the tuple once.