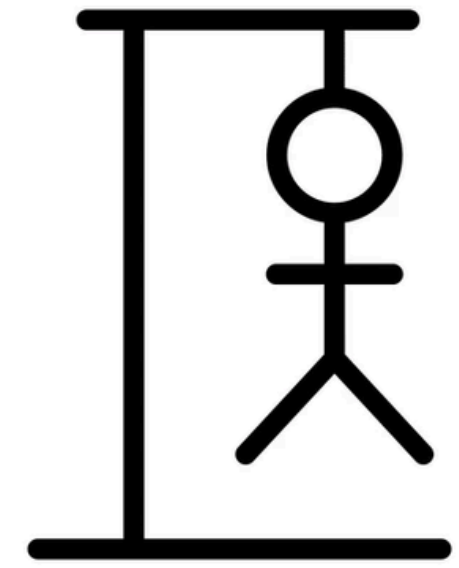




Adivina la



P A L A B R A

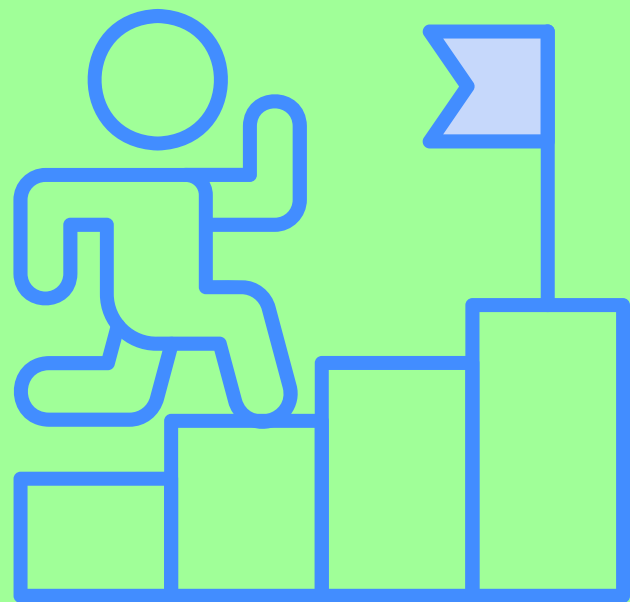
Juego Del Ahorcado



El ahorcado es un juego de adivinanza en el que se pone a prueba conocimientos de un jugador en base el vocabulario, ortografía, memoria y concentración para conseguir una victoria o una derrota, descubriendo la palabra secreta que sea seleccionada.

El objetivo del ahorcado es descubrir una palabra secreta antes de quedarse sin intentos.

- **Un jugador elige una palabra secreta y la representa con guines bajos para cada letra segun el tamaño de la palabra.**
- **El otro jugador debe adivinar las letras de la palabra, una por una.**
- **Si la letra está en la palabra, se revela en su posición correcta.**
- **Si la letra no está en la palabra, se añade una parte del dibujo del ahorcado.**
- **El jugador gana si adivina la palabra antes de que se complete el dibujo.**
- **Pierde si se forma el dibujo completo sin haber descubierto la palabra.**



Importación del modulo random

Este módulo proporciona funciones para generar números aleatorios y seleccionar elementos al azar. En este caso, se usará `random.choice()` para elegir una palabra aleatoria de la lista.

```
import random
```

Lista de palabras

Aquí se define una lista de palabras que serán utilizadas en el juego. Se han elegido palabras educativas y llamativas.

```
# Lista de palabras almacenadas el juego
palabras = ["descubrir", "explorar", "aventura", "misterio", "descubrimiento", "progreso"]
```

Selección de palabra secreta

- `random.choice(palabras)`: Escoge una palabra aleatoria de la lista.
- `len(palabra_secreta)`: Calcula la cantidad de letras de la palabra seleccionada.

```
# Seleccionamos una palabra al azar de la lista
palabra_secreta = random.choice(palabras)
longitud = len(palabra_secreta)
```





¿Estás listo?



Inicialización del juego

- palabra_mostrada: Es una lista que almacena los guiones bajos _ para ocultar la palabra secreta.
- letras_adivinadas: Contiene las letras que el usuario ha ingresado.
- errores: Contador de intentos fallidos.
- max_errores: Número máximo de intentos antes de perder el juego.

```
# inicio del juego
palabra_mostrada = ["_"] * longitud # Representación oculta de la palabra
letras_adivinadas = "" # Almacena las letras que el usuario ya intentó
errores = 0 # Contador de errores
max_errores = 6 # Número máximo de intentos antes de perder
```

Bucle principal del juego

- Se ejecuta mientras el jugador tenga intentos disponibles (errores < max_errores).
- También se detendrá si el jugador adivina la palabra completa ("_" in palabra_mostrada evalúa si aún quedan letras sin adivinar).

```
# Bucle principal del juego, se ejecuta hasta que se acaben los intentos o se adivine la palabra
while errores < max_errores and "_" in palabra_mostrada:
    print("Palabra: ", " ".join(palabra_mostrada)) # Muestra el estado actual de la palabra
    letra = input("Adivina una letra: ").lower() # Solicita una letra y la convierte a minúscula
```

Mostrar la palabra oculta y pedir ingreso de letras

- `print("Palabra: ", " ".join(palabra_mostrada)):` Muestra la palabra en progreso con espacios entre las letras.
- `input("Adivina una letra: ").lower():` Pide una letra al usuario y la convierte a minúscula.

```
print("Palabra: ", " ".join(palabra_mostrada)) # Muestra el estado actual de la palabra
letra = input("Adivina una letra: ").lower() # Solicita una letra y la convierte a minúscula
```

Verificar si la letra ya ha sido ingresada

Si el usuario ya ingresó la letra anteriormente, se muestra un mensaje y se usa `continue` para saltar el resto del código en la iteración actual.

```
if letra in letras_adivinadas: # Verifica si la letra ya fue ingresada
    print("Ya intentaste con esa letra.")
    continue # Salta al siguiente ciclo sin procesar la letra nuevamente
```


Registrar la letra ingresada

Se añade la letra ingresada a la cadena `letras_adivinadas` para evitar que el usuario la ingrese de nuevo.

```
letras_adivinadas += letra # Agrega la letra a la lista de intentos
```

Verificar si la letra está en la palabra secreta

Si la letra está en la palabra:

- Se recorre la palabra con un `for` para encontrar todas las coincidencias.
- Se reemplazan los guiones bajos `_` en `palabra_mostrada` por la letra correcta.
- Se muestra un mensaje de éxito.

```
if letra in palabra_secreta: # Si la letra está en la palabra secreta
    for i in range(longitud): # Recorre la palabra para revelar las coincidencias
        if palabra_secreta[i] == letra:
            palabra_mostrada[i] = letra
    print("¡Bien hecho!")
```

Actualizar el contador de errores

Si la letra no está en la palabra secreta:

- Se incrementa errores en 1.
- Se muestra un mensaje con los intentos restantes.

```
else: # Si la letra no está en la palabra secreta
    errores += 1 # Aumenta el contador de errores
    print(f"Incorrecto. Intentos restantes: {max_errores - errores}")
```

Comprobar el resultado del juego

- Si no quedan _, significa que el jugador adivinó la palabra y gana el juego.
- Si se alcanzan los intentos máximos (max_errores), el jugador pierde y se muestra la palabra correcta.

```
# Verifica si el jugador ganó o perdió
if "_" not in palabra_mostrada:
    print("¡Felicidades! Adivinaste la palabra:", "".join(palabra_mostrada)) # Mensaje de victoria
else:
    print("¡Has perdido! La palabra era:", palabra_secreta) # Mensaje de intentos agotados y muestra la palabra secreta
```

Felicidades



¡Gracias por jugar!