CSC148, Summer 2016

build Rational class

(do it before looking at any sample solution)

(do it before going to Lab02)

Here is a description of rational numbers, the fractions we learned in grade school:

> *Rational numbers are ratios of two integers p/q, where p is called the numerator and q is called the denominator. The denominator q is non-zero. Operations on rational numbers include addition, multiplication, and comparisons: >, <, ≤, ≥, =, etc.*

Find the most important noun (good candidate for a class...), its most important attributes, and operations that sort of noun should support.

On the back of this sheet are steps taken from the recipe for designing Python classes. You'll be using this in more depth next week during lab, but for now we'll use it as a guide to creating the Rational class. Here are exercises to carry out on paper, perhaps you'll need some scrap paper as well as this hand-out. You are, of course, welcome, to continue the exercise on a computer in Python.

1. Carry out steps 1 and 2 of Part 1 for the description of rational numbers above.
2. Carry out steps 3 and 4 of Part 1 for the description of rational numbers above.
3. Carry out step 3 of Part 2 for the description of rational numbers above.

Part 1: Define the API for the class

1. Class name and description. Pick a noun to be the name of the class, write a one-line summary of what that class represents, and (optionally) write a longer, more detailed description.
2. Example. Write some simple examples of client code that uses your class.
3. Public methods. Decide what services your class should provide. For each, define the API for a method that will provide the action. Use the first four steps of the Function Design Recipe to do so:
   (1) Header
   (2) Type Contract
   (3) Example
   (4) Description
4. Public attributes. Decide what data you would like client code to be able to access without calling a method. Add a section to your class docstring after the longer description, specifying the type, name, and description of each of these attributes.

Part 2: Implement the class

1. Internal attributes. Define the type, name, and description of each of the internal attributes (if there are any). Put this in a class comment (using the hash symbol) below the class docstring.
2. Representation invariants. Add a section to your internal class comment containing any representation invariants.
3. Public methods. Use the last two steps of the function design recipe to implement all of the methods:
   (5) Code the Body
   (6) Test your Method