Evil Hangman

LAB # 2

By

Corey Henry

***“On my honor, as a Mississippi State University student, I have neither***

***given nor received unauthorized assistance on this academic work.”***

Signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CSE-1384-06-201430 Intermediate Computer Programming

Class Section # 6

Jesse Farek

2/3/2015

**Analysis and Conclusions**

This lab was using classes to excute a video game called hangman and allow the computer to cheat which was already provided for you. I used a function to run the program and asked the user how many letters they would like in their would, then I used a for loop to create the blanks for the letters based on the amount they had chosen. I used lists to keep track of the blanks for the letter and how many letter they had gotten wrong and to display them. Loops, lists, and the ability to call a class was very crucial to this lab. I had a very difficult time understanding what the class already did for me and what I had to do. I also used if statements a lot in my program. If I could do it over again I would use less complicated coding. It could have been more efficient and less complicated with simple terms instead of loops. I also could have imported the classes instead of pasting the code into my program.

Analysis Questions

1. The program was a lot easier to write with the word class already written, it would have required a lot more understanding of classes then was needed in this lab, and we would have had to learn new techniques to be able to do what the class did like calling the dictionary.
2. I would not change anything; the class worked very well and fitted the program perfectly.
3. The hangman made it easy as well, although I would have done it different it made it easier because I didn’t have to draw the hangman.
4. I would have added the list that creates the bottom to the hangman so it is already do for you and I wouldn’t have to repeat myself.
5. I learned a lot about creating objects of a class and how to use and navigate through them.

Source Code:

#Corey Henry #Date Assigned: 27jan15

# #

#Course CSE 1384 Sec 06 #Date Due: Feb 3

#File name: Lab2-Evil Hangman

#

#Program description- create a game of evil hangman

#create main function to run program

def main():

#get the number of letters

number\_of\_letters = int(input('Welcome to Hangman, please choose the number of letters you wish to play with: '))

#set object

display = Hangman()

#set guesses to 0

guesses = 0

#create empty list for the wrong letters

wrong\_letters = []

#create the base that will show all the letters

base = []

letters = Word(number\_of\_letters)

#loop that creates the ammount of spaces for the ammount of letters chosen

for each in range(number\_of\_letters):

base.append('\_ ')

#loop for the ammounnt of guesses you are allowed

while guesses <= 7:

cheat = letters.get\_word()

#option to choose word or letter guess

guess = input('Would you like to guess a letter or the word? ')

#if statements that split the difference between letter or word guesses

if guess == 'letter':

letter = input('What is the letter you would like to guess? ')

returned\_value = letters.guess\_letter(letter)

#if the letter is not in the word, add to the guess count and add to the list of wrong letters and print the wrong letters.

if returned\_value[0] == -1:

display.guessed\_wrong()

wrong\_letters.append(letter)

print(wrong\_letters)

guesses += 1

else:

for each\_index in returned\_value:

base[each\_index]=(letter)

win =''

#create a loop to make a string format of the list base

for each\_index in base:

win += (each\_index)

#use the win and cheat to determine if you win the game

if win == cheat:

print('Congrats you win!')

quit()

print(base)

display.print\_hangman()

if guess == 'word':

word = input('What word do you think it is? ')

returned\_answer = letters.guess\_word(word)

if guesses == 7:

cheat = letters.get\_word()

print('Sorry you lose, the word was ',cheat)

quit()

class Hangman:

#Builds the gallows

def \_\_init\_\_(self):

self.gallows = [[' ', ' ', '\_', '\_', '\_', ' '],

[' ', '|', ' ', ' ', '|', ' '],

[' ', '|', ' ', ' ', ' ', ' '],

[' ', '|', ' ', ' ', ' ', ' '],

[' ', '|', ' ', ' ', ' ', ' '],

[' ', '|', ' ', ' ', ' ', ' '],

['\_', '|', '\_', ' ', ' ', ' ']]

self.guesses = 0

#Increments the number of incorrect guesses

def guessed\_wrong(self):

self.guesses += 1

if self.guesses > 8:

raise ValueError("The man has been hung. No more guesses are allowed")

#Prints the hangman

def print\_hangman(self):

#Update the gallows

if self.guesses >= 1:

#Add head

self.gallows[2][4] = 'O'

if self.guesses >= 2:

#Add neck

self.gallows[3][4] = '|'

if self.guesses >= 3:

#Add left arm

self.gallows[3][3] = '\\'

if self.guesses >= 4:

#Add right arm

self.gallows[3][5] = '/'

if self.guesses >= 5:

#Add torso

self.gallows[4][4] = '|'

if self.guesses >= 6:

#Add left leg

self.gallows[5][3] = '/'

if self.guesses >= 7:

#Add right leg

self.gallows[5][5] = '\\'

#Print the hangman

for each in self.gallows:

one\_line = ""

for each\_char in each:

one\_line += each\_char

print(one\_line)

import urllib.request

import random

class Word:

#Initializes the dictionary for all possible words of the correct lenght

def \_\_init\_\_(self, number\_of\_letters):

if type(number\_of\_letters) is not int:

raise TypeError ("Words need a length that is an integer data type.")

page = urllib.request.urlopen("http://nifty.stanford.edu/2011/schwarz-evil-hangman/dictionary.txt")

text = page.read().decode("utf8")

all\_words = text.split()

self.dictionary = []

#Build dictionary

for each in all\_words:

if len(each) == number\_of\_letters:

self.dictionary.append(each)

page.close()

if len(self.dictionary) < 1:

raise ValueError ("There are no words of that length in the dictionary.")

#print(self.dictionary)

#Models guessing a letter. It requires a character. It returns a

#list of positions the letter in the word. If the letter is not in the

#word, it returns a list that contains a -1.

def guess\_letter(self, letter):

if len(letter) > 1:

raise ValueError ("Must send a single letter.")

#Create an empty list to hold the positions of the letter in the word

positions = []

#Evil hangman rules

if len(self.dictionary) > 1:

temp = []

#Build a list that contains all words that have the guessed letter

for each\_word in self.dictionary:

if letter in each\_word:

temp.append(each\_word)

#If all words with guessed letter can be removed and still have words

#left in the list of possible words, remove them

if len(temp) < len(self.dictionary):

for each\_word in temp:

self.dictionary.remove(each\_word)

#If all words in list of possibles contain guessed letter, choose a

#word randomly

else:

random\_position = random.randrange(len(self.dictionary))

word = self.dictionary[random\_position]

self.dictionary = [word]

#Regular hangman rules

if len(self.dictionary) == 1:

word = self.dictionary[0]

#Record positions where the letter is found

index = 0

while index < len(word):

if word[index] == letter:

positions.append(index)

index += 1

#If no letter was found

if len(positions) == 0:

positions.append(-1)

return positions

#Models guessing a word. It requires a string. It returns a either

#a True (the guessed word is correct) or False.

def guess\_word(self, word):

if len(self.dictionary) > 1:

self.dictionary.remove(word)

return False

if word == self.dictionary[0]:

return True

#Returns the word or the first word in the list of possible words

def get\_word(self):

return self.dictionary[0]

main()