Final Project - Extra Credit

Goal: This is an extra credit problem (10 points) for your final project. You need to create a working 2x2 matrix-class that uses operator overload for addition, subtraction, multiplication, and "raising to a power". You *must* use the given "main" for your code. You should only design a class that works with the given code. Do not modify the given code at all.

Each code should contain extensive comments in the style discussed in class and output should be clearly displayed and labeled. Every single class should have a large block of comments describing how the class works. Similarly, the start of each problem should have a detail description of what the code does. You are welcome to comment in your own style, but you may want to look at some standard Python commenting styles (Google 'numpy style docstrings'). You *will* be graded on your programming style. Classes, functions, and lists should all be used when it makes sense. Duplicate code should be minimized.

Extra Credit:
Create a class called *Matrix2x2* . This code should have data elements *a, b, c* and *d* , corresponding to the elements of a 2x2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The class should have the following function elements:

- *__ init __* : Constrctor which takes in *self* and four values, corresponding to *a,b,c,d* for the 2x2 matrix
- *__ add __* : Takes in *self* and a second instance of the 2x2 matrix. Returns the result of the addition operation applied to the two matrices. I will give you this code (without comments):
  ```
  def __add__(self,second):
      return(Matrix2x2(self.a+second.a,self.b+second.b,self.c+second.c,self.d+second.d))
  ```
- *__ sub __* : Takes in *self* and a second instance of the 2x2 matrix. Returns the result of the subtraction operation applied to the two matrices (the *self* matrix minus the second matrix)
- *__ mul __* : Takes in *self* and a second instance of the 2x2 matrix. Returns the result of the multiplication operation applied to the two matrices (the matrix corresponding to *self* times the other matrix, multiplied on the right)
- *__ pow __* : Takes in *self* and a non-zero integer. Returns the matrix raised to the power. Negative powers should return the inversion and power of the matrix (recall $A^{-3} = \left(A^{-1}\right)^3$, or the inverse of A cubed). If the matrix is not invertible or if the 2nd argument is not a non-zero integer, a 2x2 matrix of all *math.nan* should be returned.
- *__ str __* : Takes in *self* and returns the matrix formatted as desired.

Incomplete example output is as follows:

```
Addition: A+B
[1.00 2.00]
[3.00 4.00]
+
[4.00 3.00]
[2.00 1.00]
=
[5.00 5.00]
[5.00 5.00]
Inverse with powers: A^-3 = (A^-1)^3
[1.00 2.00]
[3.00 4.00]
^-3
=
```

```
[-14.75 6.75]
[10.12 -4.62]
```