Homework 2

Due Friday, October 8, 2021

Math 205

1. You can find $\sqrt{73} = 8.54400374531753\ldots$ in Python using

   ```
   x = 73**.5
   print(x)
   ```

   and you can find the closest integer using

   ```
   print(round(x))
   ```

   How can you find the closest rational number to $\sqrt{73}$ that's of the form $m/2$? Of the form $m/3$? $m/4$?

2. Write a loop starting with

   ```
   for n in range(1,51):
       [blah blah blah]
   ```

   where for each n, you find the closest rational number to $\sqrt{73}$ that's of the form m/n; then you print the numerator m, the denominator n, the decimal expansion of m/n, and difference between m/n and $\sqrt{73}$ so you can see how close you're getting.

3. Of course you can get better and better rational approximations to $\sqrt{73}$ by allowing bigger and bigger denominators:

   $$\frac{8}{1} = 8 \qquad \frac{85}{10} = 8.5 \qquad \frac{854}{100} = 8.54 \qquad \frac{8544}{1000} = 8.544 \qquad \ldots$$

   But some denominators give unexpectedly good approximations, in the sense that they get much closer to $\sqrt{73}$ than other nearby denominators. Do you see some examples of this in your list?

4. You could say that $299/35$ is a better approximation to $\sqrt{73}$ than $94/11$, because the difference is less. But the difference is only about 20% less, at the cost of a denominator that's more than three times as big, so in another sense you could say that $94/11$ is better.

   How could you make this second sense of "better" precise? We want a statement like "m_1/n_1 is better than m_2/n_2 if [some quantity obtained from m_1 and n_1] is less than [the same quantity obtained from m_2 and n_2]," and the quantity is something that the computer can find pretty easily.

5. Write a `for` loop that does the following: for each n from 1 to 100, if the rational approximation m/n was better (in the sense of problem 4) than all previous approximations, then print m and n, and the decimal expansion of m/n, and if there's any other information you're interested in, print that too. Otherwise continue without printing anything. Once that's working well, go up to 10000.

6. Try the same thing with the square root of another two-digit number. Try the cube root of a two- or three-digit number. Try e and π using

   ```
   from math import e, pi
   ```

7. Speculate: Do you think that for every x there is some *very best* rational approximation m/n, such that no approximation with any bigger denominator is better in the sense that we're discussing? Or do you think there's always a better one further down the line? We'll return to this next week.