Notes / Road Map:

By you

def Richardson(f = name of func,x = point, can also add params like eps = accuracy wanted):

...calculate dfdx = ...

...[have to do extrapolation and junk, as a black box method...]

return dfdx, nsteps, error

By user

def f(x):

...

return value

Notes on h and derivs:

D(h) shows you the importance of picking the step h and how to change your step to optimize.

D(h) = [f(x+h) - f(x-h)] / (2h) -> the "work horse"

But we don't know the optimal h!

D(h) = D(0) + ... h^2 + ... h^4 + stuff

D(0) means lim[D(h)], where h -> 0 = df/dx

So how to take D(h) and have it approach D(0).

D(0) = D(h) - higher order derivs we don't have...

Let's plot is D(h) vs. h\_c

So we choose some starting h\_0 = 0.1 \* x\_c, where x\_c = f(x) / f'(x), so let's say x\_c ~ x

So then we take h\_n = h\_0 / 2^n

For plot / interpolation, need to start with say 4 points.

Now create interpolating function (eg. scipy)

Now extrapolate that out to 0.

But still can't get higher order derivs. Can we get these higher order derivs?

We want to calculate deriv to specific accuracy.

We want |df/dx - D(0)| < epsilon ~ 10^-6, 10^-3, or 10^-10

But what is proxy for D(0)?

1) Look at diff between D(h = 0) - D(h = h3), how much we had to extrapolate!

2) Richardson's way: Let's add a point, h = h4. Now we have 5 points, update the interpolating function, and then get a new D(0)

-> Subtracting dfdx(n-1) - dfdx(n), where dfdx(n) = D\_n(0)

If no convergence after ~20-50 iterations say no convergence...stop routine and say "Routine did not converge, would recommend using different accuracy"