CIT 590 Assignment 1: Lunar Lander

Fall 2016, David Matuszek

Purposes of this assignment

- To get you started programming in Python.
- To get you started with the IDLE Integrated Development Environment.

General idea of the assignment

Lunar Lander is one of the earliest computer games. With a proper choice of initial values, it is fairly interesting to play, even as a text-only program.

You are in a lunar module, some distance above the Moon's surface. Gravity is pulling you toward the Moon at an ever-increasing rate of speed. You have a limited amount of fuel to use, and each time you burn fuel, you reduce your speed by a certain amount. If you burn the right amount of fuel at the right time, you can land safely (that is, when you reach the Moon's surface, you are not going *too* fast).

The math

This game is not timed; you can take as long as you want to enter numbers. Each time you enter a number, one second of "game time" will have passed. In other words, 1 turn = 1 second.

At each turn, you are told:

- Your **altitude** in meters above the moon
- Your **velocity** toward the Moon in meters/second (this number will be negative if you are moving away)
- How much **fuel** you have remaining (in liters)

Each of these quantities should be in variables with appropriate names. The program should specify initial values for these variables. A reasonable choice is an altitude of 1000.0 meters, a velocity of 0.0 (neither rising nor falling), and 1000.0 liters of fuel.

You then get to specify:

- How much **fuel** to burn
    - Zero is a legal value (and may be all you have!)
    - If you ask to burn more fuel than you have, burn all that you have
    - If you ask to burn a negative amount of fuel, it should be treated as zero

The game ends when your altitude becomes zero or negative. A safe landing occurs if your speed is under 10 meters/second. Otherwise, you blast a new crater in the moons surface.

At each turn, you will need to do the following calculations:

- Your **velocity** *increases* by 1.6 meters/second, due to the acceleration of gravity; but in additon,
- Your **velocity** *decreases* by an amount proportional to the amount of fuel you just burned (zero at the first turn). That is, velocity decreases by **some constant** times the amount of fuel. A constant of 0.15 makes the game fairly easy to win.
- Your **altitude** *decreases* by your velocity multiplied by the amount of "time" each turn takes. (Each turn takes 1 second, so we don't actually need to do this multiplication.)
- Your **fuel** *decreases* by the amount you burn, but of course you cannot burn more than you have, so the amount of fuel never becomes negative. (If you ask to burn more than you have, burn only as much as you do have, and set the amount remaining to zero.)

After these calculations, you need to determine if you have "landed." You have landed if your altitude is *less than or equal to* zero (it is highly improbable that you would ever get *exactly* zero). A *safe* landing is one where your velocity is not more than 10 meters/second.

Either way, the game ends when you have landed. If you have landed safely, adjust your altitude to be zero before printing out the final numbers, along with a congratulatory message. If you didn't land safely, use your velocity to print out how deep a crater you have just blasted in the lunar surface.

After each game, ask the user if they want to play again. Any response that begins with `'y'` or `'Y'` means "yes," any response that begins with `'n'` or `'N'` means "no," and for any other answer you should ask again.

Note: We *talk* about values using their measurement units, for example, "1.6 meters/second." In our *programming*, however, we don't use the units, just the numbers, for example `1.6`. It is your responsibility as a programmer to know what units you are using. (The $125 million 1999 Mars orbiter was lost because Lockheed Martin used English units in its programming, while NASA expected metric units.)

Details

As you can imagine, it could take my TAs a lot of time to check out your game. Accordingly, I want you to turn in a file named `readme.txt`, which provides (1) a list of numbers to enter in order to win the game by landing safely, and (2) a list of numbers which will cause you to lose the game. (The latter list should be very easy to find.)

Due date

Zip and turn in your `LunarLander.py` file and your `readme.txt` file by **Tuesday midnight, September 6.** However, you should read the special instructions about the first assignment.

Only files submitted to Canvas will be accepted. I do not accept assignments submitted by email.