Using your `Rational` class for representing rational numbers, write a function `square_root_rational` which takes an input rational number `x` and returns the square root of `x` to absolute precision `abs_tol`. Your function should return a `Rational` number instance as output. Here is an example,

```
>>> square_root_rational(Rational(1112,3),abs_tol=Rational(1,1000)) # output is `Rational` instance
 10093849/524288
```

Here is your function signature: `square_root_rational(x,abs_tol=Rational(1,1000))`.

Hint: Use the `bisection` algorithm to compute the square root.

Hint There are infinite solutions within a given error tolerance because rational numbers are dense i.e, you can always find a rational number arbitrarily close to a real number. Also, we use `abs(estimate - sqrt(x))` to measure the error because `sqrt(x)` is what you are estimating.

Please put your Python code in a Python script file and upload it. Please retain your submitted source files! Remember to use all the best practices we discussed in class. You can use any module in the Python standard library, but third-party modules (e.g., Numpy, Pandas) are restricted to those **explicitly** mentioned in the problem description.

Tips:

- After you have submitted your file, do **not** use the browser back or reload buttons to navigate or open the page in multiple browser tabs, as this may cause your `attempts` to decrease unexpectedly. It may take up to thirty seconds for your code to be processed, so please be **patient**.
- If you find yourself back at the main page without any feedback or change in your `attempts` then it means that your code timed out or crashed in some unexpected way.
- Ensure that your development environment does not presume the existence of certain packages for the autograder. The autograder does not have anything other than the standard library and those third-party libraries **explicitly** named in the problem description.
- Do not leave extraneous statements in your code like test cases, print statements, or anything else besides what is needed to evaluate your submission because the the autograder will spend its limited time executing those lines, which may result in unexpected crashes or timeouts.

浏览... 0706.py Upload Python source code file

Correct! Back to assignments. functional points = 15 /15 and validation points = 5/5