proj05: Hangman

Goal:
Implement a function, hangman(), that will start up and carry out an interactive Hangman game between a player and the computer.

Specifications:
1.The computer must select a word at random from the list of available words that was provided in words.txt. The functions for loading the word list and selecting a random word have already been provided for you in proj05.py.

2. The game must be interactive: it should let a player know how many letters the word the computer has picked contains and ask the user to supply guesses. The user should receive feedback immediately after each guess. You should also display to the user the partially guessed word so far, as well as either the letters that the player has already guessed or letters that the player has yet to guess.

3. A player is allowed some number of guesses. Once you understand how the game works, pick a number that seems reasonable to you. Make sure to remind the player of how many guesses s/he has left after each turn.

4. A player loses a guess only when s/he guesses incorrectly.

5. The game should end when the player constructs the full word or runs out of guesses. If the player runs out of guesses (s/he "loses"), reveal the word to the player when the game ends.

The output of an example game may look like this:

```
Welcome to the game, Hangman!
I am thinking of a word that is 4 letters long.
-------------
You have 8 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter: a
Good guess: _a _ _
-------------
You have 8 guesses left.
Available letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter: s
Oops! That letter is not in my word: _a _ _
-------------
You have 7 guesses left.
Available letters: bcdefghijklmnopqrtuvwxyz
Please guess a letter: t
Good guess: ta _t
-------------
You have 7 guesses left.
Available letters: bcdefghijklmnopqruvwxyz
Please guess a letter: r
Oops! That letter is not in my word: ta _t
```

```
--------------
You have 6 guesses left.
Available letters: bcdefghijklmnopquvwxyz
Please guess a letter: m
Oops! That letter is not in my word: ta _t
--------------
You have 5 guesses left.
Available letters: bdefghijklmnopquvwxyz
Please guess a letter: c
Good guess: tact
Congratulations, you won!
```

Hints:

- You should start by using the provided functions to load the words and pick a random one. It is helpful to turn the letter in this word into a list so that you can compare it to a list of blanks that gets filled in as users guess letters.

  For example, if the word is "cat" → ["c", "a", "t"] can be compared to ["_", "_", "_"]. If the user guesses "t", you can find the index of t in the word. Then, you can set the blank at that index to t.

- To create a list of blanks to represent the word to the user, you can create an empty list:

  guess_list = []

  Then, for each letter in the word, you can append a blank. This gives you the same number of blanks as letters in the word.

- Consider using string.lowercase to represent the letters the user has NOT guessed yet. string.lowercase is a built in string of every lowercase letter in the alphabet. You could say:

  ```
  var = string.lowercase
  print var
  ```

  You would see every lowercase letter printed out in one string.
  Then, when a user guesses a letter, you could remove it from the string you made (in this case, called var).

  To remove their user's guess from the var, you can use replace and replace the letter with nothing:

  ```
  var = var.replace(user_guess, "")
  ```

  Note: you can name a variable anything – not just var – this is just an example.