CMPSC 102
Discrete Structures
Fall 2018

Practical 2: Calculating Cube and Forth Roots of n
14^{th} Sept 2018

Refer to your notes, slides and sample Python code from this week.

Summary

In this practical, you will be creating Python code to calculate cube- and forth-roots of a value (n) using Newton's Method. Note: your code will very similar to the Python code previously discussed in class that approximated the square root of n. Use that code, in addition to your sides from this week, to complete this practical.

GitHub Starter Link

https://classroom.github.com/a/lzNvpaBt

To use this link, please follow the steps below.

- Click on the link and accept the assignment.
- Once the importing task has completed, click on the created assignment link which will take you to your newly created GitHub repository for this lab.
- Clone this repository (bearing your name) and work on the lab locally.
- As you are working on your lab, you are to commit and push regularly. You can use the following commands to add a single file, you must be in the directory where the file is located (or add the path to the file in the command):
  - `git commit <nameOfFile> -m ‘‘Your notes about commit here’’`
  - `git push`

  Alternatively, you can use the following commands to add multiple files from your repository:
  - `git add -A`
  - `git commit -m ‘‘Your notes about commit here’’`
  - `git push`

Some Background on Square Roots

The general equation for the approximation of a square root is the following.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(n)}$$

Table 1: The breakdown of calculations necessary to approximate the square root of $a = 2$.

$a = 2$ (find square root of a)	$f(x) = x^2 - 2$ (function)
$x_1 = 1.0$ (initial guess)	$f'(x) = 2x$ (derivative)

Using the details of Table 3, we are able to calculate the first step in the approximation of the square root, $x_1 = 1.5$, which is then placed back into the general equation as shown in Table 2. Note: You might find it helpful to find the third power of a number, n, by using Python's syntax, $n**3$. For example, $2**3 = 8$.

$$\begin{aligned} x_1 &= 1.0 - \frac{f(1.0)}{f'(1.0)} \\ &= 1.0 - \frac{(1.0)^2 - 2}{2 * (1.0)} \\ &= 1.0 - \frac{1.0 - 2}{2} \\ &= 1.0 - \frac{-1.0}{2} \\ &= \frac{3.0}{2} \\ &= 1.5 \end{aligned}$$

Table 2: The steps for approximation.

Guess			Approx. root
x_n	$f(x) = x_n^2 - 2$	$f'(x_n) = 2x$	$x_n - \frac{f(x_n)}{f'(x_n)}$
1	-1	2	$1 - \frac{-1}{2} = \frac{3}{2} = 1.5$
$\frac{3}{2}$	$\frac{1}{4} = 0.25$	3.0	$\frac{3}{2} - \frac{(\frac{1}{4})}{3} = \frac{17}{12} = 1.4167$
$\frac{17}{12}$	$\frac{1}{144}$	$\frac{17}{6}$	$\frac{17}{6} - \frac{\frac{1}{144}}{\frac{17}{6}} = \frac{577}{408} = 1.4142$

What to do for this practical

You are to locate the empty source code file `src/rootFinder.py` into which you are to place code to complete this task. In this practical, you are to write working Python code to approximate the cube and forth roots of an arbitrarily chosen value a. Use the logic and structure of the square root calculation code from class to complete your code. There will be three functions for this work; one for the square root `NM2()`, one for the cube root `NM3()`, and one for the forth root `NM4()`. Your output will look like that of Figure 1.

As an extra challenge, you could design an input system to ask the user for a value for which a root is calculated.

Table 3: The functions, derivatives and values to approximate the roots of some value, a.

$\sqrt[3]{a}$	$a = 175616$ (find cube root of a)	$f(x) = x^3 - a$ (function)
	$x_1 = 1.0$ (initial guess)	$f'(x) = 3x^2$ (derivative)
$\sqrt[4]{a}$	$a = 9834496$ (find forth root of a)	$f(x) = x^4 - a$ (function)
	$x_1 = 1.0$ (initial guess)	$f'(x) = 4x^3$ (derivative)

General guidelines

Click on the repository link above in red. You will find the broken Python code in the `src/` directory of your repository. You are to reassemble this code, determine where the tabs should be found and then test your work by manually calculating root values. Alternatively, you could use Python's built-in functions to test your outputs. For example, if you wanted to check your calculations for $\sqrt[3]{27}$ and $\sqrt[4]{256}$), your Python shell commands would look like the following.

```
Finding square root of :  2
 Initial values:  n =  2 and guess =  1.0
 Square root result :  1.4142156862745099

Finding cube root of :  175616
Approx guess :  1.0
 Initial values:  n =  175616 and guess =  1.0
 Cube root result :  56.00000000040617

Finding forth root of :  9834496
 Initial values:  n =  9834496 and guess =  1.0
 Forth root result :  56.0
```

Figure 1: The output of the program to compute the cube and forth root approximations of values, 175616 and 9834496, respectively. You should try other numbers and add print statements to your code to get an idea about how the approximations are converging to the actual value of the root.

```
Python 3.6.5 (default, Apr  1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> print(math.pow(27,1/3))
3.0
>>> print(math.pow(256,1/4))
4.0
```

Once you are satisfied with the correctness of your work, push the code to GitHub using the commands written at the beginning of this document.