COMP 6411: ASSIGNMENT 1

Important Info:

1. *Feel free to talk to other students about the assignment. That's not a problem. However, when you write the code, you must do this yourself. Source code cannot be shared under any circumstance. This is considered to be plagiarism. At the end of the term, before final grades are submitted, I will scan all coding assignments during the term to look for plagiarized work. These will be sent to the university for investigation, regardless of the grade that was originally assigned. Do NOT put yourself in this position.*

2. *Assignments must be submitted on time in order to received full value. Appropriate late penalties (< 1 hour = 10%, < 24 hours = 25%) will be applied, as appropriate.*

3. *The graders will be using a Python 3.x interpreter to test your code. Make sure that your code is written and tested in a version 3.x environment, not a 2.x environment.*

DESCRIPTION: In this assignment, you will have a chance to gain experience with the Python programming language. Python is an easy to use, dynamically type Object Oriented language. It is syntactically similar to C-style languages like C++, C# and Java, but is somewhat simpler to understand.

You will be working with basic objects and various Python data structures in order to develop a simple command-line game that allows people to guess English words. The idea is as follows:

The user will be asked to guess 4-letter words. To make this realistic, we must utilize a large list of possible words. For this purpose, I have provided a text file containing 4030 4-letter English words. Your Python application will read the file from disk and load all strings into a data structure of your choice. To simplify things, you should place the file with the 4-letter words in the same folder as your Python files.

Your program will randomly select a string and ask the user to guess the string. Initially, the string will be listed as "----". The user will have the choice to try to guess the string directly or to guess a letter at a time. The initial screen might look like this:

```
** The great guessing game **

Current Guess: ----

g = guess, t = tell me, l for a letter, and q to quit
```

If the *letter* option is chosen ("l"), the user will enter a possible letter and the program will indicate how many such letters there are in this string. It will then replace one or more of the "-" values with this letter and redisplay the result. It might look like this, if 'a' is chosen:

```
l
Enter a letter:
a
You found 1 letters

Current Guess: --a-

g = guess, t = tell me, l for a letter, and q to quit
```

This process continues until the user either gets the answer right or gives up. In both cases, the proper string will be displayed and an appropriate message will be printed to the screen. At this point, a new string will be randomly selected from the database and the game will continue with a new guessing round. Of course, "quit" can be selected at any time.

To make the game more interesting, you must keep score. We will assume that a user can play up to one hundred games (you will get tired of this after 2 or 3 rounds). For each game/word, you must do the following. Each letter in the English language is used with a certain frequency ("e" is the most common letter, "z" the least). The table of frequencies is given later in this document.

The letters that are still blank at the time of a correct guess will be summed together to give a total. Basically, the point value for a given letter is equal to its frequency. Note that it is easiest to guess a word if you first uncover the most common letters. However, this leaves the least points in the remaining word. Of course, the number of letters that you turn over also affects your score. So you should divide the sum by the number of times you request a letter. The more letters you must turn over, the lower your total score. Finally, an incorrect guess costs you 10% of your score for the current word.

What happens if you give up? Then you should lose points. Here, the total points lost should simply be the sum the uncovered letters.

The idea is that your score from each round will be recorded, along with the original word, the status of the game (correct guess or not), the number of guesses required, and the final score for that game. Once the user has finished playing, they will select quit. At that point, you will print a short report that summarizes the info. It would look something like this, assuming the user had played four games:

```
Game         Word        Status      Bad Guesses      Missed Letters      Score

----         ----        ------      -----------      --------------      -----

1            yule        Success     0                2                   2.02

2            dipt        Gave up     0                4                   -17.96

3            laid        Success     2                1                   12.20

4            defi        Success     1                1                   5.83


Final Score: 2.09
```

In this assignment, you really only need three classes. The first will be called **guess** and will represent the game itself (including the menu). The second will be called **game** and will maintain information about a specific game. These game objects will be stored in a Python data structure, as required. The third class will be called **stringDatabase** and will be responsible for all disk I/O. You can add whatever code you think you need there.

Note that you will have to use Python's random number module to generate the random numbers. In addition, proper Python-style documentation **must** be provided for the system. In other words, it should be possible to generate a full web-based API for the application using *pydoc* (i.e., the Python version of JavaDocs). To be clear, you don't need a lot of documentation, just same basic info about the main classes and methods - enough to show the grader that you provided some of your own documentation.

Letter	Frequency
a	8.17%
b	1.49%
c	2.78%
d	4.25%
e	12.70%
f	2.23%
g	2.02%
h	6.09%
i	6.97%
j	0.15%
k	0.77%
l	4.03%
m	2.41%
n	6.75%
o	7.51%
p	1.93%
q	0.10%
r	5.99%
s	6.33%
t	9.06%
u	2.76%
v	0.98%
w	2.36%
x	0.15%
y	1.97%
z	0.07%

So that's the basic idea. If you haven't used Python before, you will find that it is a very accessible language with a lot of documentation and supporting materials online. If you like to code, this assignment should actually be fun.

DELIVERABLES: Your submission will have exactly three source files: `guess.py`, `game.py`, and `stringDatabase.py`. The program itself will be run from `guess.py`. Do not include the `four_letters.txt` file since the marker will already have this.

You should organize your source files into a folder called `a1_studentID` (for example, a1_4539800). Your .py source files will be placed inside this folder. You should also include an empty file called `__init__.py` (double underscores) in the main folder. This file is used to

designate that the contents of the folder should be seen as a Python package (a little like a Java package). Having distinct packages defined for each student will make grading much easier.

Once you are ready to submit, place the package folder (not just the folder contents but the folder itself) into a zip file. The name of the zip file will consist of "a1" + last name + first name + student ID + ".zip", using the underscore character "_" as the separator. For example, if your name is John Smith and your ID is "123456", then your zip file would be combined into a file called a1_Smith_John_123456.zip". The final zip file will be submitted through the course web site on Moodle. You simply upload the file using the link on the assignment web page.

Good Luck