Project 1

Evil Hangman

Evil Hangman is a popular homework assignment thought up by Professor Keith Schwarz of Stanford University.

"Evil Hangman is an assignment in which students write a computer program that cheats at the classic game *Hangman*. Normally, when programming a computer to play Hangman, the computer picks a single word and accurately represents it as the human player tries to guess all of the letters in the word. In Evil Hangman, the computer instead maintains a list of every word in the English language, then continuously pares down the word list to try to dodge the player's guesses as much as possible. Human players tend to fare terribly against this conniving silicon foe!"

The algorithm that drives Evil Hangman is fairly straight forward. The computer begins by maintaining a list of all words in the English language of a particular length. Whenever the player guesses, the computer partitions the words into "word families" based on the positions of the guessed letters in the words. For example, if the full word list is ECHO, HEAL, BEST, and LAZY and the player guesses the letter 'E', then there would be three word families:

- E---, containing **E**CHO.
- -E--, containing H**E**AL and B**E**ST.
- ----, containing LAZY.

Once the words are partitioned into word families, the computer can pick the largest of these families to use as its remaining word list. It then reveals the letters in the positions indicated by the word family. In this case, the computer would pick the family -E-- and would reveal an E in the second position of the word.

The goal of this assignment is for you to get practice selecting the best python data structure to solve a problem. For example, what is the best data structure to hold onto each word family pattern and the list of words that correspond to it. Is it a queue? Probably not. Would a dictionary with the word family pattern, -E-- as the key and the set of words that correspond to it as the value do the trick? Probably.

Project Set Up

1. Download the starter files, hangman.py and play_hangman.py and set them up in Pycharm or an IDE of your choice.
2. Add dictionary.txt to the directory of the project.

Project Instructions

1. You have been given two files – hangman.py and play_hangman.py. play_hangman.py is the starting point of the game. It asks the user if they would like to play hangman and then calls the play method on the Hangman object.
2. play_hangman.py has been implemented and completed already. You do not modify this file. Please read it and follow along with what it is doing. hangman.py is the file you will complete and submit.
3. hangman.py includes the Hangman class. The game implementation should be done in this class. You will need to implement the play method to complete the assignment. You should create and use additional class methods as well so your code is modular and easy to understand.
4. Submit hangman.py only.

Project Details

1. When the game begins, the user should be prompted for a word length, the number of guesses they would like and whether they'd like to see a list of the remaining words. Displaying the list of remaining words helps with debugging and grading.
2. Once that information is received, underscores _ the length of the word should be displayed along with the number of guesses left.
3. The user should then be prompted to enter a letter.
4. If the letter is not in the word, guesses left is decremented and letters guessed are displayed.
5. If the letter is in the word, guesses left is not decremented, letters guessed are displayed and the underscores are updated.
6. In details 4 and 5, if show remaining words is selected then the remaining words should be shown.
7. If the user guesses a letter they've already guessed, they should be prompted to enter another a letter because they've already guessed it.
8. If the user guesses enough letters for the word, they are congratulated. If not, they are told they lost. In either case, user should be prompted to play again.

Advice, Tips, and Tricks

There is no "right way" to go about writing this program, but some design decisions are much better than others. Here are some general tips and tricks that might be useful:

1. *Letter position matters just as much as letter frequency*. When computing word families, it's not enough to count the number of times a particular letter appears in a word; you also have to consider their positions. For example, "BEER" and "HERE" are in

two different families even though they both have two E's in them. Consequently, representing word families as numbers representing the frequency of the letter in the word will get you into trouble.

2. *Creating word families* - One way to do this would be to scan over the word list and create a new string for each word family to be used as a key in the dictionary. For example if the player guesses ‘O’ and the word list contains [ALLY, COOL, GOOD] there are 2 word families here , - - - - and – O O -. Loop through each word and create a new string or list with a ‘-‘ element if the letter is not an O or O if the letter is an O. This new list or string would be the key and the list of words with that pattern is the value.

Deliverables

Submit hangman.py, and any other source files you created.

Grading

60 points

The program works and a user can play a game of evil hangman.

30 points

All requirements of the program as described in project details are implemented.

User is re-prompted if any incorrect value is entered. The program should not throw exceptions. These should be caught and handled.

User is prompted for number of guesses and the option to show or not show the list of remaining words.

Number of guesses left, letters guessed and blanked out version of the word are displayed.

Remaining words are displayed if user selected that option.

Word is displayed after the user runs out of guesses.

If user wins, the user is congratulated.

10 points

Is your code pythonic? – do variable and method names use Python conventions.

Compilation: If your code does not compile, you will receive a 0 on the project. Once notified, you will have 3 days to fix the project to receive partial credit, up to 75%.

Late Policy

3 points is deducted for each day the homework is late. You do not need to ask for an extension, you can always submit late with the 3 point per day penalty.