Activities: Module 3

August 30, 2019

1. The solutions of a quadratic equation $ax^2 + bx + c = 0$ are given by

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

so the solution can be a complex number is the number under the square root is negative.

Write a program that asks the user to enter the values for a, for b, and then for c. The program then prints the two solutions. Use exponentiation by 0.5 to calculate the square root. If the "discriminant" $b^2 - 4ac$ is negative, Python will happily calculate and return complex solutions.

Here are two trial runs:

```
enter a 2
enter b 2
enter c 2
(-0.49999999999999994+0.8660254037844386j) (-0.5-0.8660254037844386j)
```

gives two complex solutions, whereas

```
enter a 1
enter b -5
enter c 6
3.0 2.0
```

gives to real solutions.

2. The Ralph Newton method allows for the fast approximation to the solutions of algebraic equations. Its derivation is a topic in the Calculus sequence, but we can use it here without knowing the rationale. To calculate the cubed root of a number a, we start with an initial guess x_0 and then improve on the guess by calculating

$$x_{n+1} = (2x_n + a/x_n^2)/3.$$

Write a script that sets the variable a to a value entered by the user, that then sets the initial guess x_0 = 1, and then prints out the first ten successive improvements x_n according to the formula above as well as their cubed root.

The output with $a = 7$ should be

```
enter a number: 7
1 1
3.0 27.0
2.259259259259259 11.531829497535941
1.963308018221572 7.567724629348746
1.9142127541656009 7.014078412195178
1.9129320405969417 7.000009417131073
```

```
1.912931182772774 7.000000000004225
1.912931182772389 6.999999999999998
1.912931182772389 6.999999999999998
1.912931182772389 6.999999999999998
1.912931182772389 6.999999999999998
```

You can see that we need urgently a way to repeat the same instruction

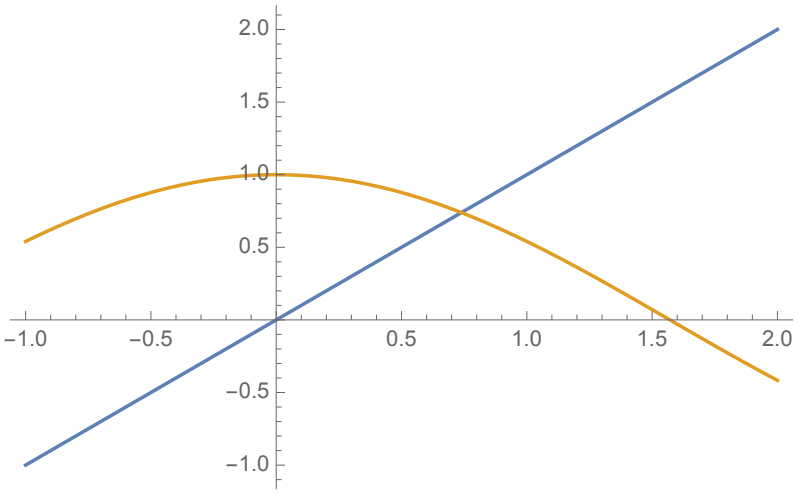
```
x = (2*x+a/x**2)/3
```

as well as a way to automatically decide that we have had enough iterations. This will be the topic of next week.

3. To use trigonometric functions, we use the math module. The first line of our script says

```
import math
```

and then we can obtain the sine of a variable `x` by saying `math.sin(x)` and the cosine of a variable `x` by saying `math.cos(x)`. Assume that we want to solve the equation `x = cos(x)`. As you can see from plotting, it has only one solution.



Raphson Newton approximates the solution by starting with a guess $x = 1$ and then improving the guess by updating x to

$$x_{\text{new}} = x - \frac{x - \cos(x)}{1 + \sin(x)}.$$

Write a script that prints the first 7 guesses and the cosine of these guesses. Here is what I got:

```
1 0.5403023058681398
0.7503638678402439 0.7314407940181265
0.7391128909113617 0.7390664350123708
0.7390851333852839 0.7390851331005635
0.7390851332151606 0.7390851332151607
0.7390851332151607 0.7390851332151607
0.7390851332151607 0.7390851332151607
0.7390851332151607 0.7390851332151607
```