Informe Teórico – Trabajo Práctico Integrador (Programación 1)

Gestión de Datos de Países en Python

Datos del Estudiante

- **Alumno:** Jerónimo Coronel
- **Profesora:** Cintia Ragoni
- **Materia:** Programación 1
- **Carrera:** Tecnicatura Universitaria en Programación
- **Institución:** Universidad Tecnológica Nacional – Facultad Regional Mendoza
- **Año:** 2025
- **Comisión**: 1PRO4

---

1. Introducción

El presente trabajo práctico integrador tiene como propósito desarrollar un programa en **Python** que permita **gestionar información sobre países**, posibilitando realizar búsquedas, filtrados, ordenamientos, estadísticas, y operaciones de alta y edición de datos, todo ello utilizando un archivo CSV como medio de almacenamiento persistente.

El proyecto se diseñó siguiendo los principios de **programación estructurada y modular**, aplicando conceptos de **listas, diccionarios, funciones, estructuras de control, manejo de archivos y validaciones**, tal como se exige en el marco de la asignatura **Programación 1**.

---

2. Objetivos del Proyecto

- Implementar un programa funcional y robusto para la gestión de datos.

- Utilizar estructuras de datos adecuadas para representar la información.
- Aplicar algoritmos de búsqueda, ordenamiento y cálculo estadístico.
- Incorporar validaciones y manejo de errores para garantizar estabilidad.
- Diseñar un menú interactivo claro y amigable para el usuario.

---

3. Descripción General del Sistema

El programa gestiona la información contenida en un archivo llamado `paises.csv`, donde cada fila representa un país con los siguientes campos:

- **nombre**
- **poblacion**
- **superficie**
- **continente**

El sistema ofrece un menú interactivo con las siguientes funcionalidades:

1. **Buscar país:** busca coincidencias parciales o exactas por nombre.
2. **Filtrar países:** por continente, rango de población o rango de superficie.
3. **Ordenar países:** por nombre, población o superficie (ascendente/descendente).
4. **Mostrar estadísticas:** calcula máximos, mínimos, promedios y conteos.
5. **Agregar país:** permite añadir nuevos registros al archivo CSV.
6. **Editar país:** posibilita modificar los datos de un país existente.
7. **Guardar y salir:** guarda los cambios y finaliza el programa.
8. **Salir sin guardar:** permite abandonar sin modificar el archivo.
9. **Crear CSV de ejemplo:** genera un archivo inicial con datos precargados.

El diseño modular del programa permite mantener un código limpio, fácil de entender y de mantener.

---

4. Estructuras y Conceptos Utilizados

4.1. Listas

Las listas se emplean para almacenar todos los países leídos desde el CSV.
Cada elemento de la lista es un **diccionario** que contiene la información de un país.
Esta estructura permite recorrer, filtrar y ordenar los elementos de manera eficiente.

4.2. Diccionarios

Cada país se representa como un diccionario con las claves:

```
{
    "nombre": "Argentina",
    "poblacion": 45376763,
    "superficie": 2780400,
    "continente": "América"
}
```

Los diccionarios permiten un acceso directo y semánticamente claro a los datos mediante sus claves.

4.3. Funciones

El código se encuentra completamente **modularizado**, garantizando el principio de **una función – una responsabilidad**.
Ejemplos:

- `load_paises()`: lee y valida los datos del CSV.
- `save_paises()`: guarda los cambios realizados.
- `buscar_pais()`, `filtrar_por_continente()`, `ordenar_paises()`: implementan la lógica de cada operación.
- `mostrar_estadisticas()`: calcula y muestra los resultados numéricos.

4.4. Condicionales

Las estructuras `if`, `elif` y `else` se utilizan en todo el programa para controlar el flujo lógico, validar entradas del usuario, seleccionar opciones del menú y verificar condiciones específicas (por ejemplo, que un país no se repita al ser agregado).

4.5. Estructuras repetitivas

Se utilizan bucles `while` y `for`:

- `while` mantiene activo el menú principal hasta que el usuario decida salir.
- `for` se utiliza para recorrer la lista de países y aplicar filtros, cálculos o búsquedas.

No se emplea recursión, cumpliendo con la restricción establecida.

---

5. Algoritmos Implementados

5.1. Algoritmo de Búsqueda – Búsqueda Lineal

Para encontrar coincidencias en nombres de países, el programa utiliza **búsqueda lineal**, recorriendo la lista elemento por elemento hasta hallar coincidencias.

- **Ventajas:** simple de implementar, suficiente para listas pequeñas o medianas.
- **Complejidad temporal:** O(n).

Uso en el código:

```
encontrados = [p for p in paises if termino.lower() in
p["nombre"].lower()]
```

-

5.2. Algoritmo de Ordenamiento – Timsort

El ordenamiento de países se realiza con la función integrada `sorted()`, la cual utiliza **Timsort**, un algoritmo híbrido que combina **Merge Sort** e **Insertion Sort**, optimizando la eficiencia en datos parcialmente ordenados.

- **Complejidad promedio:** O(n log n).

Uso en el código:

```
resultado = sorted(paises, key=lambda x: x["poblacion"],
reverse=True)
```

-

El uso de Timsort garantiza velocidad y estabilidad, siendo el algoritmo estándar de ordenamiento en Python.

---

6. Validaciones y Manejo de Errores

El programa contempla múltiples validaciones y casos borde:

- **Entradas vacías:** no se permiten campos en blanco.
- **Valores numéricos:** se verifica que población y superficie sean enteros positivos.
- **Duplicados:** no se permite agregar países con nombres repetidos.
- **Archivos faltantes:** si `paises.csv` no existe, el sistema inicia con lista vacía o permite generar el CSV de ejemplo.
- **Errores de formato:** las filas mal formateadas se omiten sin interrumpir la ejecución.
- **Excepciones controladas:** se manejan errores como `FileNotFoundError`, `PermissionError` y `ValueError`.

Los mensajes de salida informan claramente el tipo de error o éxito mediante prefijos: `ERROR:`, `ADVERTENCIA:`, `ÉXITO:` e `INFO:`.

---

7. Descripción de las Funcionalidades

1. **Búsqueda de países:**
   Permite localizar países por coincidencia parcial o exacta de nombre, mostrando los resultados en formato tabulado.

2. **Filtrado:**
   - Por continente.
   - Por rango de población.
   - Por rango de superficie.
     Utiliza comparaciones condicionales dentro de comprensiones de listas.

3. **Ordenamiento:**
   Permite ordenar alfabéticamente o por valores numéricos en ambos sentidos

(ascendente o descendente).

4. **Estadísticas:**
   Calcula:
   - País con mayor y menor población.
   - Promedio de población y superficie.
   - Cantidad de países por continente.
5. **Alta y Edición:**
   Los datos nuevos o modificados se guardan automáticamente en el archivo CSV, garantizando persistencia.
6. **Interfaz de menú:**
   Presenta una estructura visual clara con numeración, líneas divisorias y mensajes consistentes.

---

8. Casos Borde Considerados

- **Archivo CSV vacío o inexistente:** se crea automáticamente o se inicia sin datos.
- **Nombres repetidos:** se impide la duplicación.
- **Rangos invertidos (máximo < mínimo):** el programa los intercambia automáticamente.
- **Campos inválidos al editar:** los valores se mantienen y se informa al usuario.
- **Interrupción por teclado:** se captura con `KeyboardInterrupt` para salir de forma segura.

---

9. Conclusiones

El desarrollo de este trabajo permitió aplicar de forma práctica los principales conceptos de la programación estructurada en Python, fortaleciendo la comprensión de las estructuras de datos y la lógica de control.

Se logró construir un sistema **robusto, validado y modular**, cumpliendo todos los objetivos propuestos.

El uso de **listas, diccionarios, funciones y manejo de archivos CSV** permitió desarrollar una aplicación realista y escalable, demostrando la utilidad de los fundamentos teóricos en un entorno práctico.

Además, se reforzó la importancia de:

- La **modularización** como estrategia de orden y mantenimiento.
- El **manejo de errores** para mejorar la experiencia del usuario.
- La **persistencia de datos** mediante archivos CSV como solución simple pero eficaz.

Este proyecto no solo consolida los conocimientos adquiridos, sino que también sienta las bases para futuros desarrollos que involucren bases de datos y programación orientada a objetos.

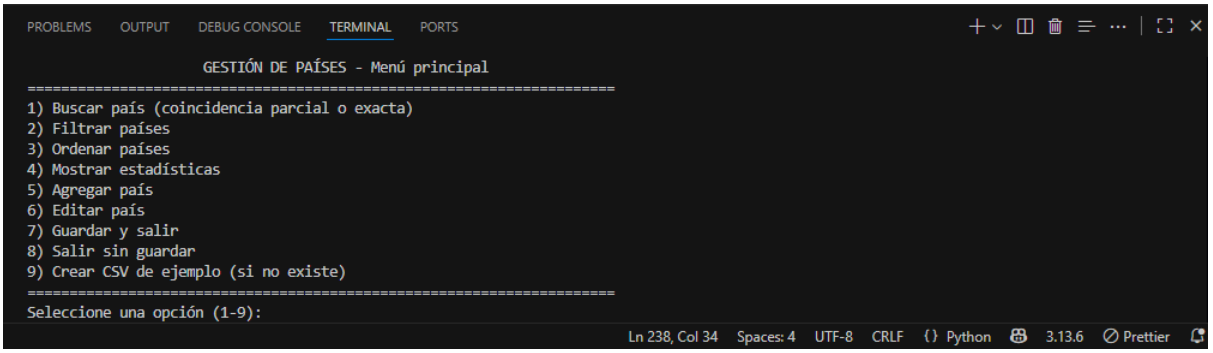
---

10. Bibliografía y Fuentes Consultadas

- Python Software Foundation. *Python 3.12 Documentation.* https://docs.python.org/3/
- Downey, A. (2015). *Think Python: How to Think Like a Computer Scientist.* O'Reilly Media.
- Sweigart, A. (2020). *Automate the Boring Stuff with Python.* No Starch Press.
- UTN – Facultad Regional Mendoza. *Apuntes de cátedra: Programación 1 (2025).*
- Documentación oficial del módulo `csv` de Python.

---

11. Capturas del programa



```
================================================================
Seleccione una opción (1-9): 5

--- Agregar país ---
Nombre del país: Argelia
Población (enteros): 200000
Superficie en km² (enteros): 345424
Continente: africa
ÉXITO: País 'Argelia' agregado y guardado correctamente.

================================================================
```

```
 Seleccione una opción (1-9): 2

 --- Filtrar países ---
 1) Por continente
 2) Por rango de población
 3) Por rango de superficie
 4) Volver al menú principal
 Opción:
```

```
================================================================
Seleccione una opción (1-9): 4

--- Estadísticas ---
INFO: País con mayor población: Brasil (213,993,437)
INFO: País con menor población: Argelia (200,000)
INFO: Promedio de población: 78,144,233.33
INFO: Promedio de superficie: 2,085,181.50 km²

Cantidad de países por continente:
  america: 1
```

---

12. Link repositorio GitHub

https://github.com/jeronimocoronel784-hue/Integrador_Coronel_1PRO4