Assignment Description:

Sometimes you will be given a program that someone else has written, and you will be asked to fix, update and enhance that program. In this assignment you will start with an existing implementation of the classify triangle program that will be given to you. You will also be given a starter test program that tests the classify triangle program, but those tests are not complete.

- These are the two files: Triangle.py and TestTriangle.py
  - ***Triangle.py*** is a starter implementation of the triangle classification program.
  - ***TestTriangle.py*** contains a starter set of unittest test cases to test the classifyTriangle() function in the file Triangle.py file.

In order to determine if the program is correctly implemented, you will need to update the set of test cases in the test program. You will need to update the test program until you feel that your tests adequately test all of the conditions. Then you should run the complete set of tests against the original triangle program to see how correct the triangle program is. Capture and then report on those results in a formal test report described below. For this first part you should not make any changes to the classify triangle program. You should only change the test program.

Based on the results of your initial tests, you will then update the classify triangle program to fix all defects. Continue to run the test cases as you fix defects until all of the defects have been fixed. Run one final execution of the test program and capture and then report on those results in a formal test report described below.

Note that you should NOT simply replace the logic with your logic from Assignment 1. Test teams typically don't have the luxury of rewriting code from scratch and instead must fix what's delivered to the test team.

Part 0:

Create a new GitHub repository for this assignment. When creating the repository, you should choose for it to be "public" repository and choose to include a README file. Give your new repository a meaningful name. In our example here we will name our repository "Triangle567" but you can name it anything (as long as it makes sense for this assignment). Also for this example, we assume that the GitHub ID is "tsmith567"

After you have created the repository in GitHub, you can now download this repository to your local environment (e.g. your laptop).

To do this you need to "clone" the repository.

```
$ git clone https://github.com/tsmith567/Triangle567.git
```

```
$ cd Triangle567/
```

The Triangle567 folder should contain the file "README.md".

Next you should upload and commit both of these files, Triangle.py and TestTriangle.py in their original form to the new GitHub repo Then any changes you make to these programs will also be committed to the same GitHub repo.

Here are the steps to do this:

- *copy your two Triangle source files to your local repository folder,*
- *then add and commit them to git, and t*
- *hen push those files up to your repository on GitHub.*

a. Copy Triangle.py and TestTriangle.py to the Triangle567 folder.

b. Next you should run these commands to add and commit the changes to your local repository on your laptop:

```
$ git add TestTriangle.py Triangle.py
```

```
$ git commit -m "add triangle code"
```

c. Finally, you should run this command to push the changes to GitHub:

```
$ git push
```

If you look at your GitHub repo you should see the files:

Part 1:

1. Review the Triangle.py file which includes the classifyTriangle() function implemented in Python.
2. Enhance the set of test cases for the Triangle problem that adequately tests the classifyTriangle() function to find problems. The test cases will be in a separate test program file called TestTriangle.py . You should not fix any bugs in Triangle.py at this time, just make changes to TestTriangle.py
3. Run your test set against the classifyTriangle() function by running:
    - python -m unittest TestTriangle
4. Create a test report in the format specified below. This report shows the results of testing the **initial** classifyTriangle() implementation.
5. Commit and push your changes to the TestTriangle.py program to GitHub

Part 2:

1. After you've completed part 1 that defines your test set and after running it against the buggy classifyTriangle() function, update the logic in classifytTriangle() to fix all of the logic bugs you found by code inspection and with your test cases.
2. Run the same test set on your improved classifyTriangle() function and create a test report on your improved logic
3. Commit and push your changes to the Triangle.py program to GitHub

Author: Christian Chaneski

Summary:

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5
Tests Planned	22	22	26	24	22
Tests Passed	0	17	17	20	22
Defects Found	5	3	3	2	0
Defects Fixed	0	2	0	1	2

- This assignment really helped me to look over how I completed the classify triangle assignment for the last homework and made me think of ways in which I could improve upon my past design for the classify triangle code. This assignment helped to show me the importance of utilizing multiple test cases in order to rigorously test my code and root out any important errors that I may have missed.

Honor Pledge: *I Pledge My Honor That I Have Abided By The Stevens Honor System.*

Detailed Results:

- Besides the obvious use of test cases to complete this homework assignment, I also utilized my knowledge of basic geometry in order to design tests that could challenge the extent of my program and root out any possible problems that could be addressed.
- The program was designed to test for specific restraints that ensured the data that was tested was considered a triangle and was made with lengths that were integers.
- I utilized a variety of inputs that were variants of each type of triangle in order to test the different ways that each type of triangle could be inputted into the program.
- I feel that the results of my code ended in the way that I had expected it to turn out, and I feel that given the simplicity of this program, the test cases that I had used were sufficient to test for the possible inputs and results that would be expected for such a program.