Daily Coding Problem #129

Problem

Given a real number n, find the square root of n. For example, given `n = 9`, return 3.

Solution

This is a classic question that was first solved by Heron of Alexandria after the first century.

Alexandra's algorithm starts with a `guess` and iteratively improves until convergence. In each iteration, we improve our `guess` by averaging `guess` and `n / guess`. This formula comes from the fact that if `guess` is an overestimate, then `n / guess` would be an underestimate. For example, If n is 9, then a `guess` of 4 is an overestimate and `9 / 4` is an underestimate. On the other hand, if `guess` is an underestimate, then `n / guess` is an overestimate. The process converges when `guess` is3 which is equal to `9 / 3`. For the full proof, please see here.

```
def squareroot(n, error=0.00001):
    guess = 1

    while abs(guess ** 2  - n) >= error:
        guess = (guess + n / guess) / 2.0
    return guess
```

A more realistic answer, in an interview setting, would be to use binary search. We can pick an underestimate `lo = 0` and an overestimate `hi = n` to start. And we can keep the loop invariant that the true `squareroot(n)` would always lie between `[lo, hi]`. To do this, we see if `guess = (lo + hi) / 2` is an overestimate, and if it is, bring the `hi` down to `guess`. Otherwise, we bring the `lo` up to `guess`. The loop finishes when `guess ** 2` is very close to `n` (plus or minus `error):

```
def squareroot(n, error=0.00001):
    lo = 0.0
    hi = n
    guess = (lo + hi) / 2.0

    while abs(guess ** 2  - n) >= error:
        if guess ** 2  > n:
            hi = guess
        else:
            lo = guess
        guess = (lo + hi) / 2.0

    return guess
```

Privacy Policy

Terms of Service

Press