Fraction Testing and Debugging

It's very common for software developers to revisit code that you (or someone else) wrote in the past. Recall that you defined a class in Homework 02 to support arithmetic on fractions stored as class instances. You'll be using that code for this assignment.

Your assignment this week has **four parts**:

1. Copy your Homework 02 Python file to a new file, e.g. HW03-YourName.py. Rename the methods in your Fraction class to use Python's magic methods for __add__(), __sub__(), __mul__(), __truediv(), __eq__(). You should be able to simply rename the existing methods to use the corresponding magic method name. You'll find an example from my solution in Canvas.

2. Add new methods to your Fraction class to support:

- __ne__(self, other) # Not equal
- __lt__(self, other) # less than
- __le__(self, other) # less than or equal to
- __gt__(self, other) # greater than
- __ge__(self, other) # greater than or equal to

Each of these methods should compare self and other and return True or False.

3. Use Python's unittest to replace the test cases from your Homework 02 submission with unittest tests. Be sure to test every feature enough to convince yourself (and me) that you have adequately tested your solution for adding, subtracting, multiplying, and dividing fractions along with ==, !=, <, <=, >, >=. You should include at least one test for each method. You may want to start with the test suite you created for Homework 2 and replace your print statements with the appropriate calls to the UnitTest methods discussed in the lecture. Be sure to use the inline operator syntax that is supported now that you've defined the magic methods, e.g.

```
>>> f1 = Fraction(3, 4)

>>> f2 = Fraction(1, 2)

>>> (f1 + f2) >= f2
```

You must use Python's unittest module and include test cases for every method in your Fraction class, including raising an exception on denominator == 0. Include a call to unittest.main(exit=False, verbosity=2) to run the tests.

Submit both your code and a screen dump of running your unittest test suite.

4. Along with the new methods and test suite, you will also get some experience using the Python debugger. Submit a screen dump of debugging one of methods in your

Fractions class with VS Code, PyCharm (or your debugger of choice). You should include at least the following features:

a. Set a breakpoint and run the debugger to stop at that breakpoint

b. Use VS Code's Variable Explorer (or equivalent) to display all of the local variables.

See the lecture notes for a sample screen shot of using the debugger.