Universidad Autónoma de Baja California

Alumno: Colin Reyes Brian Gabriel

Matrícula: 379494

Materia: Phyton

Docente: Pedro Nuñez Yepiz

Periodo: Agosto - Diciembre

Grado/semestre: 3er semestre

Grupo: 432

Tema: Actividad 6

Fecha: 22/09/25

INTRODUCCIÓN

El propósito de esta actividad es aplicar conceptos de programación relacionados con la generación de números aleatorios, listas y control de ciclos, mediante la creación de juegos interactivos. Estas prácticas permiten reforzar la lógica de programación, la validación de entradas y la interacción con el usuario, al mismo tiempo que se desarrollan algoritmos que solucionan problemas de manera práctica y entretenida.

TEORÍA

Módulo random: En Python, el módulo random permite generar números aleatorios de diferentes maneras. Por ejemplo, random.randint(a, b) genera un número entero aleatorio entre a y b, y random.choice(lista) selecciona un elemento aleatorio de una lista.

Listas sin duplicados: Son listas en las que cada elemento aparece únicamente una vez. Se pueden crear evitando duplicados manualmente (usando ciclos y comprobaciones) o utilizando funciones de random como random.sample(range, n) para generar directamente n elementos únicos de un rango, o mezclando una lista con random.shuffle() para reordenarla aleatoriamente sin repetir elementos.

EJERCICIO 1: Adivina el Número (3 Intentos)

[16] 0s
```
import random
```

[18] 0s
```
def jugar():
    numero = random.randint(1, 10)
    intentos = 3

    for i in range(1, intentos + 1):
        try:
            guess = int(input(f"Intento {i}: Ingresa un número entre 1 y 10: "))

            if guess < 1 or guess > 10:
                print("Debes ingresar un número dentro del rango (1-10).")
                continue

            if guess == numero:
                print("¡Ganaste!")
                return True
            elif guess < numero:
                print("El número es mayor.")
            else:
                print("El número es menor.")

        except ValueError:
            print("Entrada inválida, ingresa un número entero.")

    print(f"Perdiste. El número era {numero}.")
    return False
```

[20] 0s
```
def main():
    ganados = 0
    perdidos = 0

    while True:
        if jugar():
            ganados += 1
        else:
            perdidos += 1

        print(f"\nJuegos ganados: {ganados} | Perdidos: {perdidos}")

        otra = input("\n¿Quieres jugar de nuevo? (s/n): ").lower()
        if otra != "s":
            print("Gracias por jugar. ¡Hasta luego!")
            break
```

[21] 55s
```
if __name__ == "__main__":
    main()
```

```
Intento 1: Ingresa un número entre 1 y 10: 5
El número es menor.
Intento 2: Ingresa un número entre 1 y 10: 2
El número es mayor.
Intento 3: Ingresa un número entre 1 y 10: 4
El número es menor.
Perdiste. El número era 3.
```

```
Perdiste. El número era 3.

Juegos ganados: 0 | Perdidos: 1

¿Quieres jugar de nuevo? (s/n): s
Intento 1: Ingresa un número entre 1 y 10: 4
El número es mayor.
Intento 2: Ingresa un número entre 1 y 10: 8
El número es menor.
Intento 3: Ingresa un número entre 1 y 10: 7
El número es menor.
Perdiste. El número era 5.

Juegos ganados: 0 | Perdidos: 2

¿Quieres jugar de nuevo? (s/n): n
Gracias por jugar. ¡Hasta luego!
```

EJERCICIO 2: Busca Número en Lista (3 Versiones)

```
[2] ✓ 0s
import random
random.seed(42)
```

```
[3] ✓ 0s
def llena_lista_shufle():
    lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    random.shuffle(lista)
    return lista
```

```
[4] ✓ 0s
def llena_lista_while():
    lista = []
    i = 0
    while i < 10:
        num = random.randint(1, 10)
        if num in lista:
            continue
        lista.append(num)
        i += 1
    return lista
```

```
[9] ✓ 0s
def llena_lista_sample():
    lista = random.sample(range(1, 11), 10)
    return lista
```

```
[6] ✓ 0s
lista = llena_lista_shufle()
print(lista)

[8, 4, 3, 9, 6, 7, 10, 5, 1, 2]
```

```
[7] ✓ 0s
lista = llena_lista_while()
print(lista)

[10, 7, 1, 2, 4, 9, 8, 5, 3, 6]
```

```
[12] ✓ 0s
listaa = llena_lista_sample()
print(listaa)

[10, 6, 4, 9, 1, 7, 2, 8, 5, 3]
```

```
[13] ✓ 0s
def jugar(lista):
    print("Lista generada:", lista)

    numero = random.choice(lista)
    print("El número a buscar es:", numero)

    intentos = 3
    for i in range(1, intentos + 1):
        try:
            indice = int(input(f"¿En qué índice está? (Intento {i}): "))
            if indice < 0 or indice >= len(lista):
                print("Índice fuera de rango. Intenta de nuevo.")
                continue

            if lista[indice] == numero:
                print(f"¡Ganaste! El {numero} está en el índice {indice}.")
                return
            else:
                print("Incorrecto.")
                print("Incorrecto.")
        except ValueError:
            print("⚠ Entrada inválida, escribe un número.")

    print(f"Perdiste. El {numero} estaba en el índice {lista.index(numero)}.")
```

```
[15] ✓ 42s
if __name__ == "__main__":
    print("\n--- Versión con shuffle ---")
    lista_shuffle = llena_lista_shufle()
    jugar(lista_shuffle)

    print("\n--- Versión con while ---")
    lista_while = llena_lista_while()
    jugar(lista_while)

    print("\n--- Versión con sample ---")
    lista_sample = llena_lista_sample()
    jugar(lista_sample)

--- Versión con shuffle ---
Lista generada: [1, 8, 9, 10, 3, 6, 4, 5, 7, 2]
El número a buscar es: 10
¿En qué índice está? (Intento 1): 4
Incorrecto.
¿En qué índice está? (Intento 2): 2
Incorrecto.
¿En qué índice está? (Intento 3): 6
Incorrecto.
Perdiste. El 10 estaba en el índice 3.
```

```
--- Versión con while ---
Lista generada: [5, 2, 10, 3, 9, 4, 8, 7, 6, 1]
El número a buscar es: 3
¿En qué índice está? (Intento 1): 2
Incorrecto.
¿En qué índice está? (Intento 2): 5
Incorrecto.
¿En qué índice está? (Intento 3): 7
Incorrecto.
Perdiste. El 3 estaba en el índice 3.

--- Versión con sample ---
Lista generada: [1, 6, 7, 3, 10, 2, 8, 9, 5, 4]
El número a buscar es: 9
¿En qué índice está? (Intento 1): 2
Incorrecto.
¿En qué índice está? (Intento 2): 5
Incorrecto.
¿En qué índice está? (Intento 3): 7
¡Ganaste! El 9 está en el índice 7.
```

CONCLUSIONES

Los programas desarrollados permiten aplicar conceptos fundamentales de programación, como la generación de números aleatorios, manejo de listas, control de ciclos y validación de entradas con try-except. A través de los juegos interactivos, se evidencia la importancia de diseñar algoritmos claros y estructurados que permitan al usuario interactuar de manera efectiva, recibir retroalimentación inmediata y experimentar con la lógica de selección y comparación de datos. Estas prácticas fortalecen la comprensión de estructuras de control y fomentan la capacidad de crear soluciones prácticas y entretenidas a problemas específicos.