Gabriela Onelli
Dr. Saremi
SSW 567
09/11/21

Homework One: Deliverable Three

I pledge my honor that I have abided by the Stevens Honor System. - Gabriela Onelli

1. The main challenges boiled down to two things, remembering how python worked after working in Java/javascript for the past year or so and getting my unit test to actually pull from my other python file. Apparently python doesn't like my naming conventions, so sad.
2. I thought the requirements were fine, not really as formal or fully fleshed out as they would need to be in a professional environment, but fine enough for a basic homework assignment. It could have used a little more specification on what to do if the function was passed in triangle sides that were impossible, like if someone input 0 for all the side lengths obviously this isn't a triangle, but we were given no specification as to what to do with this. Did the 'client' want us to prompt the user for better inputs? Or just state that this was not a triangle? That was less clear and as such I just left it out entirely so it could act as a bug for the system and demonstrate my test script's ability to catch bugs.
3. I didn't encounter any problems with the tools, I took CS 115 a while back and we used IDLE Python for all our coding homeworks. As such I've very familiar with the environment and very comfortable working with it.
4. Since this was a simple homework assignment, thus carrying very little risk should something go wrong, I didn't go overboard with the testing. For the criteria I needed the function to show the four features I made work correctly and the one bug I put in on purpose. As such it had to test the classify_triangle function against an example of a right triangle, equilateral triangle, scalene triangle, and an isosceles triangle. The four expected test inputs, as well as an example of something that was clearly not a triangle (which was my purposeful bug in the program, the inability to recognize this input as invalid) had to be checked. As such the test script was split into 5 sections with each section testing one specific output expectation. Each section had two different triangles with different lengths it would test, both of which should return as the correct type of triangle. I did this because testing more than one triangle would show the function wasn't just hardcoded to accept a specific input and to show it wasn't just a fluke with the specific numbers chosen. The only addition was added to the sections where triangle side lengths were different numbers and the order of the lengths could be noticeably changed. This way I could, for example, make sure that my classify_triangle would recognize a right triangle whatever order the side lengths were written in. I then ran the test script with the expectation that it would run 16 tests with only 3 failures coming from the function's inability to recognize invalid triangle input. In theory I could have tested several more triangles for each section, however I felt that would be excessive given the risk associated with failure on this script and the simplicity of the classify_triangle itself.