
# PROYECTO INTEGRADOR DEL AULA

FLOR ALBA TOVAR PEREA

JULIO MARIO RAMOS LÓPEZ

GERALDIN PAOLA PALACIO MÁRQUEZ

FACULTAD DE INGENIERÍA

2024

# INTRODUCCIÓN

En el mundo de la programación, recrear juegos clásicos en formato digital resulta no sólo fascinante, sino también educativo. En este proyecto, creamos un juego de mesa inspirado en la dinámica de escaleras y serpientes, con desafiantes adivinanzas. Utilizando el lenguaje de programación de Python, seguimos un proceso meticuloso y estructurado en diversas etapas ejecutadas con precisión. Esto incluye la preparación de elementos esenciales como el diseño y creación del tablero, la definición de las escaleras y serpientes, definiendo la temática del juego, así como la compilación de un banco de adivinanzas y diccionarios de palabras para desafiar a los jugadores. Además, determinamos niveles de dificultad y desarrollamos el sistema de puntajes, asegurando así una experiencia de juego fluida y cautivadora.

# OBJETIVO GENERAL Y ESPECÍFICOS

Desarrollar un juego de mesa en consola inspirado en la dinámica de escaleras y serpientes, enriquecido con desafiantes adivinanzas, utilizando el lenguaje de programación Python. Este proyecto tiene como propósito brindar a los usuarios una experiencia de entretenimiento y aprendizaje, teniendo en cuenta los siguientes objetivos, para asegurar su excelencia y funcionamiento óptimo:

- Diseñar una experiencia de usuario inmersiva y atractiva que asegure la participación y el disfrute del juego de mesa de escaleras y serpientes, complementado con enigmas y desafíos interactivos.

- Desarrollar la funcionalidad completa del juego de mesa en consola, asegurando su jugabilidad óptima mediante la implementación eficiente de algoritmos en Python.

- Implementar pruebas exhaustivas y validaciones rigurosas para garantizar el correcto funcionamiento de todas las características del juego, así como para identificar y corregir posibles errores o fallos de rendimiento.

# MARCO TEÓRICO

[2] La creación de juegos es un proceso complejo y multifacético que va más allá de la simple generación de ideas y programación. Requiere atención meticulosa y un enfoque estratégico para superar los desafíos inherentes y ofrecer una experiencia de juego satisfactoria. En primer lugar, la definición de una temática clara y visualmente atractiva es esencial para captar la atención del jugador desde el principio. Esta temática establece el tono del juego y sirve como guía para el diseño de niveles, la narrativa y la mecánica de juego, asegurando la coherencia en todas las facetas del juego. Además, el diseño de niveles es crucial para la progresión del jugador y su experiencia general. Los niveles deben ofrecer un equilibrio entre desafío y recompensa, estimulando la habilidad del jugador y proporcionando un sentido de logro a medida que avanza. En esta línea, la implementación de algoritmos eficientes garantiza un rendimiento óptimo del juego en términos de velocidad y respuesta, optimizando el código y gestionando recursos como la memoria y la potencia de procesamiento. Sin embargo, las pruebas y validaciones son etapas cruciales para asegurar que el juego funcione correctamente en todas las situaciones y plataformas previstas, identificando y solucionando posibles errores o problemas. También es importante considerar el lenguaje de programación más viable. [1] En este caso, Python se destaca por su capacidad de prototipado rápido, automatización de tareas repetitivas, implementación de algoritmos complejos, optimización de rendimiento y pruebas automatizadas. Todo esto permite un desarrollo ágil, eficiente y flexible en todas las etapas del proceso de creación de juegos.

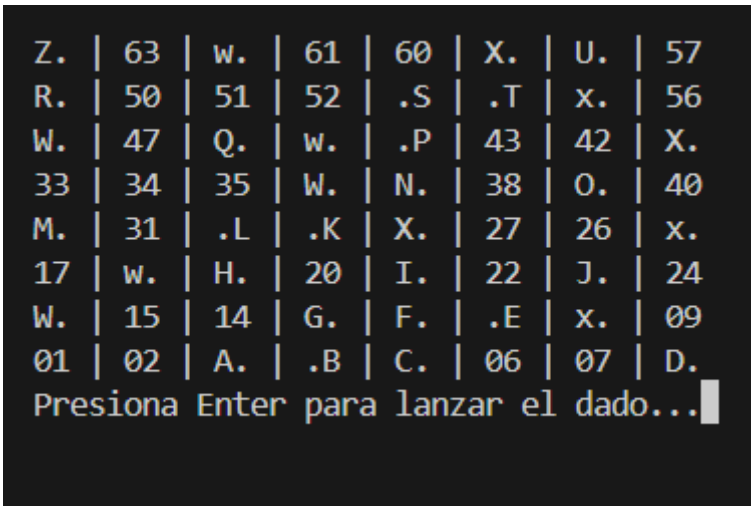
# DESARROLLO DEL PROYECTO

A continuación, vamos a presentar el desarrollo de la estrategia ADIP del proyecto:

**ANÁLISIS:**

- **Descripción:** El proyecto implica desarrollar un juego de mesa utilizando el lenguaje de programación Python, que simula un tablero compuesto por 64 casillas. Estas acciones pueden involucrar desafíos, escaleras o serpientes, añadiendo complejidad y emoción al juego. El propósito principal radica en fomentar el dominio del léxico y la ortografía a través de la interacción y la resolución de situaciones diversas dentro del juego.

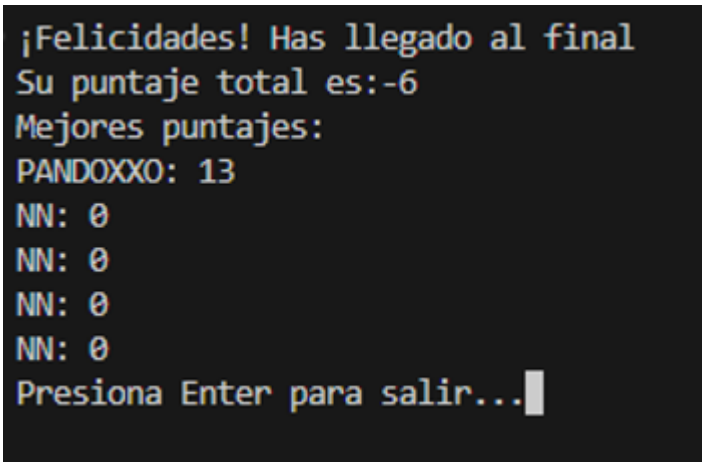
- **Condición inicial:** Al comenzar el juego, la pantalla muestra el tablero sin que el usuario haya lanzado el dado aún. Además, el historial de puntuaciones está vacío o contiene registros de partidas anteriores.




**Estado inicial**

*Representación visual del estado inicial del juego.*

- **Condición final:** Una vez que el usuario llega a la casilla 64 o ha usado todos los intentos permitidos, el juego llega a su fin natural. En este punto, se evalúa el puntaje final obtenido por el jugador. Para mantener un registro de los mejores desempeños en el juego, se compara este puntaje con los registros históricos previamente almacenados. Este proceso permite determinar si el puntaje del jugador actual merece ser incluido entre los cinco mejores puntajes registrados hasta el momento.




**Estado final**

*Representación visual del estado final del juego.*

**DISEÑO:**

- **Solución de alto nivel:**

    1. Inicialización:

        ➔ Crear la matriz del tablero de juego con dimensiones 8x8.

        ➔ Se inicializan los diccionarios para las escaleras/serpientes, adivinanzas y palabras.

    2. Lectura de archivos:

        ➔ Leer el archivo que contiene las letras y se asignan los valores a la matriz del tablero.

        ➔ Leer el archivo de escaleras y serpientes y almacenar las posiciones y tipos en un diccionario.

        ➔ Leer el archivo de adivinanzas y almacenar las preguntas y respuestas en un diccionario.

        ➔ Leer el archivo de palabras y almacenar las opciones de palabras en un diccionario.

    3. Selección de dificultad:

        ➔ Se presenta al jugador un menú donde puede elegir entre el modo normal o el modo de adivinanzas.

        ➔ Se solicita al jugador que ingrese un nombre de usuario o apodo (nickname).

    4. Juego principal:

        ➔ Se muestra en pantalla el tablero de juego.

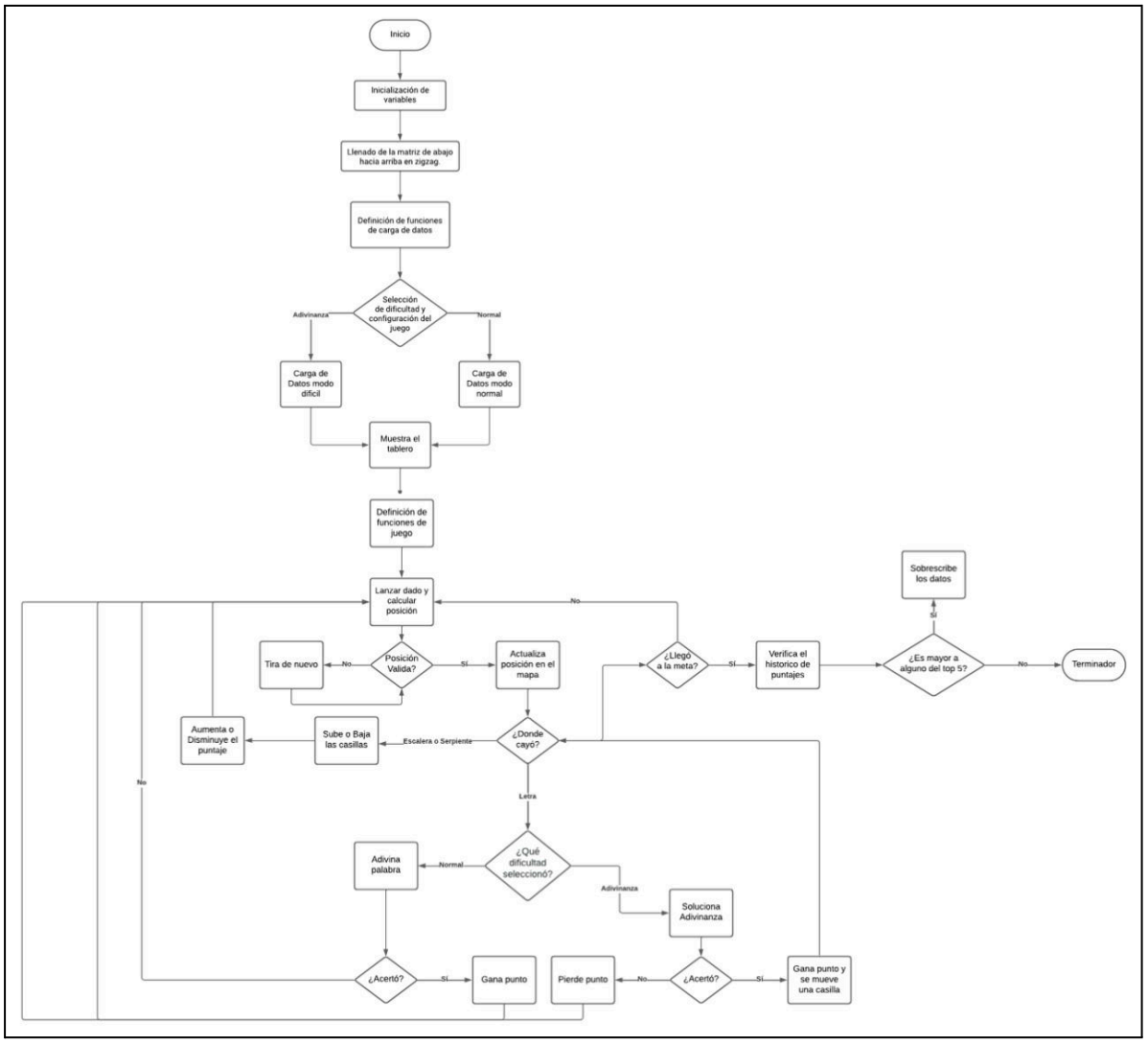
        ➔ Se lanza el dado y el jugador se mueve según el resultado obtenido.

        ➔ Se comprueba si la casilla en la que ha caído el jugador contiene una letra; en caso afirmativo, se le pide que ingrese una palabra o resuelva una adivinanza.

- ➔ Se verifica si el jugador ha caído en una escalera o serpiente y se le mueve a la posición correspondiente.
- ➔ Este proceso se repite hasta que el jugador alcance la última casilla del tablero.

5. Fin del juego:
   - ➔ Se muestra al jugador su puntaje final.
   - ➔ Se actualiza y muestra la tabla de las puntuaciones más altas.
   - ➔ Salir del juego.

**Diagrama de flujo:**



**IMPLEMENTACIÓN:**

El archivo principal se llama main.py y se registra el siguiente código:

```
import os
import random
import time
# Crear una matriz vacía de 8x8
matrix = [[0 for j in range(8)] for i in range(8)]
# Crear un diccionario para almacenar las posiciones de escaleras y serpientes
escaleras_serpientes = {}
# Crear un diccionario para almacenar las adivinanzas
adivinanzas = {}
# Crear un diccionario para almacenar los paises
diccionario_paises = {}
# Llenar la matriz de abajo hacia arriba
contador = 1
for i in range(7, -1, -1):
    if (7 - i) % 2 == 0:  # Fila par en zigzag
        for j in range(8):
            matrix[i][j] = contador
            contador += 1
    else:  # Fila impar en zigzag
        for j in range(7, -1, -1):
            matrix[i][j] = contador
            contador += 1
#Limpia la consola
def limpiar_consola():
        os.system('cls')
```

```
# Leer datos del archivo y asignar valores a la matriz
def leer_letras(archivo):
    with open(archivo, 'r') as file:
        for line in file:
            fila, columna, valor = line.split()
            matrix[int(fila)][int(columna)] = valor

# Leer el archivo y guardar las posiciones
def leer_Serpientes_Escaleras(archivo):
    with open(archivo, 'r') as file:
        for line in file:
            parts = line.split()
            if len(parts) == 6:
                tipo_inicio, inicio_fila, inicio_columna, tipo_fin, fin_fila, fin_columna = parts
                inicio = (int(inicio_fila), int(inicio_columna))
                fin = (int(fin_fila), int(fin_columna))
                if tipo_inicio.upper() == 'W':
                    escaleras_serpientes[inicio] = ('escalera', fin)
                    matrix[inicio[0]][inicio[1]] = tipo_inicio + '.'
                    matrix[fin[0]][fin[1]] = tipo_fin + '.'
                elif tipo_inicio.upper() == 'X':
                    escaleras_serpientes[inicio] = ('serpiente', fin)
                    matrix[inicio[0]][inicio[1]] = tipo_inicio + '.'
                    matrix[fin[0]][fin[1]] = tipo_fin + '.'
```

```
# Leer el archivo y guardar las adivinanzas
def leer_adivinanzas(archivo):
    with open(archivo, "r", encoding="utf-8") as archivo:
        for linea in archivo:
            partes = linea.strip().split(";")
            if len(partes) == 3:
                letra, animal, adivinanza = partes
                adivinanzas[letra] = (animal, adivinanza)
            else:
                print(f"Advertencia: la línea '{linea.strip()}' no tiene el formato esperado.")

def lectura_normal(nombre_archivo):
    with open(nombre_archivo, 'r') as archivo:
        for linea in archivo:
            letra, paises = linea.strip().split(';')
            diccionario_paises[letra] = paises.split(', ')
    return diccionario_paises
```

```
menu = 0
while menu not in [1, 2]:
    menu = int(input("SELECCIONE UNA DIFICULTAD:\n1)NORMAL\n2)ADIVINANZAS\n3)INFORMACIÓN\n-"))
    if menu == 1:
        limpiar_consola()
        usuario_normal = input("Ingrese su NICKNAME: ")
        puntaje_normal = 0
        leer_letras("letras.txt")
        leer_Serpientes_Escaleras("normal_Serpientes_Escaleras.txt")
        lectura_normal("palabras_normal.txt")
        limpiar_consola()
        break  # Salir del bucle después de seleccionar una dificultad
    elif menu == 2:
        limpiar_consola()
        usuario_dificil = input("Ingrese su NICKNAME: ")
        puntaje_dificil = 0
        leer_letras("letras.txt")
        leer_Serpientes_Escaleras("dificil_Serpientes_Escaleras.txt")
        leer_adivinanzas("adivinanzas_dificil.txt")
        limpiar_consola()
        break  # Salir del bucle después de seleccionar una dificultad
    elif menu == 3:
        print("""
Piensa y Juega es un juego al estilo de Escaleras y Serpientes que te sorprenderá con algunos retos de palabras. En
nuestro modo normal, necesitarás tener conocimiento sobre países o ciudades, y objetos para escribir la palabra
correcta según la letra y la señalización: si la letra está precedida por un punto, significa que la palabra a
colocar es un objeto; si el punto viene después, es un país o una ciudad.

Por otro lado, en nuestro modo de juego más difícil, Adivinanzas, tendrás que resolver acertijos cuya respuesta
comienza con la letra en la que caíste.

Piensa y Juega es un juego muy competitivo, por lo que nuestro sistema de puntos beneficia a los jugadores más
conocedores. En el modo normal, ganarás +1 si tomas una escalera y +1 si aciertas una palabra; no te preocupes, en
este modo nada te quitará tus puntos. En el modo Adivinanzas, ganarás +1 si tomas una escalera y +1 si aciertas el
acertijo, además de moverte una casilla adicional. Sin embargo, en este modo, perderás -1 si caes en una serpiente.

Piensa y Juega te brinda la posibilidad de crear tus propios mapas con tus propias combinaciones de escaleras,
serpientes, letras, adivinanzas y palabras. Anímate a crear tus propias partidas y compite por entrar en uno de los
mejores puntajes. ¡Disfruta del juego!
""")
        # No hay 'break' aquí, por lo que el bucle continuará
    else:
        print("Por favor, seleccione una opción válida.")
```

```
# Imprimir la matriz
for row in matrix:
    print(row)

def lanzar_dado():
    return random.randint(1, 6)

def mostrar_adivinanza(letra):
    global puntaje_dificil
    if letra in adivinanzas:
        animal, adivinanza = adivinanzas[letra]
        print(f"Adivinanza: {adivinanza}")
        respuesta = input("¿Quién soy? ").strip()
        if respuesta.lower() == animal.lower():
            print("¡Correcto! Avanzas una casilla adicional.")
            time.sleep(2)
            puntaje_dificil+=1
            limpiar_consola()
            return True
        else:
            print("Incorrecto. Te quedas en la misma casilla.")
            time.sleep(2)
            puntaje_dificil-=1
            limpiar_consola()
            return False

def comprobar_palabra(diccionario, letra):
    global puntaje_normal
    opciones = diccionario.get(letra)
    if opciones:
        print(f"Empieza por la {letra}")
        respuesta = input("--").strip()
        opciones_minusculas = [opcion.lower() for opcion in opciones]
        if respuesta.lower() in opciones_minusculas:
            puntaje_normal +=1
            print("¡Correcto!")
        else:
            print("Incorrecto")
    else:
        print("La letra no tiene opciones asociadas.")
```

```
def mover_jugador(posicion, pasos, valor_anterior):
    fila, columna = posicion
    for _ in range(pasos):
        if fila == 0 and columna == 0:  # El jugador ya está en la meta
            break
        if (7 - fila) % 2 == 0:  # Fila par en zigzag
            if columna < 7:
                columna += 1
            else:
                fila -= 1
        else:  # Fila impar en zigzag
            if columna > 0:
                columna -= 1
            else:
                fila -= 1
    return fila, columna, valor_anterior

def verificar_casilla_adivinanza(posicion_jugador, matrix):
    fila, columna = posicion_jugador
    valor_casilla = matrix[fila][columna]
    if str(valor_casilla) in adivinanzas:
        print(f"Has caído en la casilla {valor_casilla}")
        if mostrar_adivinanza(str(valor_casilla)):
            # Mover al jugador una casilla adicional si responde correctamente
            fila, columna, _ = mover_jugador(posicion_jugador, 1, valor_anterior)
            posicion_jugador = (fila, columna)
            # Verificar si la nueva casilla también tiene una adivinanza
            if posicion_jugador == (0,0):
                return posicion_jugador
            else:
                posicion_jugador = verificar_casilla_adivinanza(posicion_jugador, matrix)
    return posicion_jugador

def actualizar_puntajes(nombre_archivo, nuevo_nombre, nuevo_puntaje):
    nombres = []
    puntajes = []

    # Leer los puntajes actuales del archivo
    with open(nombre_archivo, "r") as archivo:
        for linea in archivo:
            nombre, puntaje = linea.strip().split("  ")
            nombres.append(nombre)
            puntajes.append(int(puntaje))
```

```
# Insertar el nuevo puntaje en la posición correcta
    insertado = False
    for i, puntaje in enumerate(puntajes):
        if nuevo_puntaje > puntaje:
            nombres.insert(i, nuevo_nombre)
            puntajes.insert(i, nuevo_puntaje)
            insertado = True
            break

    # Si el nuevo puntaje no es mayor que ninguno existente, añadirlo al final
    if not insertado:
        nombres.append(nuevo_nombre)
        puntajes.append(nuevo_puntaje)

    # Asegurarse de que solo haya 5 puntajes en la lista
    nombres = nombres[:5]
    puntajes = puntajes[:5]

    # Volver a escribir los puntajes actualizados en el archivo
    with open(nombre_archivo, "w") as archivo:
        for nombre, puntaje in zip(nombres, puntajes):
            archivo.write(f"{nombre}  {puntaje}\n")

    # Mostrar los puntajes en la consola
    print("Mejores puntajes:")
    for nombre, puntaje in zip(nombres, puntajes):
        print(f"{nombre}: {puntaje}")

# Posición inicial del jugador y valor anterior
posicion_jugador = (7, 0)
valor_anterior = matrix[posicion_jugador[0]][posicion_jugador[1]]
matrix[posicion_jugador[0]][posicion_jugador[1]] = '*'
```

```
# Bucle de juego
game = True
while game:
    input("Presiona Enter para lanzar el dado...")
    limpiar_consola()
    dado = lanzar_dado()
    print(f"Has sacado un {dado}")

    # Calcular la nueva posición sin mover al jugador todavía
    nueva_fila, nueva_columna, _ = mover_jugador(posicion_jugador, dado, valor_anterior)

    # Verificar si el jugador se pasaría de la meta
    if nueva_fila < 0 or nueva_columna < 0:
        print("Te has pasado de la meta. Tira de nuevo.")
        continue  # El jugador tira de nuevo el dado sin moverse

    # Extraer el valor de la casilla donde caerá el jugador
    valor_casilla = matrix[nueva_fila][nueva_columna]

    # Restaurar el valor anterior en la posición actual del jugador
    matrix[posicion_jugador[0]][posicion_jugador[1]] = valor_anterior

    # Mover el jugador y obtener el valor anterior de la nueva posición
    fila, columna, valor_anterior = mover_jugador(posicion_jugador, dado, valor_anterior)
    posicion_jugador = (fila, columna)
    if(menu==1):
        comprobar_palabra(diccionario_paises,str(valor_casilla))
    elif(menu==2):
        posicion_jugador = verificar_casilla_adivinanza(posicion_jugador, matrix)
    # Marcar la nueva posición del jugador en la matriz y almacenar el valor anterior
    valor_anterior = matrix[posicion_jugador[0]][posicion_jugador[1]]
    matrix[posicion_jugador[0]][posicion_jugador[1]] = '*'
    print(f"Te mueves a la posición {posicion_jugador}")
```

```
# Verificar si el jugador ha llegado a una escalera o serpiente
    if posicion_jugador in escaleras_serpientes:
        tipo, nueva_posicion = escaleras_serpientes[posicion_jugador]
        if(menu==1):
            if tipo =='W':
                puntaje_normal+=1
        elif(menu==2):
            if tipo =='W':
                puntaje_dificil+=1
            elif tipo == 'X':
                puntaje_dificil-=1
        print(f"¡Has encontrado una {tipo}! Te mueves a la posición {nueva_posicion}")

        # Borrar la posición actual del jugador en la matriz
        matrix[posicion_jugador[0]][posicion_jugador[1]] = valor_anterior

        # Actualizar la posición del jugador y obtener el valor anterior de la nueva posición
        posicion_jugador = nueva_posicion
        valor_anterior = matrix[posicion_jugador[0]][posicion_jugador[1]]

        # Marcar la nueva posición del jugador en la matriz
        matrix[posicion_jugador[0]][posicion_jugador[1]] = '*'
        # Imprimir la matriz
        for row in matrix:
        print(" | ".join(str(cell) for cell in row))

    # Verificar si el jugador ha llegado al final
    if posicion_jugador == (0, 0):
        limpiar_consola()
        print("¡Felicidades! Has llegado al final")
        if(menu==1):
            print(f"Su puntaje total es:{puntaje_normal}")
            actualizar_puntajes("puntajes_normal.txt",usuario_normal,puntaje_normal)
        elif(menu==2):
            print(f"Su puntaje total es:{puntaje_dificil}")
            actualizar_puntajes("puntajes_dificil.txt",usuario_dificil,puntaje_dificil)
        game = False
        break

input("Presiona Enter para salir...")
```

**PRUEBAS:**


La palabra debe ser un objeto que empiece por la letra E
--Esfero
Incorrecto
SELECCIONE UNA DIFICULTAD 1 O 2:
1)NORMAL
2)ADIVINANZAS
3)INFORMACIÓN
-2
¡Felicidades! Has llegado al final
Su puntaje total es:-1
Mejores puntajes:
PANDOXXO: 13
NN: 0
NN: 0
NN: 0
NN: 0
Presiona Enter para salir...
Has sacado un 4
La letra no tiene opciones asociadas.
Te mueves a la posición (6, 1)
Z. | 63 | w. | 61 | 60 | X. | U. | 57
R. | 50 | 51 | 52 | .S | .T | x. | 56
W. | 47 | Q. | w. | .P | 43 | 42 | X.
33 | 34 | 35 | W. | N. | 38 | O. | 40
M. | 31 | .L | .K | X. | 27 | 26 | x.
17 | w. | H. | 20 | I. | 22 | J. | 24
W. | * | 14 | G. | F. | .E | x. | 09
01 | 02 | A. | .B | C. | 06 | 07 | D.
Presiona Enter para lanzar el dado...
Has sacado un 2
Has caído en la casilla A.
Adivinanza: Blanco en el pecho, amarillo en la espalda, el rey de la selva que todo lo acecha.
¿Quién soy? aguila
¡Correcto! Avanzas una casilla adicional.
Has sacado un 2
La palabra debe ser un país o ciudad que empiece por la letra I
--India
¡Correcto!
Has sacado un 3
Te mueves a la posición (0, 5)
¡Has encontrado una serpiente! Te mueves a la posición (1, 6)
Z. | 63 | X. | 61 | 60 | X. | U. | 57
R. | 50 | 51 | 52 | .S | .T | * | 56
x. | 47 | Q. | X. | .P | 43 | 42 | X.
33 | 34 | 35 | x. | N. | 38 | O. | 40
M. | 31 | .L | .K | X. | 27 | 26 | x.
17 | X. | H. | 20 | I. | 22 | J. | 24
x. | 15 | 14 | G. | F. | .E | x. | 09
01 | 02 | A. | .B | C. | 06 | 07 | D.
Presiona Enter para lanzar el dado...
Has sacado un 1
La letra no tiene opciones asociadas.
Te mueves a la posición (7, 1)
Z. | 63 | w. | 61 | 60 | X. | U. | 57
R. | 50 | 51 | 52 | .S | .T | x. | 56
W. | 47 | Q. | w. | .P | 43 | 42 | X.
33 | 34 | 35 | W. | N. | 38 | O. | 40
M. | 31 | .L | .K | X. | 27 | 26 | x.
17 | w. | H. | 20 | I. | 22 | J. | 24
W. | 15 | 14 | G. | F. | .E | x. | 09
01 | * | A. | .B | C. | 06 | 07 | D.
Presiona Enter para lanzar el dado...
Has sacado un 3
La palabra debe ser un objeto que empiece por la letra L
--lulo
Incorrecto
Ingrese su NICKNAME: Flor
¡Felicidades! Has llegado al final
Su puntaje total es:2
Mejores puntajes:
Pandoxxo: 6
Flor: 5
Flor: 2
Crew: 1
NN: 0
Presiona Enter para salir...
Has sacado un 5
Te mueves a la posición (0, 1)
Z. | * | X. | 61 | 60 | X. | U. | 57
R. | 50 | 51 | 52 | .S | .T | x. | 56
x. | 47 | Q. | X. | .P | 43 | 42 | X.
33 | 34 | 35 | x. | N. | 38 | O. | 40
M. | 31 | .L | .K | X. | 27 | 26 | x.
17 | X. | H. | 20 | I. | 22 | J. | 24
x. | 15 | 14 | G. | F. | .E | x. | 09
01 | 02 | A. | .B | C. | 06 | 07 | D.
Presiona Enter para lanzar el dado...
Has sacado un 4
Has caído en la casilla Z.
Adivinanza: En el bosque me deslizo, con mi cola peluda. Astuto y veloz, ¿quién soy?
¿Quién soy? zorro
¡Correcto! Avanzas una casilla adicional.

# SOLUCIÓN A LAS PREGUNTAS

1. ¿Qué estructura de datos seleccionaron para almacenar los puntajes históricos?

- Se utilizan listas (nombres y puntajes) para almacenar los nombres y puntajes de los jugadores. Los puntajes históricos se leen y actualizan en un archivo de texto (puntajes_normal.txt o puntajes_dificil.txt), donde cada línea contiene el nombre del jugador y su puntaje, separados por dos espacios.

2. ¿Cómo se ganan puntos en el juego?

- Dependiendo del nivel que se haya seleccionado en el juego:

  En el modo normal, los puntos se ganan de la siguiente manera:

  ➔ +1 punto por tomar una escalera.

  ➔ +1 punto por acertar una palabra.

  En el modo de adivinanzas, los puntos se ganan de la siguiente manera:

  ➔ +1 punto por tomar una escalera.

  ➔ +1 punto por acertar una adivinanza, y además se avanza una casilla adicional.

  ➔ -1 punto por caer en una serpiente.

3. ¿Cómo se valida que el dato que ingresa el usuario sea en realidad una palabra real?

- La validación se realiza mediante la comparación de la respuesta del usuario con una lista de opciones válidas almacenadas en el diccionario diccionario_paises o adivinanzas. Si la respuesta del usuario está en la lista de opciones asociadas a la letra en la que cae, se considera una palabra real y se otorgan puntos según corresponda.

4. ¿Cuáles son las funciones más importantes del programa y cuál es la función más usada?

- Las funciones más importantes del programa son:
  - ➔ mover_jugador(): Controla el movimiento del jugador en la matriz del juego.
  - ➔ verificar_casilla_adivinanza(): Verifica si el jugador ha caído en una casilla con una adivinanza y maneja la lógica correspondiente.
  - ➔ actualizar_puntajes(): Actualiza y mantiene los puntajes históricos de los jugadores.
  - ➔ La función más usada es mover_jugador(), ya que se llama en cada turno del jugador para actualizar su posición en el tablero.

# CONCLUSIÓN

La creación de juegos es un proceso que involucra una serie de pasos complejos que deben ser cuidadosamente planificados y ejecutados. Desde el inicio con la definición de la temática y la mecánica del juego, hasta la implementación de algoritmos eficientes y las pruebas exhaustivas para garantizar su calidad, cada etapa es fundamental para asegurar una experiencia de juego satisfactoria para los jugadores.

# REFERENCIAS

1. Rossum, G . y Drake, F.L. (2010). The Python Library Reference. Release 2.6.4.
2. El Blog Python. (s.f.). Guía paso a paso para crear videojuegos con Python. Recuperado de https://elblogpython.com/tecnologia/guia-paso-a-paso-para-crear-videojuegos-con-python/