6.149 Project 1

`http://web.mit.edu/6.149/www/materials.html`

What to complete

Due: Friday, January 23, 2015 at 5 p.m.

`hangman.py`, which contains your implementation of hangman.

Project 1: The Game of Hangman

This document provides a step-by-step approach to implementing the game of hangman. While you may choose to modify your approach, you are required to write code in accordance with all function specifications laid out in `hangman.py`. Doing so will enable us to run automated test cases on your code if necessary.

1. We need to be able to check whether all of some facts are True, as well as whether at least one fact is True. For example, is every element (or at least one element) in a given list less than 6? Write the following warm-up exercises in a new file.

   a. Write a function that contains a `for` loop that will determine if every element in a variable `some_list` is less than 6.

   b. Check your code using `some_list = [1,5,3,4]` and on `some_list = [5,3,7,5]`.

   c. Now, write a new function that contains a `for` loop that will determine if at least one element in a list is less than 6. This function should return either `True` or `False`.

   d. Check your code using `[7,8,7,9]` and `[7,2,5,8]`

2. Download the files `hangman.py` and `words.txt` from the course website. Save them in the same folder. Before writing any code, run `hangman.py` in IDLE and make sure it prints:

```
Loading word list from file...
   55900 words loaded.
Enter play_hangman() to play a game of hangman!
```

We're going to start by storing the state of the game in variables at the top of the function `play_hangman`. The state is a complete description of all the information about the game. In the Nims game from Checkoff 4, the state would be:

- The current player
- How many stones are in the pile

For hangman, we need to store 3 pieces of information:

- `secret_word`: The word they are trying to guess (string).
- `letters_guessed`: The letters that they have guessed so far (list).
- `mistakes_made`: The number of incorrect guesses they've made so far (int).

For now, we've set `secret_word` to be "claptrap" in the `hangman()` function. Once we've finished our program and got it working, then we'll change the line `secret_word = 'claptrap'` to be `secret_word = get_word()`. `get_word` is a function that is already written for you that pulls a random word from the file `words.txt`. "claptrap" was selected because it's reasonably long and has duplicate letters - hopefully that will allow us to catch any bugs we might make.

Question: Why can't we use `len(letters_guessed)` for `mistakes_made`?

Note the constant variable underneath the helper code:

```
MAX_GUESSES = 6
```

Constant just means that we won't change it. This isn't enforced by the compiler, so be careful not to accidentally change the value of `MAX_GUESSES`. If we want to customize our hangman game after finishing the code, playing around with `MAX_GUESSES` would be a good place to start.

3a. Type this code into the shell:

```
for i in "hello":
    print i
for i in ['a',True,123]:
    print i
```

Make sure you understand how to interate over both strings and lists with for loops.

3b. We're going to 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, complete the function `is_word_guessed`. `is_word_guessed` takes two parameters - `letters_guessed` (a list) and `secret_word` (a string) - and returns `True` if the player has successfully guessed the word, and `False` otherwise.

Remembering that our `secret_word` is "claptrap," if `letters_guessed` is

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

`is_word_guessed()` will return `True`. If `letters_guessed` is

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

`word_guessed()` will return `False`.

Hint: There are two variables you could loop over - the letters in `secret_word` or the letters in `letters_guessed`. Which one do you want to loop over? One of them will be a lot easier than the other. You can use code from the first problem.

4a. Type this code into the shell:

```
word = "Hello"
lower_word = word.lower()
print word
word_as_list = ["H","e","l","l","o"]
word_as_string = "".join(word_as_list)
print word_as_string
```

Note that `join` is a string method. When we discussed dot notation, we learned that a string must always precede the dot when using a string method. The string we are passing as the first argument to the `join` method is `""`, the empty string. The second argument is `word_as_list`, in parentheses. `join` takes a list as input, and concatenates each element into a string, with elements separated by the argument passed in before the period. Since we passed in the empty string before the period, there is no separation between the elements.

4b. What lines of code belong in the missing spaces to achieve the desired outcome?

```
>>> List1 = ['H','e','a','r']
>>> ???????
>>> ???????
>>> print string1
hear
```

5. Now complete the function `print_guessed` that takes `secret_word` and `letters_guessed` as arguments. `print_guessed` prints a string that contains the secret word with a dash (`-`) in place of letters that have not been guessed yet.

If `letters_guessed` is `[]` and the `secret_word` is "claptrap," `print_guessed` will print `--------`.

If `letters_guessed` is `['a','p']`, `print_guessed` will print `--ap--ap`.

If `letters_guessed` is `['a','l','m','c','e','t','r','p','n']`, `print_guessed` will print `claptrap`.

Hint: One way to implement this function 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.

6. Now, write the main game code. Your code should meet the following requirements:

- At the start of every turn, print the number of guesses left and the player's current progress on guessing the secret word
- Take off two guesses if the player guesses r, s, t, l, n, or e and that letter is not in the word; otherwise, take off one guess
- Accept capital letters (you'll probably want to make them lowercase)
- Don't accept non-letters; prompt the user for new input
- If the user enters a letter he or she has already guessed, you should not take off a guess and instead prompt the user for new input
- At the end of the game, print whether the player won or lost
- If the user loses, reveal the secret word before exiting the program

Here's a rough sketch of pseudocode - you will need to add detail in your code file:

```
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?
```

Once your code is implemented, uncomment

```
# secret_word  = get_word()
```

so you can get a new secret word each time.

7. Congratulations! You've finished the game. Now we want to make it look pretty so everyone else will be as impressed as we are `:D`. Create a new file for any extensions so we can grade your original hangman solution without any distractions. Polish your game a bit using these ideas, or any of your own:

1. 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 to guess the full word:`) You may want to take off extra guesses if they enter an incorrect word.
2. Optional: ASCII GRAPHICS! On the `materials` section of the course webpage, you can find two files called `hangman_lib.py` and `hangman_lib_demo.py`. The first contains a set of ASCII graphics that you can use in your code; the second shows how to use the package. You can insert these into your Hangman game to make it much more exciting. **Be sure to credit the author of the graphics library by having your code print out "Art created by sk" at the beginning or end of your program.**
   Hint: Remember to add the line `import hangman_lib` at the top of your code. Do not copy the graphics directly into your code!
3. Optional: Modify your `print_guessed` function to also print out the letters the user has **not** yet guessed.