CSE 231
Fall 2008

Programming Project 4

This assignment is worth 40 points (4% of the course grade) and must be completed and turned in before midnight on Monday, September 22nd.

Assignment Overview

Hangman is a popular word game. In this game, the player is given some number of blanks representing the name of a movie or an actor and he/she has to guess the name using at most K number of chances.

A good website to play this game is : http://www.hangman.no/. Select English as the language, then "Play game" and finally choose a category in the list and get started. Here, the number of chances you get is 10 (i.e. k = 10), for your project it will be only 6.

Task

Your task is to implement the Hangman game in Python. Before implementing the game, please play the game on the website mentioned above. It would help you understand the project.

Program Specifications:

1) Output a brief description of the game of hangman and how to play
2) Ask the user to enter the word or phrase that will be guessed (have a friend enter the phrase for you if you want to be surprised)
3) Output the appropriate number of dashes and spaces to represent the phrase (make sure it's clear how many letters are in each word and how many words there are)
4) Continuously read guesses of a letter from the user and fill in the corresponding blanks if the letter is in the word, otherwise report that the user has made an incorrect guess.
5) Each turn you will display the phrase as dashes but with any already guessed letters filled in, as well as which letters have been incorrectly guessed so far and how many guesses the user has remaining.
6) Your program should allow the user to make a total of k=6 guesses.
7) You MUST use at least 3 string methods or operators in a **useful** manner (several examples that I used are given in the notes below). If you wish to use lists in your project that is fine, as long as you meet this requirement.

Assignment Notes:

If the letter has already been guessed, output a message to the player and ask for input again.
If the guess entered is not an alphabetic letter, output a message and ask for input again.

If the letter is present in the word to be guessed, fill in the blanks appropriately with this particular letter. If the complete name has been guessed, the game is over - player wins the game. Output a message telling the player they have won and quit the game.

If the letter/digit is not present in the word to be guessed, give a message to the player indicating that the guess is incorrect and remaining number of chances is one less. If remaining number of chances is 0 (zero), the game is over - player loses the game. Output a message that they have lost and what the correct word was. Quit the game.

Deliverables

You must use Handin to turn in the file proj04.py – this is your source code solution; be sure to include your section, the date, the project number and comments describing your code. Please be sure to use the specified file name, and save a copy of your proj04.py file to your H: drive as a backup.

Tips/Hints:

This project can all be done with strings, use help(str) in the python shell window to see all of the string methods that may be useful. Some of the ones I used in the example program are below:

1) The string concatenation operator "+". In order to keep track of the incorrect guesses, you could initialize a blank string and use "+" to update the string with the incorrect guesses.

2) The membership operator "in" would be useful to check if a particular letter/digit has already been guessed (correctly or incorrectly).

3) String slicing is useful to insert a letter in a string. For example if I have x = "hello" and I wanted to put a 'Z' in the middle of the word, I could write:
x = x[0:3] + 'Z' + x[3:]
or if I wanted to 'Z' to replace the 'e' I could write:
x = x[0:1] + 'Z' + x[2:]
remember string indexing using slices includes the start position but not the end position, so x[0:2] is "he" but does not include the 'l' at index 2.

4) lower() can be used to change a string to lowercase – make sure you find the letter the user enters whether it's lower or uppercase in the guess word.

5) find() returns the index at which the first instance of a substring is found in a string, for example if x="hello"
x.find('e') returns 1 and x.find('l') returns 2.

6) remember if there is an apostrophe (') or a dash (-) in the original phrase you should output it instead of a dash, the user doesn't have to guess those.

7) Try the example program with many different inputs and make sure your program behaves similarly!

Sample output of the program (next page):

```
please enter phrase to guess: Honey-Bee
Chances Remaining   :  6
Missed Letters/Digits :  None

_ _ _ _ _ - _ _ _

Your guess (letters only): e
Chances Remaining   :  6
Missed Letters/Digits :  None

_ _ _ e _ - _ e e

Your guess (letters only): b
Chances Remaining   :  6
Missed Letters/Digits :  None

_ _ _ e _ - B e e

Your guess (letters only): 3
Not a valid character. Please enter a letter .

Your guess (letters only): b
You have already tried this letter or digit. Guess again!


Your guess (letters only): z
This character is not present in the name.
Chances Remaining   :  5
Missed Letters/Digits :  z

_ _ _ e _ - B e e

Your guess (letters only): i
This character is not present in the name.
Chances Remaining   :  4
Missed Letters/Digits :  zi

_ _ _ e _ - B e e

Your guess (letters only):
```

Extra Credit (10pts):

Note: extra credit is an all or nothing deal, you either do the whole thing and get 10 points, or you don't get any extra credit at all. As always if your program doesn't run, you get 0 points for the whole thing, so make sure you save a copy of your working program before you start adding in the extra credit. FINISH everything else in the project first, then attempt this. 10 points is a lot, and this is meant to be hard, you can ask for help with specific errors or coding questions, but you must figure out the logic and solve the problem completely on your own.

Write-up: In order to get **any credit** for the extra coding that you do, you must write a description of your algorithm and solution in your own words. That is, describe how your solution works in English, especially how you make sure and choose letters with the proper frequency distribution. This write-up should be complete but concise, between ½ and 1 typewritten page. You will turn this in with your project under the filename "proj04xc.txt".

General idea: After the user enters the word to guess, ask the user if they want to choose the letters to guess or if they want the computer to do it. For the first option (they choose to enter the letters themselves), the program should continue just like normal. You must implement the extra code to have the computer choose the letters for the 2nd option.

Program flow: Each time the word is displayed with dashes and letters guessed so far, you will ask the user if they would like to solve the puzzle, if they enter 'y', then ask them what they think the word is. If they enter the correct word, they win and the game is over, otherwise tell them they are wrong and continue with the computer guessing letters until the user again chooses to try and guess the word or the computer uses up its 6 chances at guessing letters.

Details of computer letter-guessing: You will need to import the *random* module in order to have the computer choose a random letter. The computer should never make a guessing mistake, that is it should never try to choose a letter it has already guessed, and obviously it should never try to choose a non-alphabetic letter. Certainly it will sometimes guess letters that aren't in the word. In order to increase the computer's chances, however, it will not guess letters completely randomly from the alphabet. It will use some information about the distribution of letters used commonly in the English language.

Distribution: Wikipedia discusses letter distribution here:
http://en.wikipedia.org/wiki/Letter_frequencies
There are many frequencies mentioned, but we will use the basic morse code distribution for this project, which states letter distribution has the pattern **e it san hurdm wgvlfbk opjxcz yq** where the letters in each group are equally common as one another, and the groups are ordered from left to right as most common to least common. For this project we want, therefore, for the computer to guess 'e' the most, and 'y' and 'q' the least. Specifically we want the computer to guess the first group of letters ('e') 7 times as often as the last group ('y' or 'q'), the second group of letters ('i' or 't'), 6 times as often as the last group, the third group 5 times as often, etc… So the table of frequencies looks like this:

e	i,t	s,a,n	h,u,r,d,m	w,g,v,l,f,b,k	o,p,j,x,c,z	y,q
7x	6x	5x	4x	3x	2x	1x

There are different ways of using methods from the random module and this distribution information to make sure you select the letter ‘e’ 7x as often as the letters ‘y’ or ‘q’, it will be up to you to figure this out.

Hints: start by getting everything working with the computer just choosing letters in order through the alphabet, then figure out how to get the computer to choose a random letter, and finally figure out how to get the computer to choose a letter using the proper distribution.

Deliverables: Your extra code will just be part of your regular program file, proj04.py, and you will also turn in your written document as described above, proj04.txt