Class 4. Exercises and Challenge problems

1) Class 4 Exercise: Modify the candy.txt file
Write a program that would allow the user to modify the quantity of some candy in the candy.txt inventory file. First ask for their input ("Enter candy name") and also ask for the new quantity ("Enter the new quantity for this candy"). Then read the file line by line until you find that candy name and replace the original quantity with the new one.
(If this sounds difficult, go through the class example files of findcandy.py and modifycandy.py. Your code will likely need a few modifications from those files but it's a good starting point!)

- Bonus would be to use control flow to determine whether the user input is valid (Make sure the user input candy name is actually in the inventory, etc.)
- Remember to use the os module for renaming and deleting files.

<Challenge problems: Optional>
2) Simulate mastermind
(See http://www.wikihow.com/Play-Mastermind for the actual board game for those who don't know how to play)
How to Play Mastermind (revised version for our purposes)

- The computer (codemaker) uses code pegs of six different "colors" (ABCDEF) to create a 4-color code. For example, BCFD or ADCB. (Assume colors cannot repeat. So AABB would be illegal)
- The goal of the game is for the codebreaker (the user) to correctly determine both the four colors selected and their position in the code.
- The codebreaker tries to guess the pattern, in both order and color, within 12 turns.
- Each guess is made by typing in the console.
- Then, the codemaker provides feedback by printing out 0 to 4 scoring pegs
    - A black scoring peg is placed for each code peg that is correct in both color and position.
    - A white peg is placed for a correct color peg placed in the wrong position.
    - The order of black/white pegs do not matter. For example, the codemaker can just print out "1B, 1W" or "1W, 1B" meaning the codebreaker received one black and one white peg for their recent guess.
    - Once feedback is provided, another guess is made.
    - This continues until either the codebreaker guesses correctly or 12 incorrect guesses are made. At this point, the codemaker should either declare a win or loss for the codebreaker.
    - Assume the codebreaker is not dumb and never puts in a 3 or 5 color code instead of 4.
    - The codemaker should always print the complete "board" state so the codebreaker can see the entire history of guesses made.
    - Example of a "board" after two tries from the user

1. input ABCD: 1B, 1W
2. input ADEF: 1B, 3W

Suggested step by step process

1. First start out creating and testing your code for comparing two 4-color codes and then correctly returning black and white pegs. That is the crucial part of your code.
2. Then, hardcode a codemaker's code into the program and work on having the codebreaker input the guess 12 times or until he/she gets it right.
3. Finally, work on randomly generating the codemaker's code.

If you have time, you can make it even more similar to the actual game by allowing colors to repeat in the codemaker's code.

3) Use lists to solve this puzzle.

Jim Loy poses this puzzle: I have before me three numeric palindromes (numbers that read the same backward and forward, like 838). The first is two digits long, the second is three digits long, and when we add those two numbers together, we get the third number, which is four digits long. What are the three numbers?

- Hint 1: All variations of numeric palindromes can be easily created by making strings of digits first.
- Hint 2: Use lists!