Actividad 7: Gestión de Datos con Diccionarios y Menú Interactivo

Nombre: Alejandro Ramirez Cruz

Matricula: 379551

Fecha: Lunes 27 septiembre de 2025

Ejercicio 1: Menú Interactivo para Gestión de Alumnos

Descripción:

Desarrollar un programa que permita gestionar datos de alumnos mediante un menú con las siguientes opciones:

1. Leer Diccionario: Generar un diccionario con datos de un alumno, pidiendo al usuario los datos manualmente.
2. Generar Diccionario: Crear un diccionario con datos de un alumno a partir de listas y valores aleatorios.
3. Imprimir Diccionario: Mostrar los datos del alumno en formato de registro.
4. Salir: Finalizar el programa.

Explicación del problema:

Se requiere un sistema que permita registrar y consultar la información de alumnos de manera sencilla y segura. Para ello, se implementan funciones de validación que aseguran que los datos introducidos por el usuario sean correctos. Posteriormente, se crean funciones específicas para cada opción del menú. Además, para evitar que el programa falle ante entradas incorrectas o errores inesperados, se utiliza la estructura try-except.

Solución:

La solución se implementa dividiendo el programa en bloques de funciones y un menú principal:

- Validación:
  Se crean funciones que verifican que el usuario ingrese datos correctos.
  Para ello se emplea un bucle while True, que mantiene la solicitud activa hasta que la persona ingrese un valor válido.
  También se utilizan métodos como:
  - .strip(): elimina espacios en blanco al inicio y al final de la cadena.
  - .upper(): convierte el texto a mayúsculas, útil para estandarizar respuestas como `H` o `M`.
- Gestión de alumnos:
  Se crean funciones que permiten leer los datos directamente del teclado, o generan información aleatoria a partir de listas predefinidas y la biblioteca random.
- Impresión:
  Una función muestra los datos del alumno en formato legible, utilizando f-strings para dar claridad al resultado.
- Menú principal:
  El programa utiliza un bucle while que mantiene activo el menú hasta que el usuario decida salir.
  En cada iteración se muestran las opciones y se emplea .strip() para evitar errores si el usuario agrega espacios, garantizando que la entrada sea procesada correctamente.
- Confirmación de salida:
  Para finalizar el programa, se solicita al usuario que confirme si desea salir (`SI` o `NO`). Se emplea `.strip()` y `.upper()` nuevamente para asegurar que, sin importar cómo escriba la respuesta, sea interpretada de forma correcta.
- Control de errores:
  El uso de try-except en la funciones y en el menú principal evita que el programa se interrumpa en caso de un error inesperado.

```
import random  # Importar bibloteca para las acciones aleatorias y al azar
```

```
# Validacion de entradas

def validar_entero(mensaje): # Se usa "mensaje" como parametro para reutilizar la funcion y pedir difernete texto
    while True:
        try:
            valor = int(input(mensaje))
            return valor
        except ValueError:
            print("Error: Debe ingresar un número entero, intente de nuevo")
```

```
def validar_nombre(mensaje):
    while True:
```

```
    while True:
        # . strip() elimina espacios
        nombre = input(mensaje).strip()
        # Comprueba que la cadena no este vacia
        if nombre:
            return nombre
        else:
            print("Error: El nombre no puede estar vacío.")
```

```
def validar_edad(mensaje):
    while True:
        try:
            edad = int(input(mensaje))  # Usa el parámetro mensaje
            if edad >= 18:
                return edad
            else:
                print("Error: La edad debe ser mayor o igual a 18 años")
        except ValueError:
            print("Error: Debe ingresar un número entero, intente de nuevo")
```

```
def validar_sexo(mensaje):
    while True:
        # .strip() se usa para limpiar espacio y .upper() para convertir a mayusculas
        sexo = input(mensaje).strip().upper()
        if sexo in ['H','M']:
            return sexo
        else:
            print("Error: El sexo debe ser Hombre o Mujer, intente de nuevo")
```

```
# Funcion 1: Leer datos del alumno desde el teclado

def leer_diccionario():
    print("\n" + "="*100) # Crea un salto de linea y lo une a "=" que imprime una salida mas limpia
    print("LEER DICCIONARIO: INGRESO MANUAL")

    # Solicitar datos con validacion y mensaje de c/u
    id_alumno = validar_entero("Ingrese ID: ")
    nombre = validar_nombre("Ingrese nombre(s) 1-2: ")
    apellido_paterno = validar_nombre("Ingrese apellido paterno: ")
    apellido_materno = validar_nombre("Ingrese apellido materno: ")
    edad = validar_edad("Ingrese edad: ")
    sexo = validar_sexo("Ingrese sexo (H/M): ")

    # Crear lista con los datos
    lista_datos = [id_alumno, nombre, apellido_paterno, apellido_materno, edad, sexo]

    # Crear diccionario a partir de las lista
    # Utiliza [#] donde cada clave esta guardada en esa posición de la lista
    diccionario_alumno = {
    'ID': lista_datos[0],
    'Nombre': lista_datos[1],
    'Apellido paterno': lista_datos[2],
    'Apellido materno': lista_datos[3],
    'Edad': lista_datos[4],
    'Sexo': lista_datos[5],
    }

    print("¡Diccionario creado exitosamente!")
    return diccionario_alumno
```

```
# Función 2: Generar Diccionario con datos automáticamente

def generar_diccionario():
    print("\n" + "="*100) # Crea un salto de linea y lo une a "=" que imprime una salida mas limpia
    print("GENERAR DICCIONARIO: DATOS AUTOMÁTICOS")

    # Creación de listas con datos para generar automaticamente
    nombres = ["Cristiano Ronaldo", "Lionel Andres", "Neymar", "Alisha", "Sofía", "Ericka", "Andrea"]
    apellidos_paternos = ["Dos Santos", "Messi"," Da Silva","Martinez", "Gonzales", "Perez", "López"]
    apellidos_maternos = ["Aveiro", "Cuccittini", "Santos", "Sanchéz", "Torres", "García", "Ramirez"]
    sexos = ["H","M"]

    # Generación de datos aleatorios
    id_alumno = random.randint(1000,4000)
    nombre = random.choice(nombres)
    apellido_paterno = random.choice(apellidos_paternos)
    apellido_materno = random.choice(apellidos_maternos)
    edad = random.randint(18,100)
    sexo = random.choice(sexos)

```

```
    # Crear lista con los datos
    lista_datos = [id_alumno, nombre, apellido_paterno, apellido_materno, edad, sexo]

    # Crear diccionario a partir de las lista
    # Utiliza [#] donde cada clave esta guardada en esa posición de la lista
    diccionario_alumno = {
    'ID': lista_datos[0],
    'Nombre': lista_datos[1],
    'Apellido paterno': lista_datos[2],
    'Apellido materno': lista_datos[3],
    'Edad': lista_datos[4],
    'Sexo': lista_datos[5],
    }

    print("¡Diccionario creado exitosamente!")
    return diccionario_alumno
```

```
# Funcion 3: Imprime los datos en formato legible

def imprimir_diccionario(diccionario_alumno): # Se usa diccionario_alumno para imprimir esta ficha
    print("\n" + "="*100) # Crea un salto de linea y lo une a "=" que imprime una salida mas limpia
    print("IMPRIMIR DICCIONARIO")

    # Verifica que el diccionario tenga datos disponibles
    if diccionario_alumno:
        print("=== DATOS DEL ALUMNO ===")
        print(f"ID: {diccionario_alumno['ID']}")
        print(f"Nombre: {diccionario_alumno['Nombre']}")
        print(f"Apellidos: {diccionario_alumno['Apellido paterno']} {diccionario_alumno['Apellido materno']}")
        print(f"Edad: {diccionario_alumno['Edad']}")
        print(f"Sexo: {diccionario_alumno['Sexo']}")
    else:
        print("Error: No hay datos disponibles")
```

```
# Funcion 4: Salir del programa

def confirmar_salida():

    while True:
        # .strip() se usa para limpiar espacio y .upper() para convertir a mayusculas
        usuario = input("¿Desea salir (SI/NO): ").strip().upper()
        if usuario == 'SI':
            return True
        elif usuario == 'NO':
            return False
        else:
            print("Error: Ingrese 'SI' para salir o 'NO' para continuar")
```

```
# Mostrar el menu interactivo

def mostrar_menu():
    print("\n" + "="*100) # Crea un salto de linea y lo une a "=" que imprime una salida mas limpia
    print("=== MENÚ ===")
    print("1. Leer Diccionario")
    print("2. Generar Diccionario")
    print("3. Imprimir Diccionario")
    print("0. Salir")
```

```
# Funcion Principal

def menu_principal():

    # Variable que guarda la ficha del alumno. None es que esta vacia
    diccionario_actual = None

    while True:
        try:
            mostrar_menu()
            opcion = input("Seleccione una opcion (0-3): ").strip()

            if opcion == '1':
                diccionario_actual = leer_diccionario()
            elif opcion == '2':
                diccionario_actual = generar_diccionario()
            elif opcion == '3':
                imprimir_diccionario(diccionario_actual)
            elif opcion == '0':
                if confirmar_salida():
```

```
                    print("Gracias por usar el sistema")
                    break
                else:
                    print("Continuando...")
            else:
                print("Error: Opción no valida")
                continue
        except KeyboardInterrupt:
            print("\n\n⚠️  Programa interrumpido por el usuario.")
            break
```

```
# Ejecuta el programa al ser llamada la función
menu_principal()
```