Note: this has the flavor of a Track A "Mathematical Applications" assignment, but you are welcome to do this for credit whether you intend to complete Track A or Track B. You do not need to complete this assignment if you complete 2W instead.

PIC 16, Winter 2018 – Assignment 1F

(Week 1, Friday)

Assigned 1/12/2018. Code (a single `.py` file) due by the end of class 1/19/2018 on CCLE. *Also* hand in a printout of this document with the self-assessment portion filled out by the end of class on 1/19/2018.

In this assignment, you will write a Python program to solve an equation using Newton's method (AKA the Newton-Raphson Method), an iterative procedure for finding a root (zero) of a function.

Algorithm (to be implemented in Python)

Given a function $f(x)$, an initial guess x_0 of the solution to the equation $f(x) = 0$, and a desired tolerance ϵ to which the equation is to be satisfied (i.e. $|f(x)| \leq \varepsilon$), perform the following:

- Calculate the "residual" (error) of your guess $f(x_i)$.
- If the magnitude of the residual is less than or equal to ε, i.e. $|f(x_i)| \leq \varepsilon$, you are done because x_i is a good approximation of the root! Otherwise,
- Calculate a correction to your guess $\Delta_i = -\frac{f(x_i)}{f'(x_i)}$ (where $f'(x)$ is the function's derivative) and update your guess according to the formula $x_{i+1} = x_i + \Delta_i$.
- Repeat.

(Which did you find more helpful? Should I list only one?)

See here for an example of Newton's method in action and here for more about the theory. The equations might look slightly different in the video compared to those above, but the algorithm is exactly the same. Note that the speaker in the first video stops when $f(x) = 0.002$ because he gets tired of repeating the process. Since you won't be doing this by hand, you'll have the program continue until $|f(x)| \leq 0.0001$.

You can hard-code (`def`) the function, its derivative, the initial guess, and the tolerance in your code; you don't need to get user input. The only required output (`print`) of your code is *one* root of the function. Test your code with the following functions $f(x)$, their derivatives $f'(x)$, and initial guesses x_0:

$$f(x) = x^2 - 1, f'(x) = 2x, x_0 = 3$$

$$f(x) = x^2 - 1, f'(x) = 2x, x_0 = -0.5$$

$$f(x) = \sin(x), f'(x) = \cos x\,, x_0 = 2$$

$$f(x) = \ln(x) - 1, f'(x) = 1/x, x_0 = 1.5$$

We haven't covered how to use math functions, so search the internet. Here's a hint: you'll need to `import math` before calling any math functions or constants, then call them like `math.sin(math.pi/2)`. We'll talk more about importing "modules", like the `math` module, later.

Self-Assessment

Write the roots *provided by your code* next to each example above. Even though multiple roots may exist, Newton's method returns at most one per initial guess, so you only need to write one. Use a calculator to check the roots provided by your Python code and indicate which were correct/incorrect above.
If you can't get your code working for any of the cases above, there is no partial credit; you get 0%. Ask course staff for help to make sure that doesn't happen! If your code works for some of the functions above, but not all, you'll get 75% - there's probably a simple bug. If it works for all the cases above, you'll get 100%. Please write your expected grade for this assignment here:

If you have time, put the algorithm in a "`solve`" function that can be called like:
`print solve(lambda x: [x**2-1, 2*x], 3, 0.0001)`