day35_hangman.py (20 points) 12/9/19

Today and tomorrow we are going to create a hangman game. If you recall, hangman is a game where you guess letters, trying to discover a secret word. The idea is to guess all the hidden letters before you get "hanged", and each wrong guess gets you a little bit closer to the end. A sample run is at the end of this document.

There are many ways to make a hangman game, and I'm going to suggest one way to you that I have found works well. You can do it a different way if you want.

Store the secret word in a variable, then create a list containing "_" underline characters, the same number of entries as your word has letters. When the user guesses a letter, replace the blank in the right spot with the letter the user guessed correctly, printing out the results each time. Keep track of how many bad guesses the user makes and end the game after 9 failed guesses.

Make the hangman program initially without printing any hanging art. Just report the number of guesses they have remaining after each turn. You can make the program print art later if you have time.

Let's say the user inputs their guess into a variable called guess. You can quickly check if their entry is in the secret word like this:

```
if guess in secretWord:
```

Then, you'd need to find exactly where the letter occurs. A great way to do this would be to use a for loop to check every letter in the word. Remember, a letter may occur more than once in a word and all instances of the letter should be shown after a guess. The following code would do this; you'll obviously have to adapt it to your program for it to work:

```
for x in range(len(secretWord)):
    if secretWord[x]==guess:
        blanks[x]=guess
```

Here is a checklist:

- Set up a list with a bunch of words in it. Choose a word randomly from the list to be the secret word.
- Make the secret word and all guesses uppercase.
- Create a list with the right number of blanks in it. Print this list out each turn before you ask the user to make a guess. Replace blanks with letters when the user guesses a correct letter.
- Print the blanks and already-guessed letters after each turn nicely with a space between each. See sample output below.
- Track and report the number of bad guesses, end the game after 9 bad guesses. Tell the user what the word was if they lose.
- If there is a letter repeated in the word, all instances of it get replaced when the user guesses that letter.
- Track and report the letters the user has already guessed. Print them out nicely after each turn. If they guess the same letter again don't count it against them, just tell them they already guessed that letter and ask them to try again.
- Each turn, check to see if the user has guessed all of the letters and if they have, congratulate them and end the round.
- Ask the user if they would like to go again, choose a new secret word, reset the bad guess count and all other important variables.

For the record, I know you can go out on the internet and find a hangman program written in Python without doing any work. Doing this would be cheating.

Do not copy someone else's code.

The goal of this assignment is to get you to make a larger, more complicated program than we have done so far, and to become a better programmer through the process.

I will know if you copied someone else's program. There are a million ways to write this program, and this is an opportunity for you to solve a problem your own way.

Do your own work for this project. Your project cannot be exactly the same as the person next to you or someone else in the room. Your variables should have different names. The program should be structured somewhat differently. If you get to the art stage (optional), your program should have different art.

If you turn in a version of the program from the internet or if you turn in someone else's program as your own, even if you make a few changes here or there, you will both get a zero and get a referral to the administration for cheating. So please, DO YOUR OWN WORK! You will need to be able to explain to me what your program is doing, and how it is working, so don't use any code you don't understand.

When you are done please get me to come over to check you off.

It might help to break up the program into smaller tasks that you know how to do. For example, choosing a random word from a list, creating a list with blanks in it that is the right length, asking the user to guess a letter in a while True loop, checking if the letter is in the secret word, etc. Even a large program like this is simply made up of a series of small, do-able tasks.

Optional extra credit options:

1) Have the program read a list of words from a text file (a zip file called "Dictionary.zip" is in the class worksheets folder with the entire English dictionary in it) and randomly choose a word from the file. If you want to do this I can help you read the text file in a single command, call me over.

2) After everything else is working, make your program print out ascii art of a hangman or something else fun. Some fun examples are at the end of this document.

Sample run on next page.

Sample run:.

```
We're playing Hangman.
_  _  _  _  _  _  _  _

Guess a letter: A
There's no A.
Your bad guesses so far are: A
You have 8 bad guesses left.


_  _  _  _  _  _  _  _

Guess a letter: B
_  _  _  _  _  B  _  _

Guess a letter: C
C  _  C  _  _  B  _  _

Guess a letter: E
C  _  C  _  _  B  E  _

Guess a letter: E
You already guessed  E

Guess a letter: F
There's no F.
Your bad guesses so far are: A F
You have 7 bad guesses left.

C  _  C  _  _  B  E  _

Guess a letter: S
There's no S.
Your bad guesses so far are: A F S
You have 6 bad guesses left.

C  _  C  _  _  B  E  _

Guess a letter: U
C  U  C  U  _  B  E  _

Guess a letter: M
C  U  C  U  M  B  E  _

Guess a letter: R
C  U  C  U  M  B  E  R

You win!
Again? no
Goodbye.
```

Sample art ideas on next page

Sample art ideas for extra credit when/if you finish the core program: