Homework #2 **Due: 1/28**

1. [50 points] A *root* of a function $f(x)$ is a value of x for which $f(x)$ is zero. For example, the function $f(x) = x^2 - 2$ has two roots: $\sqrt{2}$ and $-\sqrt{2}$.

   Write a Python module `rootfinder` (in a file named `rootfinder.py`, of course) that implements a rootfinding algorithm. It should export a function

   ```
   root_find(f, xlo, xhi, eps)
   ```

   that allows the user to provide an arbitrary function `f()` and two guesses, `xlo` and `xhi`, that are assumed to bracket a root. When two successive approximations of the root are within `eps` (usually a very small positive value) of each other, return the most recent approximation.

   You may implement any rootfinding technique you know. One that works pretty well in this case is called "false position", or *regula falsi* (or, in Pig Latin Latin *egularay alsifay*). This is well documented in most numerical analysis books, or you can look on the Web (courtesy of Wolfram Research) at

   ```
   http://mathworld.wolfram.com/MethodofFalsePosition.html
   ```

   Note that *regula falsi* further requires that `f(xlo)` and `f(xhi)` have differing signs. (In the self test specified below, they do.)

   If you find *regula falsi* too hard to implement, you may use any other established root finding method you know, except that doing a linear search from `xlo` to `xhi` in steps of `eps` (or any equivalent) will offend the grader's sensibilities and cause major points to be taken off.

   As a self test, make your module find (an approximate value for) the root of the polynomial

   $$-5x^3 - 4x^2 + 3x + 2$$

   that lies between 0 and 1 to six decimal places.

2. [50 points] Use your `rootfinder` module to implement a script `findroot` that concatenates all of its command line arguments (except, of course, the program name) into an expression for `f()` so that you have a rootsolver that works from the command line. You may continue to assume that `xlo` is 0, `xhi` is 1, and `eps` is 10^{-6}.

   You must use the `rootfinder` module implemented in Part 1 without modifying it.