End-of-unit game: Hangman

For this project, you will be using your knowledge of classes to create a hangman game that runs in the terminal. When you're done, you'll be able to enter a secret word into your code that a user will try to guess when running the program. Your opponent will have 6 *incorrect* guesses before the full hangman picture is drawn out and they lose. All the hangman drawings are provided to you in a list inside the class variable, self.board. The first drawing (index 0 of the list) should be displayed at the start of the game, or as long as the user has not made any incorrect letter guesses. When one incorrect guess has been made, you should display the second drawing (index 1). What you should start to notice here is that you should keep track of the x number of incorrect guesses, and then print out the x-1 index of the hangman drawing list. When the last hangman drawing is printed, the user has lost and you have won!

On each turn (meaning each guess the user gets), you will want to display the current hangman drawing, as well as the part of the secret word that the user has guessed (you can use underscores for the part of the secret word that the user has not guessed). For example, if the word is 'secret' and the user has correctly guessed only the letter 'e', the word should appear under the hangman drawing as _e_ _ e_.

What you'll need to do:
In the starter code, all of the methods that you will want to fill in for the class are outlined. Fill in these methods, keeping in mind the class variables that are outlined in self._init_(). You'll probably want to fill these in first. Think about the data structures you'll want to use, specifically how you might want to use a set (which has no repeated items). What does this help you accomplish? (i.e., you can compare the set of correctly-guessed letters to the set of letters in the word to see if the user has won)

Some strategies:
Look through all of the class methods before you start and make sure you understand what each of them are meant to do and why they are important. Try to write all of the functions except for play first, and when you think these functions are doing what you want, try to tackle play(). Having a good understanding of all the helper methods will help you use them together in play(). Finally, remember to use the keyword 'self' when accessing class variables or methods.

If you get stuck (in the play function, for example), take a step back from the code and write a list of what you want to happen, step-by-step. For example, "First, I want the user to make a guess, and then I want the computer to check if this guess is in the word. If it is in the word, the hangman drawing stays the same, and the user guesses again. If it's not in the word, the hangman drawing should change." Thinking on paper or out loud is sometimes really helpful when the coding gets overwhelming.

For Teachers:
If you feel that your student(s) don't need as much starter code, take out the TODOs INSIDE the class methods. KEEP the TODOs wrapped in triple quotes at the top of the methods.