Actividad 9: Gestión de Datos con Diccionarios y Menú Interactivo con Libreria

Nombre: Alejandro Ramirez Cruz

Matricula: 379551

Fecha: Lunes 27 octubre de 2025

Ejercicio 1: Menú Interactivo para Gestión de Trabajadores de una Frabrica

Descripción:

El objetivo es desarrollar un programa que gestione eficientemente los datos de los empleados de una fábrica, garantizando la integridad y consistencia de la información. El sistema debe permitir tanto el registro automático como la búsqueda y manipulación de datos existentes, validando correctamente cada entrada y evitando duplicidades en los identificadores (ID). Asimismo, el programa debe ofrecer opciones para guardar y recuperar la información desde archivos externos en Google Drive.

Explicación del problema:

El objetivo es desarrollar una aplicación que gestione eficientemente los datos de los empleados de una fábrica, garantizando la integridad y consistencia de la información. El sistema debe permitir tanto el registro automático como la búsqueda y manipulación de datos existentes, validando correctamente cada entrada y evitando duplicidades en los identificadores (ID). Asimismo, el programa debe ofrecer opciones para guardar y recuperar la información desde archivos externos en Google Drive.

Solución:

Se implementó una solución modular que divide el programa en funciones específicas y un menú principal.

1. Librerías externas y personalizadas: Se usan pandas para manejar y exportar datos, random para generar valores automáticos, os para archivos, y una librería propia (Libreria.py) con funciones de validación y gestión de la lista de trabajadores.
2. Validaciones: Todas las entradas son verificadas para asegurar datos correctos (números, cadenas, edades e IDs únicos).
3. Generación automática: El sistema puede crear empleados aleatorios desde listas predefinidas, facilitando pruebas y demostraciones.
4. Gestión de datos: Permite buscar por ID o apellido paterno, ordenar, eliminar registros y mostrar la información en tabla.
5. Exportación y carga: Genera y lee archivos .xlsx, .csv y .md desde Google Drive, manteniendo la persistencia de los datos.
6. Menú interactivo: Ofrece una navegación simple y validada, manteniendo la lista activa durante toda la sesión.
7. Manejo de errores: Incluye mensajes claros y confirmaciones antes de operaciones críticas, evitando errores del usuario.

En conjunto, el programa combina estructuras de datos eficientes, validaciones y persistencia, creando una herramienta práctica para la gestión de empleados.

```
# MONTAR GOOGLE DRIVE
from google.colab import drive
drive.mount('/content/drive')
```

```
# AGREGAR LA RUTA DONDE ESTA MI LIBRERIA PERSONAL
import sys
sys.path.append('/content/drive/MyDrive/Actividad_9')
```

```
#IMPORTAR LIBRERIA PROPIA
import Libreria
print(Libreria) # Verifica que la librería se haya importado correctamente
```

```
#IMPORTAR LIBRERIAS EXTRAS
from IPython.display import clear_output
import random
import json
import yaml
import pandas as pd
import time
import os
```

```python
#PROGRAMA PRINCIPAL

#FUNCIONES ADICIONALES

def buscar_por_appat(lista_trabajadores): # Busca todas las coincidencias por apellido paterno

    if Libreria.verificar_lista_vacia(lista_trabajadores): # Si la lista esta vacia regresa sin ejecutar nada
        return

    buscar_apellido = Libreria.validar_cadena("Ingrese el appellido paterno a buscar: ")

    coincidencias = []

    for trabajador in lista_trabajadores: # Recorre la lista y busca coincidencias
        if trabajador['A. Paterno'].upper() == buscar_apellido.upper():
            coincidencias.append(trabajador)

    if coincidencias:  # Imprime los resultados
        print(f"\nSe encontraron {len(coincidencias)} coincidencias con el apellido '{buscar_apellido}':")
        for i, trabajador in enumerate(coincidencias, 1):
            print(f"\nCoincidencia {i}:")
            for clave, valor in trabajador.items():
                print(f"  {clave}: {valor}")
    else:
        print(f"No se encontraron trabajadores con el apellido paterno '{buscar_apellido}'")
```

```python
def generar_archivo(lista_trabajadores):  # Genera archivos en diferentes formatos usando pandas

    if Libreria.verificar_lista_vacia(lista_trabajadores): # Si la lista esta vacia regresa sin ejecutar nada
        return lista_trabajadores

    print("\n--- GENERAR ARCHIVO ---")
    print("a) Excel (.xlsx)")
    print("b) CSV (.csv)")
    print("c) MarkDown (.md)")

    opcion = input("Seleccione el formato (a/b/c): ").lower()
    nombre_archivo = input("Ingrese el nombre del archivo (sin extensión): ").strip()

    # Convierte la lista de diccionarios a DataFrame
    data = pd.DataFrame(lista_trabajadores)

    try:  # Según la opción elegida, genera el archivo correspondiente y lo guarda en mi Drive
        if opcion == 'a':
            ruta = f'/content/drive/MyDrive/Actividad_9/{nombre_archivo}.xlsx'
            data.to_excel(ruta, index=False)
            print("Archivo Excel generado correctamente")

        elif opcion == 'b':
            ruta = f'/content/drive/MyDrive/Actividad_9/{nombre_archivo}.csv'
            data.to_csv(ruta, index=False)
            print("Archivo CSV generado correctamente'")

        elif opcion == 'c':
            ruta = f'/content/drive/MyDrive/Actividad_9/{nombre_archivo}.md'
            data.to_markdown(ruta, index=False)
            print("Archivo Markdown generado correctamente")

        else:
            print("Error: Opción no válida")

    except Exception as e:
        print(f"Error al generar archivo: {e}")

    return lista_trabajadores
```

```python
def cargar_archivo(lista_trabajadores): # Carga cualquier archivo desde el Drive
    print("\n--- CARGAR ARCHIVO ---")
    nombre_archivo = input("Ingrese el nombre del archivo (con extensión): ").strip()

    ruta = f'/content/drive/MyDrive/Actividad_9/{nombre_archivo}'

    if not os.path.exists(ruta):  # Verifica si el archivo existe
        print("El archivo no existe")
```

```
        return lista_trabajadores

    try:  # Lee el archivo según su formato
        if nombre_archivo.endswith('.csv'):
            data = pd.read_csv(ruta)
        elif nombre_archivo.endswith('.xlsx'):
            data = pd.read_excel(ruta)
        else:
            print("Formato no soportado")
            return lista_trabajadores

        # Convierte el DataFrame a lista de diccionarios
        nuevos_trabajadores = data.to_dict('records')
        ids_existentes = [t['ID'] for t in lista_trabajadores]
        trabajadores_agregados = 0

        # Agrega solo los trabajadores nuevos (evita duplicados)
        for trabajador in nuevos_trabajadores:
            if trabajador['ID'] not in ids_existentes:
                lista_trabajadores.append(trabajador)
                ids_existentes.append(trabajador['ID'])
                trabajadores_agregados += 1

        print(f"Se cargaron {trabajadores_agregados} trabajadores nuevos desde '{ruta}'")

    except Exception as e:
        print(f"Error al cargar archivo: {e}")

    return lista_trabajadores
```

```
def imprimir_archivo(): # Imprime el contenido del archivo
    print("\n--- IMPRIMIR ARCHIVO ---")
    nombre_archivo = input("Ingrese el nombre del archivo: ").strip()

    ruta = f'/content/drive/MyDrive/Actividad_9/{nombre_archivo}'

    # Verifica si el archivo existe
    if not os.path.exists(ruta):
        print(f"El archivo no existe en: {ruta}")
        return

    try: # Segun el tipo de archivo, se lee y se imprime
        if nombre_archivo.endswith('.csv'):
            data = pd.read_csv(ruta)
            print(f"\nContenido de '{ruta}':")
            print(data.to_string(index=False))
        elif nombre_archivo.endswith('.xlsx'):
            data = pd.read_excel(ruta)
            print(f"\nContenido de '{ruta}':")
            print(data.to_string(index=False))
        elif nombre_archivo.endswith('.md'):
            data = pd.read_csv(ruta) if nombre_archivo.endswith('.csv') else pd.read_excel(ruta)
            print(f"\nContenido de '{ruta}':")
            print(data.to_string(index=False))
        else:
            print("Error: Formato no soportado para visualización")
    except Exception as e:
        print(f"Error al leer archivo: {e}")
```

```
#MENU PRINCIPAL
def mostrar_menu():

    lista_trabajadores = []

    while True:
        print("\n")
        print("====SISTEMA DE GESTIÓN DE EMPLEADOS====")
        print("1.- Agregar (automático 10)")
        print("2.- Eliminar {ID}")
        print("3.- Imprimir lista (tabla)")
        print("4.- Buscar {ID}")
        print("5.- Buscar {appat} todas las coincidencias")
        print("6.- Ordenar {ID}")
        print("7.- Generar archivo {ID}")
        print("8.- Cargar archivo {ID}")
        print("9.- Imprimir archivo {ID}")
```

```
        print("10.- Borrar Toda la lista {ID}")
        print("0.- SALIR")
        print("\n")

        opcion = Libreria.validar_entero("Seleccione una opción: ") # Se pide al usuario una opción validada

        if opcion == 0: # Si el usuario desea salir le pregunta
            confirmar = input("¿Está seguro de salir? (s/n): ").lower()
            if confirmar == 's':
                print("Ha finalizado el programa...")
                break #  Sale del programa
        elif opcion == 1:
            lista_trabajadores = Libreria.agregar_automatico(lista_trabajadores)
        elif opcion == 2:
            lista_trabajadores = Libreria.eliminar_id(lista_trabajadores)
        elif opcion == 3:
            Libreria.imprimir_lista(lista_trabajadores)
        elif opcion == 4:
            Libreria.buscar_id_trabajador(lista_trabajadores)
        elif opcion == 5:
            buscar_por_appat(lista_trabajadores)
        elif opcion == 6:
            lista_trabajadores = Libreria.ordenar_lista(lista_trabajadores)
        elif opcion == 7:
            lista_trabajadores = generar_archivo(lista_trabajadores)
        elif opcion == 8:
            lista_trabajadores = cargar_archivo(lista_trabajadores)
        elif opcion == 9:
            imprimir_archivo()
        elif opcion == 10:
            lista_trabajadores = Libreria.borrar_toda_lista(lista_trabajadores)
        else:
            print("Error: Opción no válida. Intente nuevamente.")
```

```
# EJECUTAR EL PROGRAMA
if __name__ == "__main__":
    try:
        mostrar_menu()  # Ejecuta el menú principal
    except KeyboardInterrupt:
        print("\n\nPrograma interrumpido por el usuario.") # Excepcion para cuando el usario interrumpa la ejecucion
```

```
3.- Imprimir lista (tabla)
4.- Buscar {ID}
5.- Buscar {appat} todas las coincidencias
6.- Ordenar {ID}
7.- Generar archivo {ID}
8.- Cargar archivo {ID}
9.- Imprimir archivo {ID}
10.- Borrar Toda la lista {ID}
0.- SALIR


Seleccione una opción: 8

--- CARGAR ARCHIVO ---
Ingrese el nombre del archivo (con extensión): Nuevo.csv
Se cargaron 0 trabajadores nuevos desde '/content/drive/MyDrive/Actividad_9/Nuevo.csv'


====SISTEMA DE GESTIÓN DE EMPLEADOS====
```