**Guessing Game Version Requirements**

**Extended Version of the Guessing Game**

**In this version, you will:**

- **Add a score system that tracks the number of guesses.**
- **Introduce a limited number of attempts to increase the difficulty.**
- **Use nested loops to allow the player to play multiple rounds.**
- **Add a function to encapsulate game logic to improve code organization and readability.**

**Instructions to Create the Guessing Game Program:**

1. **Import the Random Module:**
    - Begin by importing the random module, which will be used to generate random numbers for the guessing game.

import random

**Define the Function play_single_round:**

- Create a function named play_single_round() that contains the logic for a single round of guessing.
- Inside this function, initialize the following variables:
    - random_number: Use random.randint(1, 100) to generate a random number between 1 and 100.
    - attempts: Set this to 0 to track the number of guesses made by the player.
    - max_attempts: Set this to 10, as the player will have up to 10 guesses.
    - guess: Initialize this to None to store the player's guesses.
- Print a message informing the player that they have 10 attempts to guess the number.

**Create a Loop for Guessing:**

- Use a while loop that runs until the player either guesses the correct number or uses all their attempts (i.e., attempts < max_attempts).
- Inside the loop:
    - Use a try-except block to handle the player's input:
        - Ask the player to enter their guess with input().
        - Convert the input to an integer (int()).
        - Increment the attempts counter by 1 after each guess.

    - Use if, elif, and else statements to check whether the guess is too low, too high, or correct:
        - If the guess is lower than the random number, print "Too low!".
        - If the guess is higher than the random number, print "Too high!".
        - If the guess is correct, print a success message and return the number of attempts it took to guess the number.
    - If a non-numeric input is entered, catch it with the except ValueError and print an error message to prompt the user for a valid number.

**Provide Feedback on Attempts:**

- After each incorrect guess, print the remaining number of attempts using max_attempts - attempts.
- If the player uses all 10 attempts without guessing correctly, print a message that the game is over and reveal the correct number.
- Return max_attempts as the score if the player fails to guess the number within 10 attempts.

**Define the play_game Function to Manage Multiple Rounds:**

- Create a new function play_game() that handles multiple rounds of the game.
- Set the total number of rounds (total_rounds = 3) and a variable to track the total score (total_score = 0).
- Print a welcome message to the player.

**Loop Through Multiple Rounds:**

- Use a for loop to iterate through each round. For each round:
    - Display the current round number.
    - Call the play_single_round() function to play one round and store the result (score).
    - Add the score of the current round to the total score.

**Display the Final Score:**

- After completing all the rounds, print a message stating that the game is over.
- Display the player's total score (sum of all attempts across rounds).
- Provide feedback based on the player's performance:
    - If the total score is less than or equal to 7 attempts per round, print "Excellent guessing skills!".
    - If the score is between 7 and 9 attempts per round, print "Good job!".
    - Otherwise, print "Better luck next time!".

**Create a Main Function to Start the Game:**

- At the end of the program, add a main guard using if __name__ == "__main__": to call the play_game() function and start the game when the script is run.