OOP 3 "*overloading operators*"

Objects are more complex than variables. Integers, decimals and strings are 'native' variables known to Python and as such, relational and arithmetic operators have all been developed and come integrated with Python. Complex newly developed objects such as your Dice and Date classes where not known by Python until you created them. As such, relational and arithmetic operators of Python do not function properly with your newly created Objects UNTIL you 'teach' and 'define' how these operators are to function.

Date Class

Using your classes already defined in your previous two assignments,'extend' them by adding some additional functionality to these classes. Add some relational and arithmetic functionality to your classes as described below. In either case, develop and include a "test" Main that will test and demonstrate the newly added functions of your Class.

ICS3U – Date Class

Add Subtraction Arithmetic Operator Overload; returning the number of days between the two dates.

Add GreaterThan, LessThan, GreaterThanEqualto, LessThenEqualTo, Equal and NotEqual Relational Operator Overloads each returning True/False appropriately.

Enrichment Opportunity*: Add an Addition overload operator which will be unique; in that two dates added together will produce a new date that will be a valid date, the difference between the two dates ahead of the greater of the two dates. For example, February 20, 2014 + February 1, 2014 will become March 1, 2014 as there are 19 days between the two original dates and 19 days after February 20th would be March 11st. Note that the order of the two dates is arbitrary to the calculation*

Fraction Class

Design a program which will test the user's ability to perform fraction calculations. It will present the user with 2 randomly created fractions and an answer (which may or may not be correct). The user has to say if the answer is right or wrong. Score is kept.

Fields: **intNumerator** and **intDenominator** (integers)
and the following methods:

__init__ initializes the fields to given values, use 0/1 as default values and call reduce

calcGCD returns the GCD of 2 given integers

reduce check to ensure that **intDenominator** is not 0, if it is, set **intNumerato**=0 and **intDenominator** =1, otherwise reduce the current fraction to lowest terms with the negative sign (if there is one) on the top.

setValue sets the current fraction with a numerator and denominator obtained via parameters; validation/reducing required.

randomize sets the current fraction with a numerator and denominator in the range of maxInt (defaulted to +/-10) via a parameter; validation/reducing required.

__str__ Returns the fraction in proper mixed number form (eg: 1 1/3 : not 4/3, 3 : not 3/1, 0 : not 0/5) as a string

calcInverse Returns the inverse fraction of the current fraction.

__eq__ Returns true if a given fraction equals the current fraction.

__add__ Returns a fraction which is the result of adding a given fraction to the current fraction

__sub__, __mul__, __div__ same idea as above; returning a fraction which is a result of the operation on a given fraction to the current fraction.

The main program needs to randomly set two fractions, using random integers between -10 and 10, to random values. It also needs to select a random operator among the operators incorporated into your class (ie. +,-,*,/). The answer is calculated (a fraction) and also a guess is generated (a fourth fraction) which is set to either the correct answer or its inverse; depending on a random integer used to select it. Present a simple fraction equation, (ie. fractionOne + fractionTwo = fractionGuess), displayed on screen and the user has to say if the equation presented is right or wrong. Score is kept.

Enrichment Opportunity**:** *There is no particular end to the test (though you should always provide the option to continue or quit after each question UNLESS you wish to prompt the user initially for a quiz of 10, 20, … questions.*

Enrichment Opportunity**:** *Adding a GUI front end to your class; you will need globals for your fractions, the operator, score and count. A setup() method uses these globals and creates the "next question and displays it. When the user clicks the right button you simply check if answer = guess (or not for wrong), update the score and call setup().*