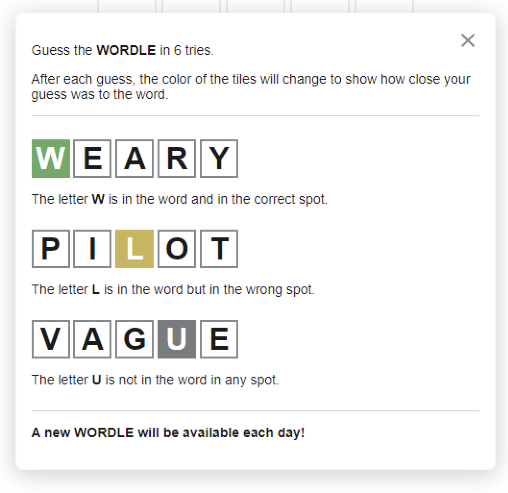
Wordle Game

You are going to create a Wordle game in Python using classes, attributes and methods.

**Rules of The Game:**

For those of you who are unfamiliar with the rules of Wordle, here is a brief explanation. The object is to guess the five-letter word chosen by the computer in 6 or fewer guesses. The user guesses a 5 letter word (it must be EXACTLY five letters). The computer checks the word against the random word it selected and redisplays the guess with the letter green if you’ve places it in the right spot, yellow if you’ve placed a correct letter but in the wrong position and grey if the letter is not in the word at all.

For example (from the Wordle website):



Note: you will NOT be using graphics in the assignment. You will use a library like colorama to display colours on the console.

**To do:**

1. Create a class diagram for the WordleGame class.
2. Create test cases for the WordleGame.
3. Code and test the WordleGame and test program.

**The Class Diagram**

BEFORE coding create a class diagram for the class. Make sure you indicate what attributes and methods are private, protected and public. Complete the table below.

|  |
| --- |
|  |
| \_\_wordList: List[str] # Private \_\_targetWord: str # Private \_\_currentAttempt: int # Private \_maxAttempts: int # Protected \_isGameOver: bool # Protected \_isGameWon: bool # Protected \_\_attempts: List[str] # Private |
| \_\_init\_\_(wordList: List[str]) # Private Constructor \_validateInput(word: str): bool # Protected \_checkWord(word: str): str # Protected \_displayBoard(): None # Protected play(): None # Public |

**Test Cases**

BEFORE coding create a test plan for the test cases you are going to use to test the program. Include things like, what you are going to do if the word is not exactly 5 characters long, if the user runs out of guesses or (if you implement this) if the word is not a valid word. Complete the table below. Add as many test cases as you think are necessary to completely test the game.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Input Data** | **Expected Output** |
| 1 | "eagle" | w |
| 2 | "alert" | First letter yellow (correct letter, wrong spot), rest grey |
| 3 | "eg" | Error: Input must be a 5-letter English word. |
| 4 | "peace" | Second and third letters yellow, fifth letter yellow, first and fourth letters grey |
| 5 | "crane" (6 guesses) | Game over: Max attempts reached |
| 6 | "12345" | Error: Input must be a 5-letter English word. |
| 7 | "great" | First, third, and fourth letters yellow, second and fifth letters grey |
| 8 | "EAGLE" | All letters green (prompts will be case insensitive) |
| 9 | "!!@#$" | Error: Invalid characters |
| 10 | "eagle" (after win) | No effect: Game already won |
| 11 | "eagle" (after loss) | No effect: Game already lost |
| 12 | "glare" | First and third letter yellow, second and fourth letters grey, and fifth letter green |
| 13 | "image" | Third and fourth letter yellow (correct letter, wrong spot), first and second letters grey, fifth letter green |
| 14 | "beady" | First, fourth and fifth letters grey, second and third letters yellow |

**The WordleGame Class**

1. The WordleGame class has at least the following attributes:
   1. secret\_word: String with the value of the secret word IN UPPER CASE.
   2. max\_attempts: int with the value of the maximum number of attempts. Default is 6.
   3. attempts\_left: int that is not passed to constructor. Set initially to the value of max\_attempts. This should be private or at least protected.
   4. guesses: list that is not passed to the constructor. Set initially as an empty list. Will have each guess added (appended) to the list. This should be private or at least protected.
2. The WordleGame class has at least the following methods:
   1. read\_file: Returns a list of the words in the provided file wordlewords.txt
   2. select\_word: Returns a string of a random word selected from the list of words read in from the file.
   3. make\_guess(guess): Checks if the guess is valid and updates the game state
   4. is\_won: Returns True if the game is won and False otherwise.
   5. is\_lost: Returns True if the game is lost (no more guesses allowed) and False otherwise.
   6. \_\_str\_\_: When object of the class is printed, displays the list of guesses, each guess on a separate line, with the appropriate colour for each letter in the guess.
   7. \_\_repr\_\_: Displays the information about the class in the form object\_name(word, max\_attempts, attempts\_left, guesses) in that format.
   8. \_\_eq\_\_: (maybe) determines if guessed word is the same as the secret\_word
   9. \_\_getitem\_\_\_: (probably) to get the individual guesses from the list
   10. \_\_setitem\_\_: (probably) to set the guess into the guesses list
   11. \_\_len\_\_: to get the length of the guesses list
   12. Other dunder methods as needed
   13. Other methods as needed

**How the Game Works**

1. You have been provided a list of 50 five-letter words. When the program starts, initialize an object of type WordleGame.
   1. The constructor should NOT have to pass anything. The constructor should call the method read\_file to read the file.
   2. Once the file is read, the constructor calls the method select\_word to select the secret word for this round. The select\_word method uses random.choice or some equivalent method from the random module to choose the word.
   3. The constructor then initializes all the attributes of the class.
2. The user is provided a welcome message and asked to enter a 5-letter word which is converted to uppercase.
3. When the word is entered, validate that it is exactly five-letters long.
4. Check if the word entered matches the secret\_word.
   1. If not, check how many letters are in the correct location within the word.
   2. Also check how many letters are in the word, but not in the correct location.
   3. Subtract one from the attempts\_left attribute

(This can be done using one or two comprehensions of some other way)

1. If the word has not been matched and there are still attempts left,
   1. Print out ALL the words from the guesses list.
   2. For EACH character in EACH word in the guesses list make the character green if it is in the right spot and yellow if it is in the word, but in the wrong spot. Be careful of duplicate letters…each letter is only indicated once. Use colorama or another python library to display the colours.
   3. Prompt the user to enter another guess
2. If the word has not been matched and there are no attempts left,
   1. Print out ALL the words from the guesses list.
   2. Display a message telling the user they have lost.
3. If the word has been matched, congratulate the user, and tell them how many guesses it took them.
4. Prompt the user if they want to play again.
5. An example of the Console window for the program is shown here.

Welcome to Wordle. I have a five-letter word in mind.

What is your guess: raise

1. RAISE

What is your guess: Flare

1. RAISE
2. FLARE

What is your guess: grADE

1. RAISE
2. FLARE
3. GRADE

What is your guess: TRADE

1. RAISE
2. FLARE
3. GRADE
4. TRADE

Excellent! You guessed it in 4 guesses

Would you like to play again (Y/N)? N

1. Some extra things to do to challenge yourself:
   1. Check the word is a valid English Language Word (try the enchant library, but it is difficult to use)
   2. Keep track of how many guesses were used for each time and provide the user with a list of the number of guesses for each round. Something like:  
      Guesses Count  
       1   
       2 \*\*\* (3)  
       3 \*\*\*\*\*\* (6)  
       4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (19)  
       5 \*\*\*\*\*\*\* (7)  
       6 \*\*\*\*\*\*\*\*\*\* (10)
   3. Save the scores to a file that you read in when the game starts so you can track the scores between games.
   4. Prompt the user for their name so you can refer to them by name.
   5. Anything else…check with me.