ATCS Command Line Based Calculator in Python

I. User Interaction

A. Program Operation

- The program operates using an infinite loop that asks the user to input a number, an expression, and then a number
- The program evaluates the expression and then returns the result

B. Accepted Values

- The program can accept four types of numbers as inputs: fractions, mixed fractions, whole numbers, and decimals, in either negative or positive form
    - **Fractions** are entered using the format: A/B
        - Spaces anywhere are not valid and will cause the program to reject the input
    - **Mixed Fractions** are entered using the format: A B/C
        - There can be exactly one space - in between the whole number and the fraction
    - **Whole Numbers** are entered using the format: A
        - Spaces anywhere are not valid and will cause the program to reject the input
    - **Decimals** are entered using the format: A.B
        - Spaces anywhere are not valid and will cause the program to reject the input

C. Accepted Operators

- The program accepts the following operators: **addition** (+), **subtraction** (-), **multiplication** (*), **division** (/), and **exponentiation** (**)
    - The program will attempt to evaluate valid expressions with these operators
    - Invalid operators or invalid expressions, such as ^ and 1/0, respectively, will be caught and rejected by the program

II. Development

A. Code

- The code was written in Sublime Text 2 text editor

B. Testing

- The program was tested in Terminal command line application

C. Formatting

- The code was imported into PyCharm text editor in order to be formatted to the correct PEP8 Python style guidelines

D. Version

- The program was written originally in Python 3.3.3. However, it was converted to run on Python 2.7.8 due to compatibility issues.

III. Strategy

A. Classes

- The program implements two classes: Fraction and Decimal
  - The **Fraction** class supports mixed numbers and fractions.
    - The class holds two values: numerator and denominator
    - The class overrides the *str, mul, float, add, sub, truediv, pow,* and *neg* methods
    - The *str* method always returns fractions in the form of A/B
  - The **Decimal** class is a subclass of Fraction
    - The class overrides the *str* method
    - The sole purpose of the class is to provide an alternative way of printing out numbers (A.B). Otherwise, it operates like a fraction
      - The format the result is printed out in depends on the format of the first number. For example, if the first number was a decimal, the program would print out the result in A.B form, whereas if it were a fraction, it would print it out in A/B form

B. Process

The program takes in each number as a raw_input. It then runs the *clean* method on the input, in which the number is an analyzed and then returned as either a Fraction, Decimal, or invalid type (None). Essentially, all input is converted to a fraction, as Decimal is a subclass of Fraction and whole numbers are represented by a fraction with a denominator of one. Then, the two numbers and the operator are sent to the *evaluate* method. There, the program analyzes the operator. It attempts to match the input to a value in a dictionary of operators. If there is a match, the program will perform the operation with the two numbers. All the operator methods are contained in the Fraction class because every number essentially is a Fraction. If the first number is a decimal, the method will merely modify the values of the decimal and then return it, so that the type is preserved and the result will be printed in the correct format. The same applies to fractions. If there are any errors along the way, like TypeErrors or ValueErrors, the program will catch them, reject the input, and continue the loop. Otherwise, the program will print out the result and continue the loop.