Universidad Tecnológica Nacional

“Trabajo Práctico Integrador de Programación 1”

Título del trabajo: *“Gestión de Datos de Países en Python: filtros, ordenamientos y estadísticas”*

Alumnos:

- *Elisabeth Cordero Campero (Comisión 3)*
- *Garcia Nadia Anahi (Comisión 5)*

Profesor: *Cinthia Rigoni*

- *Comisión 3: Brian Lara*
- *Comisión 5: Matias Torres*

Fecha de entrega: *11/11/2025*

Índice

MARCO TEÓRICO

En el presente apartado explicaremos conceptos claves para la realización del trabajo : “Gestión de Datos de Países en Python”.

1. Listas

En Python, una lista es una estructura de datos ordenada y mutable que permite almacenar una secuencia de elementos del mismo tipo e incluso de distintos, como las listas mixtas que tienen cadenas de texto (str), enteros (int) , decimales (float) y True o False (bool). También se pueden crear listas vacías y listas anidadas.

Las listas se definen mediante corchetes [] , y están separados por una coma.

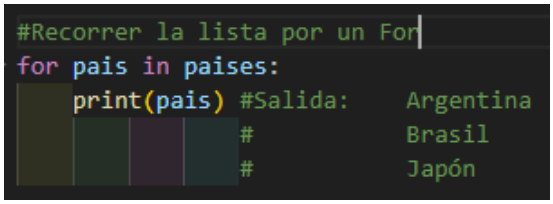
Ejemplo:

```
#Lista mixta
lista = [2, "Luciana", True, 2.5]
#Lista con mismos elementos
paises = ["Argentina", "Brasil", "Japón"]
#Listas anidadas
lista_anidada = [[1, 2, 3], [4, 5, 6]]
```

Otra característica de las listas, es que podemos acceder a los elementos por su índice y también en el caso de no querer imprimir uno por uno los elementos, los podemos recorrer por medio de un bucle for.

Ejemplos:

```
#Elemento por el indice
print(paises[0]) #Imprime: Argentina

print(paises[:3]) #Imprime ['Argentina', 'Brasil', 'Japón']

print(lista_anidada[0][1]) # Imprime 2
```



Con las listas hay múltiples funciones que pueden realizar, vamos a nombrar las más utilizadas para este proyecto:

- **Append() :** Nos permite agregar elementos a la lista original, al final.
- **In :** Con esta palabra reservada podemos buscar elementos en una lista.
- **Len ()**: Nos permite contar la cantidad de elementos que contiene una lista.
- **copy():** Crea una copia independiente de la lista original.

Las listas son fundamentales para este proyecto porque el programa almacenará la información de varios países en una lista de registros (por ejemplo, cada país será un diccionario dentro de una lista). Permiten recorrer los elementos, aplicar filtros y realizar ordenamientos de manera eficiente. Su mutabilidad es esencial para cumplir con los requisitos de "Agregar un país" y "Actualizar los datos". Para esto, utilizaremos métodos como .append() para añadir un nuevo país al final de la lista.

Al ser una secuencia ordenada, nos permitirá iterar sobre ella para realizar búsquedas, filtros y cálculos estadísticos.

Además la lista será la estructura de datos principal de nuestro programa. Contendrá la totalidad de los datos de los países cargados desde el archivo CSV.

Fuente:

- Python Software Foundation. (2024). Python Tutorial – Data Structures. https://docs.python.org/3/tutorial/datastructures.html.

- Listas y tuplas en Python – Real Python
- Material de la UTN - Listas Programación 1

2. Diccionarios

Los **diccionarios** son estructuras de datos no ordenadas y mutables (agregar o eliminar elementos una vez creado) , que almacenan pares *clave (key) - valor (value)*. Donde las claves pueden ser cualquier tipo inmutable (strings,números, tuplas). Las cadenas y números siempre pueden ser claves. A diferencia de las listas, donde los elementos se acceden por su posición, en los diccionarios se accede mediante una clave única asociada a cada valor. Gracias a esta característica de los diccionarios, nos permite acceder rápidamente a datos por su nombre en lugar de su posición.

Se definen por medio de llaves ' {} ' y separados por : y los pares entre sí por comas.

Ejemplo:

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

Las operaciones que utilizaremos en el proyecto sobre diccionario son:

- **Acceso a elementos:** Nos permite obtener el valor asociado a una clave específica.

  Ejemplo:

```
pais["nombre"] # "Argentina"
```

- **Agregar o modificar elementos:** Se puede agregar una nueva clave o cambiar el valor de una existente simplemente asignando un valor.

Ejemplo:

```
pais["idioma"] = "Español" # Agrega nueva clave "idioma" con valor "Español"
```

- **Eliminar elementos:** Con la instrucción **'del'**, se elimina una clave y su valor del diccionario.

  Ejemplo:

```
del pais["superficie"] # Elimina la clave "superficie" y su valor asociado
```

- **Recorrer diccionarios:** Con un bucle **'**for**'** y el método **'**.items()**'** se pueden recorrer las claves y valores.

  Ejemplo:

```
for clave, valor in pais.items():
    print(clave, ":", valor)
```

- **Uso de la palabra reservada in:** Permite comprobar si una clave existe en el diccionario antes de acceder a ella.

  Ejemplo:

```
if "nombre" in pais:
    print("La clave existe")
```

Otros métodos útiles:

- **.keys():** Devuelve una vista con todas las claves del diccionario.
- **.values():** Devuelve una vista con todos los valores.
- **.items():** Devuelve pares de clave–valor.
- **.get():** Permite acceder a un valor sin generar error si la clave no existe.

En el proyecto, cada país se representará como una lista de diccionario, ya que permite relacionar directamente el nombre de cada atributo con su valor. Las claves serán los nombres de sus atributos (ej. "nombre", "población", "superficie", "continente") y los valores serán los datos específicos de ese país. Esta combinación, es la más óptima para "gestionar información sobre países", ya que nos permite tener una colección de elementos (la lista) donde cada elemento (el diccionario) es fácilmente legible y sus datos son accesibles por un nombre claro (la clave).

Fuentes:

- Python Software Foundation. (2024). *Python Documentation – Mapping Types: dict* 5. Estructuras de datos — documentación de Python - 3.7.17
- Material UTN - Datos Complejos

3. Funciones

Las **funciones** son bloques de código reutilizables que nos permiten organizar y estructurar los programas. Su principal objetivo es dividir un problema complejo en partes más pequeñas, facilitando la lectura, el mantenimiento y la reutilización del código

Las características principales de las funciones son:

- Se definen con la palabra reservada **'def'**, seguida del nombre de la función y los paréntesis.
- Pueden recibir **parámetros o argumentos** y devolver valores con la instrucción **'return'**.
- Mejoran la **modularidad**, ya que cada función cumple una tarea específica.

Ejemplo de función básica:

```
def sumar(a, b):
    resultado = a + b
    return resultado
```

El uso de funciones hace que se evite la repetición de código, también lograr una estructura más clara y ordenada. Facilita la depuración y el mantenimiento del programa, así como también poder utilizar estas funciones en otros proyectos.

En nuestro proyecto las funciones serán la herramienta principal para cumplir con el requisito de "Código claro, comentado y modularizado". En lugar de escribir un único script, dividiremos el programa en múltiples funciones. Crearemos funciones específicas para cada ítem del menú. Esta "abstracción" nos permitirá tener un código más limpio, fácil de mantener y depurar.

Fuente:

- Python Software Foundation. (2024). *Defining Functions — Python Documentation.* Recuperado de: https://docs.python.org/3/tutorial/controlflow.html#defining-functions
- Programiz. (2024). *Python Functions.* Recuperado de: https://www.programiz.com/python-programming/function

4. Condicionales

Las estructuras condicionales (if, elif, else) permiten que un programa tome decisiones en función de la lógica. Son fundamentales para controlar el flujo de ejecución y realizar diferentes acciones según se cumpla o no una determinada condición.

Las Características principales:

- Se definen con la palabra reservada if, y pueden complementarse con ‘elif ‘ y ‘else’.
- Estas estructuras evalúan una expresión booleana que puede ser verdadera (True) o falsa (False).
- Permiten ejecutar distintos bloques de código según el resultado de la evaluación.

Ejemplo:

```
if pais["poblacion"] > 100000000:
    print("País muy poblado")
```

Para el proyecto las condicionales se utilizan principalmente dentro de las funciones para:

- **Validar datos ingresados** (por ejemplo, verificar si los valores son numéricos o si un campo no está vacío).
- **Comparar valores** durante los procesos de filtrado y ordenamiento.
- **Comprobar la existencia de claves** o coincidencias en diccionarios.
- **Evaluar condiciones estadísticas**, como determinar el país con mayor o menor población.

De esta manera, las estructuras condicionales garantizan el funcionamiento correcto del programa, permitiendo tomar decisiones internas sin afectar el flujo general del menú principal.

Fuentes:

- Python Software Foundation. (2024). *More Control Flow Tools — if Statements.* En: Python Documentation. Recuperado de: https://docs.python.org/3/tutorial/controlflow.html
- Material de la UTN - Estructuras condicionales

5. Estructuras Repetitivas

Las **estructuras repetitivas**, también conocidas como **bucles o iteraciones**, permiten ejecutar un mismo bloque de código varias veces mientras se cumpla una condición o hasta recorrer una secuencia completa.

Son fundamentales para automatizar tareas, recorrer listas y procesar grandes volúmenes de datos sin Los tipos principales de estructuras repetitivas son:

Bucle for:

El bucle for en Python se utiliza para recorrer los elementos de una secuencia, como una lista, cadena o rango numérico. Para este tipo, no es necesario usar un índice: el bucle accede directamente a cada elemento.

Ejemplo:

```
for pais in paises:
    print(pais["NOMBRE"])
```

En este ejemplo, la variable país toma el valor de cada elemento dentro de la lista de países. El ciclo finaliza automáticamente cuando se recorren todos los elementos de la secuencia.

El bucle for también puede usarse junto con la función range() para repetir un bloque de código una cantidad determinada de veces:

```
for i in range(5):
    print("Iteración", i)
```

Bucle while:

Se ejecuta mientras la condición especificada sea verdadera (True). Es útil cuando no se sabe cuántas veces se repetirá el ciclo , por ejemplo, para validar una entrada de usuario.

El programa usará bucles para recorrer la lista de países, aplicar filtros, sumar poblaciones o calcular promedios.

En el desarrollo del proyecto, las estructuras repetitivas tendrán un papel fundamental para garantizar que el programa funcione de manera dinámica y eficiente . Por medio de su implementación, se buscará automatizar procesos, reducir la cantidad de código manual y mantener la interacción constante con el usuario.

Se utilizará while para mostrar el menú tantas veces sea hasta tanto el usuario quiera salir.

La lectura y carga de datos, al recorrer por medio del bucle los datos que están en el archivo CSV. También en la búsqueda de la información se utilizará el bucle for, en el ordenamiento de datos y para calcular las estadísticas sumando los valores y calculando los promedios.

Fuente:

- Python Software Foundation. (2024). *More Control Flow Tools — for and while loops.* En: *The Python Tutorial* - https://docs.python.org/3/tutorial/controlflow.html
- Material UTN - Estructuras Repetitivas

6. Ordenamientos

El ordenamiento es un proceso que permite organizar los datos de una lista según un criterio determinado, como el nombre, la población o la superficie.

- **Ordenamiento manual (algoritmo de burbuja):**

  Consiste en recorrer la lista repetidas veces, comparando elementos adyacentes e intercambiándose cuando están en el orden incorrecto.

  Este método, aunque menos eficiente, resulta útil para comprender la lógica interna de los algoritmos de ordenamiento.

  Ejemplo:

```python
# Copia de la lista original
lista_ordenada = paises.copy()

# Algoritmo de ordenamiento burbuja (ascendente)
for i in range(len(lista_ordenada) - 1):
    for j in range(len(lista_ordenada) - 1 - i):
        if lista_ordenada[j]["POBLACION"] > lista_ordenada[j + 1]["POBLACION"]:
            # Intercambio de posiciones
            lista_ordenada[j], lista_ordenada[j + 1] = lista_ordenada[j + 1], lista_ordenada[j]
```

En el desarrollo del programa, las estructuras de ordenamiento **tendrán un rol esencial para mejorar la organización y presentación de los datos**. Su implementación permitirá al usuario visualizar la información de los países de manera ordenada según distintos criterios, por: Nombre, Población, Superficie . Dado que nuestra estructura es una lista de diccionarios, no basta con un ordenamiento simple. Utilizaremos métodos de ordenamiento avanzados (como la función sorted() o el método .sort()) especificando una key (clave). Esta key le indicará a Python *qué valor* del diccionario (ej. pais['poblacion']) debe usar como criterio para ordenar la lista completa, tanto de forma ascendente como descendente.

7. **Estadísticas Básicas**

Las estadísticas básicas (máximo, mínimo, promedio, conteos) permiten obtener información resumida de los datos. En Python se implementan fácilmente con funciones como max(), min(), sum().

- **sum()**: Calcula la suma total de los elementos numéricos dentro de una lista.
- **max():** Devuelve el valor máximo dentro de una lista. En el programa, se usa para encontrar el país con mayor población o superficie.
- **min():** Devuelve el valor mínimo dentro de una lista.

Ejemplo:

```
# Cálculo de estadísticas básicas
total_poblacion = sum(pais["POBLACION"] for pais in paises)
promedio_poblacion = total_poblacion / len(paises)
pais_mayor_pob = max(paises, key=lambda x: x["POBLACION"])
pais_menor_sup = min(paises, key=lambda x: x["SUPERFICIE"])
```

Para el proyecto cumplir con el requisito de "Mostrar estadísticas", aplicaremos estas operaciones sobre nuestra lista de países.

- Recorreremos la lista para encontrar el "País con mayor y menor población" (usando funciones max() y min() con una key).
- Calcularemos el "Promedio de población" y "Promedio de superficie" (sumando los valores de todos los diccionarios y dividiéndolos por el total de países, len(lista)).
- Implementaremos un conteo para determinar la "Cantidad de países por continente"

8. Archivos CSV

Los **archivos CSV (Comma Separated Values)** son archivos de texto plano en los que los datos se separan por comas, cada línea representa la fila y cada valor que está separado por las comas representan una columna. Es uno de los formatos que más se utiliza para almacenar y compartir datos, por su facilidad para trabajar con hojas de cálculos y base de datos.

Permiten mantener una persistencia de la información, permitiendo que los datos permanezcan guardados incluso cuando el programa se cierre, y puedan reutilizarse en futuras ejecuciones.

Python incluye el módulo csv , lo que permite leer, escribir y manipular archivos. Este módulo es útil para proyectos que requieren guardar y recuperar datos estructurados.

El módulo cuenta con clases específicas:

- csv.reader: para leer archivos y recorrer cada fila como una lista.
- csv.DictReader: para leer datos en forma de diccionario, accediendo a cada columna por su nombre.
- csv.writer: para escribir filas como listas.
- csv.DictWriter: para escribir registros completos como diccionarios, facilitando la escritura estructurada de datos.

Ejemplo:

```python
import csv

# Guardar datos en un archivo CSV
with open("paises.csv", "w", newline="", encoding="utf-8") as archivo:
    campos = ["NOMBRE", "POBLACION", "SUPERFICIE", "CONTINENTE"]
    escritor = csv.DictWriter(archivo, fieldnames=campos)
    escritor.writeheader()
    escritor.writerows(paises)

# Leer datos desde el archivo CSV
paises_leidos = []
with open("paises.csv", "r", encoding="utf-8") as archivo:
    lector = csv.DictReader(archivo)
    for fila in lector:
        paises_leidos.append(fila)
```

Para el proyecto el sistema debe ser capaz de "leer datos desde un archivo CSV". Utilizaremos la lectura de archivos CSV para cargar el dataset inicial en nuestra lista de diccionarios.

Para procesar cada línea del archivo, aplicaremos conceptos como el método .split(), que permite tomar un string (la línea leída) y convertirlo en una lista de strings (los datos separados por coma), los cuales luego usaremos para construir el diccionario de cada país.

Fuente:

- Python Software Foundation. (2024). *csv — CSV File Reading and Writing*. https://docs.python.org/3/library/csv.html

- Material UTN - Manejo de Archivos CSV

OBJETIVO DEL TRABAJO

El objetivo principal del trabajo fue desarrollar un programa en Python capaz de gestionar información de países de manera eficiente y organizada, aplicando los conceptos aprendidos.

A través de este proyecto se buscó que el sistema pudiera:

- Registrar nuevos países en un archivo CSV.
- Actualizar datos existentes (población y superficie).
- Buscar países por nombre o coincidencias parciales.
- Filtrar resultados según criterios específicos (continente, población, superficie).
- Ordenar los registros por distintos atributos.
- Generar estadísticas que permitan analizar la información cargada.

Además, el trabajo tuvo como objetivo la planificación del código, la comprensión del flujo de datos y la documentación adecuada del proyecto. El proyecto permitió integrar teoría y práctica, aplicando las herramientas vistas en clase para construir una solución funcional, modular y fácilmente extensible.

DISEÑO Y FLUJO DEL PROYECTO

Diagrama de flujo de operaciones principales

METODOLOGÍA UTILIZADA

Para el desarrollo del proyecto Gestión de Datos de Países, se aplicó una metodología estructurada y progresiva, que combinó el análisis teórico con la implementación práctica en Python.

Se comenzó identificando las necesidades del sistema, que debía permitir almacenar, consultar y procesar información de países de manera dinámica.

Se definieron los datos principales (nombre, población, superficie y continente) y las operaciones requeridas: ingreso, actualización, búsqueda, filtrado, ordenamiento y estadísticas.

Se elaboró un diagrama de las estructuras principales que debía tener el programa, permitiendo visualizar la lógica y corregir posibles errores antes de la codificación final.

Una vez ya con la estructura del programa fue codificado en Python utilizando estructuras de control , funciones para modularizar el código y manejo de archivos mediante la librería estándar csv. También se aplicaron validaciones de entrada para asegurar que los datos ingresados sean correctos.

Por otra parte también se integró al código el usó "try - except", estas se utilizaron específicamente en funciones de manejo de archivos: "verificar_archivo()", leer_paises_desde_archivo()" y en funciones principales como: "ingresar_paises()", "actualizar_poblacion_y_superficie()". El manejo de errores permite que el programa no se detenga ante fallos del sistema o datos incorrectos, mostrando mensajes claros al usuario. Se manejan distintos tipos de errores: "FileNotFoundError", "ValueError", "OSError/IOError" y "Exception".

También se integró al código el usó lambda en "mostrar_estadisticas()" para simplificar el código al definir una función anónima que permite comparar los países por población sin necesidad de escribir una función aparte.

Una vez finalizado el código, se realizaron pruebas con distintos conjuntos de datos para verificar la funcionalidad de cada opción del menú.

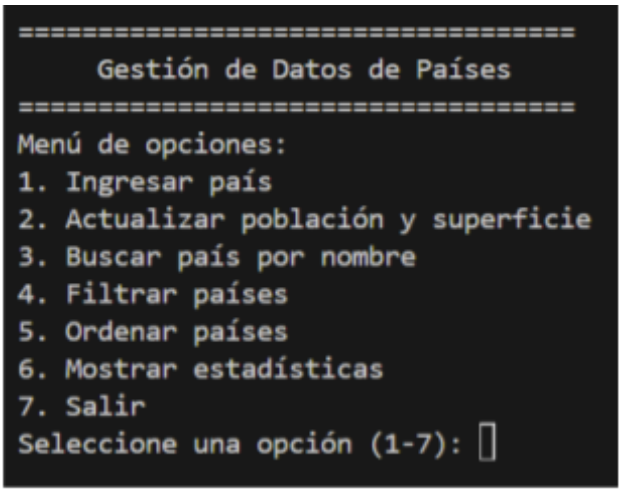
Los resultados se analizaron y se ajustaron aspectos como validaciones, formato de salida y presentación de resultados estadísticos.

Finalmente, se elaboró un archivo README que incluye la descripción del programa, instrucciones de uso, ejemplos de entrada/salida y los integrantes del proyecto.

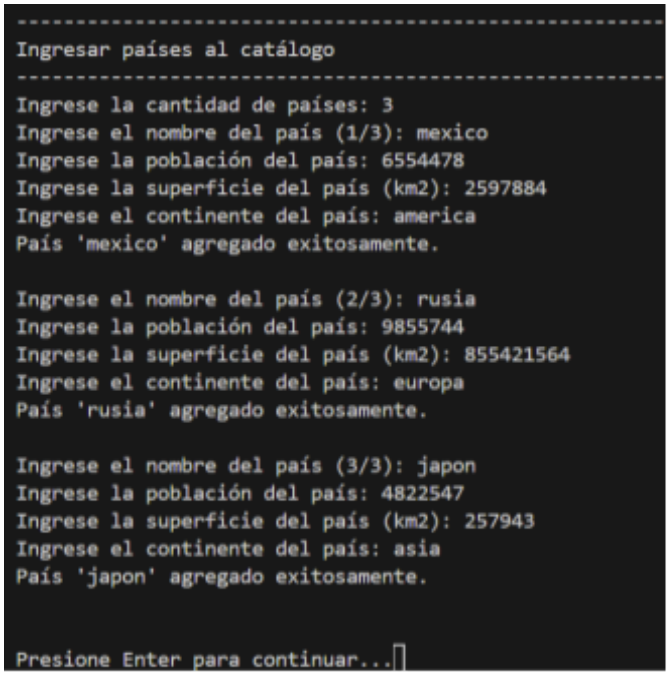
Esta metodología permitió garantizar orden, claridad y eficiencia durante todas las etapas del proyecto, desde la planificación hasta la entrega final.

RESULTADOS Y EVALUACIÓN

Capturas de ejecución del programa:



Vista de menú de opciones del programa "Gestión de Datos de Países"



Opción 1:Ingreso de países al catálogo con su respectiva población, superficie y continente

```
------------------------------------------------------------
Actualizar población y superficie de un país
------------------------------------------------------------
Ingrese el pais a actualizar:japon
Ingrese la población del país: 9985472
Ingrese la superficie del país (km²): 5687742
Población y Superficie actualizadas exitosamente!

Presione Enter para continuar...
```

Opcion 2: Permite actualizar las cantidades de población y superficie de los paises ya ingresados

```
------------------------------------------------------------------------
Filtrar países por continente, población o superficie
------------------------------------------------------------------------
1. Filtrar por continente
2. Filtrar por rango de población
3. Filtrar por rango de superficie
Seleccione una opción (1-3): 1



------------------------------------------------------------------------
Filtar paises por continente
------------------------------------------------------------------------
Ingrese el continente para filtrar: asia


NOMBRE                    POBLACION   SUPERFICIE(km²)   CONTINENTE
------------------------------------------------------------------------
India                      84000041              9541         Asia
China                    1409670000           9596961         Asia
Corea del sur              51780000            100210         Asia
Japon                       9985472           5687742         Asia

Presione Enter para continuar...
```

Opcion 4: Filtrado de países por Continente, ejemplo “Asia”

```
--------------------------------------------------------------------
Filtrar países por continente, población o superficie
--------------------------------------------------------------------
1. Filtrar por continente
2. Filtrar por rango de población
3. Filtrar por rango de superficie
Seleccione una opción (1-3): 2



--------------------------------------------------------------------
Filtar paises por rango de poblacion
--------------------------------------------------------------------
Ingrese el valor mínimo de población: 1000
Ingrese el valor máximo de población: 15000


NOMBRE                    POBLACION   SUPERFICIE(km²)   CONTINENTE
--------------------------------------------------------------------
Chile                          6554              2547      America
Francia                        4558              1587       Europa

Presione Enter para continuar...
```

Filtrado de países por Población, indicando rangos mínimos y máximos

```
--------------------------------------------------------------------
Filtrar países por continente, población o superficie
--------------------------------------------------------------------
1. Filtrar por continente
2. Filtrar por rango de población
3. Filtrar por rango de superficie
Seleccione una opción (1-3): 3



--------------------------------------------------------------------
Filtar paises por rango de superficie
--------------------------------------------------------------------
Ingrese el valor mínimo de superficie: 500
Ingrese el valor máximo de superficie: 20000


NOMBRE                    POBLACION   SUPERFICIE(km²)   CONTINENTE
--------------------------------------------------------------------
Chile                          6554              2547      America
India                      84000041              9541         Asia
Francia                        4558              1587       Europa
Argentina                  45000000              7000      America

Presione Enter para continuar...
```

Filtrado de países por superficie

```
--------------------------------------------------------------------------------
Ordenar Paises
--------------------------------------------------------------------------------
Seleccione la columna por la que desea ordenar:
1. Nombre
2. Población
3. Superficie
Opción (1-3): 1

Seleccione el orden:
1. Ascendente
2. Descendente
Opción (1-2): 2


NOMBRE                    POBLACION   SUPERFICIE(km²)   CONTINENTE
--------------------------------------------------------------------------------
Sudafrica                  60414495           1221037       Africa
Rusia                       9855744         855421564       Europa
Reino unido                67886011            243610       Europa
Peru                       34125320           1285216      America
Nigeria                   223804632            923768       Africa
Mexico                      6458245           2352144      America
Japon                       9985472           5687742         Asia
Italia                     59554023            301340       Europa
India                     840000041              9541         Asia
Francia                        4558              1587       Europa
España                     47420000            505990       Europa
Egipto                    109262178           1002450       Africa
Corea del sur              51780000            100210         Asia
Colombia                   51520000           1141748      America
China                    1409670000           9596961         Asia
Chile                          6554              2547      America
Canada                     38929902           9984670      America
Brasil                    212559417           8515767      America
Australia                  25687041           7692024      Oceania
Argentina                  45000000              7000      America
Alemania                   83240000            357022       Europa

Presione Enter para continuar...
```

Opcion 5: Ordenamiento de paises por Nombre y orden Descendente

```
--------------------------------------------------------------------------------
Ordenar Países
--------------------------------------------------------------------------------
Seleccione la columna por la que desea ordenar:
1. Nombre
2. Población
3. Superficie
Opción (1-3): 2

Seleccione el orden:
1. Ascendente
2. Descendente
Opción (1-2): 1


NOMBRE                    POBLACION   SUPERFICIE(km²)   CONTINENTE
--------------------------------------------------------------------------------
Francia                        4558              1587       Europa
Chile                          6554              2547      America
Mexico                      6458245           2352144      America
Rusia                       9855744         855421564       Europa
Japon                       9985472           5687742         Asia
Australia                  25687041           7692024      Oceania
Peru                       34125320           1285216      America
Canada                     38929902           9984670      America
Argentina                  45000000              7000      America
España                     47420000            505990       Europa
Colombia                   51520000           1141748      America
Corea del sur              51780000            100210         Asia
Italia                     59554023            301340       Europa
Sudafrica                  60414495           1221037       Africa
Reino unido                67886011            243610       Europa
Alemania                   83240000            357022       Europa
India                     840000041              9541         Asia
Egipto                    109262178           1002450       Africa
Brasil                    212559417           8515767      America
Nigeria                   223804632            923768       Africa
China                    1409670000           9596961         Asia

Presione Enter para continuar...
```

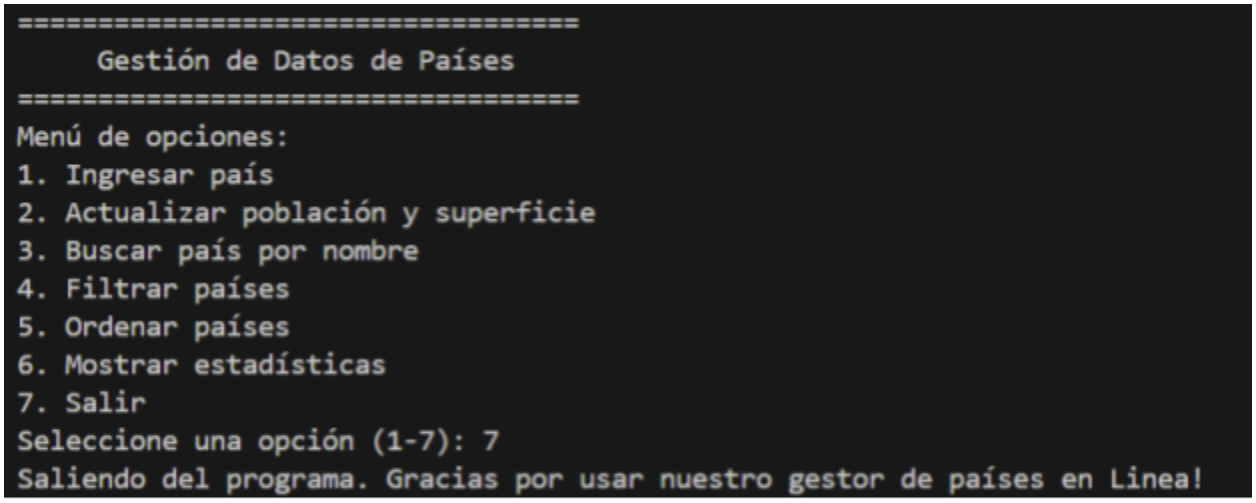
Ordenamiento de países por Población y orden Ascendente

```
------------------------------------------------------------------
Ordenar Países
------------------------------------------------------------------
Seleccione la columna por la que desea ordenar:
1. Nombre
2. Población
3. Superficie
Opción (1-3): 3

Seleccione el orden:
1. Ascendente
2. Descendente
Opción (1-2): 1


NOMBRE                    POBLACION   SUPERFICIE(km²)   CONTINENTE
------------------------------------------------------------------
Francia                        4558              1587       Europa
Chile                          6554              2547      America
Argentina                  45000000              7000      America
India                      84000041              9541         Asia
Corea del sur              51780000            100210         Asia
Reino unido                67886011            243610       Europa
Italia                     59554023            301340       Europa
Alemania                   83240000            357022       Europa
España                     47420000            505990       Europa
Nigeria                   223804632            923768       Africa
Egipto                    109262178           1002450       Africa
Colombia                   51520000           1141748      America
Sudafrica                  60414495           1221037       Africa
Peru                       34125320           1285216      America
Mexico                      6458245           2352144      America
Japon                       9985472           5687742         Asia
Australia                  25687041           7692024      Oceania
Brasil                    212559417           8515767      America
China                    1409670000           9596961         Asia
Canada                    38929902           9984670      America
Rusia                      9855744         855421564       Europa

Presione Enter para continuar...
```

Ordenamiento de países por Superficie y orden Ascendente

```
------------------------------------------------------------------
Estadísticas de Países
------------------------------------------------------------------

MAYOR Y MENOR POBLACIÓN:
- País con mayor población: China (1,409,670,000 habitantes)
- País con menor población: Francia (4,558 habitantes)

PROMEDIOS:
- Promedio de población: 125,293,506.33 habitantes
- Promedio de superficie: 43,159,711.33 km²

CANTIDAD DE PAÍSES POR CONTINENTE:
- America: 6 países
- Asia: 4 países
- Europa: 6 países
- america: 1 países
- Oceania: 1 países
- Africa: 3 países


Presione Enter para continuar...
```

Opción 6: Muestra Estadísticas de todos los países del archivo con datos como: Mayor y Menor Población, Promedios de Población y Superficie y Cantidad de Países por Continente.



Opción 7: Salir del programa

Qué cosas funcionaron bien en el proyecto:

El programa logró cumplir con todas las funcionalidades previstas: carga de países, actualización de datos, búsqueda, filtrado, ordenamiento y estadísticas.

La estructura modular facilitó la comprensión del código, y el manejo de archivos CSV funcionó de manera correcta, permitiendo conservar los datos entre ejecuciones.

Además, el menú principal resultó intuitivo y fácil de usar, ofreciendo una navegación clara para el usuario.

Qué fue difícil o qué presentó un desafío mayor:

El principal desafío fue organizar la lectura y escritura en el archivo CSV, garantizando que los datos se mantuvieran consistentes al agregar o modificar países.

También resultó complejo implementar las validaciones para evitar errores de ingreso (por ejemplo, campos vacíos o datos no numéricos).

Otro punto que exigió atención fue el filtrado y ordenamiento, donde se trabajó con distintos criterios (población, superficie, continente).

Si hubo errores o decisiones que cambiaron durante el desarrollo:

Inicialmente, el programa solo permitía búsquedas exactas, pero luego se implementó la búsqueda parcial para mejorar la experiencia del usuario.

También se modularizó la estructura de funciones para hacer el código más reutilizable y legible.

Durante las pruebas se detectaron algunos errores de formato y validación, que se corrigieron implementando controles adicionales.

Cómo fue la comunicación del equipo:

La comunicación fue constante y colaborativa. Se utilizaron herramientas como mensajería instantánea y documentos compartidos para coordinar avances.

Cada integrante asumió una parte del proyecto, y se revisó en conjunto la integración final y las pruebas antes de la entrega.

CONCLUSIONES

Con el desarrollo de este proyecto pudimos integrar y aplicar los conceptos teóricos aprendidos a lo largo del cuatrimestre, especialmente en el manejo de archivos CSV en Python, el uso de estructuras de control y la modularización del código mediante funciones.

A través de estas herramientas, logramos diseñar un programa funcional y organizado que permite gestionar información de manera dinámica y eficiente.

El trabajo en equipo fue un aspecto clave, ya que permitió coordinar tareas de programación, diseño de diagramas y elaboración de la documentación.

El uso de Python no solo facilitó la implementación lógica del programa, sino que también posibilitó representar visualmente el flujo de procesos y documentar el trabajo de manera profesional.

En conjunto, este proyecto representa una experiencia completa de aprendizaje, en la que la

organización, la colaboración y la aplicación práctica de la teoría se combinaron para lograr un resultado claro, funcional y accesible para cualquier usuario.

LINK DEL VIDEO Y REPOSITORIO DE GITHUB

https://youtu.be/fuh3kSpegFo
https://github.com/Nadia-Anahi-Garcia/UTN-Trabajo_Integrador_Programacion_1#

BIBLIOGRAFÍA

- Material de Cátedra (s.f.). *PDF Funciones - Programación I* . UTN.
- Material de Cátedra (s.f.). *PDF Listas - Programación I* . UTN.
- Material de Cátedra (s.f.). *U5_listas.ipynb*. UTN.
- Python Software Foundation. *Python 3.x Documentation* (Documentación oficial). Recuperado de https://docs.python.org/3/