El Ahorcado Multijugador

Objetivo

Escribir un programa en Python, que ofrezca a un grupo de N personas, jugar al ahorcado, respetando las reglas y condiciones que se dan a continuación.

Diccionario de Palabras

Al iniciar el programa y antes de comenzar a jugar, se debe generar un diccionario con las posibles palabras a adivinar. Para ello, deberá utilizar una función que será provista, y que le devolverá un texto. Dicho texto, debe ser procesado, de forma tal de obtener sólo palabras válidas, formadas sólo por letras; y no deben estar repetidas. Asociado a cada palabra, se debe guardar la cantidad de veces que aparece dicha palabra en el texto.

Una vez generado el diccionario, se debe mostrar por pantalla el resultado, ordenado alfabéticamente; e informar la cantidad total de palabras que hay en el diccionario.

Desarrollo del Juego

Para iniciar el juego, el programa debe preguntar la cantidad de personas que jugarán, que no deben ser más de 10.

A continuación solicitará el ingreso de los nombres de cada una de las personas.

Luego debe otorgar aleatoriamente, el orden en que jugarán los participantes, e informar el mismo.

A continuación, el programa debe solicitar la longitud de la palabra a adivinar, que no podrá ser menor a 5 caracteres. Si no hay palabras con la longitud ingresada, solicitar un nuevo número.

Cada participante tendrá que adivinar una palabra que debe ser otorgada aleatoriamente por el programa, entre todas las posibles que cumplan con la longitud ingresada.

Cada jugador, tendrá un máximo de 7 desaciertos por palabra.

En cada turno, el jugador deberá ingresar una letra. Si la letra se encuentra en la palabra, se mostrará en las posiciones donde se encuentra, y el jugador sumará 1 punto por haber acertado, y se le solicitará el ingreso de una nueva letra. Si no acierta, se le restan 2 puntos, y el turno pasa al siguiente jugador.

El primer jugador que acierte la palabra en menos de 8 intentos, es quien gana la partida; y suma 30 puntos por haber adivinado la palabra.

Si ninguno la adivina, gana el programa.

Cuando un jugador llega al máximo de desaciertos, la partida continúa con el resto de los jugadores.

En cada Turno, se debe mostrar el nombre de quien está participando, cuales son sus aciertos y desaciertos hasta el momento, y el puntaje que tiene hasta el momento.

Al finalizar cada partida, mostrar el resultado de la partida, indicando para cada jugador la palabra que debía adivinar, cantidad de aciertos y desaciertos, y el puntaje obtenido; y luego preguntar si desean jugar otra partida.

Si se juega más de una partida, acumular los resultados de las partidas y mostrar los Resultados Generales, indicando la cantidad de partidas jugadas; mostrando por jugador, los datos ordenados por Puntaje Total, junto con la cantidad de aciertos y desaciertos.

En cada nueva partida, volver a solicitar la longitud de la palabra a adivinar.

A partir de la segunda partida, el orden de los jugadores, deberá generarse nuevamente, pero esta vez, deberá tener en cuenta quien ganó la partida anterior, y colocarlo en primer lugar; para el resto de los participantes, se tendrá en cuenta el puntaje acumulado, colocando primero a los de mayor puntaje, y a igual puntaje, decidir aleatoriamente entre estos.

Validación de Datos

El ingreso de los datos debe ser validado respetando según corresponda, el tipo de dato, los rangos, o el formato que debe respetar el texto ingresado. Los nombre de los jugadores sólo podrán contener letras y espacios en blancos. Se recomiendo todo texto, convertirlo a mayúsculas.

Notas

El programa debe respetar las reglas de la programación estructurada, evitando especialmente, la redundancia de código.

Cada grupo deberá subir al campus:

1. Documento que gráficamente muestre la solución modular implementada.
   Descripción de las estructuras de datos que utilizarán, su función y su interacción. Puede ser mediante un gráfico.
   Se recomienda aplicar las técnicas mencionadas en clase, respetando la etapa de diseño, llegando a una solución modular adecuada, antes de comenzar a codificar.
   Se debe respetar el estilo de programación dado en clase, tomando como referencia el PEP 8.
2. El programa fuente de la aplicación. Cada función debe tener su correspondiente comentario compuesto por Autor: indicando el nombre del alumno que lo codificó, y una breve descripción de lo que realiza.