```
%% Class 3
classdef set_3
  methods(Static)
  %% Problem 1: Secant Method
    %
    % * input command: set_3.secant_method(13, 16, .001)
    % * output: 14.2325
    %

    function [sol] = secant_method(x_p, x_n, err_acc)
      error = abs(x_n- x_p) / x_n;
      function[y] = fxn(x)

        y = atan(exp(sin(1 / (x + 20)))) - .8;

      end



      while (error > err_acc)
        x = [x_p x_n];
        f = [fxn(x_n) -fxn(x_p)]';
        sol = (x*f) / (f(1) + f(2));
        error = abs(x_n- x_p) / x_n;
        x_p = x_n;
        x_n = sol;

      end

    end
    %% Problem 2: Fixed Point
    %
    % * input command: set_3.fixed_point(3, .001)
    % * output: 2.5222
    %

    function [x_root] = fixed_point(x_0, err_accept)

      error = 100;

      % If I understand the function's alg correctly.  We must solve
      % for x using the highest degree polynomial of x since it will
      % create a decreasing funciton as x, in a particular set of
      % directions, increases so solving for x = .5( x^3 - 11) and
      % x = (2*x + 11) / (x^2) could cause issues
      function y=gxn(x)

        y = (2*x + 11)^ (1/3);
```

```
    end



    while (error > err_accept)
        x_root = gxn(x_0);

        error = abs(x_root - x_0) / x_root;

         x_0 = x_root;

    end

    x_root = x_0;
end


%% Problem 3: Second Dimension Error
% * input command:
% >>     b2 = .9:.01:1.1;
% >>    set_3.solution_plot(b2)
% * output:
%
%
% <<../../sub/p3.png>>
%
%
function solution_plot(b2)

    x1 = -841 + 842 * b2;
    x2 = 921 - 922 * b2;
    x3 = 3 - 3 * b2;

    f1 = figure(1);
    hold on


    plot(b2, x1, '-r');

    plot(b2, x2, '-g');

    plot(b2, x3, '-b');

    legend("x1", "x2", "x3");

    hold off

    %uiwait(f1);
```

```
    end
  end
end
```