ENGR 102 - Lab Assignment #5

Spring 2020

General course learning outcomes:

- demonstrate the use of basic programming techniques in the construction of computer programs, including: techniques to collect, store, and manipulate data within a computer program; use control structures, such as conditionals and loops, in computer programs.
- apply programming techniques to solve problems in engineering, including applying conditionals and loops to implement numerical methods, such as bisection and Newton's method.
- complete a team programming assignment that ties together concepts learned in the class.

Activity 1: Even Mr. Fusion needs updates - to do in lab (team)

☑ *Write a Python program to take user input and format output in an organized manner.*
☑ *Write a Python program that utilizes a count and multiple conditionals in a while loop.*

Once again, Marty McFly has traveled back in time and is trying to get back to the future. This time there are no terrorists shooting at him and he has plenty of fuel. However, the flight system is off-line and Marty is relegated to traditional acceleration via the four wheels. But, from the time he starts accelerating the Delorean time machine, he only has 10 seconds to reach a velocity of 88 mph before Mr. Fusion will shut down to apply 2024 very important Microsoft updates. (yes, they even plague Doc Brown.)

You are tasked with writing a program to help Marty calculate if he has enough road to achieve a velocity of 88 mph. He is on a 500 ft stretch of road adjacent to the clock tower. The Delorean will start from rest and accelerate uniformly, but Marty is worried about pushing the Delorean over an acceleration of 16 ft/s^2.

Distance traveled for a uniformly accelerating vehicle is shown below, where a is acceleration (ft/s^2), t is time (s), and v_0 is the initial velocity (ft/s):

$$v = at + v_0$$

$$distance = \tfrac{1}{2}at^2 + v_0 t$$

PLAN! then CODE! Develop the steps required for your program based on the following requirements. Remember good coding practice and comments.

1. Read in the acceleration (in ft/s^2) from Marty (the user).
2. You must use a while loop to calculate the time (s) needed iteratively, not in a direct calculation. That is, start at time = 0, and increase time with each iteration. *(Use at least 0.1s increments.)*
3. Keep looping until Marty either runs out of road, or has achieved 88mph.
4. Print the resulting time (in seconds) needed to reach 88mph in a descriptive and personalized statement; or print a warning statement if Marty will not achieve the needed velocity before Mr. Fusion's updates begin or he runs out of road. Give descriptive and helpful feedback to Marty.
5. For every 120 iterations of the while loop, one banana peel is needed to support the calculations. In a descriptive statement, print how many banana peels were used during the calculations.

(continued, next page)

Activity 2: Loop Root Finding - to do in lab (team)

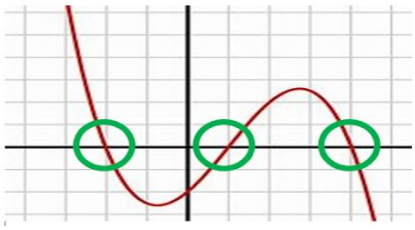
☑ *Design and create test cases for a program*
☑ *Create a program with well-defined variables and procedures for calculating the root of a polynomial.*
☑ *Apply conditionals and loops to implement the bisection method of root-finding.*

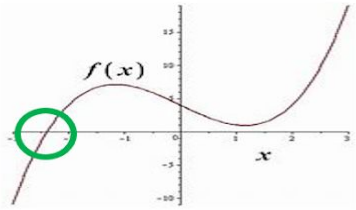
As a team, write a program to find a root of a cubic polynomial. A cubic polynomial is of the form:

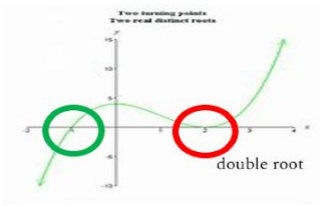
$$f(x) = Ax^3 + Bx^2 + Cx + D$$

A root of the polynomial is a value, x, such that $f(x) = 0$. For a generic cubic polynomial, there will be one local maximum, one local minimum, the curve will go off to negative infinity in one direction, and positive infinity in another.

Here are three examples:







The polynomial has some number of real roots, points at which the curve crosses the x-axis. A cubic curve will have either three roots (as above at left), one root (above, middle), or in rare cases 2 roots (as above at right). Note that roots are typically single roots where the curve is negative on one side of the root and positive on the other (a double root results in a tangent to the curve, like that above at right).

Your program should take as input the coefficients of the polynomial: A, B, C and D, along with a bound on one single root: *a*, *b*. Your program should report the value of the root, accurate to within 10^{-6}.

Note: The user is required to input a and b such that a < b, and such that the range contains exactly one single root of the polynomial.

A. PLAN! As a team, create a program that performs *bisection* (details below)* to determine the root. Develop the steps required for your program based on the following requirements:
   a. Read in the coefficients of the polynomial from the user, and an upper and lower bound around a single root of the polynomial
   b. Determine the value of that root to within 10^{-6}
   c. Print the value of the root found as a single number, and print how many iterations it took to find the root.

B. PLAN! Come up with at least four (4) test cases. These should define a curve (the 4 coefficients), as well as a large starting bound on the root (values of *a* and *b*). Your code should document the test cases as comments, near the top of the program (after your standard header).

C. CODE! Remember good coding practice and comments.

* *Bisection (Binary Search)*

Because you're looking for a single root, either f(a) or f(b) will be positive, and the other will be negative. (In other words, because a single root is a value on the x-axis where the curve goes across the axis, one side must be a positive value, and the other side must be a negative value.)

i. Find the value halfway between your initial interval [a, b]. Call this midpoint 'p'.
ii. Evaluate *f*(p). It will be either positive or negative, or (rarely) even the root itself, if *f*(p) = 0.
iii. Define a new interval, either [a,p] or [p,b]. Choose the interval which has *both* a positive and negative value.
iv. Repeat this process until the interval is less than 10^{-6}.

(continued, next page)

Activity 3: Plot that Polynomial - to do in lab (team)

☑ *Create a plot in Python.*

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.

Extend your program by using matplotlib to create a plot of the polynomial your user defined. Plot this function between the bounds (a, b) defined by your user (or as calculated if your team completed the challenge). Follow the general procedure below. We'll talk about what's going on here in much more detail in the future; for now analyze what's given below, and modify necessary pieces to plot the polynomial.

```
import numpy
import matplotlib.pyplot as plt

# Create x-values for plot
lower_range_val = -2        # Change as needed
upper_range_val = 1         # Change as needed
xvals = numpy.arange(lower_range_val, upper_range_val, 0.01)   # Values with 0.01 spacing from bounds a to b

# Create y-values for plot (from function)
yvals = 2*(xvals)**2 + 1*(xvals) + 4   # Evaluate function at every point defined by xvals. Change the function for your
                                       # case!

# Create and show plot
plt.plot(xvals, yvals)                 # Create line plot with yvals against xvals
plt.show()                             # Show the figure
```