**Name: Session:**

**Programming I**

**A Word Guess Game**

**Lab Exercise 1.14.2021**

A WORDGAME: wordGuess

For this problem, you will implement a variation of a classic word guessing game. In this problem, the second player will always be the computer, who will be picking a word at random.

In this problem, you will implement a **function**, called wordGuess that will start up and carry out an interactive Word Guess game between a player and the computer. Before we get to this function, we'll first implement a few helper functions to get you going.

For this problem, you will need the code files *wordGuess.py* and *words.txt*, which can be found on the server at location \\Ada\Data Files\Programming I\Lab Exercise 1.7.2020. These two files must be in the same folder. Open and run the file *wordGuess.py* without making any modifications to it, in order to ensure that everything is set up correctly. By "open and run" I mean do the following:

* Go to Idle. From the File menu, choose "Open".
* Find the file wordGuess.py and choose it.
* The template wordGuess.py file should now be open in Idle. From the Run menu, choose "Run Module" (or simply hit the F5 key).

The code we have given you loads in a list of words from a file. If everything is working okay, after a small delay, you should see the following printed out:

Loading word list from file...

55900 words loaded.

If you see an IOError instead (e.g., "No such file or directory"), you should change the value of the WORDLIST\_FILENAME constant (defined near the top of the file) to the **complete** pathname for the file words.txt (This will vary based on where you saved the file).

The file wordGuess.py has a number of already implemented functions you can use. You can ignore the code between the following comments, though you should read and understand how to use each helper function by reading the docstrings:

# -----------------------------------

# Helper code

# You don't need to understand this helper code,

# but you will have to know how to use the functions

# (so be sure to read the docstrings!)

.

.

.

# (end of helper code)

# -----------------------------------

You will want to do all of your coding for this problem within this file as well because you will be writing a program that depends on each function you write.

Requirements

Here are the requirements for your game:

1. The computer must select a word at random from the list of available words that was provided in *words.txt*. The functions for loading the word list and selecting a random word have already been provided for you in *wordGuess.py*.
2. The game must be interactive; the flow of the game should go as follows:

* At the start of the game, let the user know how many letters the computer's word contains.
* Ask the user to supply one guess (i.e. letter) per round.
* The user should receive feedback immediately after each guess about whether their guess appears in the computer's word.
* After each round, you should also display to the user the partially guessed word so far, as well as letters that the user has not yet guessed.

1. Some additional rules of the game:

* A user is allowed 8 guesses. Make sure to remind the user of how many guesses s/he has left after each round. Assume that players will only ever submit one character at a time (A-Z).
* A user loses a guess **only** when s/he guesses incorrectly.
* If the user guesses the same letter twice, do not take away a guess - instead, print a message letting them know they've already guessed that letter and ask them to try again.
* The game should end when the user constructs the full word or runs out of guesses. If the player runs out of guesses (s/he "loses"), reveal the word to the user when the game ends.

## wordguess PART 1: IS THE WORD GUESSED?

We'll start by writing 3 simple functions that will help us easily code the Wordguess 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 False otherwise.

Example Usage:

>>> secretWord = 'apple'

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

>>> print (isWordGuessed(secretWord, lettersGuessed))

False

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

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

'''

for i in range(len(secretWord)):

if secretWord[i] not in lettersGuessed:

return False

return True

## PRINTING OUT THE USER'S GUESS

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 fromisWordGuessed!

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\_ \_ \_ \_ ). However, 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.

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

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.

'''

guess =''

for i in range(len(secretWord)):

if secretWord[i] in lettersGuessed:

guess += secretWord[i]

else:

guess += '\_ '

return guess

## PRINTING OUT ALL AVAILABLE LETTERS

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

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

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.

'''

letters = 'abcdefghijklmnopqrstuvwxyz'

avail = ''

for i in range(len(letters)):

if letters[i] not in lettersGuessed:

avail += letters[i]

return avail

**Wordguess PART 2: THE GAME**

Now you will implement the function wordGuess, 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 wordGuess.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 wordGuess 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!).

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

def wordguess(secretWord):

'''

secretWord: string, the secret word to guess.

Starts up an interactive game of Wordguess.

\* At the start of the game, let the user know how many

letters the secretWord contains.

\* Ask the user to supply one guess (i.e. letter) per round.

\* The user should receive feedback immediately after each guess

about whether their guess appears in the computer’s word.

\* After each round, you should also display to the user the

partially guessed word so far, as well as letters that the

user has not yet guessed.

'''

lettersGuessed = []

mistakesMade = 0

guessesLeft = 8

wordGuessed = False

guessedWord = getGuessedWord(secretWord, lettersGuessed)

print("Welcome to the game Word Guess!")

print("I am thinking of a word that is " + str(len(secretWord)) + " letters long")

while guessesLeft > 0 and not wordGuessed:

print("---------------------")

print("You have " + str(guessesLeft) + " guesses left")

availableLetters = getAvailableLetters(lettersGuessed)

print("Available letters: " + availableLetters)

guess = input("Please guess a letter: ")

if guess in lettersGuessed:

print("Oops! You've already guessed that letter" + guessedWord)

else:

lettersGuessed.append(guess)

if guessedWord != getGuessedWord(secretWord, lettersGuessed):

guessedWord = getGuessedWord(secretWord, lettersGuessed)

print("Good guess: " + guessedWord)

if isWordGuessed(secretWord, lettersGuessed):

wordGuessed = True

else:

print("Oops! That letter is not in my word: " + guessedWord)

mistakesMade += 1

guessesLeft -= 1

print("---------------------" )

if guessesLeft > 0:

print("Congratulations, you won!")

else:

print("Sorry, you ran out of guesses. The word was " + secretWord)

**When you have your game working print a sample of game output, attach to this packet and turn in.**