Approximating $\sqrt{n}$ with the Babylonian Method

Motivation

This lab shows how to encode and solve a recurrence relation from mathematics using iteration in Python to approximate a real-world, real valued function. It also teaches how to quickly prototype something in Excel (if warranted).

Discussion

To approximate square root, $\sqrt{n}$, the idea is to pick an initial estimate, x_0, and then iterate with better and better estimates, x_i, using the recurrence relation:

$$x_{i+1} = \frac{1}{2}(x_i + \frac{n}{x_i})$$

To see how this works, jump into Excel (yes, a spreadsheet) and crank through a few iterations by defining cells with n and your initial estimate x_0, which can be anything you want. (It's sometimes easier to play around without having to deal with a programming language.) Then you need to define a cell that computes the above better approximation using x_i as the cell above it. I hardcoded the names in column A and the first two rows of column B. Cell B3 should be a formula that computes B4 based upon B3. Then you can extend the formula down and watch it converge on the final (correct) value for $\sqrt{125348}$. My spreadsheet looks like this:

	A	B
1	n	125348
2	x_0	20
3	x_1	3143.7
4	x_2	1591.78638
5	...	835.266564
6		492.668011
7		373.547461
8		354.554285
9		354.04556
10		354.045195
11		354.045195

Try out any nonnegative number and you'll see that it still converges, though at different rates.

There's a great deal on the web you can read to learn more about why this process works but it relies on the average (midpoint) of x and n/x getting us closer to $\sqrt{n}$. It can be shown that if x is above $\sqrt{n}$ then n/x is below $\sqrt{n}$ and the reverse is true if x is below the root. The iteration converges and does so quickly. Informally, as shown in Wikipedia, we can represent the true square root by adding an error term to our estimate:

$$\sqrt{n} = x + \epsilon$$

or,

$$n = (x + \epsilon)^2$$

Expanding, we get:

$$n = x^2 + 2x\epsilon + \epsilon^2$$

Solving for ϵ:

$$n - x^2 = \epsilon(2x + \epsilon)$$

$$\epsilon = \frac{n - x^2}{2x + \epsilon}$$

Because ϵ is much smaller than x, we can drop it from the denominator leaving us with an estimate of epsilon:

$$\epsilon = \frac{n - x^2}{2x}$$

Then we can plug it back into $x + \epsilon$ and get:

$$x := x + \epsilon = x + \frac{n - x^2}{2x} = \frac{2x^2}{2x} + \frac{n - x^2}{2x} = \frac{1}{2}\frac{x^2 + n}{x} = \frac{1}{2}(x + \frac{n}{x})$$

Which gets its back to the Babylonian formula. Since we dropped an ϵ term, this formula for x is inexact but it gets us closer to $\sqrt{n}$.

Now that you understand how this estimate works, your goal is to implement a simple Python method called sqrt() that uses the Babylonian method to approximate the square root. File `sqrt.py` is in you are repository with some starter code. You will also find unit tests in `stats/test_sqrt.py`, which you can run with:

```
$ python -m pytest test_sqrt.py
```

You may not use math.sqrt() for implementing your function, but you may use it for testing the results. Obviously.

Deliverables. Make sure that `stats/sqrt.py` is correctly committed to your repository and pushed to github.