Overview

We want to build a program that generates and plays a game of Wheel of Fortune with the user based on information the user provides.

Background

Wheel of Fortune is a popular TV-game show in which contestants try to guess a word or set of words from a given category to win money. Such contestants know the category, but have to guess individual letters they think might lead them to uncovering the word or set of words given. There is a monetary value, however, attached to each guess-- that is, every round is preceded by a spin of the "wheel of fortune" that tells the contestant the amount of money they would win if they guess the correct letters for the word. There are also cases in which the wheel could land on "bankrupt," which puts the contestant's balance at zero. Other rules-- such as a monetary deduction when guessing a vowel or a monetary gain when solving the puzzle with many unknown letters-- also apply.

Implementation

Our primary implementation lies within the GameShow class. First, we outline constant factors in the class, such as the different types of hosts (such as Lizzo, Snoop Dogg, and Steve Harvey) that are randomly chosen and the approved list of words to start the show. Then, we outline the script for the introduction of the show, using print functions, a timer, and a simple text-based I/O system where the user enters information when needed to emulate the dialogue that would occur. We also ensure that the user actually wants to begin the game by having a list of approved affirmative answers, such as “yes,” “yeet,” “yuh,” “of course,” “undoubtedly,” “sure,” “why not,” “maybe,” “oui,” “si,” “hai,” “def,” etc. Then, we randomly choose a topic for the contestant using the `random` function and by loading the script of a text-file containing the different topics and sets of words. While doing so, we randomly choose a certain amount of time the wheel takes to “spin” in order to create the effect that an actual wheel is spinning. After the word is chosen, the game begins: we first outline the script for the beginning of the game by reading from a prompt and then use the simple text-based I/O system to allow the user to input responses. We also randomly choose a monetary value associated with the guess (bankruptcy having a ⅙ probability of occurring). After evaluating the user’s responses, update the score the user has based on their responses and the randomly chosen monetary value associated with the guess. Finally, at the end of a round, the user’s score is stored in their bank. The process outlined above is repeated with a few modifications: the host does not say the introduction script and our implementation checks to make sure that we never pick a puzzle from the same category twice in a given game. At the end of all the rounds (which is, by default 3, but can be changed by modifying the `self.numRounds` variable), the user’s score (the amount of money they received) is compared to that of other highscores, which are kept in a separate text-file. If the user performed well enough, this text-file is revised to contain their name and score.

Implementation

With the downloaded files (`GameShow.py, puzzles.txt, and scoreFile.txt`), run the main python script named `GameShow.py`.

Shareability

We are comfortable with the CS41 staff sharing all the contents of this project.

Credits

Jose Calinawan Francisco and Helena Vasconcelos