Homework 03

Submit Assignment

Due Sep 27 by 10pm **Points** 100 **Submitting** a file upload

Assignment Description

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, rename it with `HW03_FirstName_LastName.py`. For example, if Benji is a student in SSW810, the Python file Benji submits will be named as `HW03_Benji_Cai.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: "Fraction")`: not equal
   - `__lt__(self, other: "Fraction")`: less than
   - `__le__(self, other: "Fraction")`: less than or equal to
   - `__gt__(self, other: "Fraction")`: greater than
   - `__ge__(self, other: "Fraction")`: 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 self.assert*() 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 assert statements 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 = Fraction(3, 4)
   >>> f2: Fraction = 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 tests for __str__(self) and raising an exception on denominator == 0.

Submit both your code and test file.

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:

    1. Set a breakpoint and run the debugger to stop at that breakpoint
    2. 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.

NOTE:

- Be sure to include docstrings at the top of every file, function, and method.
- Use type hints to specify the return type of every function/method, all instance attributes, and local variables.
- Be sure to resolve all warnings from Python/MyPy before submitting your code.
- Use f-strings rather than string concatenation.
- Address all of the issues I raised in your HW02 submission to avoid losing more points in HW03 for the same problems.

Deliverable File Structure

In this assignment, you will need to **separate your code into two files** - one for code logic and one for unit test. It's always a good practice to separate the code and the test.

You are going to have **two .py files** for submission:

1. `HW03_FirstName_LastName.py`
2. `HW03_FirstName_LastName_test.py`

`HW03_FirstName_LastName.py` should have your definition of class Fraction.

`HW03_Test_FirstName_LastName_test.py` should import your class Fraction from `HW03_FirstName_LastName, and then define your test cases to completely test your Fraction class`. See the class lecture for an example.

Top-level script environment

`HW03_FirstName_LastName.py`

Define class Fraction. You may also include your HW02 functions for the Fraction calculator, but they aren't needed in HW03.

`HW03_FirstName_LastName_test.py`

In this file, you should include `all of your test cases.`

No `print` in your implementation of class Fraction

You **should not** have any `print` statements in your implementation of class Fraction - there is no need to *print* out any information. You can raise an exception with customized string - this is a good practice. But don't `print` it out. Other programmers will want to import your class Fraction and they don't want your print statements showing up in their programs.

If your have any concern, please do not hesitate to reach out.