Exercise 1.36. Modify `fixed-point` so that it prints the sequence of approximations it generates, using the `newline` and `display` primitives shown in exercise 1.22. Then find a solution to $x^x = 1000$ by finding a fixed point of $x \mapsto {}^{\log(1000)}/_{\log(x)}$. (Use Scheme's primitive `log` procedure, which computes natural logarithms.) Compare the number of steps this takes with and without average damping. (Note that you cannot start `fixed-point` with a guess of 1, as this would cause division by $\log(1) = 0$.)

Answer. To print the sequence of approximations procedure generates, a simple strategy would be asking the evaluator to send out the new guess as soon as it is generated. Hence, all we need to do is adding the printing module at the very begining in the body of procedure `try`

```
(define (fixed-point f first-guess)
  (define (close-enough? v1 v2)
    (let ((tolerance 0.00001))
      (< (abs (- v1 v2)) tolerance)))
  (define (try guess)
    (let ((next (f guess)))
      (display next)
      (newline)
      (if (close-enough? guess next)
          next
          (try next))))
  (try first-guess))
```

Aimed with procedure `fixed-point`, we have little difficulty in finding the root of equation $x^x = 1000$ through a procedure

```
(fixed-point (lambda (x)
               (/ (log 1000)
                  (log x)))
             2.0)
```

It takes the evaluator 33 steps to obtain the result in this naive practice[1]. However, there is a much more efficient way to reach the same goal

```
(fixed-point (lambda (x)
               (average (/ (log 1000)
                           (log x))
                        x))
             2.0)
```

Obviously, with only 9 steps our `fixed-point` procedure which uses averaging damping converges much faster than the former way.

---

1. For the sake of aesthetics, the tedious output generated by the interpreter is not presented here. It is located in file *Test_for_Exercise_1.36.scm*.