AMath 483/583 — Lecture 3

This lecture:

- computing square roots
- Python demo
- git demo

Reading:

- class notes: git section
- class notes: Python sections (new)

Notes:

Computing square roots

Hardware arithmetic units can add, subtract, multiply, divide.

Other mathematical functions usually take some software.

Example: Compute $\sqrt{2} \approx 1.4142135623730951$

In most languages, sqrt(2) computes this.

```
>>> from numpy import sqrt
>>> sqrt(2.)
```


Notes:

One possible algorithm to approximate $s = \sqrt{x}$

```
s = 1.    # or some better initial guess
for k in range(kmax):
    s = 0.5 * (s + x/s)
```

where `kmax` is some maximum number of iterations.

Note: In Python, range(N) is $[0,\ 1,\ 2,\ ,\ldots,\ N-1]$.

Why this works...

If $s < \sqrt{x}$ then $x/s > \sqrt{x}$

If $s > \sqrt{x}$ then $x/s < \sqrt{x}$

In fact this is Newton's method to find root of $s^2 - x = 0$.

Notes:

Newton's method

Problem: Find a solution of $f(s) = 0$ (zero or root of f)

Idea: Given approximation $s^{[k]}$,
approximate $f(s)$ by a linear function,
the tangent line at $(s^{[k]},\ f(s^{[k]}))$.

Find unique zero of this function and use as $s^{[k+1]}$.

Updating formula:

$$s^{[k+1]} = s^{[k]} - \frac{f(s^{[k]})}{f'(s^{[k]})}$$

Notes:

Demo...

Goals:

- Develop our own version of `sqrt` function.
- Start simple and add complexity in stages.
- Illustrate some Python programming.
- Illustrate use of git to track our development

We will do this in `$UWHPSC/lectures/lecture3` directory
so you can examine the various versions later.

Notes: