CIT 590: Fall 2019
Homework 2

HW deadline as per Canvas.

This homework deals with the following topics:

- Getting user input
- Error checking
- Variables & data types
- Conditionals

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 though it is a text-only program.

You will take the role of an astronaut in the lunar module attempting to land on the moon's surface. Gravity pulls you towards the moon at an increasing rate of speed. In order to land safely, you must burn **fuel** to counter gravity's acceleration to land on the moon at a safe speed (below 10 m/s). However, be careful, you can only use so much **fuel**. And if you are too high when you run out of **fuel**, you'll inevitably crash at an unsafe speed.

The Math

This game is not timed. Rather, you will "simulate" the passage of time by one second after each number entered. The game will play out as follows:

1. The player will be given their current **altitude**, **velocity**, and **fuel**.
2. The player will specify how much **fuel** to burn in the next second.
3. The game will perform the calculations of how the player's actions change **altitude**, **velocity**, and **fuel** over the next "second".
4. If still above the moon's surface, go to step 1. Else, end the game (safely or otherwise …)

These steps will be inside of a loop that runs *while* your **altitude** is positive. To create a *while* loop, you can use the following code in your program:

```
while altitude > 0:

      #write your your code for the steps above
      #it will run indefinitely while altitude > 0
```

Note that it is unlikely you will hit zero **altitude** exactly. Instead, use **altitude** being less than or equal to zero as your termination condition.

Note that the game should set an initial **altitude**, **velocity**, and **fuel** amount. **Altitude** = 100.0 meters, **velocity** = 0.0 meters/second, and **fuel** = 100.0 liters can lead to interesting games.

To break down step 3, assume the following:

- Each turn, your **velocity** increases by 1.6 meters per second due to the force of gravity of the moon (assume **altitude** doesn't affect the force generated by gravity).
- Additionally, your **velocity** decreases by an amount proportional to the **fuel** you burn. This means you multiply **fuel** burnt times some constant to get the **velocity** change. Using a constant of 0.15 makes the game fairly easy to win. Note, a negative **velocity** means the lunar module is moving in reverse.
- Your **altitude** decreases by your **velocity** (since you calculate every second, and your **velocity** is in meters per second, you don't have to do a multiplication step for units of time)

1. Your **fuel** decreases by the amount specified by the player.

Note that it is possible that a player tries to "cheat" by specifying an illegal amount of **fuel** to burn. We want to reasonably prevent these errors as follows:

- If a player specifies burning zero **fuel**, this is fine (in fact, if they are out of **fuel**, this is their only choice).
- If a player tries to burn a negative amount of **fuel**, treat it as if they burnt zero **fuel**.
- If a player specifies to burn more **fuel** than they have, burn all their **fuel**.
- If a player specifies an invalid amount (i.e. a non-number) for **fuel**, print a friendly message and prompt the player to enter again.

The end game should specify:

- Whether the landing was safe or not, (a safe landing is one where the final **velocity** is < 10m/s)
- At what **velocity** the landing occurred,
- How many seconds the landing took, and

- How much **fuel** is left.

Additionally, the program should ask if the player wants to play again. Any response that begins with 'y' (capital or lowercase) should play again. Any response that begins with 'n' (capital or lowercase) means the user wants to exit. Any response that begins with any other character should ask the player again.

Submission

Your submission should include:

- *lunar_landing.py* - the source code for your game
  This file must include a header, in the form of a multi-line comment at the top of your script, that contains (each on its own line):
  - Your name
  - Your Penn ID
  - Statement of work. Either:
    - A list of resources you used and/or people you received help from (including TAs/Instructor)
    - A statement that you worked alone without help
  - A comma-separated list of inputs that leads to a "win"
  - A comma-separated list of inputs that leads to a "loss"

Evaluation

Correctness - 10 pts

Does the game work as expected? Did you follow the directions exactly? Is your math correct? Does the comma-separated list of inputs that leads to a "win" allow the player to win the game? (Please note that we will not be running your code on any horrible inputs.)

Comments - 5 pts

Did you add clear and descriptive comments to all non trivial lines of code?