Guess The Number Game

Python Bootcamp: Day One

The Game

A simple, classic program to build is a guessing game. A random number is generated. Then, the player makes as many guesses needed in order to guess the number correctly. To help the player make a better choice, it is usually a good idea if the player's guess is higher or lower then the generated number.

The Program

Write a program that will replicate the guessing game. The game will be split into three phases:

1. Introducing the user on how to play the game.
2. Playing the guessing game, allowing the user to play as many matches as they want.
3. Reporting the user's overall performance

Requirements

Code Structure

It is mandatory to have at least the following functions:

1. A function that prints the introduction to the user
2. A game function that plays one match of the guessing game (just one game)
3. A function that calculates and report the user's performance

Note that the three functions correspond to the three phases of the game. You can create additional functions if you find them helpful. Requirements for each individual function will be described further below.

Overall Game Environment

You are required to define a variable that will represent the maximum value that can be generated. In the example log found on Figure 1, the maximum value that can be generated is 100. Make sure that the value for the maximum value is an integer (whole) number. You can choose how big the range the player has to guess between.

You are required to keep track of the total number of games the user has played, and the total number of guesses that the user has made. You are also required to record the user's best performance. The best performance of a user is defined as *the game match with the least number of guesses*. You are to record which game match value the user performed the best, and the number of guesses they used. These values will be used in the third function, where they will be reported to the user.

function One: Introduction

The following text will be required to be printed before every game session. Note that if {max} appears below, substitute that text with the maximum value you have defined:

This program allows you to play the guessing game. It will guess a number between 1 and {max}*. You can guess as many times as needed in order to guess the correct number. The game will let you know if the right answer is higher or lower than your guess. If you want to continue playing again, type y/yes when asked.*

There should be a newline that separates the introduction from the game matches. Refer to Figure 1 to see what the actual example output looks like. Note that {max} = 100 for the example log.

function Two: A Single Game Match

In every game match, generate a random number between the range of 1 and your maximum value defined. Tell the user that the number is generated for that match. Prompt the user for a value. Save the input string, and typecast that input as an int. Check if the input is equal to the randomly generate value. If not, tell the user if the actual value is higher or lower than the value they guessed. Loop until the user gets the value correctly.

At the end of every match, prompt the user if they want to play another match. If the user types *y or yes* (ignoring case), play another match. Separate each match with a new line. Refer to Figure 1 to see what multiple matches may look like.

Don't forget the requirments mentioned in the section **Overall Game Environment**.

function Three: Performance Results

Print the following: the total game matches the user played, the total guesses the user has made overall, how many guesses per game, and the game match session with the least amount of guesses (best performance). Refer to Figure 1 to see how the performance text is printed out. Note that each individual metric is spaced from the left margins with a tab character.

Other

- When printing the number of guesses it took for the player to guess correctly for an individual match, handle the case where the user guesses it correctly within the first guess. Refer to Figure 2 to see an example.
- When printing which game has the best performance, handle the case where the best game took only 1 guess. Refer to Figure 2 to see an example.
- You are required to use a list for this project.
- Use the built-in random library to generate integer values.

Hints

- You may find it useful to learn string formatting to make it easier to do positional formatting. Here some resources about them:
  - `https://docs.python.org/3/tutorial/inputoutput.html`
  - `https://docs.python.org/3/library/stdtypes.html#old-string-formatting` (for printf-style formatting)
- Use parameter passing and return values to help keep track of values
- Write comments and pseudocode!

Figure 1: Example Log: Normal Game

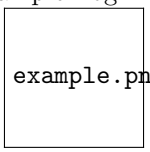

Figure 2: Example Log 2: Modified Game