Previously, we considered the function of finding the square root of a positive number. Our function was based on the bisection method, and it requires the elements to be ordered. When we did the guess, it could be either less or more of the desired result, based on this, we changed the lower or upper bound.

However, this method issues an error when trying to calculate a square root of number that is less than one.

We wrote a function with the tests of our program, we launched and studied the result, the program worked correctly for 4 and for 2

However when we tried to take a square root of 0.25. The program reported an error, in fact it did not really crashed, But it found an assert statement, because the number of iterations became more that 100. The program was looking for an answer in the interval, in which there was no right answer, so if not assert statement, she would work infinitely long

This happened because initially we presented that the answer lies between 0 and the number itself. However, this is not true for numbers of smaller 1.

For example, in the examined example, the root of 0.25 is not between 0 and 0.25 The right answer is 1/2.

----2----

Fix this error was managed by changing the initial search bounds in the following way.

It seems that we found the ideal method of finding the root, but if we are worried about the number of iterations and program execution time, this method can work for a very long time.

----3---

to solve the problem of speed of convergence we use Newton’s method. To use this method you need to determine the function: f(guess)= guess2 – x

We want to find such a guess so that f(guess)=0  
guessi+1  = guessi - f(guessi)/(2\* guessi)

Implementation on python looks like this

---4---

Tests show that such a method shows the same accuracy as the method previously considered, but it works for fewer iterations.

---5---