Actividad 6: Búsqueda de números con listas y random

Nombre: Alejandro Ramirez Cruz

Matricula: 379551

Fecha: Lunes 18 septiembre de 2025

Ejercicio 1: Adivina el número (3 intentos)

Descripción:

Crear un juego donde el usuario adivine un número aleatorio entre 1 y 10. El usuario tiene 3 intentos. Al finalizar, mostrar cuántas veces ganó o perdió.

[Explicación del problema]:

1. Generar un número aleatorio.
2. Pedir al usuario que adivine el número (3 intentos máx.).
3. Validar si el número es correcto o dar pistas ("mayor/menor").
4. Al finalizar, preguntar si desea jugar de nuevo.

Solución:

El juego de “Adivina el número” se resolvió usando la librería random, que genera un número aleatorio del 1 al 10. Con un ciclo while se controla que el jugador tenga tres intentos para adivinarlo, y otro ciclo externo controlado por bandera que permite repetir la partida si así lo quiere. Además, se usan condicionales para dar pistas al usuario, indicando si el número buscado es mayor o menor al que escribió.

También se lleva un marcador de victorias y derrotas que se actualiza al final de cada ronda. Para evitar errores si el usuario escribe letras u otros símbolos, se agregó un bloque try-except y validaciones que solo aceptan números entre 1 y 10. En conjunto, el programa mezcla aleatoriedad, ciclos y validación de entradas para hacer un juego simple, dinámico y entretenido.

```
import random    #Importar biblioteca Random para generar números aleatorios
```

```
def adivina_numero():   # Función para adivinar un número
    victorias = 0
    derrotas = 0
    jugar = "si"  # Bandera para repetir el juego

    # Ciclo while que se repite, si el usario quiere jugar de nuevo
    while jugar.lower() == "si":
        aleatorio = random.randint(1, 10)  # Número aleatorio a adivinar
        intentos = 3   # Intentos disponibles
        i = 0

        print("\n¡Adivina el número entre 1 y 10!")

        while intentos > 0:
            try:
                numero = int(input(f"Intento {i+1}: "))
            except ValueError:
                print("Error: ingresa un número válido")
                continue  # Vuelve a pedir número sin restar intentos

            if not 1 <= numero <= 10:
                print("Número fuera del rango, intenta entre 1 y 10")
                continue # Vuelve a pedir número sin gastar intento
            i += 1

            if numero == aleatorio:
                print("¡Ganaste!")
                victorias += 1
                break   # Termina la partida si gana
            elif numero < aleatorio:
                print("El número es mayor.")
            else:
                print("El número es menor.")

            intentos -= 1  # Resta un intento cada que falla

        else:
            # Se ejecuta si se gastan todos los intentos sin acertar
            print(f"Has perdido todos tus intentos. El número era {aleatorio}")
            derrotas += 1
```

```
            derrotas += 1

        # Imprime el marcador
        print(f"\nJuegos ganados: {victorias} | Perdidos: {derrotas}")

        # Pregunta al usuario si desea jugar otra vez
        jugar = input("¿Quieres jugar otra vez? (si/no): ")
```

```
# Llamado a la funcion y ejecuta el juego
adivina_numero()
```

```
Intento 1: 9
¡Ganaste!

Juegos ganados: 2 | Perdidos: 0
¿Quieres jugar otra vez? (si/no): si

¡Adivina el número entre 1 y 10!
Intento 1: 3
El número es mayor.
Intento 2: 6
El número es mayor.
Intento 3: 5
El número es mayor.
Has perdido todos tus intentos. El número era 9

Juegos ganados: 2 | Perdidos: 1
¿Quieres jugar otra vez? (si/no): si

¡Adivina el número entre 1 y 10!
Intento 1: 9
El número es menor.
Intento 2: 6
El número es mayor.
Intento 3: 8
¡Ganaste!

Juegos ganados: 3 | Perdidos: 1
¿Quieres jugar otra vez? (si/no): si

¡Adivina el número entre 1 y 10!
Intento 1: 5
El número es menor.
Intento 2: 7
El número es menor.
Intento 3: 3
El número es menor.
Has perdido todos tus intentos. El número era 1

Juegos ganados: 3 | Perdidos: 2
¿Quieres jugar otra vez? (si/no): si

¡Adivina el número entre 1 y 10!
Intento 1: 8
El número es menor.
Intento 2: 5
El número es menor.
Intento 3: 2
El número es menor.
Has perdido todos tus intentos. El número era 1

Juegos ganados: 3 | Perdidos: 3
¿Quieres jugar otra vez? (si/no): si

¡Adivina el número entre 1 y 10!
Intento 1: 1
El número es mayor.
Intento 2: 6
¡Ganaste!
```

Ejercicio 2: Busca número en lista (A)

Descripción:

Generar una lista de 10 números aleatorios únicos (1-10). Mostrar un número aleatorio de la lista y pedir al usuario que adivine su índice. El usuario tiene 3 intentos.

[Explicación del problema]:

1. Usar un ciclo while y validar duplicados manualmente.
2. Seleccionar un número aleatorio de la lista y mostrarlo.
3. Pedir al usuario el índice donde cree que está (3 intentos).
4. Mostrar mensajes de "Ganaste" o "Perdiste".

Solución:

Primero, se importa la biblioteca random, que permite generar números aleatorios entre 1 y 10 para construir la lista y elegir el número que el jugador debe adivinar. Después, se crea una función que genera la lista de 10 números únicos utilizando un ciclo while y verificando que no haya duplicados; esto permite tener un conjunto fijo de números sin repeticiones. Una vez creada la lista, se selecciona un número aleatorio de esa lista con random.choice, que será el objetivo del juego.

A continuación, se implementa otro ciclo while que controla los intentos del jugador, junto con estructuras de control para comparar la posición ingresada con la posición real del número. Se usan validaciones para asegurar que el usuario solo ingrese números dentro del rango permitido, y un bloque try-except para manejar errores si se ingresa un valor no numérico, evitando que el programa se detenga. Finalmente, se muestra un mensaje de victoria o derrota junto con la lista completa.

```
import random    #Importar biblioteca Random para generar números aleatorios
```

```
def llena_lista_while():    # Función que genera una lista del 1 al 10 sin duplicados con ciclo while
    lista_numeros = []    # Lista vacia para guardar los números

    # Se repite el ciclo hasta que los 10 números sean unicos
    while len(lista_numeros) < 10:
        num = random.randint(1, 10)    # Genera número aleatorio entre 1 y 190

        if num not in lista_numeros:    # Se agrega solo si no está repetido
            lista_numeros.append(num)

    return lista_numeros        # Regresamos lista sin duplicados
```

```
def juego_version_a():     # Función que genera el juego en su versión A
    lista = llena_lista_while()    # Lista generada sin duplicados
    num_aleatorio = random.choice(lista)
    intentos = 3
    i = 0

    print(f"\n¡En una lista de 10 posiciones, encuentra el índice donde se encuentra el numero {num_aleatorio}!")

    # Ciclo while que se detiene hasta que los intentos se hayan acabados
    while intentos > 0:
        try:
            posicion = int(input(f"Intento {i+1}: "))
        except ValueError:
            print("Error: ingresa un número válido")
            continue   # Vuelve a pedir sin gastar intento

        if posicion < 1 or posicion > 10:    # Valida el rango permitido
            print("Número fuera del rango, intenta entre 1 y 10")
            continue   # Vuelve a pedir sin gastar intento
        i += 1

        # Verifica que el número este en la posición indicada
        if lista[posicion-1] == num_aleatorio:
            print(f"¡Ganaste! el {num_aleatorio} esta en el indice {posicion}")
            print(f"Lista de números generada: {lista}")
            break    # Termina la partida si gana
        else:
            intentos -= 1

    else:
        # Se ejecuta si se gastan todos los intentos sin acertar
        indice = lista.index(num_aleatorio)   # Indice correto (1-10)
        print(f"Has perdido todos tus intentos. La posición del número era: {indice+1}")
        print(f"Lista de números generada: {lista}")
```

```
# Llamado a la funcion y ejecuta el juego
juego_version_a()
```

```
¡En una lista de 10 posiciones, encuentra el índice donde se encuentra el numero 4!
Intento 1: 6
Intento 2: 5
Intento 3: 3
Has perdido todos tus intentos. La posición del número era: 9
Lista de números generada: [6, 9, 7, 2, 10, 3, 5, 1, 4, 8]
```

Ejercicio 2: Busca número en lista (B)

Descripción:

Generar una lista de 10 números aleatorios únicos (1-10). Mostrar un número aleatorio de la lista y pedir al usuario que adivine su índice. El usuario tiene 3 intentos.

[Explicación del problema]:

1. Generar la lista con random.sample(range(1, 11), 10).
2. Seleccionar un número aleatorio de la lista y mostrarlo.
3. Pedir al usuario el índice donde cree que está (3 intentos).
4. Mostrar mensajes de "Ganaste" o "Perdiste".

Solución:

Primero, se importa la biblioteca random, que permite generar números aleatorios entre 1 y 10. A diferencia de la versión A , aquí se utiliza la función random.sample para crear directamente una lista de 10 números únicos sin necesidad de un ciclo while. Luego, se selecciona un número aleatorio de esa lista con random.choice, que será el objetivo que el jugador debe encontrar.

El juego funciona con un ciclo while que controla los intentos disponibles (tres por partida). Se usan estructuras de control para comparar la posición ingresada por el jugador con la posición real del número. Se implementan validaciones para que el usuario solo pueda ingresar números entre 1 y 10, y un bloque try-except para manejar entradas no numéricas, evitando errores. Al final, se muestra un mensaje de victoria o derrota junto con la lista completa.

```
import random    #Importar biblioteca Random para generar números aleatorios
```

```
def llena_lista_sample():      # Función que genera una lista de números del 1 al 10 sin duplicados
    lista_numeros= random.sample(range(1,11),10)   # Toma 10 números únicos del 1 al 10 en orden

    return lista_numeros  # Regresamos lista sin números duplicados
```

```
def juego_version_b():     # Función que genera el juego en su versión B
    lista = llena_lista_sample()        # Lista generada sin duplicados
    num_aleatorio = random.choice(lista)
    intentos = 3
    i = 0

    print(f"\n¡En una lista de 10 posiciones, encuentra el índice donde se encuentra el numero {num_aleatorio}!")

    # Ciclo while que se detiene hasta que los intentos se hayan acabados
    while intentos > 0:
        try:
            posicion = int(input(f"Intento {i+1}: "))
        except ValueError:
            print("Error: ingresa un número válido")
            continue   # Vuelve a pedir sin gastar intento

        if posicion < 1 or posicion > 10:    # Valida el rango permitido
            print("Número fuera del rango, intenta entre 1 y 10")
            continue   # Vuelve a pedir sin gastar intento
        i += 1

        # Verifica que el número este en la posición indicada
        if lista[posicion-1] == num_aleatorio:
            print(f"¡Ganaste! el {num_aleatorio} esta en el indice {posicion}")
            print(f"Lista de números generada: {lista}")
            break    # Termina la partida si gana
        else:
            intentos -= 1

    else:
        # Se ejecuta si se gastan todos los intentos sin acertar
        indice = lista.index(num_aleatorio)   # Indice correto (1-10)
        print(f"Has perdido todos tus intentos. La posición del número era: {indice+1}")
        print(f"Lista de números generada: {lista}")
```

```
# Llamado a la funcion y ejecuta el juego
juego_version_b()
```

```
¡En una lista de 10 posiciones, encuentra el índice donde se encuentra el numero 4!
Intento 1: 9
Intento 2: 3
Intento 3: 7
Has perdido todos tus intentos. La posición del número era: 4
Lista de números generada: [10, 8, 6, 4, 1, 5, 7, 2, 9, 3]
```

Ejercicio 2: Busca número en lista (C)

Descripción:

Generar una lista de 10 números aleatorios únicos (1-10). Mostrar un número aleatorio de la lista y pedir al usuario que adivine su índice. El usuario tiene 3 intentos.

[Explicación del problema]:

1. Crear una lista ordenada y mezclarla con shuffle().
2. Seleccionar un número aleatorio de la lista y mostrarlo.
3. Pedir al usuario el índice donde cree que está (3 intentos).
4. Mostrar mensajes de "Ganaste" o "Perdiste".

Solución:

Primero, se importa la biblioteca random, que permite generar números aleatorios. En esta versión C, se crea una lista ordenada del 1 al 10 y luego se mezcla usando random.shuffle, lo que garantiza que todos los números estén presentes pero en un orden aleatorio, es decir sin un orden fijo. Después, se selecciona un número aleatorio de la lista con random.choice, que será el objetivo que el jugador debe encontrar.

El juego utiliza un ciclo while para controlar los tres intentos disponibles. Se implementan condicionales (if-elif-else) para verificar si la posición ingresada por el jugador coincide con la del número seleccionado y dar retroalimentación. También se incluyen validaciones para asegurar que solo se ingresen números del 1 al 10 y un bloque try-except para manejar entradas no numéricas. Al final, se muestra un mensaje de victoria o derrota junto con la lista completa.

```
import random     #Importar biblioteca Random para generar números aleatorios
```

```
def llena_lista_shuffle():      # Función que genera una lista de números del 1 al 10 sin duplicados
    lista_numeros= [1,2,3,4,5,6,7,8,9,10]    # Llenamos una lista ordenada del 1 al 10 sin repetidos
    random.shuffle(lista_numeros)   # Mezclamos la lista de manera aleatoria (sin duplicados)

    return lista_numeros    # Regresamos lista con números mezclados
```

```
def juego_version_c():     # Función que genera el juego en su versión C
    lista = llena_lista_shuffle()      # Lista generada sin duplicados
    num_aleatorio = random.choice(lista)
    intentos = 3
    i = 0

    print(f"\n¡En una lista de 10 posiciones, encuentra el índice donde se encuentra el numero {num_aleatorio}!")

    # Ciclo while que se detiene hasta que los intentos se hayan acabados
    while intentos > 0:
        try:
            posicion = int(input(f"Intento {i+1}: "))
        except ValueError:
            print("Error: ingresa un número válido")
            continue   # Vuelve a pedir sin gastar intento

        if posicion < 1 or posicion > 10:    # Valida el rango permitido
            print("Número fuera del rango, intenta entre 1 y 10")
            continue   # Vuelve a pedir sin gastar intento
        i += 1

        # Verifica que el número este en la posición indicada
        if lista[posicion-1] == num_aleatorio:
            print(f"¡Ganaste! el {num_aleatorio} esta en el indice {posicion}")
            print(f"Lista de números generada: {lista}")
            break    # Termina la partida si gana
        else:
            intentos -= 1

    else:
        # Se ejecuta si se gastan todos los intentos sin acertar
        indice = lista.index(num_aleatorio)   # Indice correto (1-10)
        print(f"Has perdido todos tus intentos. La posición del número era: {indice+1}")
        print(f"Lista de números generada: {lista}")
```

```
# Llamado a la funcion y ejecuta el juego
juego_version_c()
```

```
¡En una lista de 10 posiciones, encuentra el índice donde se encuentra el numero 9!
Intento 1: 5
Intento 2: 6
Intento 3: 8
Has perdido todos tus intentos. La posición del número era: 4
```

```
Lista de números generada: [10, 1, 5, 9, 3, 7, 4, 6, 2, 8]
```

CONCLUSIONES

En esta práctica se profundizó el uso del módulo random y la manip
listas sin duplicados, lo cual permitió desarrollar juegos interac
múltiples intentos y validación de entradas. Comprendí que las lis
la organización y legibilidad del código, y que algunas funciones
tareas complejas mientras que otras requieren un razonamiento lógi
funcionar correctamente.
Además, el manejo de números aleatorios y rangos con range() ofrec
herramientas eficientes para resolver problemas. La aplicación de
buenas prácticas de documentación ayudó a comprender mejor la lógi
programa, fortaleciendo mis habilidades técnicas y mi capacidad pa
estructurar soluciones de manera clara y organizada.

En esta práctica se profundizó el uso del módulo random y la manipulación de listas sin duplicados, lo cual permitió desarrollar juegos interactivos con múltiples intentos y validación de entradas. Comprendí que las listas facilitan la organización y legibilidad del código, y que algunas funciones simplifican tareas complejas mientras que otras requieren un razonamiento lógico para funcionar correctamente. Además, el manejo de números aleatorios y rangos con range() ofrece herramientas eficientes para resolver problemas. La aplicación de comentarios y buenas prácticas de documentación ayudó a comprender mejor la lógica del programa, fortaleciendo mis habilidades técnicas y mi capacidad para estructurar soluciones de manera clara y organizada.