CS109 Assignment 4 - Wordle with a GUI

Due date and time: Thursday April 4, 11 pm

50 Points: 50 (40 for correctness, 10 for program hygiene. Follow the program hygiene guidelines as explained in PEP 8. https://peps.python.org/pep-0008/)

Starter File: `wordle_gui.py` Available on Canvas. You must use this starter file.

Dictionary Files: Available on Canvas. The **`secret_words.txt`** file contains the words that may be picked as the secret word. The **`other_valid_words.txt`** file contains the words that are accepted as valid input by the program, but will not be picked as the secret word for this version of the program. Assume these files are in the current working directory when your program is running. If you need help determining the current working directory, import the **`os`** module to your program with the **`import os`** statement and execute the following statement in your program:
`print(os.getcwd())`

Sample Files: Available on Canvas. The **`master_mind.py`** and **`letter_typing.py`** programs will prove **very useful** as examples for creating a Tkinter interface. This code is available on Canvas -> Files -> Sample Code

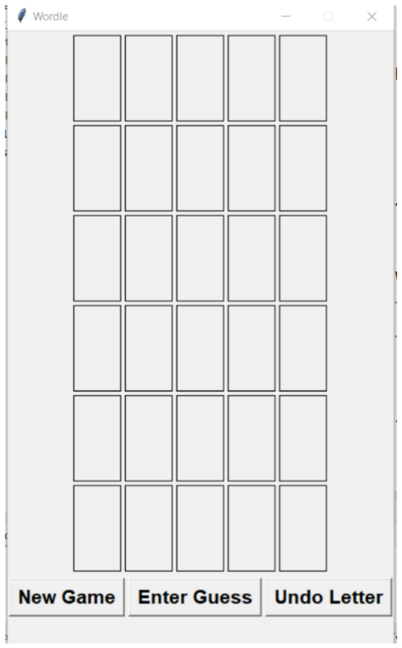
Submission: File name must be: **`wordle_gui.py`** For this assignment turn in your **`wordle_gui.py`** program to Canvas Assignment 4. We are not using GradeScope this time.

Your program may use features up to and including those from Python version 3.7, the version installed on the CS department public lab machines. If you use features from versions of Python beyond 3.7 your program may not work when graded.

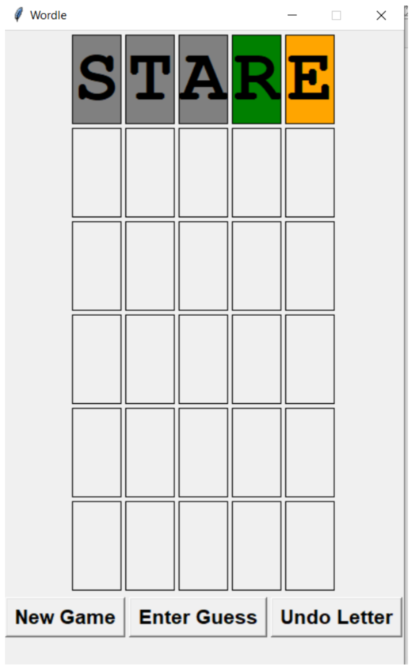
The program must be your own work. You may not copy code from any source. You may use and adapt the example code and your Wordle code from assignment 1. Copying code from another source will result in an academic dishonesty case filed with the Student Conduct and Academic Integrity.
https://deanofstudents.utexas.edu/conduct/

Program: Complete the **`wordle_gui.py`** program. The game allows a user to play Wordle on a laptop or desktop machine. See the list of features to implement below. I recommend converting your code from assignment 1 into a class similar to the approach in the mastermind example. By way of comparison my suggested solution (with one minor extra feature) is a total of 325 lines, including all of the blank lines.

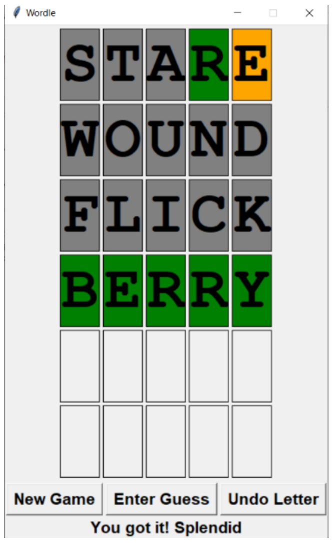
Start of game:



First guess (secret word is BERRY):



Game progression. Note the feedback is "You got it!" and the same feedback based on the number of guesses required, exactly the same as assignment 1:



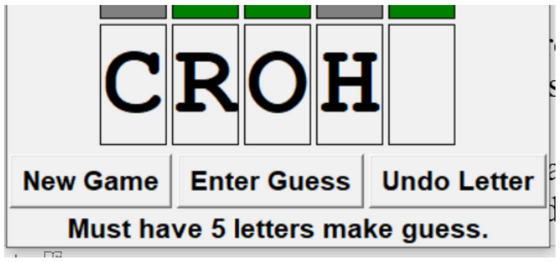
You got it! Splendid

Required Features:

1. Responds to letters pressed, upper and lower case. Letters pressed are the only keypresses that must be responded to. Letter presses cause the letter to appear in the next open label if appropriate.

2. Displays letters in UI as upper case.

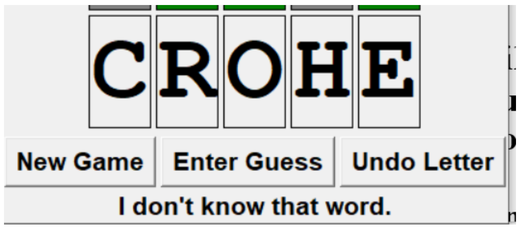
3. If less than 5 letters entered and the user presses Enter Guess, an error message is displayed:



Must have 5 letters make guess.

4. If 5 letters are entered and the user presses the Enter Guess button the UI responds with the feedback if the word is in the dictionary.

5. If the word entered is not in the dictionary, display a message. No feedback provided. User must undo letters.

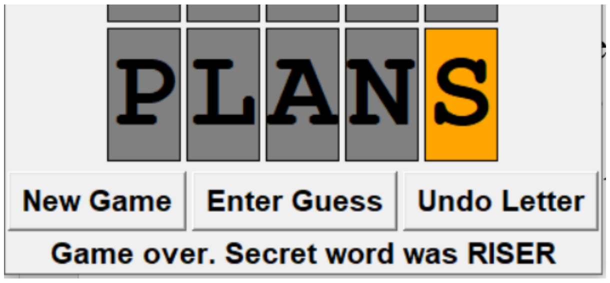


I don't know that word.

6. If the Undo Letter button is pressed and there are one or more letters in the current guess the rightmost letter is undone.

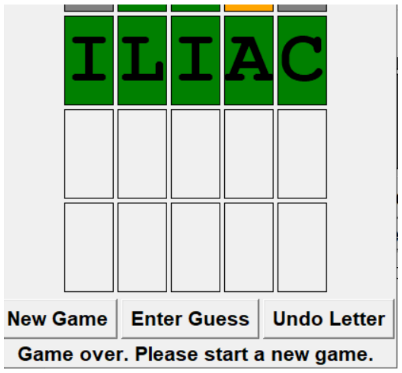
   If the Undo Letter button is pressed, but there are no letters in the current row, nothing happens, no error message, and letters from the previous row that already has feedback are not altered.

7. If the user doesn't solve the puzzle show them the secret word.



Game over. Secret word was RISER

8. If the user tries to enter letters or make a guess, display a reminder to start a new game.



Game over. Please start a new game.

9. Start a new game whenever the New Game button is pressed. The New Game starts even if the current game is not over. You are not required to warn the user if they are in the middle of a game.

10. The final version you turn in must the initial seed of the random module to 3252024. This line of code is in the shell file. This should lead to the first three words being AFFIX, PROXY, APING. This "feature" is to make grading easier.

    My version of the program uses the **`random.choice(sequence)`** function on the list of possible secret words. Recall a list is a kind of sequence.

    Functions for sequences:

    random.choice(seq): Return a random element from the non-empty sequence seq. If seq is empty, raises IndexError.

11. **In addition to the required features you must add one new feature and document it at the top of your program.**