Name: Kevin Ferreras

Deliverable 3

* One of the challenges that I encountered was thinking about the situation in which classify\_triangle() were to receive a negative input as a length of a side and how the function would go about handling it. My solution to the negative-side situation was to create a separate function that classify\_triangle() would call to check if any of the inputs are negative. Another challenge I encountered was thinking about the situation in which one side did not make a complete triangle. Similarly, my solution to this was to create a separate function that classify\_triangle() would call in order to make sure that the inputs adhered to the triangle property: *The sum of the length of any two sides of a triangle is greater than the length of the third side* , which would avoid the incomplete-triangle situation.
* I found that the requirements for this assignment were vague/incomplete. For example:
  + The requirements do not specify whether the parameters that are to be passed into classify\_triangle() should be integers, floats, or strings
  + The requirements do not specify where the parameters will be coming from; will the program require user input? Will it be a data dump? Will it have to read from a file? Etc.
  + For the string that is supposed to be returned by classify\_triangle(), the requirements do not specify how the string should read or be formatted, leaving the assumption that the function should just “spit out” *scalene*, *isosceles*, *equilateral*, or *right*
* Given prior experience with VS Code, Github, & Unittest, I did not encounter any challenges with these development tools.
* My criteria/approach for my test cases was to have at least one general case and one edge case.