```
%{
A recursively defined function for computing square roots using Heron
function [Sqr_out,diff]=my_sqrt(x_in,x_0,num)
format long;
Sqr_out=x_0;

if num==0 %end the recursion when at depth zero
    Sqr_out=x_0;
    diff=abs(floor(log10(abs(Sqr_out-sqrt(x_in)))));
else %not at depth zero then recall our function
    Sqr_out=0.5*(my_sqrt(x_in,Sqr_out,num-1)+x_in/my_sqrt(x_in,Sqr_out,num-1));
    if Sqr_out-sqrt(x_in)>=1
        diff=0;
    else
        diff=abs(floor(log10(abs(Sqr_out-sqrt(x_in)))));
    end
end
end
%}
function [Sqr_out, err, nIter,nAgree,L1,L2,L3,L4]=my_sqrt(x_in,x_0,tol)
format long;
nIter=0;
err=abs(x_in-x_0^2);
Sqr_out=x_0;
nAgree=count_zeros(abs(Sqr_out-sqrt(x_in)));
L1=[];
L2=[];
L3=[];
L4=[];
L1=[L1,Sqr_out];
L2=[L2,err];
L3=[L3,nIter];
L4=[L4,nAgree];
while err > tol
    Sqr_out=0.5*(Sqr_out+x_in/Sqr_out);
    err=abs(x_in-Sqr_out^2);
    nIter=nIter+1;
    nAgree=count_zeros(abs(Sqr_out-sqrt(x_in)));
    L1=[L1,Sqr_out];
    L2=[L2,err];
    L3=[L3,nIter];
    L4=[L4,nAgree];
end
end

function nZeros=count_zeros(dec_in)
format long;
nZeros=0;
x=dec_in;
while x<1
    nZeros=nZeros+1;
    x=x*10;
end
end
```