

## Trabajo Integrador Final

Gestión de Datos de Países en Python: filtros, ordenamientos y estadísticas

**Alumnos – Comisión n°8:** Montenegro Dahyana, Dominguez Matias

**Roles en el Proyecto:**

- **Matías Domínguez:** Desarrollo e implementación de la lógica principal del menú y las operaciones numéricas (Estadísticas, Ordenamiento por algoritmo de intercambio). Responsable de la función validar\_entrada\_numerica.
- **Dahyana Montenegro:** Desarrollo de las funciones de persistencia de datos (Carga y Guardado del CSV), implementación de la estructura de control (match/case) y funciones de entrada/salida. Responsable de la documentación técnica y el **Informe Teórico**.

**Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.**

### Programación I

**Docente Titular**

Sebastian Bruselario

**Docente Tutor**

Virginia Cimino

**Tabla de Contenido**

Introducción y Objetivo del Trabajo .....	3
Estructuras de Datos: La Base del Sistema .....	4
Modularización y Control de Flujo.....	5
Persistencia de Datos y Validaciones.....	6
Ejecución del Programa .....	7, 8, 9
Conclusiones.....	10
Fuentes Bibliográficas.....	11

## 1. Introducción y Objetivo del Trabajo

El proyecto que desarrollamos es una aplicación de consola en Python que permite gestionar información de distintos países, como su nombre, población, superficie y continente.

El objetivo principal fue integrar y practicar todos los temas vistos durante la materia Programación 1: estructuras de datos, funciones, condicionales, bucles, validaciones, ordenamientos y cálculos estadísticos.

El sistema utiliza un archivo CSV llamado paises.csv, donde se guardan los datos para que no se pierdan al cerrar el programa. Además, buscamos que el código sea claro, fácil de leer y esté organizado en funciones, para que cada parte cumpla una tarea específica y el programa sea sencillo de mantener y entender.

## 2. Estructuras de Datos: La Base del Sistema

Se emplearon dos estructuras fundamentales: listas y diccionarios.

- **Lista:** Contiene todos los registros de países y permite recorrerlos fácilmente con bucles for. Es ideal para realizar búsquedas, filtros y aplicar el algoritmo de ordenamiento.
- **Diccionario:** Cada país se representa con un diccionario (por ejemplo: {'nombre': 'Argentina', 'poblacion': 45376763,...}). Esto hace que el código sea mucho más fácil de leer y entender.

### 3. Modularización y Control de Flujo

#### Funciones y Modularización

Para hacer el programa más ordenado, decidimos separar el código en distintas funciones. Cada una se encarga de una tarea específica. Por ejemplo:

- `agregar_pais()` agrega un nuevo país.
- `guardar_datos()` se ocupa de guardar los cambios en el archivo CSV.
- `ordenar_paises()` organiza la lista según lo que elija el usuario (por nombre, población o superficie).

Si algo no funciona, podemos revisar solo la parte del código que corresponde a esa función sin tocar todo lo demás. También usamos una sola lista general llamada `paises`, que se pasa como **parámetro** entre las funciones. Eso nos ayudó a mantener el control de los datos sin repetir variables.

#### Estructuras Condicionales y Repetitivas

En el programa usamos bucles y condicionales para manejar el flujo del sistema. El menú principal funciona con un `while True`, lo que hace que el programa siga corriendo hasta que el usuario elija salir. Los bucles `for` se usan para recorrer la lista de países y hacer distintas acciones, como mostrar los datos, filtrar por continente o calcular estadísticas. Los condicionales `if`, `elif` y `else` sirven para tomar decisiones, por ejemplo, para validar que el usuario no deje campos vacíos o que ingrese números donde corresponde.

#### 4. Persistencia de Datos y Validaciones

##### 4.1.

###### Persistencia de Datos (CSV)

Para que la información no se pierda al cerrar el programa, usamos un archivo CSV llamado paises.csv. Al iniciar, el sistema lee ese archivo (utilizando el módulo csv) y carga todos los países guardados. Si el archivo todavía no existe, el programa crea una lista inicial con algunos países de ejemplo para empezar a usarlo igual. Cada vez que el usuario agrega o modifica un país, el programa vuelve a escribir el archivo completo, guardando todos los cambios.

##### 4.2.

###### Validaciones de Datos

En esta parte nos aseguramos de que los datos ingresados sean correctos antes de guardarlos. Para los campos numéricos como población y superficie usamos una función auxiliar (`validar_entrada_numerica`) que aplica el método `.isdigit()`, que verifica que el valor sea un número. También usamos condicionales if para evitar que el usuario deje campos vacíos o repita un país que ya está cargado. Estas validaciones hacen que el programa sea más estable y fácil de usar, evitando errores que podrían detener la ejecución.

##### 4.3.

###### Algoritmo de Ordenamiento y Estadísticas

Para ordenar los países, implementamos un algoritmo de intercambio (tipo burbuja). Este algoritmo compara los valores de dos países seguidos y los intercambia si están en el orden incorrecto. El usuario puede elegir ordenar por nombre, población o superficie, de forma ascendente o descendente.

Además, el sistema calcula distintas estadísticas:

- El país con mayor y menor población.
- El promedio de población y de superficie.
- La cantidad de países por continente.

Todo esto se hace recorriendo la lista con bucles y acumulando los datos necesarios.

## 5. Ejecución del Programa (Capturas de Pantalla)

Figura 1. Menú principal del sistema con las opciones disponibles para gestionar los países:

```
gestion_paises.py
gestion_paises.py ...
def main():
    match opcion:
        case "1":
            agregar_pais(paises)
        case "2":
            mostrar_paises(paises)
        case "3":
            buscar_pais_menu(paises)
        case "4":
            actualizar_pais(paises)
        case "5":
            filtrar_continente(paises)
        case "6":
            filtrar_por_rango(paises)
        case "7":
            ordenar_paises(paises)
        case "8":
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\dahya\OneDrive\Escritorio\gestion_paises.python> & C:/Users/dahya/AppData/Local/Programs/Python/Python313/python.exe c:/users/dahya/OneDrive/Escritorio/gestion_paises.python/gestion_paises.py
```

```
*** Gestión de Países: Programa Iniciado ***

--- SISTEMA DE GESTIÓN DE PAÍSES ---
1. Agregar país
2. Ver todos los países
3. Buscar país por nombre
4. Actualizar población/superficie
5. Filtrar por continente
6. Filtrar por rango (Población/Superficie)
7. Ordenar países
8. Estadísticas
9. Salir (Guarda el CSV)
```

```
Seleccione opción: 1
```

LN 382, Col 11 Spaces: 4 UTF-8 CR LF () Python 3.13.7 Claude

Figura 2. Ejemplo de uso de la función *agregar\_pais()*, donde se agrega un nuevo país con todos sus datos:

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows a folder named "GESTION\_PAISES.PYTHON" containing "datos\_países.csv" and "gestion\_países.py".
- Code Editor:** Displays the content of "gestion\_países.py". The code defines a main function that filters continents, ranges, or orders countries, saves them to a CSV, and exits. It also handles invalid input and calls a statistics function.
- Terminal:** Shows a menu of options:
  - Actualizar población/superficie
  - Filtrar por continente
  - Filtrar por rango (Población/Superficie)
  - Ordenar países
  - Estadísticas
  - Salir (Guarda el CSV)A prompt "Seleccione opción: 1" is followed by a separator line and the instruction "... Agregar País ...".

Nombre: Irlanda  
Población (solo números): 5308000  
Superficie en km² (solo números): 70273  
Continente: Europa  
Registro exitoso: El país 'Irlanda' ya está agregado a la lista.  
Guardado Listo. Los cambios están a salvo en 'datos\_países.csv'.
- Bottom Status Bar:** Shows line 382, column 11, spaces 4, UTF-8 encoding, CR LF line endings, Python as the language, and file numbers 3.13.7 and Claude.

Figura 3. Lista completa de países almacenados en el sistema:

```

File Edit Selection View Go Run Terminal Help ⏎ ↻ 🔍 ... gestion_paises.py
EXPLORER ... gestion_paises.py > ...
GESTION_PAISES.PYTHON
  datos_paises.csv
  gestion_paises.py
344 def main():
345     filtrar_continente(paises)
346     case "6":
347         filtrar_por_rango(paises)
348     case "":
349         ordenar_paises(paises)
350     case "8":
351         mostrar_estadisticas(paises)
352     case "9":
353         guardar_paises_en_csv(paises, NOMBRE_ARCHIVO_CSV)
354         print("Cerrando sistema y guardando datos. ¡Adios!")
355         ejecutando = False
356     case _:
357         print("ERROR: Escribe un número válido del 1 al 9.")
358
359 if __name__ == "__main__":
360     main()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
8. Estadísticas
9. Salir (Guarda el CSV)
-----
Selecione opción: 2

--- Lista de Países ---
1. Argentina - Población: 45,376,763 hab. - Superficie: 2,780,400 km² - Continente: América
2. Brasil - Población: 213,993,437 hab. - Superficie: 8,515,767 km² - Continente: América
3. Chile - Población: 19,116,291 hab. - Superficie: 756,102 km² - Continente: América
4. España - Población: 47,351,567 hab. - Superficie: 505,990 km² - Continente: Europa
5. Francia - Población: 66,750,000 hab. - Superficie: 551,600 km² - Continente: Europa
6. Alemania - Población: 125,800,000 hab. - Superficie: 377,975 km² - Continente: Europa
7. Australia - Población: 25,920,000 hab. - Superficie: 7,692,024 km² - Continente: Oceanía
8. Irlanda - Población: 5,308,000 hab. - Superficie: 70,773 km² - Continente: Europa

--- SISTEMA DE GESTIÓN DE PAÍSES ---
1. Agregar país
2. Ver todos los países
Ln 382, Col 11 Spaces:4 UTF-8 CRLF {} Python 3.13.7 Claude

```

Figura 4. Resultado de una búsqueda parcial y filtrado por continente.:

```

File Edit Selection View Go Run Terminal Help ⏎ ↻ 🔍 ... gestion_paises.py
EXPLORER ... gestion_paises.py > ...
GESTION_PAISES.PYTHON
  datos_paises.csv
  gestion_paises.py
344 def main():
345     filtrar_continente(paises)
346     case "6":
347         filtrar_por_rango(paises)
348     case "":
349         ordenar_paises(paises)
350     case "8":
351         mostrar_estadisticas(paises)
352     case "9":
353         guardar_paises_en_csv(paises, NOMBRE_ARCHIVO_CSV)
354         print("Cerrando sistema y guardando datos. ¡Adios!")
355         ejecutando = False
356     case _:
357         print("ERROR: Escribe un número válido del 1 al 9.")
358
359 if __name__ == "__main__":
360     main()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
6. Filtrar por rango (Población/Superficie)
7. Ordenar países
8. Estadísticas
9. Salir (Guarda el CSV)
-----
Selecione opción: 5

--- Filtrar por continente ---
¿Qué continente quieres ver?: Europa
Países encontrados en 'Europa':
-> España, Población: 47,351,567
-> Francia, Población: 67,750,000
-> Irlanda, Población: 5,308,000

--- SISTEMA DE GESTIÓN DE PAÍSES ---
1. Agregar país
2. Ver todos los países
Ln 382, Col 11 Spaces:4 UTF-8 CRLF {} Python 3.13.7 Claude

```

Figura 5. Ejemplo de ordenamiento de países por población en orden descendente.

The screenshot shows a Python code editor interface. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar for 'gestion\_paises.py'. The left sidebar shows a project structure under 'GESTION\_PAISES.PYTHON' with files 'datos\_paises.csv' and 'gestion\_paises.py'. The main code area displays a portion of the 'main()' function:

```

344 def main():
    ...
    case "7":
        ordenar_paises(paises)
    ...

```

Below the code, a terminal window is open with the following text:

```

7. Ordenar países
8. Estadísticas
9. Salir (Guarda el CSV)
-----
Selecione opción: 7
    ... Ordenar Paises ...
1. Nombre (A-Z)
2. Nombre (Z-A)
3. Población (Menor a Mayor)
4. Población (Mayor a Menor)
5. Superficie (Menor a Mayor)
6. Superficie (Mayor a Menor)
Elige como quieres ordenar (1-6): 1
Ordenamiento completado: La lista va por nombre.
    ... Lista de Países ...
1. Argentina - Población: 45,376,763 hab. - Superficie: 2,780,400 km² - Continente: América
2. Australia - Población: 25,920,000 hab. - Superficie: 7,692,024 km² - Continente: Oceanía
3. Brasil - Población: 213,993,437 hab. - Superficie: 8,515,767 km² - Continente: América
4. Chile - Población: 17,200,000 hab. - Superficie: 7,275,000 km² - Continente: América
5. Egipto - Población: 47,351,267 hab. - Superficie: 506,900 km² - Continente: África
6. Francia - Población: 67,750,000 hab. - Superficie: 551,695 km² - Continente: Europa
7. Irlanda - Población: 5,308,000 hab. - Superficie: 70,273 km² - Continente: Europa
8. Japón - Población: 125,800,000 hab. - Superficie: 377,975 km² - Continente: Asia

```

The status bar at the bottom indicates 'In 382, Col 11 Spaces:4 UTF-8 CRLF {} Python 3.13.7 Claude'.

Figura 6. Estadísticas generales del sistema: país con mayor y menor población, promedios y cantidad de países por continente:

The screenshot shows a Python code editor interface, similar to Figure 6. The menu bar, sidebar, and code area are identical. The terminal window now displays the results of the 'filtrar\_por\_rango' function:

```

6. Filtrar por rango (Población/Superficie)
7. Ordenar países
8. Estadísticas
9. Salir (Guarda el CSV)
-----
Selecione opción: 8
    ... Estadísticas ...
Total de países registrados: 8
Total de países registrados: 8
País con más gente: **Brasil** (213,993,437 hab.)
País con menos gente: **Irlanda** (5,308,000 hab.)
Promedio de población: 68,826,996 hab.
Promedio de superficie: 2,656,278 km²

Países por Continente:
- América: 3 países
- Oceanía: 1 países
- Europa: 3 países
- Asia: 1 países
    ... SISTEMA DE GESTIÓN DE PAÍSES ...
1. Agregar país

```

The status bar at the bottom indicates 'In 382, Col 11 Spaces:4 UTF-8 CRLF {} Python 3.13.7 Claude'.

## 5. Conclusiones

Durante el desarrollo aplicamos varias buenas prácticas de programación, como mantener el código modular, usar nombres claros para las variables y agregar comentarios que ayuden a entender qué hace cada parte del programa. El programa fue probado con distintos casos (agregar, modificar, buscar y filtrar países) y funcionó correctamente en todos.

- **Conexión entre estructuras:** Aprendimos que las listas son muy útiles para manejar varios registros y que los diccionarios permiten guardar los datos de cada país de una forma más clara y ordenada.
- **Persistencia y validaciones:** Al usar archivos CSV y validar los datos manualmente, logramos que el programa sea más sólido y fácil de usar.
- **Algoritmia:** Implementar el algoritmo de ordenamiento nos ayudó a entender mejor cómo funcionan los bucles y las comparaciones dentro de un proceso lógico.
- **Trabajo en equipo:** Dividir las tareas y revisar el código entre los integrantes nos permitió mejorar la calidad final y aprender de los errores de forma conjunta.

En conclusión, el proyecto fue una buena experiencia para aplicar todo lo aprendido en Programación 1.

## 6. Fuentes Bibliográficas

- Videos del Aula Virtual de Programación I – UTN San Nicolás (2025).
- Downey, A. (2015). Think Python: Cómo pensar como un científico informático. Versión en español e inglés.
- Sweigart, A. (2020). Automate the Boring Stuff with Python. No Starch Press.
- Python Software Foundation. (2025). Documentación oficial de Python 3.12. Disponible en:  
<https://docs.python.org/3/>.

*Montenegro, Dahyana y Domínguez, Matías. (2025). Video Tutorial: Explicación y Demostración del Trabajo Práctico Integrador (TPI) de Programación I. UTN San Nicolás. Disponible en:  
<https://www.youtube.com/watch?v=AQ6DC8qH3ps>*