Challenge Assignment: Hangman

The game of Hangman: a game for two in which one player tries to guess the letters of a word, and failed attempts are recorded by drawing a gallows and someone hanging on it, line by line.

Summary

You will write a program using the python language to create a simple console game of the word guessing game called hangman. Your program should read from a file containing a list of words, choose a random word, and allow the user to guess letters. The game should only allow the user to guess wrong 6 times before they lose, drawing more of the man each time. If the user guess the full word without guessing wrong 6 times, they win!

Detailed Instructions

Download the compiled version of the game here. Then unzip the file named hangmanCompiled.zip. You should see a folder with two files inside one called hangman.pyc and one called words.txt. To run the game in the terminal by changing the directory to the hangmanCompiled folder and typing “python hangman.pyc”. This will show you how the game should work, and you will write the code to mimic this game.

To begin your version, first create a folder named “hangman” and copy the words.txt file into that folder. Then use text wrangler to save a new file called “hangman.py” into the hangman folder.

Copy the following starter code into your hangman.py game:

```
def get_words():
    pass
def pick_word(words):
    pass
def new_game(words):
    pass
def is_word_guessed(game):
    pass
def is_game_over(game):
    pass
def guess_letter(game, letter):
    pass
def display_picture(game):
    pass
def display_word(game):
    pass
def display_guessed_letters(game):
    pass
def display_status(game):
    pass

def main():
    words = get_words() #get long list of words
    keep_playing = 'y'
    while keep_playing == 'y':
        game = new_game(words) #get a list with the game data  list, string, int
        display_status(game)
        while not is_game_over(game):
            guess = raw_input("next guess? ")
            guess = guess.strip()
            if len(guess) > 0:
                guess = guess.lower()
                game = guess_letter(game, guess[0])
            display_status(game)

        if is_word_guessed(game):
            print "You win!"
        else:
            print "The word is:", game[1]
            print "You died."

        print "Would you like to play again? y/n: "
        keep_playing = raw_input()
    print "Good bye."
    return

if __name__ == "__main__":
    main()
```

Function Descriptions:

`get_words()`
This function has no input parameters. It uses the open() command to read all the words from the words.txt file. It should fill an empty list named “words” with all the words from the file using a for loop to append all the words in the file to the list one at a time. It returns the words list. (Don’t forget to close the file)

`pick_word(words)`
This function has one input parameter for the words list. It uses the random library (your file should import random at the top) to choose a random word from the list. Simply use the randrange function to choose a random number within the length of the list, then index the list at that position. Return the random word.

`new_game(words)`
This function has one input parameter for the words list. It should return a list with the following three items: a list of the guessed letters (starts as an empty list []), the word they are trying to guess (retrieved using the `pick_word` function), and an integer to keep track of the number of misses.

`is_word_guessed(game)`
This function has one input parameter for the game list. It should read the guessed letters list from the game list, and the word from the game list. Using a for loop it should check that all the letters in the word have been guessed. If all the letters in the word are in the guessed letters list than return True, otherwise if not all the letters in the word are in the guessed letters list return False.

`is_game_over(game)`
This function has one input parameter for the game list. It should read the number of misses from the game list and if the number of misses is greater than 6 it should return True. Otherwise it should return the value (True or False) of calling the is_word_guessed(game) function (so if the word is guessed the game is over). Otherwise return False.

`guess_letter(game, letter)`
This function has two input parameters, one for the game list and one for the letter currently being guessed. First the letter should be appended (added) to the guessed letters list in the game list so that the letter is remembered. Next, use a for loop to check if the letter is in the word in the game list, if it isn’t then add 1 to the misses in the game list. Return the updated game list.

`display_picture(game)`

This function has one input parameter for the game list. Based on the number of misses from the game list, this function should print the visual representation of the number of misses by drawing a stick figure man on a gallows using characters from the keyboard.
example:

```
 -----
 |   !
 |   O
 |  /|\
 |  / \
 |
```

`display_word(game)`
This function has one input parameter for the game list. This function will print the current state of the guessed word. Using a for loop, loop through each letter in the word from the game list, and if the letter in the guessed letters list from the game list then print the actual letter, otherwise print an underscore (_) signifying an unguessed letter.
examples:

```
_ l v e _ _ o r l d
p r _ g r a m m i _ g
```

`display_guessed_letters(game)`
This function has one input parameter for the game list. This function simply prints the list of guessed letters from the game list.

`display_status(game)`
This function has one input parameter for the game list. This function should call the display_picture(game), display_word(game), and display_guessed_letters(game) function in that order. You may need to add a few extra print commands for spacing.