Project 1: Wordle

www.shortl.io/cs0-project1

Background and Rules

In the game Wordle, the user is trying to guess a secret 5 letter word. The user will type in a 5 letter guess, and the computer will share information about how close the guess is to the actual answer.

- If a letter in the guess exactly matches a letter in the answer (same letter and correct position), the letter will be marked green.
- If a letter in the guess almost matches a letter in the answer (same letter, but incorrect position), the letter will be marked yellow.
- If a letter in the guess doesn't match a letter in the answer (guessed letter doesn't exist in answer), the letter will be marked red. I know the actual game marks the letter gray, but we will mark it red to be super clear.

The user will use this information to then make another guess, and will keep guessing until they guess the answer (they win), or they run out of their 6 guesses (they lose).

For example, if the secret answer is TOWEL and the user guesses LOWER, the computer should print out LOWER.

- The O, W, and E are all green because the user guessed those letters in the correct position.
- The L is yellow because L is the last letter in the answer, but the user guessed it in the first position.
- The R is red because it does not appear in the answer.

You can also play the game online at https://www.nytimes.com/games/wordle/index.html to see more examples. (unlimited practice use this link: https://wordplay.com/)

Hard Mode

For this project, we will be implementing the game in hard mode. In hard mode, if a user guesses a letter in a previous guess that was either yellow or green, they must use it in their next guess (it needn't be in the same position).

For example, if the user guesses LOWER and the computer prints out LOWER (L is in yellow; O, W, E are in green; R is in red), the user must reuse the letters L, O, W, and E in their subsequent guesses. In invalid guess will not count towards one of the user's 6 guesses; see this section.

Why Wordle as a Project?

This project requires using everything we have learned so far this semester.

Concept	**Related Lab / HW**
Printing output	Lab 1.1, Lab1.2 and HW 1
Getting user input	Lab 1.1 - Lab 5
Data Types & Variables	Lab 2 and HW 1
Arithmetic Operators	Lab 2 and HW 1
Logical Operators	Lab 3 and HW 1
If / else statements	Lab 3 and HW 1
String access (i.e. get first character in a string)	Lab 5
Loops	Lab 4
Lists	TBD :)

If you feel uncomfortable with any of the topics, I would encourage you to look back at your lab / HW, rewatch the pre-lecture videos, and review the corresponding slides on the course schedule. Or come into office hours!

This is also a cool project because after just 4 weeks of coding, you will already be able to make a game that sold for over 1 million dollars to the New York Times (link).

Differences from Real Game

1. In our project, we will be doing the user input and output in the terminal (as opposed to an online web browser like the real game).
2. **We will also assume that no answer or guess has duplicate letters** (i.e. a guess like TOTEM is not allowed because it has 2 Ts). Allowing duplicates makes the project more complicated, and requires additional data structures like arrays, which we haven't learned yet. We will also assume that all guesses will be all capitalized.
3. You do not need to check if any of the guesses are real words.
4. In hard mode, a user does not need to reuse green letters *in the same position*. As long as they use all green/yellow letters from their previous guesses in their new guess (regardless of which position they're in), their guess is valid.

How to Get Started?

Much like the number guessing and word guessing games you did in Labs 4 and 5, this problem seems daunting and overwhelming at first glance. The mark of a good engineer is being able to break a complex project into smaller, more manageable parts.

To get started, finish the Wordle Starter Assignment on Codio. Before doing any coding, you should completely fill out the assignment. If you would like to run through your doc with a staff member, please sign up for office hours.

Timeline

Wednesday, Oct 12	Project released
Sunday, Oct 23	Starter assignment due
Sunday, Oct 30	Project code due

Grading and Requirements

Starter Assignment Submission (20 points)

You will receive points for submitting your starter assignment and filling in the questions correctly. We do not expect your final project to follow the same exact logic you outline in your starter assignment.

Code Correctness (80 points)

Your code will need to do the following things correctly

- Print each letter in the guess as green, yellow, or red
- Allow user to keep guessing until they get the answer or run out of guesses
- Enforce that the user uses any correct letters (yellow or green) from their previous guess in their subsequent guesses
- Let the user know if they won or lost.
    - If they won, also mention how many guesses it took.
    - If they lost, also mention what the answer was

Code Style (Extra Credit: 10 points)

You must write at least **four** functions to break up your code into more readable parts. You can write more if you would like. The required functions are the following:

(5 Points total for correct naming of functions)

1. `is_valid_guess(guess, required_letters)`: Checks if the input guess is valid, i.e. it contains all of the letters in the list required_letters.
2. `get_guess(required_letters)`: Prompts the user to input a guess until they enter a valid guess that contains all of the `required_letters`. Returns the valid guess.
3. `print_guess(secret_word, guess)`: Prints the guess to the screen in the correct colors. This function returns a list of letters that are found in both secret_word and guess. (aka `required_letters`)
4. `is_game_over(secret_word, guess, tries_left)`: Returns True if the game is over and False otherwise. This function should also print the appropriate message IF the game is over.

(5 Points total for descriptive comments and use of snake_case)

Your code must contain descriptive comments.
Variable and function names must be descriptive and follow the preferred naming convention.

- snake_case for variable names
- snake_case for function names

Sample Game Flow

Below is a sample game flow. **Please make sure your code exactly uses the text as shown in the game flow; our autograding relies on this.**

1. Prompt the user to enter a secret word with the message `Enter the secret word:`
    a. You do not need to check that the secret word is in all caps and does not include any duplicate letters; you may assume that all words follow these rules.
2. Prompt the user to input their guess with the message `Enter your guess:`
    a. If the user's guess is invalid (i.e. it does not reuse a green or yellow letter from the previous guesses), say `Your guess must contain all yellow and green letters from your previous guesses.` and then ask the user to input their guess with the message `Enter your guess:` This should not count towards of the user's 6 guesses.
    b. You do not need to check that their guess is in all caps and does not include any duplicate letters, you may assume all guesses follow these rules.
3. If the user guesses correctly, print `Correct! You got it in {number_of_tries} tries!`
4. else [they don't get it right], do the following:
    a. Output their guess with the letters colored according to the coloring scheme described here. We have provided functions that can do this for you.
    b. Prompt the user to input their guess with the message `Enter your guess:`
    c. Repeat the step above until the user guesses correctly OR until they use up all 6 tries

5. If the user doesn't guess correctly in six tries, print `Game over! The correct word is CODER`.

Here are some sample games. **The text in italics and highlights are not in the actual game output, it is just there to provide more context. Note that these are only sample games, your solutions will be graded on more test cases than is provided.**

GAME 1: User wins in 5 guesses.

Computer: `Enter the secret word:`
User: `QUOTE`
Computer: `Enter your guess:`
User: `TWICE`
Computer: `TWICE` *This is guess 1. (T is yellow, WIC are red, E is green)*
Computer: `Enter your guess:`
User: `AUNTS`
Computer: `Your guess must contain all yellow and green letters from your previous guesses.` *This was not guess 2 since it was invalid; it did not have E in it.*
Computer: `Enter your guess:`
User: `TUNES`
Computer: `TUNES` *This is guess 2. (T and E are yellow, U is green, N and S are red)*
Computer: `Enter your guess:`
User: `MUTED`
Computer: `MUTED` *This is guess 3. (M and D are red, U is green, T and E are yellow)*
Computer: `Enter your guess:`
User: `TONES`
Computer: `Your guess must contain all yellow and green letters from your previous guesses.` *This was not guess 4 since it was invalid; it did not have U in it.*
Computer: `Enter your guess:`
User: `QUITE`
Computer: `QUITE` *This is guess 4. (Q, U, T, and E are green, I is red)*
Computer: `Enter your guess:`
User: `QUOTE`
Computer: `Correct! You got it in 5 tries!`

GAME 2: User loses.

Computer: `Enter the secret word:`
User: `POINT`
Computer: `Enter your guess:`
User: `ADIEU`
Computer: `ADIEU` *This is guess 1. (A, D, E, and U are red, I is green)*

Computer: `Enter your guess:`
User: `TWICE`
Computer: `TWICE` *This is guess 2. (T is yellow, I is green, W, C, and E are red)*
Computer: `Enter your guess:`
User: `EXIST`
Computer: `EXIST` *This is guess 3. (I and T are green, E, X, and S are red)*
Computer: `Enter your guess:`
User: `ORBIT`
Computer: `ORBIT` *This is guess 4. (O and I are yellow, T is green, R and B are red)*
Computer: `Enter your guess:`
User: `BRBIT`
Computer: `BRBIT` *This is guess 5. (B, R, and B are red, I is yellow, and T is green)*
Computer: `Enter your guess:`
User: `BRBIT`
Computer: `BRBIT` *This is guess 6. (B, R, and B are red, I is yellow, and T is green)*
Computer: `Game over! The correct word is POINT.`