Com S 127x
Fall 2016
Assignment 5

Due Date: Wednesday, November 16, 11:59 pm (midnight)

"Late" deadline (25% penalty): Thursday, November 17, 11:59 pm

General information

This assignment is to be done on your own. See the Academic Dishonesty policy in the syllabus for details.

If you need help, see your instructor or one of the TAs. Lots of help is also available through the Piazza discussions.

Note: This assignment does not have to be turned in until after the exam on 11/14, but it's a good idea to do it before that since it provides more practice with lists and strings, which are covered on the exam.

Introduction

The purpose of this assignment is to give you some more practice with lists, to put together an application involving several components, and to work with global variables.

This is a relatively short assignment in which you will implement the letter-guessing game known as "hangman". The idea of the game is that there is a *hidden word* that the player tries to guess one letter at a time. There is also a *displayed word* in which each letter in the hidden word is shown as a dash. If a correct letter is guessed, then those positions of the displayed word are filled in with the correct letter. There is a maximum number of guesses. (Traditionally, an incorrect guess results in a body part being added to a figure on a scaffold. If the figure is completed before the player guesses all the letters in the word, the player loses.)

You'll implement two modules: **`hangman.py`** and **`words.py`**. The **`hangman.py`** module keeps track of the *state* of the game as it is played (the secret word, which letters have been guessed, how many wrong guesses, and so on). The **`words.py`** module is just used at the beginning to randomly generate a secret word. To create a complete application, there is a third module **`hangman_ui.py`**, that is an example of a user interface for the game. *The hangman_ui.py module is provided for you and you should not modify it.*

A screenshot from the sample text-based user interface is shown below.

```
  ____
  1   |
  o   |
 /|\  |
  /\  |
=======

Word so far:    - p a t - - a
Letters already guessed: t, e, h, a, n, p, m, k, i
You have 0 guesses left
Sorry, you've lost.
The word was spatula
```

The module hangman.py

The hangman.py module contains the game state using *global variables*. In general, global variables should not be accessed directly from outside the module; instead, the module provides a set of functions for interacting with the game. Here are the specified functions:

`start(hidden_word, max_guesses)`

Initializes all global variables for a new game using the given hidden word and maximum number of guesses.

`guess_letter(letter)`

Checks whether the given letter occurs in the hidden word and updates the state accordingly. Specifically, if the letter occurs in the hidden word and has not yet been guessed, then all occurrences of the letter are revealed in the displayed word, and the function returns True. Otherwise the function returns false. In either case, the letter is added to the list of guessed letters.

`get_displayed_word()`

Returns the displayed word as a list of characters.

`get_hidden_word()`

Returns the hidden word.

`get_guesses_left()`

Returns the number of guesses that the player has left.

`get_guessed_letters()`

Returns the letters that have already been guessed, as a list of characters

`is_over()`

Returns True if the game is over, False otherwise.

`is_won()`

Returns True if the game is over and the player has won, False otherwise.

To see how these functions would be used in a game, see the excerpt below from the sample user interface. (The call to **`get_word()`** is explained in the next section. The function **`display_game()`** is not shown here, but it just prints out the current state of the game showing the displayed word and the letters guessed so far. See **`hangman_ui.py`** for details.)

```
hidden = words.get_word()
hangman.start(hidden, 6)
while (not hangman.is_over()):
    display_game()
    letter = input("Enter your guess: ")
    correct = hangman.guess_letter(letter)
    if correct:
        print("Good guess!")
    else:
        print("Nope.")
    if hangman.is_won():
        print("You win!")
    elif hangman.is_over():
        display_game()
        print("Sorry, you've lost.")

print("The word was " + hangman.get_hidden_word())
```

The sample code includes a skeleton file **`hangman.py`** with declarations for some global variables that might be useful to have. However, you can change these as you see fit.

The module words.py

This module has two functions for generating random words. The first one is already implemented for you.

`get_word()`

Returns a randomly selected word from a fixed list of the names of kitchen items.

`get_word_from_file(filename)`

Returns a randomly selected word from the given file. The file must have a single word on each line.

The sample UI hangman_ui.py

This module is just an example of how the two modules above could be made into a complete game with a simple text-based user interface.

If you have questions

For questions, please see the Piazza Q & A pages and click on the folder **`hw5`**. If you don't find your question answered, then create a new post with your question. Try to state the question or topic clearly in the title of your post, and attach the tag **`hw5`**. *But remember, do not post any source code for the classes that are to be turned in.* It is fine to post source code for general Python examples that are not being turned in. (In the Piazza editor, use the button labeled "pre" to have code formatted the way you typed it.)

If you have a question that absolutely cannot be asked without showing part of your source code, make the post "private" so that only the instructors and TAs can see it.

Getting started and testing

Before you start this assignment, make sure you understand the examples **`number_game.py`** and **`number_game_ui.py`**. See the topics page for November 7.

The best way to develop code is to start with simple test cases and check them as you write the code. This will be much easier than trying to check what's happening by running the UI over and over again. You might begin with just the basics of the **`start()`** function to correctly initialize all the global variables. For example, you could write a simple test case like this:

```
hangman.start("foo", 6)
print(hangman.get_displayed_word())   # Expected ['-', '-', '-']
print(hangman.get_hidden_word())      # Expected 'foo'
print(hangman.get_guessed_letters())  # Expected []
print(hangman.guesses_left())         # Expected 6
print(hangman.is_over())              # Expected False
print(hangman.is_won())               # Expected False
```

Remember, you can easily make a list of `'-'` characters for the displayed word using
`['-'] * len(hidden_word)`

Next, what happens as you guess letters that are *not* in the word? The function returns False, the list of guessed letters is updated, and the number of remaining guesses goes down:

```
result = hangman.guess_letter('x')
print(result)                         # Expected False
print(hangman.get_guessed_letters()) # Expected ['x']
print(hangman.guesses_left())        # Expected 5
```

Also, the displayed word does *not* change.

```
print(hangman.get_displayed_word())  # Expected ['-', '-', '-']
```

Now if you guess a letter that *is* in the word, the function returns True, the list of guessed letters is updated, and the displayed word is updated:

```
result = hangman.guess_letter('o')
print(result)                         # Expected True
print(hangman.get_guessed_letters()) # Expected ['x', 'o']
print(hangman.get_displayed_word())  # Expected ['-', 'o', 'o']
```

And you should be able to verify that there are still 5 guesses remaining.

Documentation and style

Include comments at the top of each file with

- your name, and
- a brief statement of what the program does

For each *function* you write, include a docstring that states what it does. For each global variable that you define, include a comment explaining what it represents.

What to turn in

Turn in the two files `hangman.py` and `words.py`.
Upload your two files to the Homework 2 submission link on Blackboard. Be sure to CHECK your submission history and make sure that you successfully submitted both files. **Remember you can't just upload the files, you have to click the "Submit" button!**