1. Informe Teórico

Institución: Universidad Tecnológica Nacional
Materia: Programación 1
Trabajo Integrador
Integrantes:

- Chiara Diaz
- FERNANDO GERMÁN LUDUEÑA GÓMEZ

Docente: Brian Esteban Lara Campos
Año lectivo: 2025

1.2. Índice

1. Introducción
2. Objetivos del sistema
3. Descripción general del programa
4. Diseño del sistema
   4.1. Diagrama general
   4.2. Módulos y responsabilidades
5. Desarrollo y tecnologías utilizadas
   5.1. Uso de archivos CSV
   5.2. Funciones y modularización
   5.3. Uso de diccionarios, listas y lambdas
6. Marco teórico
7. Pruebas y capturas de pantalla
8. Conclusiones
9. Bibliografía

---

1. Introducción

En este trabajo práctico desarrollamos un sistema en Python para la gestión de información sobre países utilizando archivos CSV.
El sistema permite agregar, actualizar, buscar, filtrar y ordenar países, además de mostrar estadísticas básicas.
El objetivo principal fue aplicar los conceptos vistos en clase: manejo de archivos,

estructuras de datos (listas y diccionarios), funciones, funciones lambda y validación de entradas.

2. Objetivos del sistema

- Implementar un programa modular en Python que gestione información de países.
- Utilizar un archivo CSV para almacenar y persistir los datos.
- Aplicar funciones, docstrings y comentarios para mejorar la claridad del código.
- Utilizar funciones lambda para ordenar y procesar datos.
- Practicar el diseño de un menú interactivo para el usuario.

3. Descripción general del programa

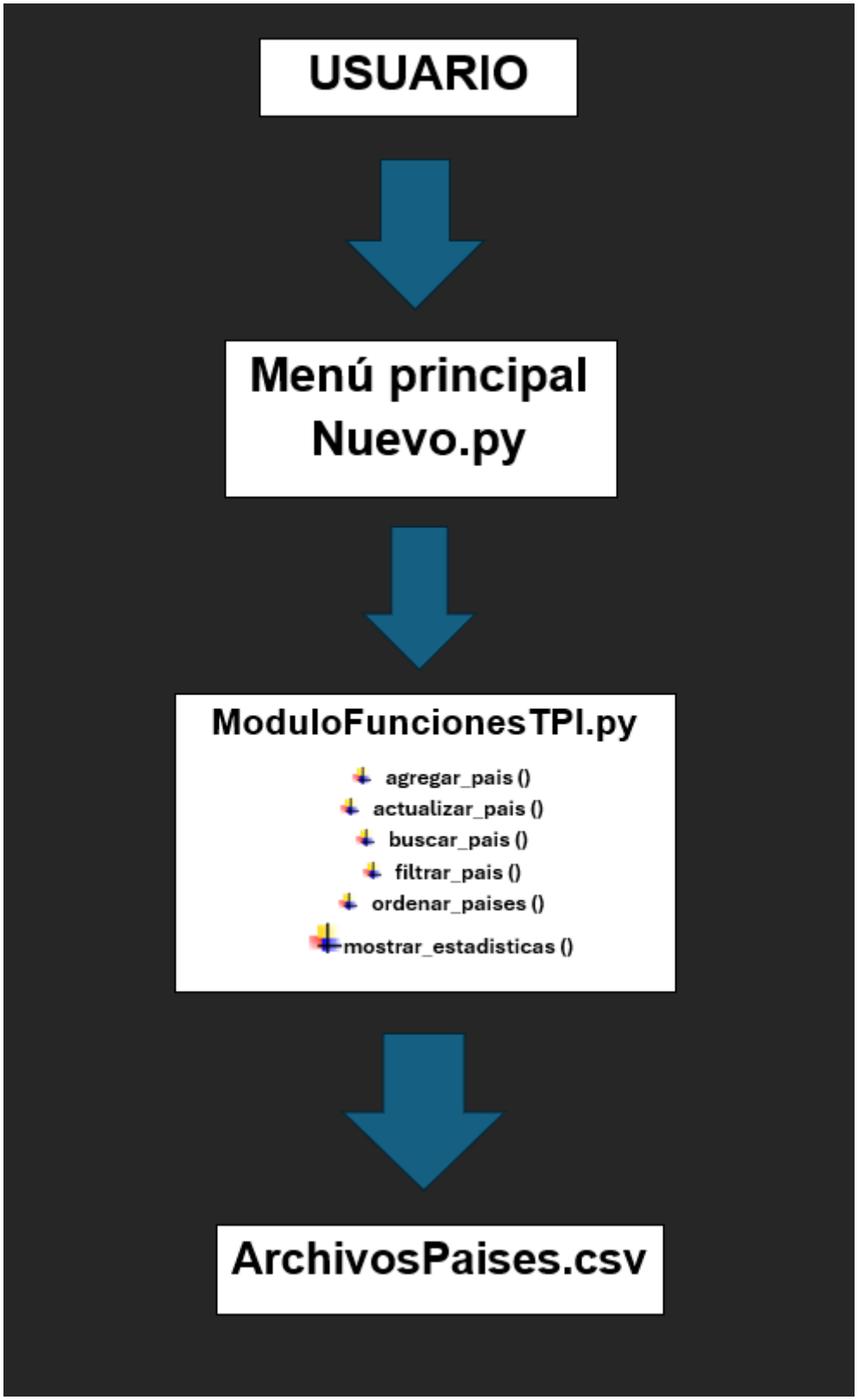
El sistema se compone de dos módulos principales:

- **Nuevo.py**: contiene el menú principal. Desde aquí el usuario interactúa con el sistema, elige las opciones y se llaman a las funciones correspondientes.
- **FuncionesTPI.py**: contiene la lógica del programa. Incluye las funciones para agregar, actualizar, buscar, filtrar, ordenar países y mostrar estadísticas, trabajando sobre el archivo paises.csv.

El usuario interactúa desde la consola, y todas las operaciones se reflejan en el archivo CSV, que actúa como base de datos simple.

4. Diseño del sistema

4.1. Diagrama visual (esquema)



El diseño se basa en la separación entre interfaz de usuario y lógica de negocio. Nuevo.py se encarga de mostrar el menú y capturar las opciones, mientras que

FuncionesTPI.py implementa las operaciones concretas sobre los datos, que se almacenan en paises.csv.

4.2. Módulos y responsabilidades

- **Nuevo.py**
    - Muestra el menú principal.
    - Valida la opción ingresada por el usuario.
    - Llama a la función correspondiente de FuncionesTPI.py.
- **FuncionesTPI.py**
    - Verifica la existencia del archivo paises.csv y lo inicializa.
    - Define las funciones:
        - mostrar_paises()
        - agregar_pais()
        - actualizar_pais()
        - buscar_pais()
        - filtrar_pais()
        - ordenar_paises()
        - mostrar_estadisticas()

5. Desarrollo y tecnologías utilizadas

5.1. Uso de archivos CSV

Utilizamos el archivo paises.csv para almacenar los datos de manera persistente. Para manejarlo, usamos el módulo estándar csv de Python con DictReader y DictWriter, lo que nos permite trabajar con diccionarios en lugar de listas de posiciones, facilitando la lectura del código.

5.2. Funciones y modularización

Definimos funciones específicas para cada operación del menú (agregar, actualizar, buscar, etc.).
Esto mejora la organización del código, evita repeticiones y permite que el programa principal (Nuevo.py) sea más simple y legible.

5.3. Estructuras de datos y lambdas

- Utilizamos **listas** para mantener en memoria el conjunto de países leídos desde el archivo.

- Cada país se representa como un **diccionario** con campos: nombre, población, superficie y continente.
- Empleamos **funciones lambda** junto con sort, max y min para ordenar y obtener el país con mayor y menor población, lo cual simplifica mucho el código.

6. Marco teórico

En el desarrollo del sistema de gestión de países se aplicaron diversas herramientas y conceptos de Python cuya elección puede justificarse directamente a partir de los fundamentos teóricos presentados en *Think Python* de Allen Downey. Cada estructura, módulo y decisión de diseño responde a motivos concretos relacionados con la abstracción, la modularidad, la claridad del código y la prevención de errores, principios que el libro establece como esenciales en la programación.

En primer lugar, el programa se diseñó de manera modular utilizando funciones. Downey sostiene que una función es una "secuencia nombrada de instrucciones que realiza una computación" y destaca que dividir un problema grande en funciones más pequeñas permite **manejar la complejidad**, mejorar la **legibilidad** y evitar la **repetición de código**. Siguiendo esta teoría, el sistema separa las responsabilidades en dos archivos: Nuevo.py como menú principal y FuncionesTPI.py como módulo que contiene la lógica del programa. Esta decisión es superior a escribir todo el código en un único bloque secuencial, ya que facilita el mantenimiento y la ampliación futura del sistema, coincidiendo con el principio de *abstracción* presentado por Downey.

En cuanto al almacenamiento de datos, se eligió un archivo **CSV** en lugar de un archivo de texto plano (TXT) o una base de datos. Downey explica que los archivos de texto simples son útiles, pero cuando la información tiene una estructura tabular es preferible utilizar formatos organizados que eviten realizar parsing manual. El formato CSV permite almacenar los países en filas y columnas de forma clara y uniforme, mientras que un archivo TXT requeriría separar manualmente los valores, lo cual aumenta las posibilidades de error y contradice la recomendación del autor de evitar "hacer a mano lo que Python puede automatizar mediante sus bibliotecas". Además, el módulo csv de Python permite leer y escribir datos como diccionarios, lo que simplifica la lógica y reduce el riesgo de inconsistencias.

La elección de **diccionarios** como representación de cada país también se fundamenta en la teoría del libro. Downey explica que los diccionarios son estructuras ideales cuando los datos poseen una relación clave–valor,

permitiendo acceder a la información de forma directa y semánticamente clara. En el sistema, un país tiene atributos definidos como nombre, población, superficie y continente. Al utilizar diccionarios, estos campos pueden identificarse por nombre, lo que es mucho más legible que usar índices numéricos en listas o tuplas. Además, a diferencia de las tuplas, los diccionarios permiten modificar valores, algo indispensable para funciones como la actualización de registros. Por estas razones, esta estructura de datos es la más adecuada según la teoría propuesta por Downey.

El uso de **listas** para almacenar el conjunto de países también está alineado con los fundamentos del autor. Downey señala que las listas son estructuras flexibles que permiten recorrer elementos, ordenarlos y filtrarlos, lo cual es coherente con las operaciones que realiza el sistema (filtros por población o continente, ordenamientos por diferentes criterios, recorridos completos, etc.). Alternativas como los conjuntos (*sets*) no permiten mantener un orden ni realizar operaciones de clasificación de forma eficiente, mientras que otras estructuras más complejas resultan innecesarias para este tipo de aplicación.

Otra herramienta importante utilizada en el programa es el uso de **funciones lambda** para ordenar y obtener estadísticas. Según Downey, las lambdas permiten definir pequeñas funciones anónimas "sobre la marcha", especialmente cuando se van a usar una sola vez, evitando crear funciones completas solo para un criterio de ordenamiento. Gracias a esto, operaciones como ordenar países por nombre, población o superficie pueden implementarse de manera clara y concisa. Frente a la alternativa de escribir una función completa por cada criterio, el uso de lambdas es más directo, limpio y coherente con el paradigma de funciones como objetos de primera clase que desarrolla el libro.

El sistema implementa un menú interactivo basado en un bucle while True, lo cual también está respaldado por la teoría de Downey sobre las estructuras de control. El autor explica que los bucles son apropiados cuando se requiere repetir una acción hasta que el usuario decida detenerse, mientras que la recursividad —aunque útil en problemas matemáticos— no es adecuada para interacción continua, ya que genera acumulación de llamadas y dificultades en el control del flujo. Por lo tanto, la estructura basada en un bucle infinito controlado por condiciones internas es la opción más estable y segura para un menú repetitivo.

Respecto a la validación de datos, Downey resalta en el capítulo de *debugging* que la mejor estrategia para evitar errores es prevenirlos antes de que ocurran. Por esta razón, el sistema valida que la población y la superficie sean numéricas antes de procesarlas, evitando que la conversión a entero o las operaciones posteriores generen fallos. Esto es preferible a permitir que el error ocurra y luego manejar

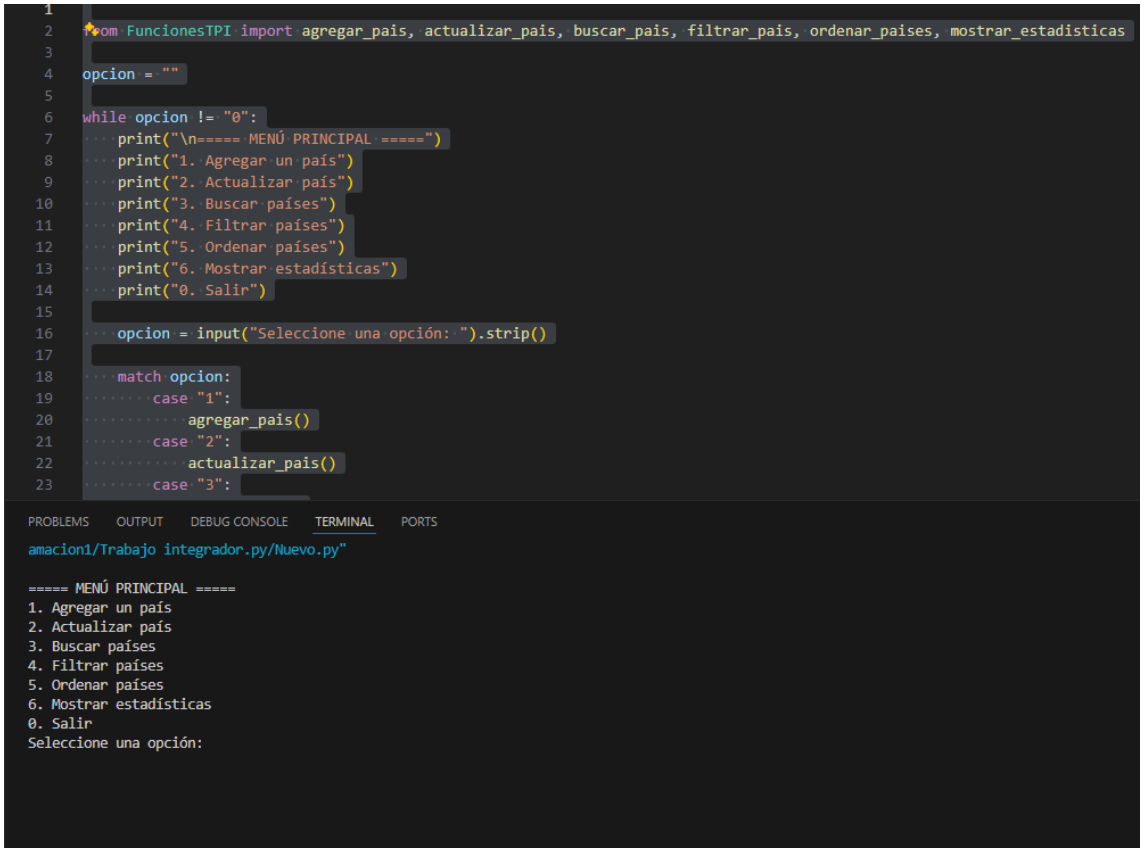
excepciones, ya que la prevención produce programas más robustos y predecibles, en línea con las buenas prácticas recomendadas por el autor.

Finalmente, el sistema utiliza entrada y salida por consola con input() y print(). Downey enfatiza que esta es la forma más simple y efectiva de comenzar a resolver problemas, ya que permite concentrarse en la lógica sin distraerse con el diseño de interfaces gráficas, las cuales requieren conocimientos adicionales y no aportan valor al objetivo pedagógico de comprender la estructura y funcionamiento de los programas.

En conjunto, todas estas decisiones reflejan un diseño coherente con los principios fundamentales del libro *Think Python*. Cada herramienta elegida — funciones, modularización, CSV, diccionarios, listas, lambdas, bucles y validación de entrada— no solo cumple con los requerimientos del trabajo, sino que se justifica teóricamente frente a alternativas menos adecuadas según la propia bibliografía.

7. Pruebas y capturas de pantalla

- Menú principal en ejecución.



- Ejemplo de **agregar país** (entrada por teclado + resultado en el listado).

```
===== MENÚ PRINCIPAL =====
1. Agregar un país
2. Actualizar país
3. Buscar países
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
0. Salir
Seleccione una opción: 1
-----AGREGAR UN PAÍS-----
Nombre del país: China
Cantidad de habitantes: 1400000000
Superficie del país: 9597000
Continente: Asia
País China agregado exitosamente.
```

- Ejemplo de **búsqueda** de país.

```
===== MENÚ PRINCIPAL =====
1. Agregar un país
2. Actualizar país
3. Buscar países
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
0. Salir
Seleccione una opción: 3
-----Buscar País-----
Ingrese una parte o todo el nombre del país: Arg
Se encontraron 1 resultado(s):

Argentina - America | 47000000 hab | 2780000 km²
```

- Ejemplo de **filtro** por continente o por rango de población.

```
===== MENÚ PRINCIPAL =====
1. Agregar un país
2. Actualizar país
3. Buscar países
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
0. Salir
Seleccione una opción: 4
-----Filtrar País-----
1. Por continente
2. Por rango de población
3. Por rango de superficie
Elija una opción: 3
Superficie mínima (KM): 176000
Superficie máxima (KM): 756000
Se encontraron 9 resultado(s):

Chile - America | 19200000 hab | 756000 km²
Uruguay - America | 3500000 hab | 176000 km²
Espana - Europa | 47500000 hab | 505000 km²
Alemania - Europa | 83100000 hab | 357000 km²
Francia - Europa | 67000000 hab | 643000 km²
Italia - Europa | 59500000 hab | 301000 km²
Reino Unido - Europa | 67000000 hab | 243000 km²
Japon - Asia | 125000000 hab | 378000 km²
Nueva Zelanda - Oceania | 5200000 hab | 268000 km²
```

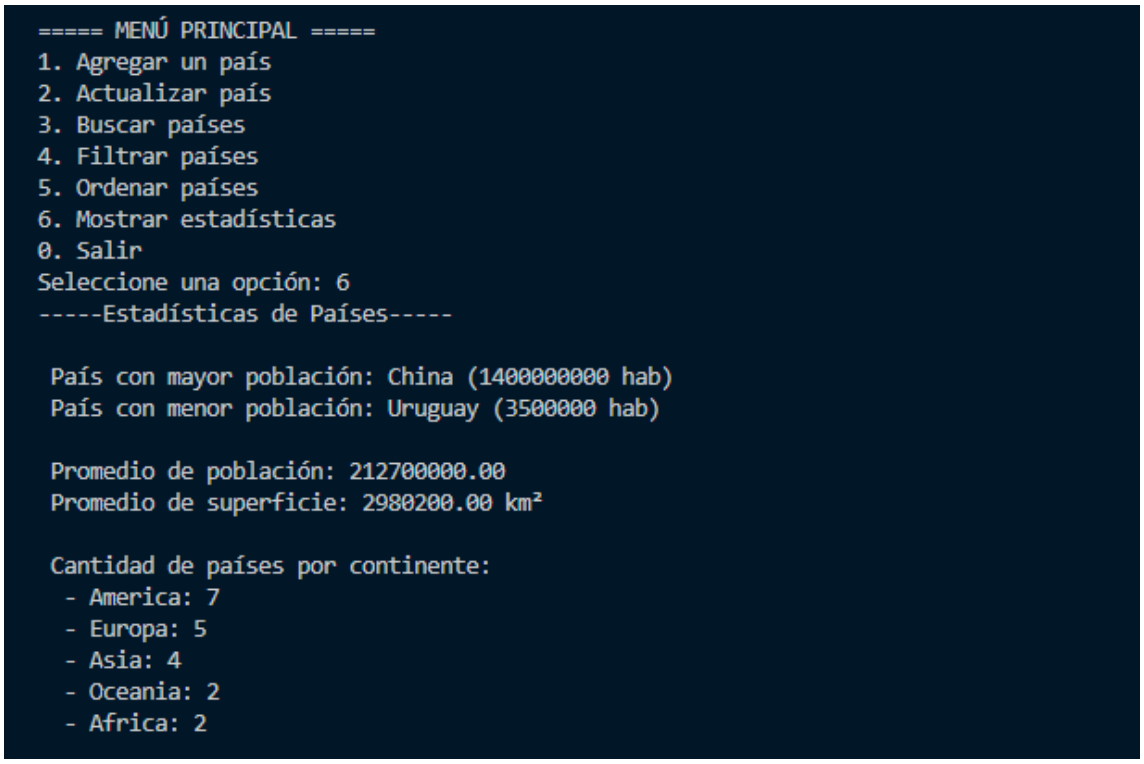
```
===== MENÚ PRINCIPAL =====
1. Agregar un país
2. Actualizar país
3. Buscar países
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
0. Salir
Seleccione una opción: 4
-----Filtrar País-----
1. Por continente
2. Por rango de población
3. Por rango de superficie
Elija una opción: 2
Población mínima: 3500000
Población máxima: 19200000
Se encontraron 3 resultado(s):

Chile - America | 19200000 hab | 756000 km²
Uruguay - America | 3500000 hab | 176000 km²
Nueva Zelanda - Oceania | 5200000 hab | 268000 km²
```

- Ejemplo de **ordenamiento**

```
===== MENÚ PRINCIPAL =====
1. Agregar un país
2. Actualizar país
3. Buscar países
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
0. Salir
Seleccione una opción: 5
-----Ordenar Países-----
1. Por nombre
2. Por población
3. Por superficie
Elija una opción: 2
¿Orden Ascendente (A) o Descendente (D)? a

Países ordenados:

Uruguay - America | 3500000 hab | 176000 km²
Nueva Zelanda - Oceania | 5200000 hab | 268000 km²
Chile - America | 19200000 hab | 756000 km²
Australia - Oceania | 26000000 hab | 7692000 km²
Canada - America | 38000000 hab | 9985000 km²
Argentina - America | 47000000 hab | 2780000 km²
Espana - Europa | 47500000 hab | 505000 km²
Corea del Sur - Asia | 52000000 hab | 100000 km²
Italia - Europa | 59500000 hab | 301000 km²
Sudafrica - Africa | 60000000 hab | 1221000 km²
Francia - Europa | 67000000 hab | 643000 km²
Reino Unido - Europa | 67000000 hab | 243000 km²
Alemania - Europa | 83100000 hab | 357000 km²
Egipto - Africa | 102000000 hab | 1001000 km²
Japon - Asia | 125000000 hab | 378000 km²
Mexico - America | 127000000 hab | 1964000 km²
Brasil - America | 214000000 hab | 8516000 km²
Estados Unidos - America | 331000000 hab | 9834000 km²
India - Asia | 1380000000 hab | 3287000 km²
China - Asia | 1400000000 hab | 9597000 km²
```

```
===== MENÚ PRINCIPAL =====
1. Agregar un país
2. Actualizar país
3. Buscar países
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
0. Salir
Seleccione una opción: 5
-----Ordenar Países-----
1. Por nombre
2. Por población
3. Por superficie
Elija una opción: 1
¿Orden Ascendente (A) o Descendente (D)? a

Países ordenados:

Alemania - Europa | 83100000 hab | 357000 km²
Argentina - America | 47000000 hab | 2780000 km²
Australia - Oceania | 26000000 hab | 7692000 km²
Brasil - America | 214000000 hab | 8516000 km²
Canada - America | 38000000 hab | 9985000 km²
Chile - America | 19200000 hab | 756000 km²
China - Asia | 1400000000 hab | 9597000 km²
Corea del Sur - Asia | 52000000 hab | 100000 km²
Egipto - Africa | 102000000 hab | 1001000 km²
Espana - Europa | 47500000 hab | 505000 km²
Estados Unidos - America | 331000000 hab | 9834000 km²
Francia - Europa | 67000000 hab | 643000 km²
India - Asia | 1380000000 hab | 3287000 km²
Italia - Europa | 59500000 hab | 301000 km²
Japon - Asia | 125000000 hab | 378000 km²
Mexico - America | 127000000 hab | 1964000 km²
Nueva Zelanda - Oceania | 5200000 hab | 268000 km²
Reino Unido - Europa | 67000000 hab | 243000 km²
Sudafrica - Africa | 60000000 hab | 1221000 km²
Uruguay - America | 3500000 hab | 176000 km²
```

- Ejemplo de **estadísticas** mostradas en pantalla.



8. Conclusiones

El desarrollo de este sistema nos permitió aplicar conceptos de programación estructurada en Python, trabajar con archivos CSV y practicar la separación entre interfaz y lógica del programa.
Además, comprendimos la importancia de la validación de datos, la documentación del código mediante docstrings y comentarios, y el uso de funciones lambda para simplificar operaciones de ordenamiento y cálculo de estadísticas.

9. Bibliografía

Ejemplo:

- Downey, A. (2015). *Think Python: How to Think Like a Computer Scientist*.
- Documentación oficial de Python: módulo csv.
- Apuntes de cátedra de la materia