

Trabajo Práctico Integrador Programación

[Marco Teórico]

Alumnos participantes

Calvo, Belén y Scherer, Marcelo

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Programación I

Docente Titular

Rigoni, Cintia y Hualpa, Ramiro

31 de octubre del 2025

Tabla de contenido

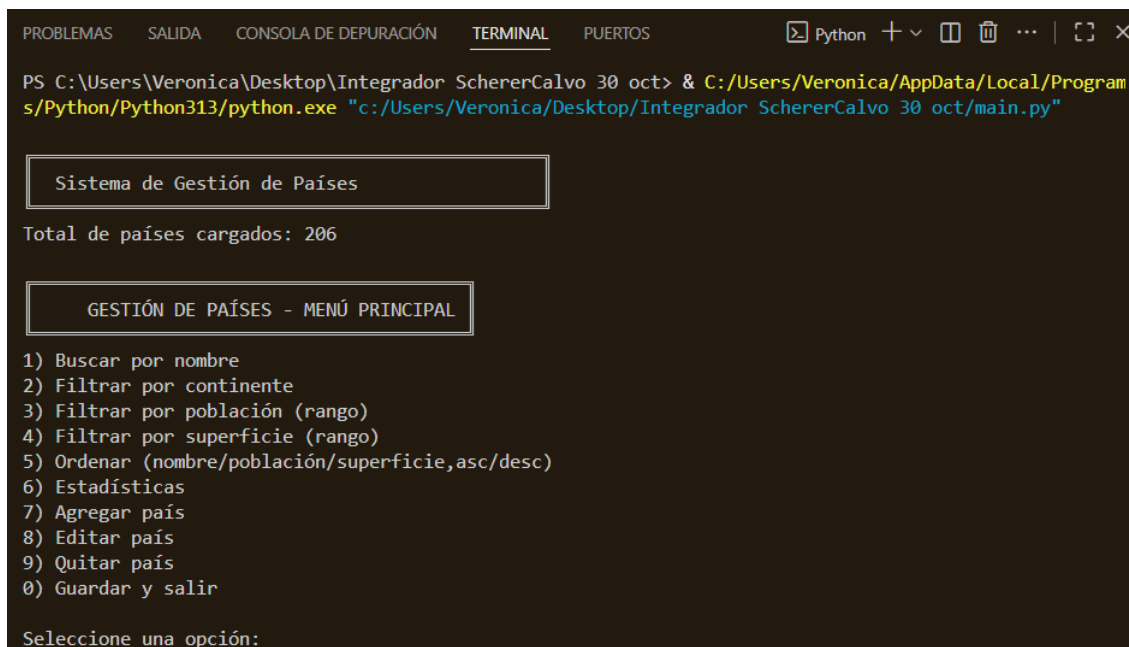
Introducción	3
Objetivos	4
¿Qué es una lista?	5
Diccionarios	6
Funciones y modularización	8
Condicionales	10
Bucles	12
Ordenamiento	14
Estadísticas básicas	17
Archivos CSV	20
Operaciones CRUD	24
Conclusiones	29
Reflexión final	32
Referencia	33

Trabajo Práctico Integrador

1. Introducción

Este proyecto consiste en un sistema para gestionar información de países usando Python. El objetivo es aplicar lo que aprendimos en Programación 1: listas, diccionarios, funciones, condicionales, bucles, archivos CSV y cálculos básicos.

El sistema permite buscar países, filtrarlos por diferentes criterios, ordenarlos, editarlos y ver estadísticas sobre población y superficie.



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  Python + - [ ] [ ] ... [ ] [ ] x

PS C:\Users\Veronica\Desktop\Integrador SchererCalvo 30 oct> & C:/Users/Veronica/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Veronica/Desktop/Integrador SchererCalvo 30 oct/main.py"

Sistema de Gestión de Países

Total de países cargados: 206

GESTIÓN DE PAÍSES - MENÚ PRINCIPAL

1) Buscar por nombre
2) Filtrar por continente
3) Filtrar por población (rango)
4) Filtrar por superficie (rango)
5) Ordenar (nombre/población/superficie,asc/desc)
6) Estadísticas
7) Agregar país
8) Editar país
9) Quitar país
0) Guardar y salir

Seleccione una opción:
```

Objetivos

-Gestión Completa de Datos (CRUD)

Implementar un sistema completo de manejo de información que permita:

- **Crear:** Agregar nuevos países con validación de datos
- **Leer:** Cargar y visualizar información desde archivo CSV
- **Actualizar:** Modificar la lista de países en memoria
- **Eliminar:** Quitar países del sistema

-Búsqueda y Análisis de Información

Proporcionar herramientas flexibles para consultar y analizar datos mediante:

- **Búsqueda por texto:** Coincidencia parcial insensible a acentos y mayúsculas
- **Filtros múltiples:** Por continente, rangos de población y superficie
- **Ordenamiento:** Por diferentes criterios (nombre, población, superficie) en orden ascendente o descendente
- **Estadísticas:** Cálculo de máximos, mínimos, promedios y distribución por continentes

-Programación Modular y Estructurada

Demostrar buenas prácticas de programación mediante:

- **Modularización:** Separación de responsabilidades en 6 módulos distintos
- **Reutilización:** Funciones genéricas y utilidades compartidas
- **Mantenibilidad:** Código organizado, documentado y fácil de modificar
- **Sintaxis moderna:** Uso de match-case (Python 3.10+)
- **Interfaz intuitiva:** Menús claros con navegación fluida y validación de entradas

2. ¿Qué es una lista?

Una lista en Python es una colección ordenada que puede guardar varios elementos. Se puede modificar (agregar o sacar elementos) mientras el programa está corriendo. Las listas mantienen el orden en que agregamos los elementos y podemos acceder a ellos por su posición usando números (índices).

Las características principales de las listas son:

- Son ordenadas: los elementos tienen una posición específica
- Son mutables: podemos cambiarlas después de crearlas
- Pueden contener cualquier tipo de dato
- Se pueden recorrer con bucles

Cómo lo usamos en el proyecto

En nuestro proyecto, la lista principal se llama países y guarda TODOS los países del sistema. Cada país es un diccionario, entonces tenemos una lista de diccionarios.

Cuando el programa arranca, carga el archivo CSV y va agregando cada país válido a esta lista. Durante la ejecución, si el usuario agrega un nuevo país, lo sumamos a la lista con `append()`. Si elimina uno, lo sacamos de la lista.

Las operaciones con listas que usamos son:

- **Agregar elementos:** cuando cargamos países del CSV o el usuario agrega uno nuevo
- **Contar elementos:** para saber cuántos países hay en total usando `len()`
- **Recorrer:** con bucles `for` para buscar, filtrar u ordenar países
- **Modificar toda la lista:** cuando eliminamos un país, reemplazamos la lista completa

Las listas fueron fundamentales porque necesitábamos una estructura dinámica que creciera o se achicara según las acciones del usuario, sin tener que definir un tamaño fijo al principio.

3. DICCIONARIOS

¿Qué es un diccionario?

Un diccionario es como una agenda o un directorio: tiene pares de clave-valor. En vez de buscar elementos por su posición numérica (como en las listas), los buscamos por un nombre descriptivo llamado "clave".

Por ejemplo, en una agenda buscamos un teléfono por el nombre de la persona (clave), no por "la tercera entrada de la página 5". Esto hace que el código sea mucho más claro y fácil de entender.

Las características de los diccionarios son:

- Usan pares clave-valor

- Las claves deben ser únicas
- Son mutables (se pueden modificar)
- El acceso a valores por clave es muy rápido

Cómo representamos países

Cada país en nuestro sistema se representa como un diccionario con 5 claves:

- **nombre:** el nombre del país (texto)
- **poblacion:** cantidad de habitantes (número entero)
- **superficie:** área territorial en kilómetros cuadrados (número entero)
- **continente:** continente al que pertenece (texto)
- **reconocido_onu:** si está reconocido por la ONU o no (verdadero/falso)

Elegimos usar diccionarios porque:

- Es más claro escribir pais["nombre"] que tener que recordar que el nombre está en la posición 0
- Podemos agregar campos nuevos en el futuro sin romper el código existente
- Coincide perfectamente con cómo funciona el archivo CSV: las columnas del CSV se mapean a las claves del diccionario

Cuando mostramos un país o calculamos estadísticas, accedemos a sus datos usando las claves, lo que hace que el código sea legible y fácil de mantener.

4. FUNCIONES Y MODULARIZACIÓN

¿Qué es una función?

Una función es un bloque de código reutilizable que hace UNA tarea específica. Es como una máquina que recibe información (parámetros), hace algo con ella, y devuelve un resultado.

La idea principal es dividir un programa grande en pedazos más chicos que sean más fáciles de entender, arreglar y reutilizar. Esto se llama "modularización".

Las ventajas de usar funciones son:

- **Reutilización:** podemos usar la misma función varias veces
- **Organización:** cada función tiene una responsabilidad clara
- **Mantenimiento:** si hay un error, sabemos exactamente dónde buscar
- **Trabajo en equipo:** cada persona puede trabajar en funciones diferentes

Cómo organizamos el código

Dividimos todo el proyecto en varios archivos según su función específica:

`funciones.py`: Se encarga de cargar y guardar el archivo CSV. Contiene funciones para abrir el archivo, leer los datos, validarlos y escribirlos de vuelta.

`funciones_operaciones.py`: Tiene todas las operaciones principales: buscar países por nombre, filtrar por continente, filtrar por rangos de población o superficie, ordenar por diferentes criterios, y calcular estadísticas.

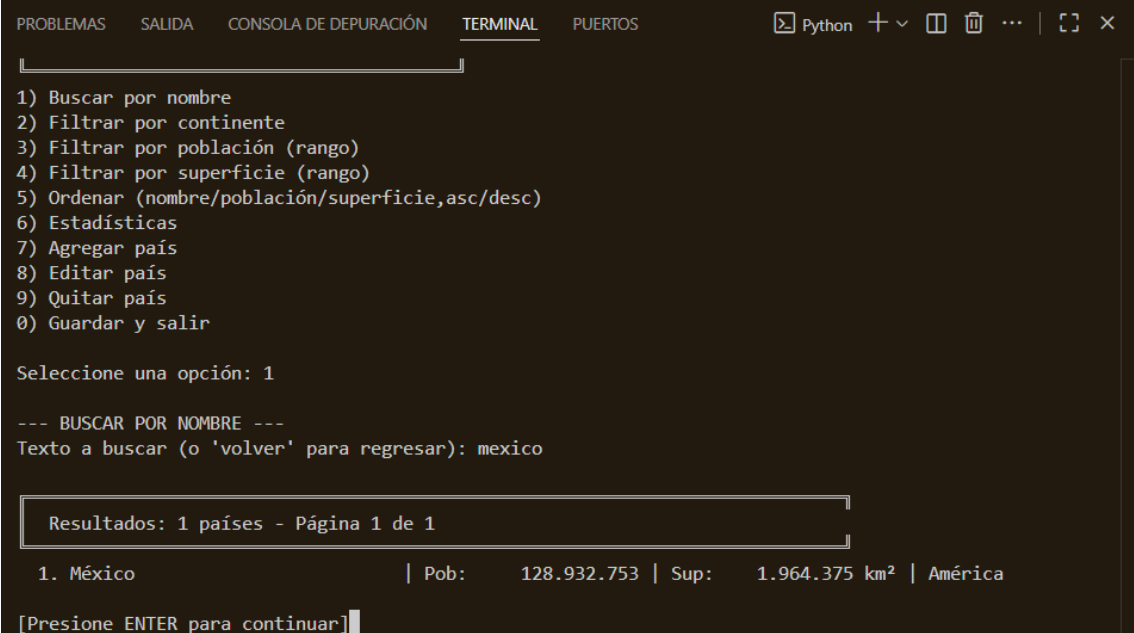
`funciones_altas_bajas.py`: Maneja agregar, editar y eliminar países. Incluye validaciones para evitar duplicados y verificar que los datos sean correctos.

`mostrar_lista.py`: Se encarga de mostrar los resultados con paginación, para que cuando haya muchos países no se llene toda la pantalla.

`menu.py`: Muestra el menú principal con todas las opciones disponibles.

`utilidades.py`: Contiene funciones auxiliares de validación, como pedir números enteros al usuario, pausar la pantalla, o preguntar si quiere hacer otra operación.

Esta organización hizo que el desarrollo fuera mucho más manejable. Por ejemplo, si la búsqueda no funcionaba bien, sabíamos que teníamos que revisar solo `funciones_operaciones.py`, no todo el proyecto. Además, nos permitió trabajar en paralelo: uno podía mejorar la visualización mientras el otro arreglaba el sistema de filtros.



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  Python + - [ ] [X] ... [ ] [X]
```

```
1) Buscar por nombre
2) Filtrar por continente
3) Filtrar por población (rango)
4) Filtrar por superficie (rango)
5) Ordenar (nombre/población/superficie,asc/desc)
6) Estadísticas
7) Agregar país
8) Editar país
9) Quitar país
0) Guardar y salir

Seleccione una opción: 1

--- BUSCAR POR NOMBRE ---
Texto a buscar (o 'volver' para regresar): mexico

Resultados: 1 países - Página 1 de 1

1. México | Pob: 128.932.753 | Sup: 1.964.375 km² | América

[Presione ENTER para continuar]
```

5. CONDICIONALES

¿Qué son los condicionales?

Los condicionales son estructuras que le permiten al programa tomar decisiones: "si pasa esto, hacé aquello; si no, hacé otra cosa". Son fundamentales para que el programa se comporte de manera inteligente según diferentes situaciones.

Python tiene varias formas de escribir condicionales:

- **if:** ejecuta código solo si se cumple una condición
- **elif:** verifica otra condición si la primera fue falsa
- **else:** ejecuta código si ninguna condición anterior fue verdadera
- **match-case:** la versión moderna para cuando hay muchas opciones (disponible desde Python 3.10)

Cómo los usamos en el proyecto

Usamos condicionales en muchas partes del sistema:

En validaciones: Antes de agregar un país, verificamos que el nombre no esté vacío, que la población sea positiva, que la superficie sea válida. Si algo está mal, rechazamos los datos.

En los menús principales: Usamos la estructura match-case porque teníamos **10 opciones** en el menú (del 0 al 9) y quedaba mucho más claro que usar muchos if-elif-else seguidos. Cada opción del menú ejecuta una función diferente.

En el submenú de estadísticas: También usamos match-case para que el usuario elija qué estadística quiere ver: mayor población, menor población, promedios, distribución por continentes, o todas juntas.

En edición de países: Cuando el usuario elige editar un país, usamos match-case para preguntarle qué campo quiere modificar (nombre, población, superficie, continente o reconocimiento ONU).

En filtros: Cuando el usuario filtra por rangos (población o superficie), usamos condicionales para verificar si cada país cumple con los límites mínimos y máximos que pidió.

En navegación: Para saber si el usuario quiere volver al menú principal o seguir en un submenú, usamos if para evaluar su respuesta.

El match-case fue especialmente útil porque hace que el código de los menús sea más fácil de leer. En vez de tener un montón de elif uno debajo del otro, cada opción queda claramente separada.

```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
0) Guardar y salir

Seleccione una opción: 2

--- FILTRAR POR CONTINENTE ---
Continentes disponibles: América, Europa, Asia, África, Oceanía
Continente (o 'volver' para regresar): Oceanía

Resultados: 16 países - Página 1 de 2

1. Australia          | Pob:    25.687.041 | Sup:    7.692.024 km² | Oceanía
2. Fiyi               | Pob:     896.445 | Sup:     18.272 km² | Oceanía
3. Islas Marshall     | Pob:     59.190 | Sup:       181 km² | Oceanía
4. Islas Salomón      | Pob:    686.884 | Sup:    28.896 km² | Oceanía
5. Kiribati           | Pob:    119.449 | Sup:     811 km² | Oceanía
6. Micronesia         | Pob:    548.914 | Sup:     702 km² | Oceanía
7. Nauru              | Pob:     10.824 | Sup:      21 km² | Oceanía
8. Nueva Zelanda      | Pob:   5.084.300 | Sup:   268.021 km² | Oceanía
9. Palaos             | Pob:     18.094 | Sup:      459 km² | Oceanía
10. Papúa Nueva Guinea | Pob:   8.947.024 | Sup:   462.840 km² | Oceanía

Navegación:
[S]iguiente página | [A]nterior página | [Número] para ir a una página
[V]olver (para salir de la visualización)

Opción: 

```

6. BUCLES (ESTRUCTURAS REPETITIVAS)

¿Qué son los bucles?

Los bucles son estructuras que repiten un bloque de código varias veces. Python tiene dos tipos principales:

while: Repite el código MIENTRAS una condición sea verdadera. Es útil cuando no sabemos exactamente cuántas veces vamos a repetir algo.

for: Repite el código PARA CADA elemento de una colección (lista, archivo, rango de números, etc.). Es útil cuando queremos procesar todos los elementos de algo.

Dentro de los bucles podemos usar:

- **break:** para salir del bucle inmediatamente
- **continue:** para saltar a la siguiente repetición

Cómo los usamos en el proyecto

Bucles while para menús: El programa principal tiene un while True que se repite infinitamente hasta que el usuario elige la opción 9 (salir). Cada submenú también tiene su propio while para que el usuario pueda hacer varias operaciones seguidas sin volver al menú principal.

Por ejemplo, en el submenú de búsqueda, el usuario puede buscar "Argentina", ver los resultados, buscar "Brasil", ver esos resultados, y así hasta que escriba "volver".

Bucles for para procesar datos:

Cuando cargamos el archivo CSV, usamos un for para leer cada línea del archivo, validar los datos y agregarlos a la lista de países.

Para buscar o filtrar países, recorreremos toda la lista con un for, evaluamos cada país, y si cumple con lo que buscamos, lo agregamos a los resultados.

Para calcular estadísticas (promedios, sumas), usamos for para recorrer todos los países y acumular los valores.

Para mostrar resultados con paginación, usamos for con range() para mostrar solo los países de la página actual (por ejemplo, del 0 al 9 para la primera página, del 10 al 19 para la segunda, etc.).

Bucles con validación: En las funciones de utilidades, usamos while para pedir datos al usuario hasta que ingrese algo válido. Por ejemplo, si pedimos un número y el usuario escribe texto, el bucle se repite hasta que ingrese un número válido.

Los bucles fueron esenciales porque la mayor parte del programa consiste en repetir operaciones: procesar archivos línea por línea, recorrer listas buscando elementos, mostrar menús hasta que el usuario decida salir.

```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
Python + - [ ] [X] ... [ ] [X]

Seleccione una opción: 3

--- FILTRAR POR POBLACIÓN (RANGO) ---
¿Desea realizar un filtrado? (si/no): si
Dejá vacío un límite si no querés aplicarlo.
Población mínima: 100000000
Población máxima:

Resultados: 14 países - Página 1 de 2

1. Bangladesh | Pob: 164.689.383 | Sup: 147.570 km² | Asia
2. Brasil | Pob: 213.993.437 | Sup: 8.515.767 km² | América
3. China | Pob: 1.439.323.776 | Sup: 9.596.961 km² | Asia
4. Egipto | Pob: 102.334.404 | Sup: 1.002.450 km² | África
5. Estados Unidos | Pob: 331.893.745 | Sup: 9.833.517 km² | América
6. Etiopía | Pob: 114.963.588 | Sup: 1.104.300 km² | África
7. Filipinas | Pob: 109.581.078 | Sup: 300.000 km² | Asia
8. India | Pob: 1.380.004.385 | Sup: 3.287.263 km² | Asia
9. Indonesia | Pob: 273.523.615 | Sup: 1.904.569 km² | Asia
10. Japón | Pob: 125.800.000 | Sup: 377.975 km² | Asia

Navegación:
[S]iguiente página | [A]nterior página | [Número] para ir a una página
[V]olver (para salir de la visualización)

Opción:

```

7. ORDENAMIENTOS

¿Qué es el ordenamiento?

Ordenar es organizar elementos de una colección según un criterio específico, ya sea de menor a mayor (ascendente) o de mayor a menor (descendente).

Python tiene una función integrada llamada `sorted()` que hace todo el trabajo pesado por nosotros. Solo tenemos que decirle qué queremos ordenar y según qué criterio.

Los parámetros importantes de `sorted()` son:

- **key**: una función que extrae el valor por el cual queremos ordenar
- **reverse**: False para orden ascendente, True para descendente

Cómo lo implementamos

Creamos una función llamada `ordenar()` que recibe tres parámetros: la lista de países, la clave por la cual ordenar (nombre, población o superficie), y si queremos orden descendente o no.

Para que `sorted()` sepa cómo ordenar diccionarios, le pasamos una función auxiliar que extrae el valor de cada país según la clave elegida. Por ejemplo, si el usuario quiere ordenar por población, la función auxiliar devuelve `pais["poblacion"]` para cada país.

Decidimos NO usar funciones lambda (aunque sería más corto) porque las funciones auxiliares explícitas son más fáciles de entender y depurar, especialmente para un trabajo de primer año.

Ejemplos de ordenamiento que implementamos:

- **Por nombre alfabético:** ordena de Afganistán a Zimbabue
- **Por población descendente:** muestra primero China, India, Estados Unidos (los más poblados)
- **Por superficie ascendente:** muestra primero Vaticano, Mónaco, San Marino (los más chicos)

El ordenamiento fue importante porque con 206 países, encontrar el más grande o el más chico manualmente sería imposible. Además, ver los países ordenados ayuda a identificar patrones y comparar datos más fácilmente.

```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
Python + -  [Iconos de interfaz]

0) Guardar y salir

Seleccione una opción: 5

--- ORDENAR PAÍSES ---
Criterios disponibles: nombre, población, superficie
Criterio de ordenamiento (o 'volver' para regresar): poblacion
Orden (asc/desc): desc

Resultados: 206 países - Página 1 de 21

1. China | Pob: 1.439.323.776 | Sup: 9.596.961 km² | Asia
2. India | Pob: 1.380.004.385 | Sup: 3.287.263 km² | Asia
3. Estados Unidos | Pob: 331.893.745 | Sup: 9.833.517 km² | América
4. Indonesia | Pob: 273.523.615 | Sup: 1.904.569 km² | Asia
5. Pakistán | Pob: 220.892.340 | Sup: 881.913 km² | Asia
6. Brasil | Pob: 213.993.437 | Sup: 8.515.767 km² | América
7. Nigeria | Pob: 206.139.589 | Sup: 923.768 km² | África
8. Bangladesh | Pob: 164.689.383 | Sup: 147.570 km² | Asia
9. Rusia | Pob: 145.912.025 | Sup: 17.098.242 km² | Europa
10. México | Pob: 128.932.753 | Sup: 1.964.375 km² | América

Navegación:
[S]iguiente página | [A]nterior página | [Número] para ir a una página
[V]olver (para salir de la visualización)

Opción:

```

```

[2] Volver al menú principal
Seleccione una opción: 1

--- ORDENAR PAÍSES ---
Criterios disponibles: nombre, población, superficie
Criterio de ordenamiento (o 'volver' para regresar): nombre
Orden (asc/desc): asc

Resultados: 206 países - Página 1 de 21

1. Abjasia | Pob: 250.000 | Sup: 8.660 km² | Europa [No reconocido por ONU]
2. Afganistán | Pob: 38.928.346 | Sup: 652.230 km² | Asia
3. Albania | Pob: 2.877.797 | Sup: 28.748 km² | Europa
4. Alemania | Pob: 83.149.300 | Sup: 357.022 km² | Europa
5. Andorra | Pob: 77.265 | Sup: 468 km² | Europa
6. Angola | Pob: 32.866.272 | Sup: 1.246.700 km² | África
7. Antigua y Barbuda | Pob: 97.929 | Sup: 442 km² | América
8. Arabia Saudita | Pob: 34.813.871 | Sup: 2.149.690 km² | Asia
9. Argelia | Pob: 43.851.044 | Sup: 2.381.741 km² | África
10. Argentina | Pob: 45.376.763 | Sup: 2.780.400 km² | América

Navegación:
[S]iguiente página | [A]nterior página | [Número] para ir a una página
[V]olver (para salir de la visualización)

```


8. ESTADÍSTICAS BÁSICAS

¿Qué son las estadísticas básicas?

Las estadísticas básicas son cálculos matemáticos simples que nos ayudan a resumir y entender grandes conjuntos de datos. En vez de mirar 206 países uno por uno, calculamos valores que nos dan una idea general.

Las estadísticas que implementamos son:

- **Máximo:** el valor más grande
- **Mínimo:** el valor más pequeño
- **Promedio:** la suma de todos los valores dividida por la cantidad
- **Conteo por categorías:** cuántos elementos caen en cada grupo

Cómo calculamos las estadísticas

Nuestro sistema calcula 5 tipos de estadísticas sobre los países:

1. País con mayor población: Usamos la función `max()` de Python que busca automáticamente el valor más grande. Le pasamos una función auxiliar que extrae la población de cada país para compararlos. El resultado es el diccionario completo del país más poblado (China con 1,439 millones de habitantes).

2. País con menor población: Similar al anterior pero con `min()`. Encuentra el país menos poblado del sistema (en nuestro caso, Sealand con solo 30 habitantes).

3. Promedio de población: Recorremos todos los países con un bucle `for`, vamos sumando las poblaciones en una variable, y al final dividimos ese total por la cantidad de países. La fórmula es: $\text{Promedio} = \text{Suma Total} / \text{Cantidad de Países}$.

4. Promedio de superficie: Exactamente igual que el promedio de población, pero sumando las superficies en vez de las poblaciones.

5. Cantidad de países por continente: Usamos un diccionario vacío como contador. Recorremos todos los países, y para cada uno miramos su continente. Si ese continente ya está en el diccionario contador, le sumamos 1. Si no está, lo agregamos con valor 1. Al final tenemos un diccionario que dice cuántos países hay en Asia, Europa, África, América y Oceanía.

Submenú interactivo de estadísticas: En vez de mostrar todas las estadísticas de una, creamos un menú donde el usuario elige qué quiere ver. Puede ver solo el país más poblado, solo los promedios, solo la distribución por continentes, o todas juntas. Esto hace que el programa sea más flexible y que la pantalla no se llene de información que tal vez no interesa.

Las estadísticas son útiles porque nos permiten responder preguntas como "¿cuál es el país más grande?", "¿cuántos países hay en promedio en cada continente?", "¿hay más países en Asia o en Europa?" sin tener que revisar manualmente los 206

registros.

```
GESTIÓN DE PAÍSES - MENÚ PRINCIPAL

1) Buscar por nombre
2) Filtrar por continente
3) Filtrar por población (rango)
4) Filtrar por superficie (rango)
5) Ordenar (nombre/población/superficie,asc/desc)
6) Estadísticas
7) Agregar país
8) Editar país
9) Quitar país
0) Guardar y salir

Seleccione una opción: 6

--- ESTADÍSTICAS GENERALES ---
¿Qué estadística desea ver?
[1] País con mayor población
[2] País con menor población
[3] Promedios (población y superficie)
[4] Cantidad de países por continente
[5] Todas las estadísticas
[6] Volver al menú principal
Seleccione una opción: █
```

```
--- ESTADÍSTICAS GENERALES ---
¿Qué estadística desea ver?
[1] País con mayor población
[2] País con menor población
[3] Promedios (población y superficie)
[4] Cantidad de países por continente
[5] Todas las estadísticas
[6] Volver al menú principal
Seleccione una opción: 1

País con mayor población: China (1,439,323,776 habitantes)

¿Qué desea hacer?
[1] Realizar otra operación en este submenú
[2] Volver al menú principal
Seleccione una opción: █
```

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

[2] Volver al menú principal
Seleccione una opción: 1

--- ESTADÍSTICAS GENERALES ---
¿Qué estadística desea ver?
[1] País con mayor población
[2] País con menor población
[3] Promedios (población y superficie)
[4] Cantidad de países por continente
[5] Todas las estadísticas
[6] Volver al menú principal
Seleccione una opción: 5

País con mayor población: China (1,439,323,776 habitantes)
País con menor población: Sealand (30 habitantes)
Promedio de población: 37,860,470.45 habitantes
Promedio de superficie: 653,469.60 km²

Cantidad de países por continente:
• Asia: 49 países
• Europa: 49 países
• África: 57 países
• América: 35 países
• Oceanía: 16 países
```

9. ARCHIVOS CSV

¿Qué es un archivo CSV?

CSV significa "Comma-Separated Values" (valores separados por comas). Es un formato de archivo de texto plano donde cada línea representa una fila de datos, y los valores dentro de cada fila se separan con comas.

La primera línea suele tener los nombres de las columnas (encabezados), y las líneas siguientes tienen los datos. Es como una tabla de Excel pero guardada en formato simple de texto.

Ventajas del formato CSV:

- Es fácil de abrir con cualquier editor de texto
- Funciona en cualquier sistema operativo

- Es compatible con Excel, Google Sheets y la mayoría de programas
- Ocupa poco espacio
- Se puede versionar con Git

Por qué usamos encoding UTF-8

El encoding (codificación) es la forma en que las letras se convierten en números para guardarlas en archivos. Si usamos la codificación incorrecta, los acentos y caracteres especiales se ven mal.

UTF-8 es importante porque:

- Soporta todos los caracteres especiales: tildes (á, é, í, ó, ú), ñ, símbolos
- Funciona con cualquier idioma del mundo
- Es el estándar en internet y en Python moderno

Sin UTF-8, "México" se vería como "M?xico" o algo peor. Como nuestro archivo tiene países de todo el mundo con nombres en diferentes idiomas, UTF-8 era obligatorio.

Cómo cargamos el archivo CSV

Usamos el módulo csv de Python (que viene incluido, no hay que instalarlo) para leer el archivo. Específicamente usamos csv.DictReader que convierte cada fila en un diccionario automáticamente, usando los encabezados como claves.

El proceso de carga es:

1. Abrir el archivo con open() en modo lectura, usando UTF-8
2. Crear un lector de diccionarios con csv.DictReader
3. Recorrer cada fila del archivo con un bucle for
4. Limpiar cada fila (quitar espacios en blanco, normalizar nombres de columnas)

5. Validar los datos (convertir textos a números, verificar valores mínimos)
6. Si la fila es válida, agregarla a la lista de países
7. Si hay error en alguna fila, descartarla y seguir con la siguiente

Si el archivo no existe, el programa lo crea automáticamente con los encabezados correctos para que no se rompa.

Cómo guardamos cambios

Cuando el usuario elige la opción 9 (Guardar y salir), escribimos toda la lista de países de vuelta al archivo CSV.

Usamos `csv.DictWriter` que hace el trabajo inverso: convierte diccionarios en filas de CSV. El proceso es:

1. Abrir el archivo en modo escritura (esto borra el contenido anterior)
2. Crear un escritor de diccionarios especificando el orden de las columnas
3. Escribir la fila de encabezados
4. Recorrer todos los países y escribir cada uno como una fila
5. Convertir el campo booleano `reconocido_onu` a texto ("si" o "no") para que sea legible

Validación de datos

Los datos que vienen del CSV son todos texto. Tenemos que convertirlos a los tipos correctos:

- Los nombres quedan como texto
- La población se convierte a número entero
- La superficie también se convierte a entero (si viene con decimales, la redondeamos)
- El continente queda como texto

- El reconocimiento ONU se convierte a verdadero/falso

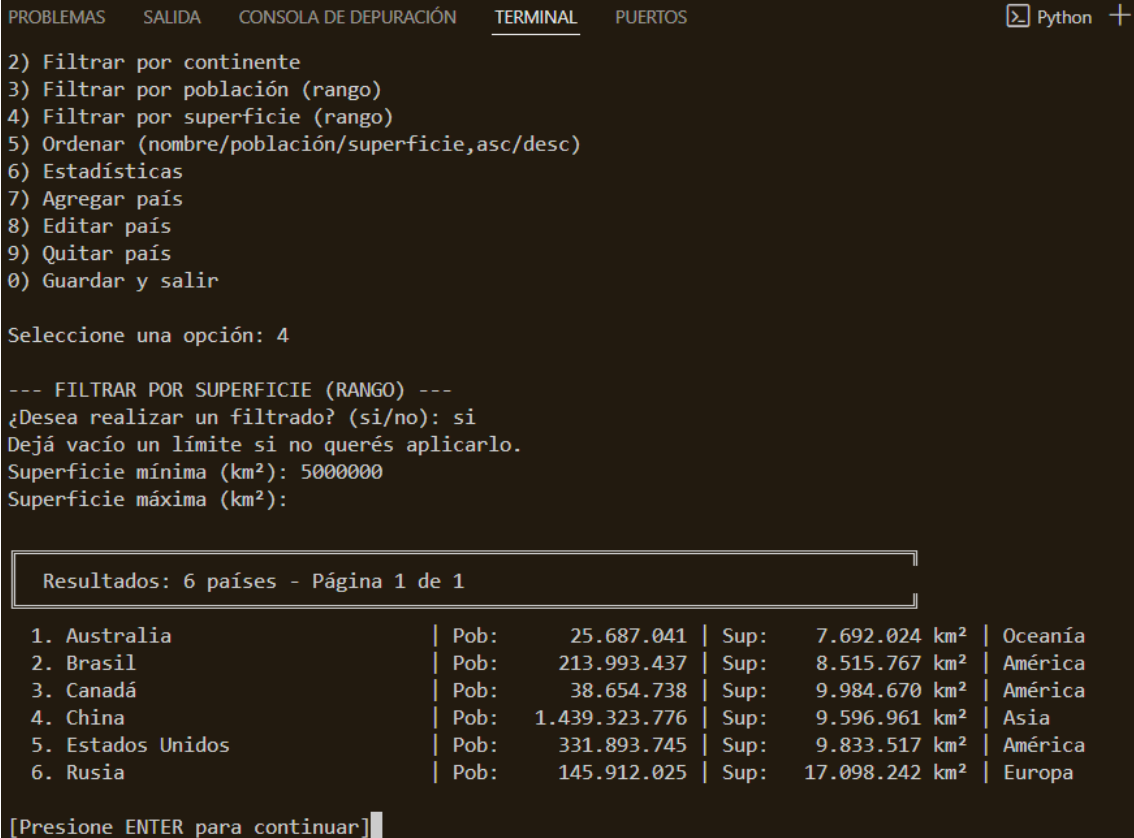
También validamos que:

- Los nombres no estén vacíos
- Los números no sean negativos
- Todos los campos requeridos estén presentes

Si algún dato no pasa las validaciones, descartamos esa fila completa del CSV

para evitar que el programa se rompa con datos incorrectos.

El manejo de archivos CSV fue una de las partes más importantes del proyecto porque es lo que permite que los datos persistan entre ejecuciones. Sin esto, cada vez que cerráramos el programa perderíamos todos los cambios.



```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
Python +

2) Filtrar por continente
3) Filtrar por población (rango)
4) Filtrar por superficie (rango)
5) Ordenar (nombre/población/superficie,asc/desc)
6) Estadísticas
7) Agregar país
8) Editar país
9) Quitar país
0) Guardar y salir

Seleccione una opción: 4

--- FILTRAR POR SUPERFICIE (RANGO) ---
¿Desea realizar un filtrado? (si/no): si
Dejá vacío un límite si no querés aplicarlo.
Superficie mínima (km²): 5000000
Superficie máxima (km²):

Resultados: 6 países - Página 1 de 1

1. Australia | Pob: 25.687.041 | Sup: 7.692.024 km² | Oceanía
2. Brasil | Pob: 213.993.437 | Sup: 8.515.767 km² | América
3. Canadá | Pob: 38.654.738 | Sup: 9.984.670 km² | América
4. China | Pob: 1.439.323.776 | Sup: 9.596.961 km² | Asia
5. Estados Unidos | Pob: 331.893.745 | Sup: 9.833.517 km² | América
6. Rusia | Pob: 145.912.025 | Sup: 17.098.242 km² | Europa

[Presione ENTER para continuar]

```

9.1 OPERACIONES CRUD

Agregar, Editar y Eliminar Países

Nuestro sistema permite gestionar completamente los datos de países mediante operaciones CRUD (Create, Read, Update, Delete):

Agregar países nuevos: El usuario puede ingresar un país con todos sus datos (nombre, población, superficie, continente). Además, **el sistema pregunta si el país está reconocido por la ONU**, guardando esta información como verdadero o falso. Esto permite distinguir entre países oficialmente reconocidos y territorios con reconocimiento limitado.


```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

GESTIÓN DE PAÍSES - MENÚ PRINCIPAL

1) Buscar por nombre
2) Filtrar por continente
3) Filtrar por población (rango)
4) Filtrar por superficie (rango)
5) Ordenar (nombre/población/superficie,asc/desc)
6) Estadísticas
7) Agregar país
8) Editar país
9) Quitar país
0) Guardar y salir

Seleccione una opción: 7

--- AGREGAR NUEVO PAÍS ---
Nombre del país: PaisTest
Población: 5000000
Superficie (km²): 100000
Continente: América
¿El país es reconocido por la ONU? (si/no): no
✓ País 'PaisTest' agregado exitosamente.

¿Qué desea hacer?
  [1] Realizar otra operación en este submenú
  [2] Volver al menú principal
Seleccione una opción: █
```

Editar países existentes: Esta fue una funcionalidad que agregamos para hacer el sistema más completo. Permite buscar un país por su nombre exacto y modificar cualquiera de sus campos: nombre, población, superficie, continente o estado de reconocimiento ONU. El sistema muestra primero los datos actuales y luego pregunta qué campo se quiere cambiar.

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

9) Quitar país
0) Guardar y salir

Seleccione una opción: 8

--- EDITAR PAÍS ---
Nombre exacto del país a editar (o 'volver' para regresar): PaisTest

Datos actuales de 'PaisTest':
Población: 5,000,000
Superficie: 100,000 km²
Continente: América
Reconocido por ONU: No

¿Qué desea editar?
[1] Nombre
[2] Población
[3] Superficie
[4] Continente
[5] Reconocido por ONU
[6] Cancelar
Seleccione una opción: 2
Nueva población: 6000000
✓ Población actualizada.
```

Eliminar países: Busca un país por nombre exacto y lo quita de la lista. Si el país no existe, informa al usuario que no se encontró.

GESTIÓN DE PAÍSES - MENÚ PRINCIPAL

- 1) Buscar por nombre
- 2) Filtrar por continente
- 3) Filtrar por población (rango)
- 4) Filtrar por superficie (rango)
- 5) Ordenar (nombre/población/superficie,asc/desc)
- 6) Estadísticas
- 7) Agregar país
- 8) Editar país
- 9) Quitar país
- 0) Guardar y salir

Seleccione una opción: 9

--- ELIMINAR PAÍS ---

Nombre exacto del país a eliminar (o 'volver' para regresar): PaisTest
✓ País 'PaisTest' eliminado exitosamente.

¿Qué desea hacer?

- [1] Realizar otra operación en este submenú
- [2] Volver al menú principal

Seleccione una opción: 1

--- FILTRAR POR CONTINENTE ---

Continentes disponibles: América, Europa, Asia, África, Oceanía
Continente (o 'volver' para regresar): América

Resultados: 35 países - Página 1 de 4

1. Antigua y Barbuda	Pob:	97.929	Sup:	442 km ²	América
2. Argentina	Pob:	45.376.763	Sup:	2.780.400 km ²	América
3. Bahamas	Pob:	393.244	Sup:	13.943 km ²	América
4. Barbados	Pob:	287.375	Sup:	430 km ²	América
5. Belice	Pob:	397.628	Sup:	22.966 km ²	América
6. Bolivia	Pob:	11.673.021	Sup:	1.098.581 km ²	América
7. Brasil	Pob:	213.993.437	Sup:	8.515.767 km ²	América
8. Canadá	Pob:	38.654.738	Sup:	9.984.670 km ²	América
9. Chile	Pob:	19.458.000	Sup:	756.102 km ²	América
10. Colombia	Pob:	51.265.844	Sup:	1.141.748 km ²	América

Navegación:

[S]iguiente página | [A]nterior página | [Número] para ir a una página
[V]olver (para salir de la visualización)

Opción: s

Resultados: 35 países - Página 2 de 4

11. Costa Rica	Pob:	5.094.118	Sup:	51.100 km ²	América
----------------	------	-----------	------	------------------------	---------

Resultados: 35 países - Página 2 de 4				
11. Costa Rica	Pob:	5.094.118	Sup:	51.100 km ² América
12. Cuba	Pob:	11.326.616	Sup:	109.884 km ² América
13. Dominica	Pob:	71.986	Sup:	751 km ² América
14. Ecuador	Pob:	17.797.737	Sup:	283.561 km ² América
15. El Salvador	Pob:	6.486.205	Sup:	21.041 km ² América
16. Estados Unidos	Pob:	331.893.745	Sup:	9.833.517 km ² América
17. Granada	Pob:	112.523	Sup:	344 km ² América
18. Guatemala	Pob:	17.915.568	Sup:	108.889 km ² América
19. Guyana	Pob:	786.552	Sup:	214.969 km ² América
20. Haití	Pob:	11.402.528	Sup:	27.750 km ² América
Navegación: [S]iguiente página [A]nterior página [Número] para ir a una página [V]olver (para salir de la visualización)				

GESTIÓN DE PAÍSES - MENÚ PRINCIPAL				
1) Buscar por nombre 2) Filtrar por continente 3) Filtrar por población (rango) 4) Filtrar por superficie (rango) 5) Ordenar (nombre/población/superficie,asc/desc) 6) Estadísticas 7) Agregar país 8) Editar país 9) Quitar país 0) Guardar y salir				
Seleccione una opción: 1				
--- BUSCAR POR NOMBRE ---				
Texto a buscar (o 'volver' para regresar): taiwan				
Resultados: 1 países - Página 1 de 1				
1. Taiwán	Pob:	23.600.000	Sup:	36.000 km ² Asia [No reconocido por ONU]

10. CONCLUSIONES

¿Qué aprendimos?

Belén:

Al principio, ver la lista de consignas me resultó abrumador. No sabía bien por dónde empezar ni cómo íbamos a conectar todas las partes. Pero cuando comprendimos la importancia de la modularización —dividir el código en funciones pequeñas y específicas— todo empezó a tener sentido. Aprendí que es mucho más eficiente desarrollar una función que haga solo una cosa y probarla a fondo, que intentar resolver todo en un solo bloque.

Lo que más me costó fue trabajar con los datos del CSV. Había espacios innecesarios, valores extraños y celdas vacías, y eso hacía que el programa fallara. Ahí entendí por qué los docentes insisten tanto en no confiar ciegamente en los datos externos. Implementar funciones de limpieza y validación fue fundamental para lograr que el sistema sea estable y no se rompa ante errores de formato.

Marcelo:

En mi caso, lo que más me aportó este trabajo fue aprender a manejar correctamente listas de diccionarios. Al principio se me mezclaban las ideas, pero con la práctica entendí que es una de las formas más lógicas de representar información real dentro de un programa.

También disfruté mucho incorporar la estructura match-case en los menús. Venía acostumbrado a los clásicos if-elif-else que quedan largos y poco legibles, y esta

nueva forma me pareció mucho más ordenada. Otro punto desafiante fue el manejo de archivos: entender cómo funciona el `encoding="utf-8"` para que los caracteres especiales se vean correctamente y aplicar el bloque `try-except` con `FileNotFoundError` para crear el archivo CSV automáticamente si no existía. Ese fue un gran aprendizaje práctico.

Dificultades que enfrentamos

1. La escala del proyecto:

Al principio nos costó entender cómo unir todas las consignas. Conectar el menú con las funciones de operaciones y, a su vez, con las de carga y lectura de archivos fue nuestro primer gran desafío.

2. Modularización:

Dividir el código en seis archivos no fue tarea fácil. Tuvimos que decidir con cuidado qué función pertenecía a cada módulo para evitar errores de importación circular o funciones con responsabilidades duplicadas. Las consultas con la profesora Cintia y el apoyo de herramientas como las IAs fueron de gran ayuda en este punto.

3. Manejo de archivos (CSV automático):

Lograr que el programa no falle si el archivo CSV no existe fue complicado. No sabíamos cómo hacer que Python verificara la existencia del archivo y lo generara con los encabezados correctos en caso de que faltara. Con la ayuda de nuestro compañero **Camilo Illaños** comprendimos la lógica del `try-except` con `FileNotFoundError` y la escritura con `"w"`.

4. Paginación:

Mostrar los 206 países de una sola vez era poco práctico. La lógica para calcular los índices de “desde” y “hasta” en cada página (por ejemplo, página 1: 0–9, página 2: 10–19) requirió varios intentos, pruebas y hasta esquemas en papel antes de lograr un funcionamiento correcto.

5. Búsqueda sin acentos:

Queríamos que, al buscar “Mexico”, el programa también encontrara “México”. Investigando descubrimos el módulo `unicodedata`, que nos permitió normalizar

los textos y comparar sin acentos, mejorando notablemente la experiencia del usuario.

Si tuviéramos más tiempo, agregaríamos...

- **Una interfaz más amigable en consola:**
Usar librerías como rich o colorama para agregar color, limpiar la pantalla más prolijamente y mostrar los datos en tablas en lugar de texto plano.
- **Búsqueda por múltiples criterios:**
Permitir al usuario combinar filtros, por ejemplo: mostrar los países de “Asia” con una población mayor a 100 millones.
- **Exportación de reportes:**
Crear una función para guardar los resultados de una búsqueda o las estadísticas en un archivo de texto (.txt) o en un CSV resumido.

Reflexión final

Este trabajo nos permitió entender que programar no es memorizar comandos, sino aprender a **pensar los problemas paso a paso**. Lo importante no es saberlo todo, sino saber **cómo buscar una solución**, dividir las tareas y usar las herramientas adecuadas.

Aprendimos que no está mal no saber algo; lo que realmente importa es no quedarse trabado. Aprovechamos todos los recursos disponibles: el material de la cátedra, las consultas con la profesora, la colaboración entre compañeros y el apoyo de las inteligencias artificiales, además de las explicaciones de **Charly Cimino**. Comprobamos que en la práctica profesional, “saber buscar” es tan importante como “saber programar”.

El trabajo en equipo fue clave. Supimos organizarnos, dividir tareas y revisar el código del otro, lo que mejoró mucho el resultado final. Terminamos con un programa que funciona correctamente, cumple con todas las consignas y, sobre todo, nos permitió entender **cómo y por qué funciona cada parte**.

Más allá de la nota, nos quedamos con la satisfacción de haber aprendido de verdad.

REFERENCIAS

Anthropic. (2024). Claude 4.5 Sonnet [Large language model]. <https://claude.ai>

Cimino, C. [@CharlyCimino]. (s.f.). Python desde cero [Playlist de video].

YouTube. Recuperado el 24 de octubre de 2025,

de <https://www.youtube.com/playlist?list=PL-wATfer314-qg-aNII-1L-j2df-OH-0s>

Google. (s.f.). Gemini Pro [Large language model]. <https://gemini.google.com>

Illañes, C. (2025). Asesoramiento sobre implementación de paginación y modularización en consola [Comunicación personal]. Estudiante de Programación 1, UTN Mendoza.

OpenAI. (s.f.). ChatGPT-4 [Large language model]. <https://chat.openai.com>

Python Software Foundation. (2024). The Python Standard Library. <https://docs.python.org/3/library/>

Rigoni, C. (2025). Material de la plataforma educativa: Programación 1. [Material didáctico no publicado]. Tecnicatura Universitaria en Programación, UTN Facultad Regional Mendoza.