Hangman Game

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

Rules of The Game:

For those of you who are unfamiliar with the rules of Hangman, here is a brief explanation. The object is to guess the word chosen by the computer in 6 or fewer guesses. The computer chooses a word is display a single dash (-) for each letter in the word. The user guesses a letter that might be in the word. The computer checks the letter against the word it selected and redisplays the unknown word with all copies of the guessed letter revealed. This continues until the user either makes 6 letter guesses that are not in the word or guesses all the letters in the word.

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

To do:

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

HangmanGame
+ secret_word: str + current_word_status: str - max_attempts: int # this is word being weird idk how to fix this + attempts_left: int - guesses: set + usr: str + condemned: str
+ __str__() -> str + __repr__() -> str + __len__() -> int + read_file() -> list + select_word() -> str + make_guess(guess: str) + is_win() -> bool + is_lost() -> bool + letter_validation(guess: str) -> bool + non_guessed_letters() -> str

+ dash_temp_name() -> str
+ hangman_art() -> str
+ result_msg() -> str
+ try_again_prompt()
+ intro()
+ main()

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 a letter is guessed twice, if the user runs out of guesses or if the user attempts to enter more than one letter. Complete the table below. Add as many test cases as you think are necessary to completely test the game.

Test Plan for Hangman			
Test Case		***Input Data***	***Expected output***
1.	Input Username	john	'John' as their Username
2.	Input the person getting Hang	doe	'Doe' as their name
3.	Input Username, if they did not add a name and only press enter	'' (empty string)	'User' is their username as default
4.	Input the person getting Hang, if they did not add a name and only press enter	''	'John Doe' is the default name
5.	Enter a guess	a	'a' will be 'A' through .upper(), then be added at guesses set, and return True from the letter_validator function and minus one for the attemtps_left
	Enter a guess twice	a is inputted again	Will display an error to enduser that the guess has already been guessed
	Enter more than one guess	abcd	Will display an error to enduser that the guess has more than one character
	Enter a guess that is not an alphabetical character	1	Will display an error that the guess is not an alphabetical character
	Try again	y	The enduser will play hangman again and attempts_left is reset to 7.
	Try again	n	The enduser will leave the program

Test Plan for Hangman			
	Test Case	***Input Data***	***Expected output***
	Try again	Chars that don't y or n in the str	Will display an error stating invalid input.
	Dunder str		Display the current word status
	Dunder repr		Display every attribute
	Dunder len		Return the amount of guesses
	Read_file	hangman.txt	Read hangman.txt and return a list consisting of each word from the text file
	Select_word	Word list from read_file func.	Return a random word from the word list
	Make_guess	guess	If the guess is in the secret word, it will appear in the current word status. If not deduct attempts
	Is_win		It return True
	Is_lost		It return True
	non_guessed letters		Return a str consisting of letters that are not guess yet
	dash_temp_name		Return a str with dash multiply by the length of the secret word
	hangman_art		Return an ascii art depending on the amount of attempts left
	result_msg		Reutrn either a win or lose msg, display the secret word, and the amount of guesses it took
	intro		Return a msg, telling user their amount of attempts and initialize the read_file func, assign the secret word, and assign the current word status

The HangmanGame Class

1. The HangmanGame class has at least the following attributes:

a. secret_word: String with the value of the secret word IN UPPER CASE.
b. current_word_status: String with the guessed letters in the right spot and dashes in the not guessed spot
c. max_attempts: int with the value of the maximum number of attempts. Default is 6.
d. 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.
e. guesses: **set** that is not passed to the constructor. Initialized as an empty set. Will have each guess added (appended) to the list. This should be private or at least protected.

2. The HangmanGame class has at least the following methods:
   a. read_file: Returns a list of the words in the provided file Hangmanwords.txt
   b. select_word: Returns a string of a random word selected from the list of words read in from the file.
   c. make_guess(guess): Checks if the guess is valid and updates the game state
   d. is_won: Returns True if the game is won and False otherwise.
   e. is_lost: Returns True if the game is lost (no more guesses allowed) and False otherwise.
   f. __str__: When object of the class is printed, displays the current_word_status attribute.
   g. __repr__: Displays the information about the class in the form object_name(word, current_status_word, max_attempts, attempts_left, guesses) in that format.
   h. __eq__: (maybe) determines if guessed letter in the same as the secret_word
   i. __getitem___: (probably) to get the individual letters from the set
   j. __setitem__: (probably) to set the guessed letter into the guesses set
   k. __len__: to get the length of the guesses
   l. Other dunder methods as needed
   m. Other methods as needed

How the Game Works

1. You have been provided a list of 50 words. When the program starts, initialize an object of type HangmanGame.
   a. The constructor should NOT have to pass anything. The constructor should call the method read_file to read the file.
   b. 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. The current_word_status attribute is set to a dash for each letter in the word.
   c. The constructor then initializes all the attributes of the class.
2. The user is provided a welcome message, the selected word is displayed (current_word_status), the alphabet of NON guessed letters is displayed and the user is asked to enter a letter word which they think is in the word. The letter is converted to uppercase.
3. When the letter is entered, validate that it is a single letter (no numbers and no multi-letters) and that it has not been entered before.
4. Check if the letter entered is in the secret_word.
   a. Add the letter to the guesses set

   b. If it is, check update the current_status_word replacing a dash for each time the letter appears in the word.
   c. If it is not, subtract one from the attempts left.
5. If the word has not been completely guessed (there are still dashes in current_word_status) and the user has attempts_left.
   a. Display the current_word_status and prompt the user for another letter.
   b. Print out the list of all the letters that have been guessed (this should be calculated from a string of all the letters in the alphabet removing the letters in the guesses set).
   c. Display a message telling the user how many more guesses they are allowed.
   d. Prompt the user to enter another guess.
6. If the word has not been completely guessed and the user has no more attempts (attempts_left = 0), display a message that the user has lost the game.
7. If the word has been matched, congratulate the user, and tell them how many guesses it took them.
8. Prompt the user if they want to play again.

9. An example of the Console window for the program is shown here.

```
Welcome to Hangman. I have a word in mind.
- - - - - -
ABCDEFGHIJKLMNOPQRSTUVWXYZ
What is your guess: T

T is correct
- - T T - -
ABCDEFGHIJKLMNOPQRSUVWXYZ
You can still make 6 wrong guesses
What is your guess: a

A is NOT correct
- - T T - -
BCDEFGHIJKLMNOPQRSUVWXYZ
You can still make 5 wrong guesses
What is your guess: r

R is correct
- - T T - R
BCDEFGHIJKLMNOPQSUVWXYZ
You can still make 5 wrong guesses
What is your guess: E

E is correct
- - T T E R
BCDFGHIJKLMNOPQSUVWXYZ
You can still make 5 wrong guesses
What is your guess: O

O is NOT correct
- - T T E R
BCDEFGHIJKLMNPQRSUVWXYZ
You can still make 4 wrong guesses
What is your guess: U
```

```
U is correct
- U T T E R
BCDFGHIJKLMNOPQSVWXYZ
You can still make 4 wrong guesses
What is your guess: B

Excellent! You guessed the word It was BUTTER

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

10. Some extra things to do to challenge yourself:
    a. 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)
```

    b. Save the scores to a file that you read in when the game starts so you can track the scores between games.
    c. Prompt the user for their name so you can refer to them by name.
    d. Anything else…check with me.