Homework 7

Due Nov 22 by 11:59pm **Points** 100 **Available** after Nov 5 at 6am

Home (https://northeastern.instructure.com/courses/156930/pages/landing-page)

Resources (https://northeastern.instructure.com/courses/156930/pages/resourc

Homework 7: Classes, Objects & PyUnit

Simple Fractions

Your solution will be evaluated according to the grading rubric available on CANVAS in the Resources section.

Programming Component (100%)

Program: Simple Fraction (85%)

Files:

- **Submit**: `SimpleFraction.py`
- **Starter files**: `FractionTextPresenter.py` (https://northeastern.instructure.com/courses/156930/files/22371842?wrap=1) (https://northeastern.instructure.com/courses/156930/files/22371842/download?download_frd=1) . `FractionTurtlePresenter.py` (https://northeastern.instructure.com/courses/156930/files/22371995?wrap=1) (https://northeastern.instructure.com/courses/156930/files/22371995

`/download?download_frd=1)`

This week you'll be implementing your own simple Fraction class. The Python library has a similar class for use, but this week's homework will have you create your own simplified version.

Task Overview:
Create a `SimpleFraction` class that manages rational numbers. Your class must implement the following methods (if you want to implement more than these, that's fine – but these are the minimum operations your class must support):

`__init__(self, numerator, denominator):` Create a new `SimpleFraction` instance by supplying integer values for the numerator and denominator. Your method should also be able to handle the case where client code does not supply any parameters (we call this a "default constructor). Use default values of 1 and 1 for the numerator and denominator, so creation requests like the following should be supported by your solution:

```
    fract = SimpleFraction()
```

In general, any values pass to `__init__` that are not integers should raise a ValueError

`get_numerator(self):` returns the value of the `SimpleFraction's` numerator

`get_denominator(self):` returns the value of the `SimpleFraction's` denominator

`make_reciprocal(self):` Returns a new `SimpleFraction` instance that has the current instance's numerator as its denominator, and the current instance's denominator as its numerator. This is a NON-MUTATING method, so the current instance is not modified. A new `SimpleFraction` instance is returned

`validate(self):` Checks to ensure that the numerator and denominator are integers. If **either** of the values are NOT integers, this method raises a **ValueError**

`multiply(self, other):` Multiplies the current instance with another `SimpleFraction` OR a whole number (scalar). Returns the result. This is a NON-MUTATING method, so the current instance is not modified. A new `SimpleFraction` instance is returned

`divide(self, other):` Divides the current instance with another `SimpleFraction` OR a whole number (scalar). Returns the result. This is a NON-MUTATING method, so the current instance is not modified. A new `SimpleFraction` instance is returned

`__str__(self):` Returns a string representation of `SimpleFraction` instances

`__eq__(self, other):` Compares current `SimpleFraction` instance to another one. Returns True if they are equal, False otherwise

Additionally:

Ensure that your `SimpleFraction` class works with, and is able to be "plugged" into my `FractionTextPresenter` and `FractionTurtlePresenter` (starter code). These are "views" that allows us to represent your model on the console or on the Turtle canvas.

Notes:

- It is NOT a requirement to store SimpleFractions in their simplest reduced form. For example, a SimpleFraction 2/4 is perfectly acceptable rather than storing it as 1/2. However, those two SimpleFractions, when compared with each other, should be equal.
- Your SimpleFraction should be able to multiply and divide whole numbers as well as other SimpleFractions. So, if x is a SimpleFraction holding 1/2, x.multiply(x) would return a SimpleFraction 1/4. x.multiply(4) would return a SimpleFraction 4/2.
- Please ensure your class is named precisely as indicated (`SimpleFraction` - using PascalCase)

Program: Testing Simple Fraction (15%)

Files:

- **Submit**: `TestSimpleFraction.py`

Task Overview:
Create a SimpleFractionTest class that tests the `SimpleFraction` class you created. Your test class must use the PyUnit framework, so ensure it inherits from `unittest.TestCase`

Your grade for this portion of the homework will be determined by your test class and the sufficiency of testing you do for the methods you implemented for `SimpleFraction`.

Note: You are only required to test the features of `SimpleFraction`. You are NOT required to validate the TextPresenter or TurtlePresenter or the compatibility of SimpleFraction with either of those two components. This portion of the assignment is focused solely on validating `SimpleFraction`.