Starting Python code for each of the following is provided as **h\_8\_starting.zip**, posted next to this handout.

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 **HW 8 Assignment** link.

**[H8-1]** (**ascend3.py**) Find and print all three digit integers **abc** where **a < b < c, a,b,c** in **'0123456789'**. Your integer may have leading zeroes, as in **007**. Hint: use a triply-nested **for**-loop, each of the form **for d1 in range(10):**

**[H8-2]** (**count\_alice3.py**) Continue with refining your **H7-1** **count\_alice2.py** code, which reads and analyzes **alice.txt** and prints out all lower-cased words in the text file without words including unnecessary punctuation. Extend it by using a dictionary to count the frequency of each word, as described in HTT12 Exercise 17.

**[H8-3]** (**poker\_odds2.py,test\_poker\_odds2.py**) Continue with Lab 9's L9-4 code, implementing the functions for checking if a hand has exactly one pair, two pairs, three of a kind, four of a kind, and full house. Also implement at least 3 **py.test** tests for each of these kinds of poker hands, putting your tests in the separate file **test\_poker\_odds2.py**.

Note: the Lab 9 code for **hasOnePair()** is incorrect: it counts three-of-a-kind hands as hands having exactly one pair. Fix this in your solution...

**[H8-4]** (**readint.py**) Define a Boolean function **read\_int()** that reads an **int** value from the user, then returns it as the result of the call. Your function should defend against invalid entry by catching exceptions that occur when trying to convert a non-integer to **int** using **int()**, then handling them by printing an error message and prompting for reentry. Your function should not return until the user has entered a valid integer, with all your exception handling within the function body.

**[H8-5]** **(inventory.py**) Using a dictionary with string keys and integer values, implement a simple database for tracking the inventory of items. 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, and **Q** exits the program.

* 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 quit (**Q**) is selected, exit the program after printing out the final inventory (same as if **V** was selected).