ENGR 102 - LAB ASSIGNMENT #6B

Fall 2021

Remember to document your code with comments and print labels and units when applicable for this individual assignment. When these programs are completed, submit all files to Mimir. Remember the appropriate header information.

Deliverables:

1.) Lab6b_Act1.py
2.) Lab6b_Act2.py
3.) Lab6b_Act3.py

Program 1: Averaging Measurements

☑ *Use Python looping procedures to take user input.*
☑ *Calculate simple statistical data from a small set of measurements.*

Assume that someone has collected a set of measurements and wants some statistical data about them. Write a program that asks a user for measurements and prints the average, the maximum, and the minimum measurement. You are not to use the built-in min, max, or mean functions. Users should be allowed to enter as many measurements as they want, until entering a negative measurement. The negative measurement should not be processed, but is just used to indicate that the user has finished entering measurements.

Note: use a <u>while</u> loop for this task, and do <u>not</u> use a list to store measurements.

Program 2: Divisors

☑ *Create a program using Python looping procedures and math functions to calculate the desired output.*

For numbers from 2 to 100, print a series of lines indicating which numbers are divisors of other numbers. For each, print out "X divides Y", where X <= Y, and both X and Y are between 2 and 100. (*Hint: nested for loops*)

Example output (the first few lines will be):

```
2 divides 2
3 divides 3
2 divides 4
4 divides 4
5 divides 5
2 divides 6
3 divides 6
etc.
```

Program 3: The Angry Bird Decision

☑ *Plan and create a program that takes user input, uses Python looping procedures to calculate the required output, and format the output for the user.*

Red, our Angry Bird friend, will be shot from a catapult. You've been given the equation that projects his trajectory based on his initial velocity (v_0), his angle of launch (θ), and the acceleration due to gravity (g).

$$y = x \tan\theta - \frac{gx^2}{2{v_0}^2 \cos^2\theta}$$

You've been informed that the wall he *must* knock down is 202.4 m away.

a. Take user input for Red's initial velocity (for testing, start with 45 m/s).
b. For the input velocity, you want to figure out at what angle must your catapult be set to reach the wall at a height of 1 m (+/- 0.1m). Use Earth's gravity. Start at 0 degrees and increase the angle until the required condition is met. Not all initial velocities will have a solution; in these cases, tell the user there is no solution for Red.
c. Plot his trajectory. Use the example below for help in creating the plot. Do not create a plot if there is no solution for the velocity the user input.
d. This was on Earth, what if we move to Mars?
e. Report to the user the angle you calculated and the height of impact for both Earth and Mars.

Creating the plot of Red's trajectory:

From Week 1 you should have the matplotlib and numpy modules available in your PyCharm environment. We will be using plotting commands occasionally from this point forward, so please work with the teaching team if you still are having any issues with these modules.

Follow the general procedure below. We'll talk about what's going on here in much more detail in the near future; for now analyze what's given below, and modify necessary pieces to plot Red's trajectory.

This is an example. Some of the code below will need to be modified for your task/scenario:

```
import numpy
import matplotlib.pyplot as plt

# Create x-values for plot
lower_range_val = -2          # Lower bound for x-values. Change this value as needed.
upper_range_val = 1           # Upper bound for x-values. Change this value as needed.
xvals = numpy.arange(lower_range_val, upper_range_val, 0.01)   # Creates values with 0.01 spacing from
                                                               # lower_range_val to upper_range_val

# Creating y-values for plot (from a function)
yvals = 2*(xvals)**2 + 1*(xvals) + 4   # Evaluates function at every point defined by xvals. The function shown here
                                       # is an example.  This is not the function you will plot for this exercise.

# Create and show plot
plt.plot(xvals, yvals)                 # Creates line plot with yvals (vertical axis) plotted against xvals (horizontal axis)
plt.show()                             # Show the figure
```