Root finding with Newton's method

An algebraic expression has roots. Finding real roots (or no root) of a linear equation (y=mx+c) is easy. Finding roots of a quadratic equation ax^2 + bx + c = 0 can sometimes produce complex roots. Finding roots to a cubic equation can sometimes be easy for simple cases that can be factored into simpler terms. Finding root to a 4th degree equation ... well maybe. And that's the end of it. By one of the fundamental mathematical postulates, one can never find roots to a 5th degree (or higher) equation using the basic operation of addition, subtraction, multiplication and division. As disappointing as it sounds, a vast majority of higher order equation do have real roots, the only question is how does one find them with simpler methods.

The Newton's method is one such root finding method for a given function f(x). We start with an initial guess value (x0), compute f(x0) and f'(x0) to **estimate** the location of the next **x.** Then use this new x, let's call it x1 to determine f(x1) and f'(x1), so that the next estimate of **x** can be estimated. Then use this new x, call it x2 to determine f(x2) and f'(x2), and so on. This can go indefinitely until we arbitrarily stop the "loop" provided the next **x** turns out be to nearly the same as the previous **x**. We call it "converging" to a estimate of **x** that can be safely assigned as one of the real roots of the equation. Seems too arbitrary? Try a simple example with pen and paper.

1. Write down a simple linear function f(x) = x. We know that 0 is the root of this equation.
2. Start with an initial value x0 = 1. Find f(1) and f'(1)
3. Find x1 = x0 - f(1)/f'(1), which should be x1 = 1 - 1/1 = 0.
4. x1 = 0, which is the exact root of y=x.

Try another example f(x) = x^2 - 3x + 2. We know the roots for this quadratic equation are 1 and 2.

1. Start with x0 = 0. You can pick any value you want but sometimes you may get 1 or 2 depending on where you begin. That's the only downside of this method.
2. Find f(0) and f'(0) -> f(0) = 2, f'(0) = -3
3. Find x1 = x0 - f(0)/f'(0), which should be x1 = 0 - 2/(-3) = 2/3.
4. Compute f(x1), f'(x1), i.e. f(2/3), f'(2/3)
5. Find x2 = x1 - f(x1)/f'(x1),
6. Repeat it 3 more times and the subsequent values of x will be estimated root.

```
% Clear screen and variables
clc; clear;

max_step = 5;

x0 = 1; % set starting value to 1
root = Newton(??,??);
```

```
step    x
   1   1.0000
   2   1.0000
   3   1.0000
   4   1.0000
   5   1.0000
```

```
% Try again, this time with 2
x0 = ??;
root = ??;
```

```
step    x
   1   2.3333
   2   2.0667
   3   2.0039
   4   2.0000
   5   2.0000
```

```
% And try again, this time with x0 = 300. Did you get the
% result in 5 steps? If not, then what different
% can you do to get either 1 or 2 as the estimated root?
x0 = 300;
max_step = ??;
root ??
```

```
step    x
   1 150.7504
   2  76.1260
   3  38.8147
   4  20.1607
   5  10.8370
   6   6.1819
   7   3.8677
   8   2.7366
   9   2.2194
  10   2.0335
  11   2.0010
  12   2.0000
  13   2.0000
  14   2.0000
  15   2.0000
```

```
function [root] = Newton(??,??)

%% NEWTON Determines root using Newton's method
%% Requires initial value x0 and max_step; returns root.

    % Print header (Each col is 4 space wide)
    fprintf('%? %? \n', 'step', 'x');

    % Set x = initial x, i.e. x0
    x = ;

    % Run loop for 1 to max_step
    for ??
        % Evaluate fx = x^2 - 3x + 2
        fx = ;
        % Evaluate dfx
        dfx = ;

        % Find x_new from x, fx, dfx using the
```

```
        x_new = x - fx/dfx;

        % Update x with x_new
        x = ;

        % Print step number (i) and x.
        % Col1: integer, 4 spaces wide
        % Col2: float, 8 spaces wide with 4 decimal places

        fprintf('??\n', ??, ??);
    end

    root = x;

    fprintf('\n\n');
end
```