Data Structures (CS 1520) Homework #4 Due: March 12 (Sat) at 11:59 PM

Objective: Practice designing a program and using files and built-in Python list and dictionary.

To start the homework: Download and extract the file hw4.zip from
http://www.cs.uni.edu/~fienup/cs1520s16/homework/
The hw4.zip file contains only a single file: dictionary.txt - a fairly complete dictionary file

For this assignment you are to design and implement a program for the word game Hangman.

Standard Hangman Rules: You are probably all familiar with the game *Hangman*, but the rules are as follows:
1. One player (the computer) chooses a secret word, then writes out a number of dashes equal to the word length.
2. The other player (the human) begins guessing letters. Whenever she guesses a letter contained in the hidden word, the first player (the computer) reveals all instances of that letter in the word. Otherwise, the guess is wrong.
3. The game ends either when all the letters in the word have been revealed or when the guesser has run out of guesses.

The Assignment -- Your assignment is to design and write a computer program to play *Hangman*. In particular, your program should do the following:
0. Read the file dictionary.txt, which contains the full contents of the *Official Scrabble Player's Dictionary, Second Edition*. This file has over 120,000 words, which should be more than enough for our purposes. You should use a Python dictionary to store the words such that each key is the word length with it corresponding value being a list of all word of that length (for example 3 : ['dog', 'cat', 'the', ...])
1. Prompt the user for a word length. Reprompting as necessary until she enters a valid choice. For example, enters a number such that there's at least one word that's exactly that long (i.e., don't accept word lengths of -42 or 137, since no English words are that long)
2. Choose a secret word of the correct length at random
3. Prompt the user for a number of guesses, which must be an integer greater than zero. Don't worry about unusually large numbers of guesses – after all, having more than 26 guesses is clearly not going to help your opponent!
4. Play a game of *Hangman* using the standard Hangman rules, as described below:
   a) Print out how many guesses the user has remaining, along with any letters the player has guessed and the current blanked-out version of the word.
   b) Prompt the user for a single letter guess, reprompting until the user enters a letter that she hasn't guessed yet. Make sure that the input is exactly one character long and that it's a letter of the alphabet.
   c) Reveal the position of the guessed letter (if any) to the user.
   d) If the word doesn't contain any copies of the letter, subtract a remaining guess from the user.
   e) If the player has run out of guesses, display the word that the computer "picked".
   f) If the player correctly guesses the word, congratulate her.

Design First: Since you're building this project from scratch, you'll need to do a bit of planning to figure out what the best data structures (Python lists and dictionaries, etc.) are for the program, and how to functionally decompose the program. **I want a design document turned in as in the first couple homework assignments (structure chart and a couple sentences describing each functions -- see lab 1 description for an example).**

Warning: *Watch out for gaps in the dictionary*. When the user specifies a word length, you will need to check that there are indeed words of that length in the dictionary. You might initially assume that if the requested word length is less than the length of the longest word in the dictionary, there must be some word of that length. Unfortunately, the dictionary contains a few "gaps" The longest word in the dictionary has length 29, but there are no words of length 27 or 26. Be sure to take this into account when checking if a word length is valid.

Possible Extra credit:
Track the number of incorrect guesses using ASCII art hangman figure

Submit all necessary files (dictionary.txt, design.doc (or .pdf, .txt, ...)) with your hangman.py program file(s) as a single zipped file (called hw4.zip) electronically at

https://www.cs.uni.edu/~schafer/submit/which_course.cgi