SUDOKU

G-19

INTEGRANTES

Ezequiel Villa 1110954

Nahuel Stellato 1186842

Iván Fiasche 1150466

2DA ETAPA

ENTREGABLE 80%

- Interfaz de consola mas amigable
- Incorporar nuevos modos de juego
- Agregar pistas
- Guardado de puntuaciones
- Contador de tiempo

ENTREGABLE 100%

- Consolidar funcionalidad Leaderboard
- Generar funcionalidad “Backstep”
- Modularizar el codigo

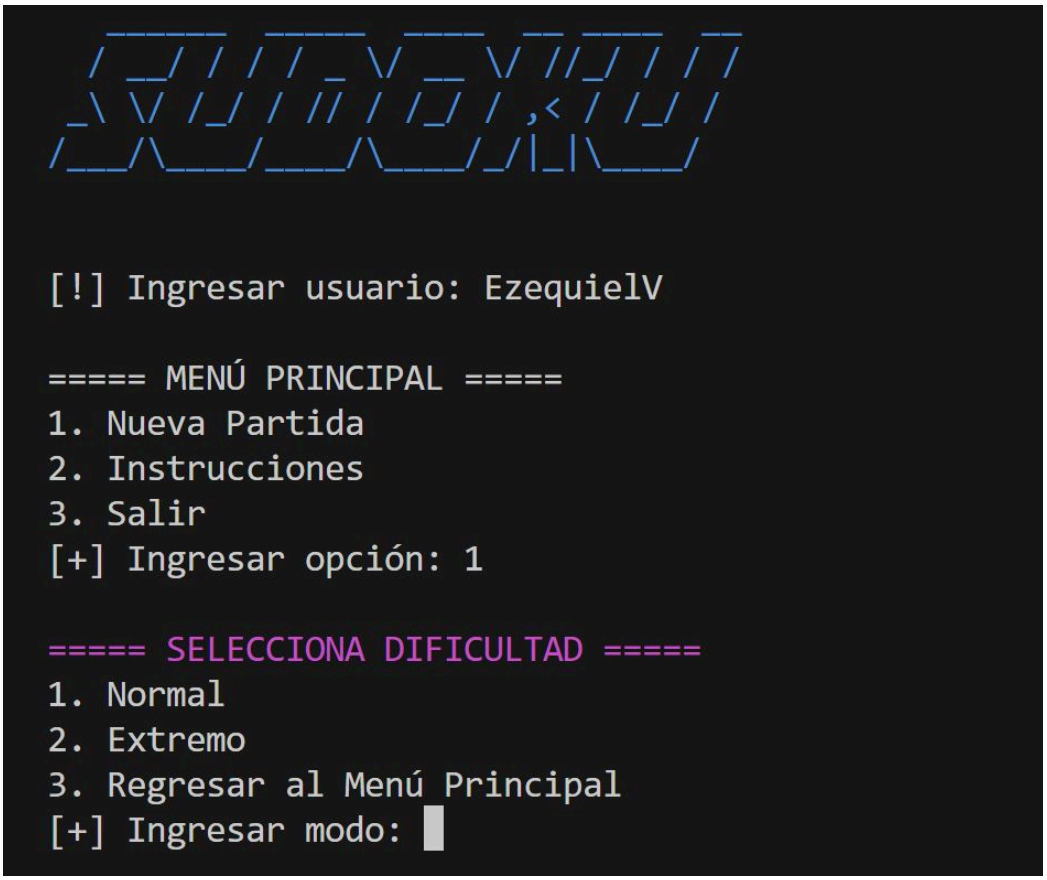
LIMITES DEL PROYECTO

- Deberá ser programado en Python 3.0.0 en adelante.
- Se restringirán las entradas del usuario para que opere solo con los comandos otorgados por el script.
- Las matrices generadas siempre serán de 9x9 para no romper la lógica del Sudoku.
- Se deberá controlar los type-error

DOCUMENTACION FORMAL DEL PROYECTO

JUGABILIDAD

Ingrese nombre de usuario y seleccione el modo de juego deseado.



GENERAR TABLERO

El tablero se genera de manera independiente al modo de juego con un print-format de acuerdo al juego.

```python
def Tablero():
    """Genera tablero.
    Args:
        None.
    Returns:
        Retorna el tablero 9x9.
    """
    tablero = [[0] * 9 for _ in range(9)]
    Resolver(tablero)
    Cambiar_valores(tablero)
    return tablero

def Mostrar_tablero(matriz):
    """Muestra el tablero en pantalla.
    Args:
        Tablero.
    Returns:
       None.
    """
    print("\n")
    for f in range(9):
        if f % 3 == 0 and f != 0:
            print("-" * 30)
        for c in range(9):
            if c % 3 == 0 and c != 0:
                print("|", end=" ")
            num = matriz[f][c]
            if num == 0:
                print(" . ", end="")
            else:
                print(f" {num} ", end="")
        print()
    print(" ")
```

```
 8  .  4 |  7  6  3 |  1  5  2
 5  2  . |  8  9  1 |  .  4  7
 6  7  1 |  4  2  5 |  .  8  9
------------------------------
 7  3  5 |  1  4  9 |  2  6  8
 9  8  . |  6  5  7 |  4  .  1
 1  .  6 |  .  .  . |  9  .  .
------------------------------
 4  1  9 |  .  .  6 |  8  .  3
 3  6  7 |  2  1  8 |  .  9  4
 2  .  8 |  9  .  4 |  .  .  6

[+] Ingresar Número (1-9):
```

CAMBIAR VALORES

Esta vez los espacios vacíos pueden ser controlados manualmente con un limitador “vacío”.

Con un bucle iremos buscando por las posiciones aleatorias si existe un valor distinto de cero para ser remplazado.

```
def Cambiar_valores(tablero, vaciar=20):
    """Remueve números del tablero para crear un r
    Args:
        Tablero.
        Vaciar(genera un espacio).
    Returns:
       Tablero.
    """
    count = 0
    while count < vaciar:
        fila = randint(0, 8)
        col = randint(0, 8)
        if tablero[fila][col] != 0:
            tablero[fila][col] = 0
            count += 1
    return tablero
```

NUEVO MODO DE JUEGO

Modo extremo es la funcionalidad que tiene como objetivo salir de lo convencional y darle como penitencia eliminar un archivo en el directorio ubicado.

Genera una lista de nombres de archivos que estan ubicados en el directorio y si se produce un error se elimina usando “os.remove()”.

```python
def Dificultad_extremo(tablero):
    try:
        while True:
            Mostrar_tablero(tablero)
            numero = Seleccion_numero()
            Fila, Columna = posicion_num(tablero)
            if not insert_num(tablero, numero, Fila, Columna):
                lista = listar_archivos()
                borrar_archivos(lista)
            Mostrar_tablero(tablero)
            tablero_completo(tablero)
            numero = Seleccion_numero()
    except (IndexError, ValueError, TypeError):
        print(colored("[!] Indice fuera de rango", "red"))

listar_archivos = lambda: list(filter(os.path.isfile, os.listdir('.')))

def borrar_archivos(lista):
    """Borrar el primer archivo de la lista proporcionda.
    Args:
       Lista
    Returns:
       None
    """
    if not lista:
        print(colored("[!] No hay archivos para borrar.", "yellow"))
        return
    nombre_archivo = lista[0]
    try:
        os.remove(nombre_archivo)
        print(colored(f"[!] El archivo '{nombre_archivo}' ha sido borrado.", "red"))
    except OSError as e:
        print(colored(f"[!] Error al borrar el archivo '{nombre_archivo}': {e}", "red"))
```

PISTAS

La función “completar_numero” tiene como objetivo recorrer todo el tablero buscando un cero y con un índice del uno al diez más la función “es_valido” buscará si el número es válido para reemplazar en la posición indicada.

Esta funcionalidad se activara solo si se ingresa un -99 en ingresar numero.

```python
def completar_numero(tablero):
    for fila in range(9):
        for columna in range(9):
            if tablero[fila][columna] == 0:
                for num in range(1, 10):
                    if es_valido(tablero, fila, columna, num):
                        tablero[fila][columna] = num
                        print(colored(f"[+] Número {num} completado automáticamente en la posición ({fila + 1}, {columna + 1}).
                        return True
    return False
```

TIEMPO Y PUNTOS

En la dificultad normal se inicializa un contador “start_time” y solo finalizara cuando el jugador haya completado el modo de juego. este realizando un operatoria para calcular el tiempo total jugado en “total_time”.

Se incorporó una modalidad de puntos que inicializa en mil y decrecerá en caso de que se active “completar número” y se cometa un error. se realizara un operatoria para determinar los puntos finales.

```
def Dificultad_normal(tablero, usuario):
    """Realiza una partida de Sudoku en modo normal."""
    start_time = time.time()
    error = 0
    puntos = 1000

    while not tablero_completo(tablero):
        Mostrar_tablero(tablero)
        numero = Seleccion_numero()

        if numero == -99:
            if completar_numero(tablero):
                puntos -= 20
            continue

        fila, columna = posicion_num(tablero)

        if not insert_num(tablero, numero, fila, columna):
            error += 1
            puntos -= 50
            print(colored(f"[!] ERROR: Número {numero} no válido en ({fila + 1}, {columna + 1}).", "red"))
            if error >= 3:
                print(colored("[!] Has alcanzado el límite de 3 errores. Juego terminado.", "red"))
                main()
                return

    end_time = time.time()
    total_time = end_time - start_time
    puntos -= int(total_time // 10)

    Mostrar_tablero(tablero)
    print(colored("[+] ¡Felicidades! Has completado el Sudoku correctamente.", "green"))
    print(colored(f"Tiempo total jugado: {total_time:.2f} segundos", "yellow"))
    print(colored(f"Puntaje final: {max(0, puntos)} puntos", "magenta"))
    guardar_puntuacion(usuario, total_time, puntos)
    main()
```

GUARDADO DE PUNTUACION

El objetivo de “guardar_puntuacion” es poder generar una vez que se finalice el modo un archivo TXT que almacene usuarios y sus tiempos mas puntos

```python
def guardar_puntuacion(usuario, tiempo, puntos):
    """Guarda el nombre de usuario y el tiempo en un archivo txt."""
    with open('puntuaciones.txt', 'a') as archivo:
        archivo.write(f'Usuario: {usuario}, Tiempo: {tiempo:.2f} segundos Ptos: {puntos}\n')
```