CIS 180 L - Intro to Python - Fall 2006
Homework Assignment #5

Due: THURSDAY, October 17th, beginning of class

Purpose: To practice with Python dictionaries

Note: as long as you meet the specifications below, you may add additional embellishments as you wish.

How to turn this in: copy the file **hw5.py** that you create to cs-server, and then submit it using ~st10/180pysubmit.

1. Create a plain-text file named **hw5.py**. Start this file with comments containing at least your name and the last-modified date for this file.

First, we would like a function that will be helpful in #2's task. Assume that you would like for a dictionary to keep track of how many of something there is. The key represents the item that we are interested in, and the value for the key is how many of that item there are.

What if we just want to add 1 to the count for some item? Then if we've seen any of that item before, we only need to add 1 to the current count. What if we have never seen it before? Then we want to add the key for that item to the dictionary, with an initial count of 1.

SO: write a function **update_ct**. It takes a dictionary and a key whose count should be updated as its arguments. If the key is already in the dictionary, it should change the value for that key by making it one bigger (it should increase its current value by 1). If the key is not already in the dictionary, it should make it a key, with a value of 1.

For example,
```
>>> myDict = {'a':3, 'b':4}
>>> hw5.update_ct(myDict, 'a')
>>> myDict
{'a': 4, 'b': 4}
>>> hw5.update_ct(myDict, 'x')
>>> myDict
{'a': 4, 'x': 1, 'b': 4}
```

2. Now, sometimes one is interested in letter frequencies (or, at least Scrabble fanatics and Wheel of Fortune fans may be... 8-)).

Write a function **letter_freq** that takes a string as its argument, and builds and returns a dictionary containing, for each alphabetic letter in that string, how many times that letter occurs. (Count the lowercase and uppercase together – that is, we want to know how many **a**'s, whether lowercase or uppercase.) Call **update_ct** in your function.

For example,
```
>>> looky = letter_freq("the rain in SPAIN stays MAINLY in the PLAIN!")
>>> looky
{'a': 5, 'e': 2, 'i': 6, 'h': 2, 'm': 1, 'l': 2, 'n': 6, 'p': 2, 's': 3,
'r': 1, 't': 3, 'y': 2}
```

3. That dictionary resulting from **letter_freq** is nice, but wouldn't it be nice to display it in some "prettier" fashion? Like a simple ASCII bar chart?

Write a function **show_counts** that takes a frequency-count dictionary – like that returned by **letter_freq** – as its argument, and it prints to the screen the letter, then how many times that letter occurs, then an X for each time that letter occurs (resulting in a rather crude, horizontal "bar chart"). **Show the letter counts in alphabetical order by letter**.

Two nice features to include, if you can:

* put some punctuation – dash or colon or something – to separate the letter from its count from its X's. (You'll see what I chose below – you can vary this according to your taste.)

* print the count formatted so that it takes 3 spaces (right-justified); recall that:
```
print "%3s"% 30
print "%3s"% 1
```
...will print:
```
 30
  1
```
...right-justified in 3 spaces.

For example,
```
>>> show_counts( letter_freq("the raaain in Spaain stays mainly in the
    plaaain"))
a -  10  :  XXXXXXXXXX
e -   2  :  XX
h -   2  :  XX
i -   6  :  XXXXXX
l -   2  :  XX
m -   1  :  X
n -   6  :  XXXXXX
p -   2  :  XX
r -   1  :  X
s -   3  :  XXX
t -   3  :  XXX
y -   2  :  XX
```

And, when you are done, submit **hw5.py** using ~st10/180pysubmit on cs-server.