Exercise A

(a) In the second lecture and first tutorial, we saw the iteration rule

$$x^{(k+1)} = \phi(x^{(k)}) = \frac{1}{2}\left(x^{(k)} + \frac{a}{x^{(k)}}\right) \tag{1}$$

Show that this is actually the Newton-Raphson iteration for solving

$$f(x,a) = x^2 - a = 0, \tag{2}$$

where $a \geq 0$. That is, the code you wrote in the first tutorial is actually an implementation of the Newton-Raphson method for solving equation (2). Submit your work to GitHub Classroom, preferably using LaTeX.

(b) Show that if $x^{(k+1)} = \phi(x^{(k)}) = x^{(k)}$ then $x^{(k)}$ is the solution to equation (2). That is, Newton-Raphson iteration would give a Newton-Raphson step $\delta x = 0$ if the initial guess is exact ($x = \sqrt{a}$). Submit your work to GitHub Classroom, preferably using LaTeX.

(c) Modify your bisection code (from the `2072U-Course-Codes` repo) to find an approximate solution to $f(x,a) = 0$. Choose appropriate values for the parameter a, the initial boundaries a_0 and b_0, the maximal number of iterations $k_{\max}$, the error tolerance ϵ_x and the residual tolerance ϵ_f. Hint: for $a > 1$, $0 < \sqrt{a} < a$ and for $a < 1$, $0 < \sqrt{a} < 1$. Submit your commented code to GitHub Classroom.

(d) Instead of using your code from tutorial 1, implement the Newton-Raphson method for equation (2), by writing a script that calls the Python function `NewtonRaphson` that can be found in the `NewtonRaphsonIteration.py` file. A starter code `NewtonRaphsonMethod.py` is included in Tutorial 2 repository. Submit your commented code to GitHub Classroom.

(e) Modify the codes from part (d) to implement the Secant Method for solving equation (2). In particular, modify the code found in `SecantIteration.py` (which is essentially a copy of `NewtonRaphsonIteration.py` with some hints) so that it computes Secant iterations rather than Newton-Raphson iterations, and write a code `SecantMethod.py` for solving equation (2); hint: modify your `NewtonRaphsonMethod.py` code so that it calls the secant iteration function `SecantIteration` (defined in the file `SecantIteration.py`). Submit your commented code to GitHub Classroom.

(f) Compare the convergence rate of the three methods, e.g. plot number of iterations vs. approximate error of all three methods on the same graph. Try using a semi-log plot. Hint: you'll have to modify your iteration functions to save and return the information from all iterations. Finally, discuss which method is more efficient for approximating $\sqrt{a}$. In particular, suppose every elementary computation (multiplication, division, addition, subtraction) costs you \$10, and you need to compute $\sqrt{5}$ to 16 digits, which method would you choose? Have one member of your group post the graph and your group's discussion of your solutions on Slack.

Exercise B: A mystery script

Have a look at the script `mystery_script.py` contained in the tutorial repository. Answer the following questions as accurately as possible.

- What does this script do?
- What is the meaning or the role of the variables used in the script? Add comments whenever you understand one of them. Submit your commented code to GitHub Classroom.
- What question can you answer by running this script (playing around with the parameters)? One member of your group should post your group's discussion of your answer.

Once you have answered these questions, formulate what you have learned about Newton-Raphson iteration.