Starting Python code for each of the following is provided as **h\_8\_starting.zip**, posted next to this handout. Partial or complete solutions to each are also provided, as described below. Use these solutions if you wish, or implement your own solutions.

At the start of each of these problems, the name of a Python file is given in **blue**: **foo.py**. You should create and save the requested source code in a file having this name. Also add a comment at the top of each giving your name.

When finished, upload each **.py** file with the specified name to the Canvas **H8 Assignment** link.

**~~[H8-1]~~** ~~(~~**~~count\_alice3.py~~**~~) The starting code uses a dictionary to analyze~~ **~~alice.txt~~** ~~and prints out the frequency of each word therein. It's based on what we did in~~ **~~[L8-3]~~**~~, although the provided code here does NOT remove punctuation: you should add this code yourself. The counting code is from our HTT book's "answer code" from HTT12 Exercise 3 - but the book's code removes all single quotes (~~**~~'~~**~~) and replaces other punctuation with empty strings (~~**~~''~~**~~) instead of the single blank (~~**~~' '~~**~~) that I recommended in class.~~

~~Modify the starting code so it removes punctuation except for single quotes, creating the first initial list of words (~~**~~allwords~~**~~).~~ ~~Then iterate through each word in~~ **~~allwords~~**~~, handling single quotes correctly and creating the final word list~~ **~~new\_allwords~~**~~. "Correctly" means (a) single quotes within words are left alone; (b) single quotes are removed if they either start or end a word, EXCEPT for the word~~ **~~'tis~~** ~~- which should be left alone; (c) words which are single quotes are removed. Use~~ **~~==~~** ~~to compare two strings for equality, since~~ **~~is~~** ~~may fail for string "clones". Hint: suggested code is given in comments.~~

~~Full credit will be given if you remove all punctuation correctly, including single quotes as described above. As a check, note there are 5 instances of this word:~~ **~~'tis~~**

**[H8-2]** (**test\_poker\_odds2.py**) Continue with Lab 9's **test\_poker\_odds.py** code, implementing the functions for checking if a hand has exactly two pairs, three of a kind, four of a kind, or a full house. The code for checking for a single pair is already implemented, and we did some of the others in class. You can earn 0.25 points of Extra Credit if you implement functions for checking additional poker hands such as flush, straight, etc.

Also implement at least 3 **pytest** tests for each of these kinds of poker hands (including any Extra Credit), putting your tests at the end of the same file. Note that the **.py** filename prefix '**test\_'** is detected by PyCharm and specially run within the **pytest** framework: you don't have to import anything in your code, but you do need to set **pytest** as the **Default Test Runner** within your project's **Preferences->Tools->Python Integrated Tools** dialog. Remember that **pytest** calls only those functions with no arguments and names that begin with '**test\_'**(again), with the framework tracking and reporting the test results of all such calls. Finally, recall that the easiest way of running your **pytest** tests is by opening the **Terminal** tab in PyCharm, then typing **pytest** in the command line therein.

**[H8-3]** **(inventory.py**) Using a dictionary with string keys and integer values, implement a simple database for tracking different items and their quantities in inventory. That is, for a dictionary **dict**, **dict['apples'] == 47** represents 47 apples in the inventory.  
  
The posted starting code provides a framework for the user to enter different commands as a single letter: **A** adds an item to the inventory, **V** views the current contents of the inventory, **R** removes some quantity of an item in the inventory, and **Q** exits the program. Specifically:

* 1. If **Add(A)** is selected, prompt the user to enter the string name **invname** and the **int** quantity **qty** of some item. If **invname** is a new item (thus, a new key for the dictionary), then add this new item and its count **amt** to inventory. Otherwise, add **amt** to the existing count of **invname** within inventory.
  2. If **View(V)** is selected, iterate over entire existing inventory, printing out the name and current quantity of each item in the existing inventory.
  3. If **Remove(R)** is selected, prompt user to enter the string name **invname** and the **int** quantity **qty** of some existing inventory item, then remove this amount from the inventory and update the dictionary. If all of the remaining quantity of some item is removed from the inventory, you should also remove the item's key from the dictionary.
  4. If **Quit(Q)** is selected, exit the program after printing out the final inventory (same as if V was selected).

All of the above except for **Remove(R)** are implemented in the starting code. Your submitted code should also implement **Remove (R)**, thus completing this application.