Project #6 – Playing around with Python

Learning objectives

- Design and implement a complete program in an interpreted language.
- Use built-in data structures in Python.
- Perform file and console I/O in Python.

Overview

The objective of this assignment is to get to know Python and a few of its built-in data types. To this end, you will write a ***Hangman game*** implementation using a provided text file as the words source.

Example of Play

When I run your program, it should appear more-or-less as follows (depending on user input, and random word selection):

```
Welcome to Hangman
What word-length would you like to play? (3 to n) ==> 5 (put n as max length)

Word: *****
You have 9 guesses remaining.
Type a letter or a word guess:  a
Sorry, there are no a's.

Word: *****
You have 8 guesses remaining.
Type a letter or a word guess: e
There is 1 e!

Word: **e**
You have 7 guesses remaining.
Type a letter or a word guess: a
You guessed a before!
```

```
Word: **e**
You have 7 guesses remaining.
Type a letter or a word guess: s
There are 2 s's!

Word: **ess
You have 6 guesses remaining.
Type a letter or a word guess: dress
Sorry, the word is not 'dress'

Word: **ess
You have 5 guesses remaining.
Type a letter or a word guess: g
There is 1 g!

Word: g*ess
You have 4 guesses remaining.
Type a letter or a word guess: guess
Congratulations, you guessed it!
Game over.
```

Specifications and suggestions

- When executed, your program should look for the file **`wordlist.txt`** in the same directory as your main program script. It should load the list of possible words from this file. If this file is not present, your program should exit gracefully.
- A reasonable way to organize your data would be a dictionary(hash) where each key is a number n, and each value is a list of words with length n letters.

  To get an idea of what I mean, you can experiment with this sort of structure:

  ```
  list1 = ['aaa', 'bbb', 'ccc']
  list2 = ['aaaa', 'bbbb', 'cccc']
  list3 = ['aaaaa', 'bbbbb', 'ccccc']
  allWords = {3 : list1, 4 : list2, 5 : list3}
  ```

```
for example:
print(allWords[4][1])    #=> bbbb
```

- Your game should be playable for any number of letters from 3 to the number of letters in the longest word in the text file.
- Number of guesses for a word of length n is (**2n -1**)
- When prompted for a letter or word guess, any input of length 1 should be treated as a letter, any input of length 3 or more should be treated as a word guess (upper or lower case), any input of length 2 or more than n, the program should re-prompt the user.
- The user wins when they correctly guess the word, or all the letters in the word have been guessed. The user loses if they have no more guesses remaining and they have not yet won.
- Each time your game is played, it should randomly select a word of the appropriate length. Playing the game repeatedly with the same number of letters should not result in the same word being repeatedly chosen.

What to Turn In

Submit a single, well-commented file with your Python code via the Pilot dropbox. The filename should be called **Project6.py**. Include your name in the header comments.

Grading

Each time I run your code it should play one game of hangman. Your code should not crash regardless of user inputs. If an input does not make sense (for example, a word length of -1 or "hello"), the program should re-prompt the user.

This project is worth 40 points. Your program will be graded in two areas:

1. **Capabilities and Correctness (35 pts)** – your program should run correctly regardless of user input, and should play the game as described above.
2. **Style and Efficiency (5 pts)** – your program should be correct with efficient use of Python classes, methods and data types. Your program should be logically organized, well-commented, and should conform to the “Coding Standards” document handed out on the first day of class (and available on the course web page under “Handouts”).