Problem 1 - Is the Word Guessed

(10/10 points)

Please read the Hangman Introduction before starting this problem. We'll start by writing 3 simple functions that will help us easily code the Hangman problem. First, implement the function isWordGuessed that takes in two parameters - a string, secretWord, and a list of letters, lettersGuessed. This function returns a boolean -True if secretWord has been guessed (ie, all the letters of secretWord are in lettersGuessed) and Falseotherwise.

def isWordGuessed(secretWord, lettersGuessed):

'''

secretWord: string, the word the user is guessing

lettersGuessed: list, what letters have been guessed so far

returns: boolean, True if all the letters of secretWord are in lettersGuessed;

False otherwise

'''

# FILL IN YOUR CODE HERE...

for x in secretWord:

if x not in lettersGuessed:

return False

return True

Problem 2 - Printing Out the User's Guess

(10/10 points)

Next, implement the function getGuessedWord that takes in two parameters - a string, secretWord, and a list of letters, lettersGuessed. This function returns a string that is comprised of letters and underscores, based on what letters in lettersGuessed are in secretWord. This shouldn't be too different from isWordGuessed!

Example Usage:

>>> secretWord = 'apple'

>>> lettersGuessed = ['e', 'i', 'k', 'p', 'r', 's']

>>> print(getGuessedWord(secretWord, lettersGuessed))

'\_ pp\_ e'

When inserting underscores into your string, it's a good idea to add at least a space after each one, so it's clear to the user how many unguessed letters are left in the string (compare the readability of \_\_\_\_ with\_ \_ \_ \_ ). This is called *usability* - it's very important, when programming, to consider the usability of your program. If users find your program difficult to understand or operate, they won't use it!

For this problem, you are free to use spacing in any way you wish - our grader will only check that the letters and underscores are in the proper order; it will not look at spacing. We do encourage you to think about usability when designing.

def getGuessedWord(secretWord, lettersGuessed):

'''

secretWord: string, the word the user is guessing

lettersGuessed: list, what letters have been guessed so far

returns: string, comprised of letters and underscores that represents

what letters in secretWord have been guessed so far.

'''

# FILL IN YOUR CODE HERE...

string = ""

for x in secretWord:

if x in lettersGuessed:

string += x

else:

string +='\_'

return string

Problem 3 - Printing Out all Available Letters

(10/10 points)

Next, implement the function getAvailableLetters that takes in one parameter - a list of letters,lettersGuessed. This function returns a string that is comprised of lowercase English letters - all lowercase English letters that are **not** in lettersGuessed.

Example Usage:

>>> lettersGuessed = ['e', 'i', 'k', 'p', 'r', 's']

>>> print(getAvailableLetters(lettersGuessed))

abcdfghjlmnoqtuvwxyz

Note that this function should return the letters in alphabetical order, as in the example above.

For this function, you may assume that all the letters in lettersGuessed are lowercase.

**Hint:** You might consider using string.ascii\_lowercase, which is a string comprised of all lowercase letters:

>>> import string

>>> print(string.ascii\_lowercase)

abcdefghijklmnopqrstuvwxyz

def getAvailableLetters(lettersGuessed):

'''

lettersGuessed: list, what letters have been guessed so far

returns: string, comprised of letters that represents what letters have not

yet been guessed.

'''

# FILL IN YOUR CODE HERE...

notGuessed = []

# notGuessed = string.ascii\_lowercase

for x in range(26):

notGuessed += chr(x + ord('a'))

for y in lettersGuessed:

notGuessed.remove(y)

string = ""

for z in notGuessed:

string += z

return string

### Problem 4 - 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 inavailableLetters, you should print a message telling them they've already guessed that - so try again!).