1. Download the file hangman_template.py from the course website; save it as hangman.py. Also download words.txt and save it in the *same place*. To make sure everything is ready, run hangman.py in IDLE. If it prints the line ‘’Loading word list form file… 55900 words loaded’’, everything is working right.
We’re going to start by storing the state of the game in variables at the top of the function play_hangman. The states are:

- secret_word: the words they are trying to guess (string).
- letters_guessed: the letters they have guess so far (list)
- mistakes_made: the number of incorrect guesses they’ve made so far (int)

You can name these something else if you’d like, but use a descriptive name. For now, we’ll set secret_word to be ‘claptrap’ to be get_word(), a function that pulls a random word from the file words.txt. This function is already defined for you. ‘claptrap’ was selected because it’s reasonably long and has duplicate letters – hopefully that will allow us to catch any bugs we might make.

2. Let's start writing code! Here's our approach: we'll write functions to take care of smaller tasks that we need to do in hangman, then use them to write the actual game itself. First, define the function word_guessed(). word_guessed() will return True if the player has successfully guessed the word, and False otherwise.

Example: If the letters guessed variable has the value

['a','l','m','c','e','t','r','p','n']

word_guessed() will return True. If the letters guessed variable has the value

['e','l','q','t','r','p','n']

word_guessed() will return False.

Hint: Obviously, you'll use a loop. There are two things you could loop over-- the letters in secret word or the letters in letters_guessed. Which one do you want to loop over? Don't just guess here, think! One of them makes sense and will be a lot easier than the other.

3. Back to Hangman. So you'll want to use the string library. Note how we have added the line **from string import *** to the top of the template. This imports all of the functions from the string library, so you can use them as if you've defined them within your own file.

4. Now define the function print_guessed() that returns a string that contains the word with a dash `-' in place of letters not guessed yet.

Example: If the letters guessed variable has the value [], the expression print print_guessed()

will print --------.

If the letters guessed variable has the value ['a','p'], the expression print print_guessed() will print --ap--ap.

If the letters guessed variable has the value ['a','l','m','c','e','t','r','p','n'], the expression print print_guessed() will print claptrap.

Hint: There are a lot of ways to go about this. One way is to iterate through secret word and append the character you want to print to a list. Then use the join function to change the list into a string: your last line will look something like return join(character list,'')

6. Now write the main game code. It helps to informally sketch out the code you want to write--this is called “pseudocode": an outline of what you are going to code that helps to guide you when you begin writing code. Here's an rough sketch of pseudocode, although you will want to expand on this:

```
continually loop:
        print "n guesses left"
        print "word"

        get letter in lowercase
        check - has letter already been guessed?
                If so, what should I do?
                If not, what should I do?
        check - is letter in word?
                If so, what should I do?
                If not, what should I do?
```

Write out some pseudocode that details what you want to do. It's a good idea to do this in comments within your code fille, so you can use this as a guideline to write your code. Hint: remember to use the break statement if you use the continual loop!

7. Congratulations! You've finished the game. Now we want to make it look pretty so everyone else will be impressed as we are :D. Polish your game a bit using the following extensions:

1. Optional: Don't use the word “claptrap" every time! Underneath the function play hangman you should see a commented line that looks like this:

secret_word = get_word()

Remove the `#' before it, and the secret word will be a new, random word each time!

2. Optional: Allow the user the option of guessing the full word early (perhaps by modifying your prompt to say something like, Enter a letter, or the word 'guess' to try and guess the full word:) Then, allow the user a try to enter in the full word)
You may want to take off 2 guesses if they enter an incorrect word...

3. Optional: Modify your print_guessed() function such that, in addition to what it already prints out, it prints out the letters the user has not yet guessed.