Piedra, Papel o Tijera - Desafío de Programación

El clásico juego de la infancia, donde dos jugadores eligen entre tres elementos y la victoria se determina según las siguientes reglas:

- **Piedra** aplasta **Tijera** → 🏆 **Gana la Piedra**
- **Tijera** corta **Papel** → 🏆 **Gana la Tijera**
- **Papel** envuelve **Piedra** → 🏆 **Gana el Papel**
- Si ambos jugadores eligen el mismo elemento, la ronda termina en **empate**.

📌 Reglas del Juego

- La partida se juega al **mejor de 3 rondas**.
- Si un jugador (humano o máquina) logra **dos aciertos seguidos**, la partida finaliza automáticamente.
- En caso de **empate en las 3 rondas**, el juego continuará hasta que haya un ganador.

📌 Funciones a desarrollar

1️⃣ `verificar_ganador_ronda(jugador, maquina) → str`

📌 **Objetivo:** Determina quién ganó la ronda según las elecciones del jugador y la máquina.

🔹 **Parámetros:**

- `jugador` (int): Elección del jugador (1 para Piedra, 2 para Papel, 3 para Tijera).
- `maquina` (int): Elección de la máquina (1 para Piedra, 2 para Papel, 3 para Tijera).

🔹 **Retorno:**

- `"Jugador"` → Si el jugador gana la ronda.
- `"Máquina"` → Si la máquina gana la ronda.
- `"Empate"` → Si ambos eligen el mismo elemento.

2️⃣ `verificar_estado_partida(aciertos_jugador, aciertos_maquina, ronda_actual) → bool`

📌 **Objetivo:** Determina si la partida sigue en curso o ya ha finalizado.

🔹 **Parámetros:**

- `aciertos_jugador` (int): Cantidad de rondas ganadas por el jugador.
- `aciertos_maquina` (int): Cantidad de rondas ganadas por la máquina.
- `ronda_actual` (int): Número de la ronda actual.

🔹 **Retorno:**

- `True` → Si la partida sigue en curso.
- `False` → Si la partida ha finalizado (porque alguien ganó dos veces seguidas o se jugaron todas las rondas).

3️⃣ `verificar_ganador_partida(aciertos_jugador, aciertos_maquina) → str`

📌 **Objetivo:** Determina quién ganó la partida comparando los aciertos finales.

🔹 **Parámetros:**

- `aciertos_jugador` (int): Cantidad de rondas ganadas por el jugador.
- `aciertos_maquina` (int): Cantidad de rondas ganadas por la máquina.

🔹 **Retorno:**

- `"Jugador"` → Si el jugador tiene más aciertos.
- `"Máquina"` → Si la máquina tiene más aciertos.

4️⃣ `mostrar_elemento(eleccion) → str`

📌 **Objetivo:** Convierte la elección numérica en un texto legible.

🔹 **Parámetros:**

- `eleccion` (int): Número de elección (1 para Piedra, 2 para Papel, 3 para Tijera).

🔹 **Retorno:**

- `"Piedra"` cuando su `elección == 1`.
- `"Papel"` cuando su `elección == 2`.
- `"Tijera"` cuando su `elección == 3`.

5️⃣ `jugar_piedra_papel_tijera() → str`

📌 **Objetivo:** Gestiona toda la lógica del juego, usando las funciones anteriores.

🔹 **Flujo de la función:**

1. Inicia una partida donde el jugador compite contra la máquina.
2. En cada ronda, el jugador elige una opción y la máquina genera una elección aleatoria.
3. Se determina el ganador de la ronda.
4. Se verifica si la partida continúa o si alguien ha ganado.
5. Al finalizar, la función devuelve quién ganó la partida (`"Jugador"` o `"Máquina"`).

📌 Requisitos del Código

✅ Todas las funciones deben estar correctamente modularizadas.
✅ Se debe validar que el jugador solo ingrese valores válidos (1, 2 o 3).
✅ Se deben manejar posibles errores de entrada de datos.
✅ Se recomienda usar `random.randint(1,3)` para la elección de la máquina.
✅ Mostrar mensajes claros en cada ronda indicando los elementos elegidos y el estado de la partida.
✅ Crear tantos módulos como considere necesario y reutilizar los propios.