Assignment: using classes with the Hangman game

In this assignment, you will transform the code from previous lab in order to use classes in a "real world" situation.

Build the `SecretWord` class

This class represents the word to be guessed. It contains an attribute: the secret word to be guessed.

The class has the following methods:

- `__init__`
  - this method is the constructor, and is called when the class is instantiated
  - if the constructor receives an argument, set the secret word to the argument received
  - if the argument is not provided (or `None`), open the file `words.txt` and set the secretword to a random word in the list
  - reuse / refactor the code from `pick_random_word`!
- `show_letters`
  - this method takes one argument: a list of letters
  - it returns a string revealing the letters provided as arguments in the secret word
  - reuse / refactor the code from `show_letters_in_word`!
- `check_letters`
  - this method takes one argument: a list of letters
  - it returns `True` if all letters of the secret word are present in the list of letters provided
  - reuse / refactor the code from `all_letters_found`!
- `check`
  - this method takes one argument: a string
  - it returns a boolean: `True` if the string provided is the secret word, `False` otherwise
- make sure your methods work **regardless** of case (case insensitive)

> You must decide on names for the instance attributes yourself!

You can check the `SecretWord` class with the test `test_secret_word.py`.

Build the `Game` class

This class represents the game being played. It has the following attributes:

- an instance of the `SecretWord` class
- an integer representing the number of turns that can still be played
- a list of letters tried by the player (start with an empty list)

> You must decide on most of the names for the instance attributes yourself!

`__init__`: the constructor

The constructor can receive an **optional** argument. This argument is the number of turns allowed for the game. Its default value is 10. You **must** have an attribute (or property) called `turns`.

Make sure you create a `SecretWord` instance, and that you initialize an empty list of "tried letters".

`play_one_round`: plays one round

This method deals with a single round of the hangman game. The program asks the player for `input`.

- The player can type one letter (and Enter)
  - if the player already tried that letter, ask again
  - add the letter to the list of `letters`
  - display the word on the screen, showing correctly guessed letters (use the `show_letters` method!)
  - remove 1 from the number of turns remaining
  - if all letters were guessed, `return True`, otherwise `return False`
- The player can type multiple letters (= a word)
  - Compare the word provided with the secret word. Use the `check` method!
  - remove 1 from the number of turns remaining
  - If the word was correctly guessed, `return True`

`play`: plays the game

This method is the main loop for the game.

At each turn of the loop:

- if there are 0 turns remaining, the player lost the game: exit the loop
- call the `play_one_round` method
- if the player won (found the word), or lost (word not found and no attempts remaining), exit the loop
- display a message informing the player whether they won or lost
- return `True` if the player won, or `False` otherwise

Submission

You should be able to play the game by running:

```
my_game = Game(10)  # for 10 turns
my_game.play()
```

Submit your `hangman.py` file to D2L.