edX (https://www.edx.org)

MITx: 6.00.1x Introduction to Computer Science and Programming Using Python

zhushun0008 (/dashboard) ▾

Courseware (/courses/MITx/6.00.1_3x/2T2014/courseware) Updates & News (/courses/MITx/6.00.1_3x/2T2014/info)

Calendar (/courses/MITx/6.00.1_3x/2T2014/89309559b0414f6d8cbef9e48ca19f4b/) Wiki (/courses/MITx/6.00.1_3x/2T2014/course_wiki)

iscussion (/courses/MITx/6.00.1_3x/2T2014/discussion/forum) Progress (/courses/MITx/6.00.1_3x/2T2014/progress)

Help

HANGMAN PART 2: THE GAME (15 points possible)

Now you will implement the function `hangman`, which takes one parameter - the `secretWord` the user is to guess. This starts up an interactive game of Hangman between the user and the computer. Be sure you take advantage of the three helper functions, `isWordGuessed`, `getGuessedWord`, and `getAvailableLetters`, that you've defined in the previous part.

Hints:

- You should start by noticing where we're using the provided functions (at the top of `ps3_hangman.py`) to load the words and pick a random one. Note that the functions `loadWords` and `chooseWord` should only be used on your local machine, not in the tutor. When you enter in your solution in the tutor, you only need to give your `hangman` function.

- Consider using `lower()` to convert user input to lower case. For example:

  ```
  guess = 'A'
  guessInLowerCase = guess.lower()
  ```

- Consider writing additional helper functions if you need them!

- There are four important pieces of information you may wish to store:

  1. `secretWord`: The word to guess.
  2. `lettersGuessed`: The letters that have been guessed so far.
  3. `mistakesMade`: The number of incorrect guesses made so far.
  4. `availableLetters`: The letters that may still be guessed. Every time a player guesses a letter, the guessed letter must be removed from `availableLetters` (and if they guess a letter that is not in `availableLetters`, you should print a message telling them they've already guessed that - so try again!).

Sample Output
The output of a winning game should look like this...
And the output of a losing game should look like this...

Note that if you choose to use the helper functions `isWordGuessed`, `getGuessedWord`, or `getAvailableLetters`, you do not need to paste your definitions in the box. We have supplied our implementations of these functions for your use in this part of the problem. If you use additional helper functions, you will need to paste those definitions here.

Your function should include calls to `raw_input` to get the user's guess.

Why does my Output Have `None` at Various Places?

```
def hangman(secretWord):
    '''
    secretWord: string, the secret word to guess.

    Starts up an interactive game of Hangman.

    * At the start of the game, let the user know how many
      letters the secretWord contains.
```

Unanswered

To make the output less verbose, we are only testing simple input cases here. When your hangman function passes these checks, paste the same code again into the next box to test it on harder input cases.

Check Save *You have used 0 of 30 submissions*

HANGMAN - COMPLEX TESTS (20 points possible)

```
# When your hangman function passes the checks in the previous
# box, paste your function definition here to test it on harder
# input cases.


```

Unanswered

Check Save *You have used 0 of 30 submissions*

Show Discussion New Post