**Programming Guide: Word-Scramble Game**

**Goal**

This guide will help you design and program a **Word-Scramble Game**. In this game, the programmer will pick a random word, scramble its letters, and challenges the player (their friend in the class) to unscramble it by typing the correct word. The game will check if the player’s guess matches the original word and provide feedback. Players can continue playing round after round until they decide to stop.

This project is meant to reinforce your understanding of Python concepts like lists, loops, if-else statements, functions, and working with the random library.

**Step 1: Import the random Library**

💡 *Why?* We’ll use the random library to shuffle letters in the word and make the game more interesting.

* Syntax: Start your program by importing random. This allows you to use its functions.

**Step 2: Create a List of Words**

💡 *Why?* The game needs a set of words for the player to unscramble.

* Syntax: Use square brackets [ ] to make a list. Separate each word with a comma and surround each word with quotation marks.
* Example: Think of words related to technology or Python programming.

**Step 3: Pick a Random Word from the List**

💡 *Why?* The game should randomly choose a word to scramble for each round.

* Syntax: Use a function from the random library to choose a single word from your list.

**Step 4: Scramble the Letters of the Word**

💡 *Why?* To create a challenge, the letters of the word need to be shuffled.

* Syntax:
  1. Convert the word (a string) into a list of letters using the list() function.
  2. Use random.shuffle() to mix up the order of the letters in the list.
  3. Join the shuffled letters back into a string to create the scrambled word.
* Remember: random.shuffle() works **in place**, so the list will change directly. That is, you don’t need to make a new variable for your shuffled list.

**Step 5: Show the Scrambled Word to the Player**

💡 *Why?* The player needs to see the scrambled version of the word.

* Syntax: Use a function to display a message with the scrambled word. Combine text and the variable holding the scrambled word.

**Step 6: Take the Player’s Guess**

💡 *Why?* The player needs to type their guess into the program.

* Syntax: Use a function that waits for the player to type their answer. Store the answer in a variable so you can check it later. (what function do we use to input things into the computer?)

**Step 7: Check the Player’s Guess**

💡 *Why?* To compare the player’s input with the original word.

* Syntax:
  1. Use an if statement to compare the player’s guess (variable) with the original word.
  2. If they match, print a message saying the player guessed correctly.
  3. Otherwise, use an else statement to print the correct word.

**Step 8: Repeat the Game**

💡 *Why?* Let the player keep playing without restarting the program.

* Syntax:
  1. Use a while loop to repeat the game.
  2. At the end of each round, ask the player if they want to play again.
  3. Use their input to decide whether the loop should stop or continue.

**Key Syntax and Patterns to Remember**

1. **Lists**:
   * Use brackets [ ] to create a list of words.
   * Use .append() to add new items to a list if needed.
2. **Random Functions**:
   * random.shuffle() shuffles a list of items (e.g., letters in a word).
   * random.choice() picks one item from a list.
3. **String Manipulation**:
   * Use a function to convert a word into a list of letters.
   * Use another function to join letters back into a word after shuffling.
4. **Input and Output**:
   * Use a function to take input from the player and another to display messages.
5. **If Statements**:
   * Use if, else, and elif to decide what happens based on the player’s input.
6. **Loops**:
   * Use a while loop to keep the game running as long as the player wants.

**Bonus Challenges**

1. Add a **score system**: Keep track of how many words the player guesses correctly.
2. Create a **timer**: Give players a set amount of time to guess the word.
3. Allow **difficulty levels**: Use longer words or fewer hints for harder rounds.

This guide lays out everything your students need to build the game while giving them room to write the code themselves and apply their knowledge. Let me know if you want to adjust anything further! 😊

```

import random

word\_list = ["excited", "friday","pilgrim", "humongous", "story", "singer", "sports", "class", "peeling", "superstition", "jungle", "canal", "speaker","forcefully", "royal", "begin", "bawling"]

# double-check your spelling:)

# choose a random word

word = random.choice(word\_list)

print(word)

# this makes the letters into a list.

word = list(word)

print(word)

# shuffle the letters in our word list

random.shuffle(word)

print(word)

# put them back into a word

word = ''.join(word)

print(word)

```