Lab #9 Grading Rubric and Instructions

This is the ninth lab for the 'Introduction to Computer Programming' course.

Please see the syllabus for information about when the work in this lab is due.

Lab Objective

The purpose of this lab is to give you practice writing unit tests with `pytest`, and to give you a (brief) introduction to the concept of "Test Driven Development."

Instructions/ Deliverables

NOTE: These tasks can be completed in any order you like. See the **Grading Items** section below for the point distribution.

CITATION: Some of the exercises found here are completely original to the instructor.

- The abbreviation 'MH' will be used to indicate these exercises. This citation will be placed next to the exercise title.
    - ex: [MH]

Reading:

- 'Check off' your notes in your Engineering Notebook for Runestone chapters 13 and 20 as well as the slides listed below with the TA/ Instructor. This should already be done before the start of the lab period.
    - `The_Runtime_Stack.pptx`
    - **NOTE**: You do not need to complete any of the exercises at the end of the chapter. However, it would be helpful to you in the long term if you were to do so.

NOTE: For this lab, you will need to install the `pytest` framework if you have not done so already:

- (PC) `pip install pytest`
- (mac) `pip3 install pytest`

NOTE: If, for whatever reason, your computer is not cooperating and you cannot install `pytest`, just ask either one of the TAs or the Instructor, and we will help you get everything working.

`myMathTests.py`: [MH]

- Consider the given math module, `myMath.py`. This module contains a variety of mathematical functions.
    - Unfortunately, many of these functions contain subtle errors that may lead to unexpected behavior if used in a production environment.
- For each function in `myMath.py`, write a series of unit tests to determine the problems in these functions.
    - **NOTE**: Not all of the functions in `myMath.py` contain errors. It is up to you to find out

which ones do, and which do not. This means you will need to manually calculate many of your test answers. Additionally, you may need to look more deeply into what each function is actually supposed to calculate.

- An example script for just the `add()` function is below:

```
from myMath import add

def test_add():
    assert add(1, 2) == 3
    assert add(-1, -1) == -2
    assert add(0, 0) == 0
```

From the terminal, you can invoke `pytest` on your unit tests in the following way:

```
pytest myMathTests.py
```

- Once you determine the problem or problems with each function, take steps to debug/ fix the code so that it functions correctly.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `myMathTests.py`.

`fourInSequenceTests.py`: [MH]

- Previously in the semester, you were given access to a file called `debugging.txt`, which was meant to help you debug your `fourInSequence.py` project.
    - While this code might have been useful in helping you complete your game, it might have also been a bit cumbersome.
- Use the code inside of `debugging.txt` to help you make actual `pytest` unit tests for your `fourInSequence.py` project.
    - Please see the instructions for the `myMathTests.py` task (above) for an example of how to make/ run `pytest` unit tests.
- You will need to make unit tests for *all* of the following functions in `fourInSequence.py`:
    - `checkForNextMoveWin()`
    - `checkAdjacent()`
    - `checkDraw()`
    - `checkWinner()`
- If your `fourInSequence.py` project had bugs, or you did not get it completed, then this is the perfect time to fix your submission if you want to get it working so you have something cool to show your friends!
    - Please note, we cannot update/ change your grade for your initial submission of `fourInSequenceTests.py` - even if you now get it working properly.
- **HINT**: Do you need all of the code inside `debugging.txt`? Or, do you just need some of it?
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `fourInSequenceTests.py`.

`rockPaperScissors.py` and `rockPaperScissorsTests.py`: [MH]

- 'Test Driven Development' is a way of writing code, wherein the programmer writes unit tests *first*, and *then* writes their real code.
    - Please read the following for more information: https://en.wikipedia.org/wiki/Test-driven_development

- For this task, you will use the idea of 'Test Driven Development' to make a simple game of 'Rock Paper Scissors.'
- The rules of 'Rock Paper Scissors' are as follows:
    - There are exactly two players - a human player, and the computer player.
    - The human player decides how many 'rounds' to play.
        - This must be an *odd* number, so that there is no chance of a draw.
    - The player who wins the most 'rounds' wins.
        - If one player has won more than half the rounds (Ex: 3/5, 4/7, etc.), then the game immediately ceases - even if there are potentially more rounds that can be played.
    - Simultaneously, each player chooses one of the following moves: Rock, Paper, or Scissors.
        - If one player chooses Rock, and the other chooses Paper, the player who chose Paper wins.
        - If one player chooses Rock, and the other chooses Scissors, the player who chose Rock wins.
        - If one player chooses Paper, and the other chooses Scissors, the player who chose Scissors wins.
        - If one player chooses Paper, and the other chooses Rock, the player who chose Paper wins.
        - If one player chooses Scissors, and the other chooses Rock, the player who chose Rock wins.
        - If one player chooses Scissors, and the other chooses Paper, the player who chose Scissors wins.
        - If both players choose the same move, the game is a draw.
- First, write unit tests for your game and save them to a file called `rockPaperScissorsTests.py`.
    - Please see the instructions for the `myMathTests.py` task (above) for an example of how to make/ run `pytest` unit tests.
    - Your game should have at least the following functions:
        - `generateComputerMove()`
            - Returns a randomly selected string: "Rock", "Paper", or "Scissors"
        - `determineWinner()`
            - Takes both moves as arguments and determines which one wins. The function then returns a string to that effect.
                - Ex: `determineWinner("Rock", "Paper")` => "Paper"
        - `playRound()`
            - Takes the human player's move as an argument, generates the computer's move, and then calculates whether the human or computer wins. The function then returns a string to that effect.
                - Ex: `playRound("Rock")` (assuming the computer picked 'Paper') => "Computer Wins!"
- After you have made your unit tests, *then* go about programming your `rockPaperScissors.py` game.
- Use a `main()` function in the way demonstrated in class.
    - Ex: `if __name__ == "__main__":` etc.
- Other than using a `main()` function, you can code your answer any way you like.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to files called `rockPaperScissors.py` and `rockPaperScissorsTests.py`.

Attendance:

- If you have completed all of your tasks for the lab, you may work on any of the 'Additional

Resources for Study' found in the Canvas announcement of the same name.
    - **NOTE**: If you leave early, you will not receive the 'attendance points' for the lab.

Optional Readings

NOTE: These readings are not required. However, they may provide a bit of interest/ insight into the broader world of Computer Science. Please complete the rest of your lab tasks before doing these readings. You do not need to take notes on these in your Engineering Notebook.

What Is TDD? Importance & Benefits - by: Suma Ganji

- Available: https://www.accelq.com/blog/tdd-test-driven-development/

5 ways to avoid being catfished - by: Malwarebytes Labs

- Available: https://www.malwarebytes.com/blog/news/2022/06/5-ways-to-avoid-being-catfished

When Should You Apply for a Summer Internship? (With 6 Tips) - by: Indeed Editorial Team

- Available: https://www.indeed.com/career-advice/finding-a-job/when-to-apply-for-summer-internship

Gaming Industry: The good the bad and the ugly - by: Priscilla Martin

- Available: https://towardsdatascience.com/gaming-industry-the-good-the-bad-and-the-ugly-47c8e1244e24

Files Provided

`myMath.py`

`debugging.txt`

Example Script

None

Example Output

None

Grading Items

- (**Reading**) Has the student read chapters 13 and 20 of the Runestone textbook, as well as the listed slides, and shown their notes in their Engineering Notebook to the TA/ Instructor?: _____ / 10
- (**myMathTests.py**) Has the student completed the task above, and saved their work to a file called `myMathTests.py`?: _____ / 30
- (**fourInSequenceTests.py**) Has the student completed the task above, and saved their work to a

file called `fourInSequenceTests.py`?: _____ / 10

- (**rockPaperScissors.py** and **rockPaperScissorsTests.py**) Has the student completed the task above, and saved their work to files called `rockPaperScissors.py` and `rockPaperScissorsTests.py`?: _____ / 30
- (**Attendance**) Did the student attend the full lab meeting in person, or did they attend the full lab meeting virtually via WebEx?: _____ / 20

TOTAL _____ / 100