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

11 de noviembre de 2025

Tabla de Contenido

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:

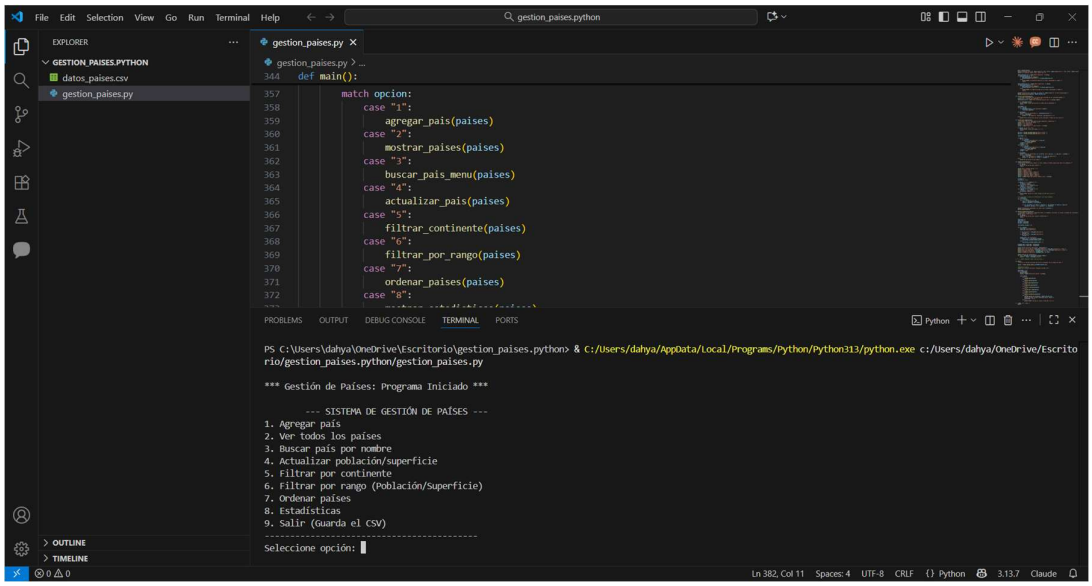


Figura 2. Ejemplo de uso de la función *agregar_pais()*, donde se agrega un nuevo país con todos sus datos:

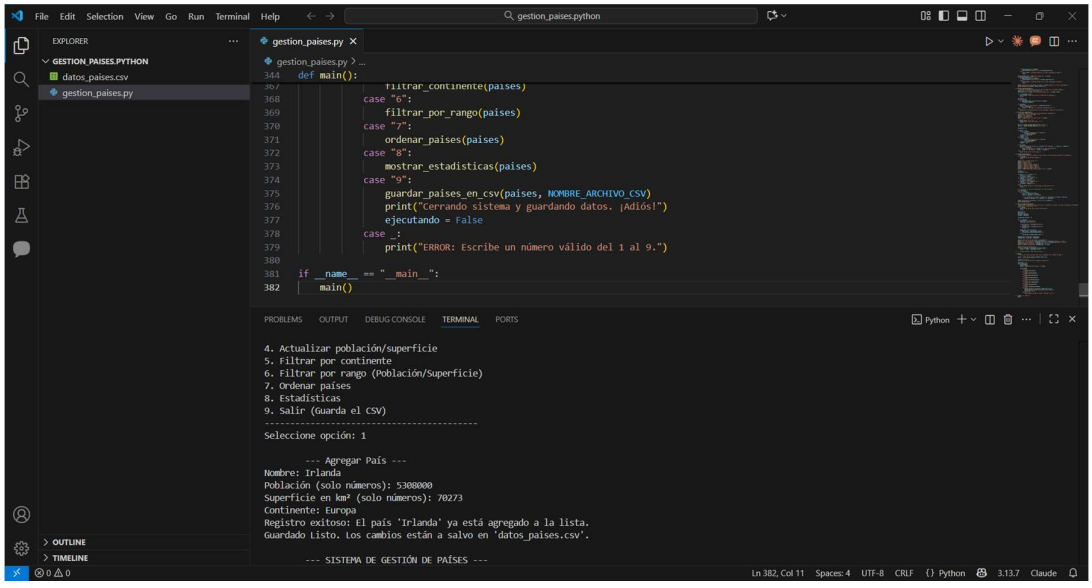


Figura 3. Lista completa de países almacenados en el sistema:

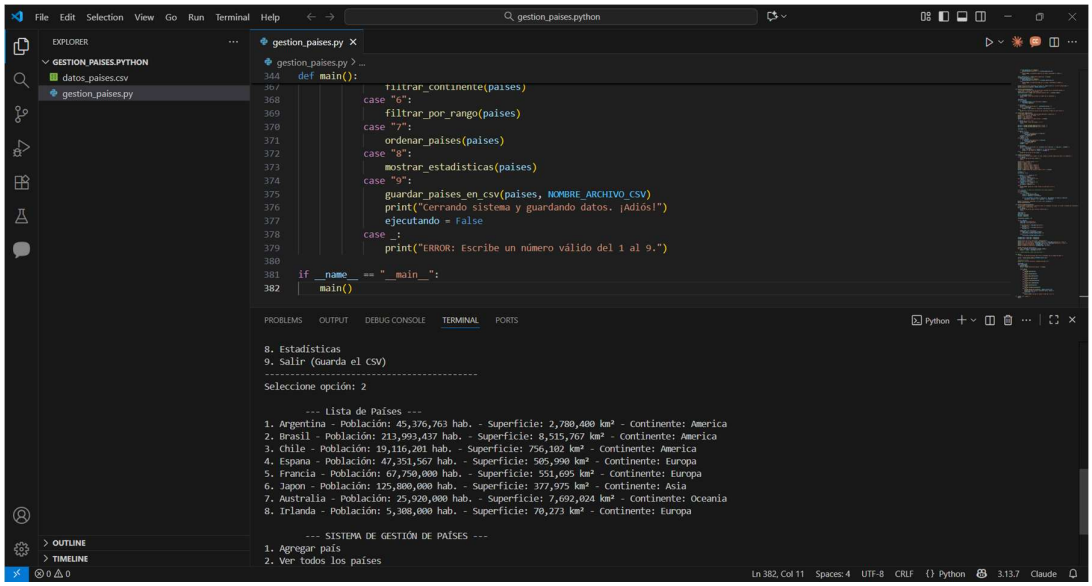


Figura 4. Resultado de una búsqueda parcial y filtrado por continente.:

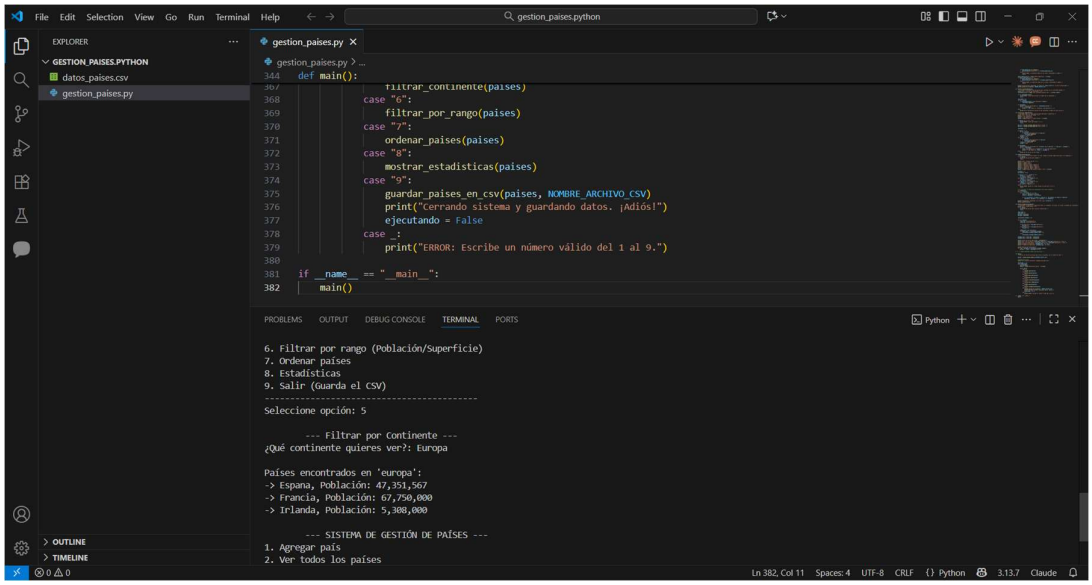


Figura 5. Ejemplo de ordenamiento de países por población en orden descendente.

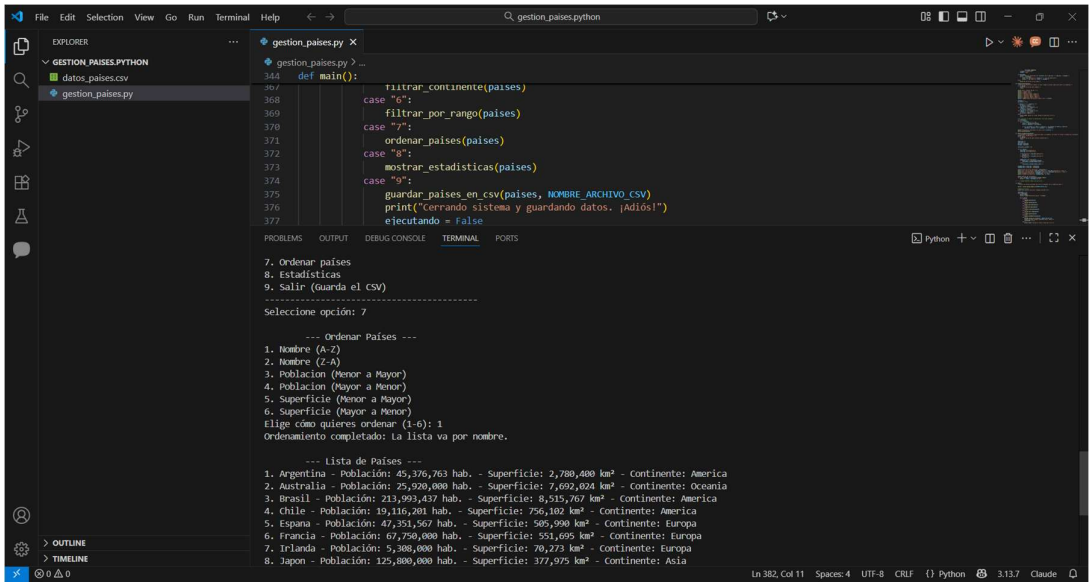
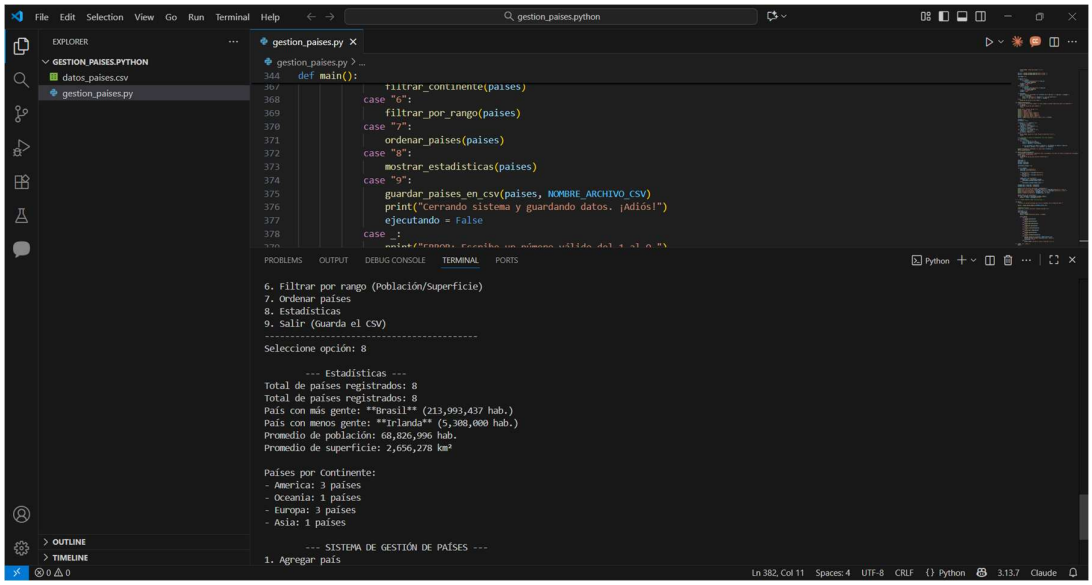


Figura 6. Estadísticas generales del sistema: país con mayor y menor población, promedios y cantidad de países por continente:



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