CS 232 – Programming in Python

Assignment #3

Deadlines

This assignment is due on **Wednesday, March 21, 2018 @ 1:00 PM (beginning of class)**

How to submit your work on the assignment:

Your assignment will be turned in as a **single** Python file (the "module") containing your Python code. The module will contain **only** the code that meets the problems' requirements. **Do not include testing code! There is NO NEED to submit a testing module!** I will run your code with my own testing module.

The File Naming Convention

Properly name the file that contains your work. The file's name will contain the following, with dashes in between each part listed below and no spaces:

* The course name, **CS232**
* Then a hyphen and the assignment number as a two-digit number, in this case **03**
* Then a hyphen and your HSU account username with your initials and a number – in this example, I'll use **jqs123** for a student named Jospehus Q. Smileyface whose HSU login ID is **jqs123**.
* Windows will automatically add a **.py** at the end of the file when saving. This may be hidden, though, depending on how Windows is configured on the computer you're using (the lab computers are configured this way by default). If you're using a Mac, you'll probably need to add a **.py** to the end of the filename when you save it.

Put all the rules together, and Jospehus Q. Smileyface's Assignment 3 file in CS 232 would be named

```
CS232-03-jqs123.py
```

Your file name will be different from this because your HSU login ID is different, but that's the **only** difference!

Submitting your file electronically

Upload all homework assignment submissions to Canvas. Find the assignment page in Canvas, and use the "Submit" button to upload the file. If you have any questions about this procedure, please contact me. All your Python functions for this assignment should be in a single **.py** file.

Canvas allows students to upload multiple versions of your file. My usual grading procedure is that the **only** version of the assignment I will grade is the **latest** version you submit. NOTE – this means that if you submit a file before the deadline, then submit an updated file after the deadline (subject to a 25% penalty), I will ONLY grade the last submission and assess the 25% penalty on the assignment! If you made a mistake and don't want the latest version to be graded, send me an email explaining which specific version you do want graded, and do so **before** the assignment is actually graded!

Documentation of your Python code

Begin the Python module (file) with at least the following opening comments:

* A comment line containing the file's name
* A comment line containing your name
* A comment line containing the date that your module was last modified

For example:

```
# CS232-03-jqs123.py
# Josephus Q. Smileyface
# Last modified: February 30, 2099
```

For this particular assignment, the signature and purpose of the function you will write have already been supplied.

PROJECT – THE GAME OF HANGMAN

Using the supplied files **hangman_template.py** and **words.txt**, complete the coding necessary to play the game of Hangman. The two files need to be kept in the same local folder when testing your code!

Rename the **hangman_template.py** file to the requested filename used for CS 232 homework submissions (explained in page 1). Place the files into the folder you will use for this assignment, and uncomment and type the full folder path into the **os.chdir()** statement near the top of the file. When submitting the assignment, you should submit only the one Python file – I don't need extra copies of the **words.txt** file!

It will help you greatly to carefully look over the code already written in **hangman_template.py** and follow the instructions contained in it. Familiarize yourself with the supplied functions, and **especially** the global variables: **secret_word** (the word to be guessed) and **letters_guessed** (a list of letters that have been guessed during that game by the user). You need to make sure the values of these global variables are properly used and also properly reset after each Hangman game – the goal is for a user to be able to play as many Hangman games as they want.
Your testing needs to include playing Hangman several times in a row to ensure it works each time.

Your **play_hangman()** function will use a local variable **wrong_guesses** to keep track of the number of wrong letter guesses. These three variables – **secret_word**, **letters_guessed**, and **wrong_guesses** – are all the information Hangman needs to keep track of during game play.

PROBLEM 1 – THE `word_guessed()` FUNCTION (20 points)

Write the function **word_guessed()** and test it thoroughly by supplying it with various values for **secret_word** and **letters_guessed**. For example, once your function is written you can test it by doing something like the following, using different words and different lists of letters guessed:

```
Loading word list from file...
   55900 words loaded.
Enter play_hangman() to play a game of hangman!
>>> secret_word = "arf"
>>> letters_guessed = ['a', 'r', 'f']
>>> word_guessed()
True
>>> letters_guessed = ['a', 'r']
>>> word_guessed()
False
```

Test it thoroughly with various words and lists of letters.
HINT: You can test to see if something is NOT in a list by using the following syntax:

if item_variable not in list_variable:

PROBLEM 2 – THE `print_guessed()` FUNCTION (20 points)

Write the function **print_guessed()** and test it thoroughly by supplying it with various values for **secret_word** and **letters_guessed**. For example, once your function is written you can test it by doing something like the following, using different words and different lists of letters guessed:

```
Loading word list from file...
   55900 words loaded.
Enter play_hangman() to play a game of hangman!
>>> secret_word = "arf"
>>> letters_guessed = ['a', 'r', 'f']
>>> print_guessed()
a r f
>>> letters_guessed = ['a', 'r']
>>> print_guessed()
a r _
```

Test it thoroughly with various words and lists of letters.

PROBLEM 3 – THE `play_hangman()` FUNCTION (60 points)

This problem will take the longest time to write, so be sure to leave enough time to write it!
This function will control the play of the game by choosing a secret word then performing a loop where, for each letter to be guessed by the user, it will:

- Ask the player for a letter, and input the letter entered
- Determine if the word has been completely guessed – if so, notify the user of their victory, reveal the word that was completely guessed, and exit the function
- If the word isn't completely guessed yet, find out whether the guessed letter is in the word **(NOTE: if a letter is guessed more than once, ignore the guess and tell the user to try again!)**
- If the letter guessed isn't in the word, then increment the **wrong_guesses** value
- Draw the updated Hangman image to the screen
- Print the current status of the Hangman word, revealing only the letters guessed
- If the player's out of guesses and is "hanged", then notify the user that they've lost, reveal the word that wasn't completely guessed, and exit the function
- If the player still has guesses left, go back to the top and ask for the next letter

Make sure to write this function so you can run **play_hangman()** repeatedly in a single session, and have each game play correctly.

For example, if the secret word is **"claptrap"**, then game play should look something like this:

```
Loading word list from file...
   55900 words loaded.
Enter play_hangman() to play a game of hangman!
>>> play_hangman()
The secret word is ready!
Guess a letter: a
     _________
    |    |
    |
    |
    |
    |
    |_________

     _ _ a _ _ _ a _
     LETTERS GUESSED: a

Guess a letter: e
    _________
    |    |
    |    O
    |
    |
    |
    |_________

    _ _ a _ _ _ a _
    LETTERS GUESSED: a e
```

And so on. Test it thoroughly, and have fun playing Hangman!