Trabajo Práctico Integrador Programación I

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

Alumnos : Marcelo Hernan Fleitas

Jonathan Nicolas Leiva

Profesores Titulares

Cinthia Rigoni

Sebastian Bruselario

Profesores Tutores

Ana Mutti

Flor Gubiotti

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Programación I

16 de Noviembre de 2025

Introducción

El siguiente trabajo presenta el marco teórico y los conceptos de programación aplicados en el desarrollo del Trabajo Práctico Integrador de la materia Programación I. El objetivo es describir las estructuras de datos y herramientas de Python, que se utilizaron para construir la aplicación de gestión de datos de países.

Objetivo del Trabajo

El objetivo principal del trabajo es elaborar una aplicación que gestione los datos principales de países incluyendo: nombre, población, superficie y continente en la plataforma Python 3.13 permitiendo al usuario en una interfaz sencilla realizar las operaciones: -añadir países; -actualizar datos de países existentes (población o superficie); - búsqueda de países por nombre con coincidencia parcial o total; - filtrado por continente, rango de población y superficie; - ordenado por nombre, población y superficie (ascendente y descendente); - estadísticas de población (país con mayor y menor población), promedio de superficie y población, cantidad de países por continente.

Desarrollo

En el presente trabajo se aplica el uso de listas, diccionarios, funciones, condicionales, y manejo de archivos CSV que se dictaron como contenidos obligatorios durante el cursado de la asignatura Programación I de la carrera Programación a Distancia de la Universidad Tecnológica Nacional donde desarrollamos los temas citados anteriormente en la plataforma Python 3.13 a partir del material teórico y didáctico comprendido en el programa de la materia.

Las listas son un tipo de estructura de datos que permiten trabajar con cualquier tipo de datos, y que además permite la modificación o sobreescritura de datos brindando un amplio abanico de opciones para trabajar sobre dichos datos, aplicando operaciones como concatenación y multiplicación de los parámetros muy útiles para la expresión de métricas y valores estadísticos como sumatoria, promedio, valor medio, mediana, valores máximos y mínimos. Dentro de las categorizaciones de listas existen las listas de diccionarios que se definen como: lista: [{"nombre": "Argentina", "población": 45376763, "superficie": 2780400, "continente": "America }] y va a ser nuestra lista principal con la que trabajará nuestro código. Al trabajar con una lista utilizamos principalmente la función .append agregando en nuestra lista “países” las distintas actualizaciones de países que ingrese el usuario.

Los diccionarios son otro tipo de estructuras de datos que utilizan la condición clave → valor, donde la palabra clave sirve de referencia para el ingreso de datos como valores. Esto implica un ordenamiento donde la clave sirve de referencia para el conjunto de datos ingresados como valores. Un ejemplo de ello es: países={“nombre”: “Brasil, “población”: 203.080.756, “superficie": 8.515.770, "continente": "America } donde las claves que utilizaremos serán: “nombre”, “población”, “superficie”, “continente”.

Las funciones son bloques de código que ejecutan una tarea específica y permite transformar una operación en un módulo que se puede llamar y ejecutar en múltiples partes sin

repetir todo el bloque sino llamando la función que retorna el resultado de ésta. Para el presente trabajo utilizamos múltiples funciones tanto built-in que están incluidas en Python y otras que definimos en nuestro código y luego las llamamos o utilizamos su argumento para otra cadena de acciones entre funciones o eventos por ejemplo para validar entradas de datos en los distintos parámetros como el nombre de un país y lo guardamos en el retorno de la función, si el nombre está vacío o no es válido lo guardamos como None que en Python genera un False por lo que no continua la ejecución de otras funciones que necesitan True para continuar con la secuencia de código. También incluimos el uso de la función .refullmatch() del módulo re que asegura el uso de caracteres válidos evitando símbolos y otros argumentos inválidos para el nombre de países para los nombres de países en español.

Los condicionales son sentencias de código que como su nombre lo indica condicionan la salida que continuará el código de acuerdo con una condición específica que se establece como parámetro para plantear distintos escenarios de ejecución en consecuencia del valor que surja de la condición. Es decir, "Si tal condición se cumple" el código resuelve el escenario planificado para esa circunstancia, sino sigue con "elif" que busca otra condición complementaria a la anterior y si ninguna de las condiciones incluidas en "if" o "elif" coinciden sigue el curso hasta "else" donde en español es igual a decir de otra manera y ejecuta el código consiguiente. Es importante también aplicar conectores lógicos que nos permiten agrupar condiciones o separar las mismas para una simplificación mayor y un correcto funcionamiento del código. En nuestro código utilizamos los conectores lógicos AND, OR y NOT que sirven de funciones lógicas de conjunción, disyunción y negación de las variables proposicionales. Un ejemplo de ello es:

```
if poblacion < 0 or superficie <= 0:
    return "Error: población debe ser ≥ 0 y superficie > 0."
```

Donde nos aseguramos de que el valor ingresado en población sea mayor a o igual a 0 o también la superficie sea mayor a 0 para validar la entrada del usuario y asegurarnos de que no se ingresen números enteros negativos serían un error ya que no existen países con poblaciones o superficies con valores negativos.

Para ordenamientos en Python se logra ordenar los datos a partir de las funciones .sort() y .sorted(). En el presente trabajo utilizamos .sorted () para ordenar los datos de la población y superficie. Y la función sort para ordenar listas como continente. Para sorted() definimos la variable obtener_poblacion(país) usando key=país y devolviendo el valor para ese país. Decidimos usar una función que se encargue de esa tarea porque también estaba la opción de usar funciones lambdas para simplificarlo, en cambio para facilitar su entendimiento y por cuestiones didácticas preferimos aplicar una solución más simple y adaptable a la consigna que vuelve más legible al código.

Las estadísticas básicas se calculan aplicando la fórmula de media aritmética donde se realiza la sumatoria de los valores de países utilizando el bucle (for) que recorre todos los países para luego dividirlo por la cantidad de países usando la función len(), en cambio para ver los países con mayor cantidad y menor cantidad de población usamos otra vez un bucle (for) que compara el país con las variables de referencia mayor y menor sobrescribiéndolo si es mayor o menor guardando los extremos de valores (mayor y menor).

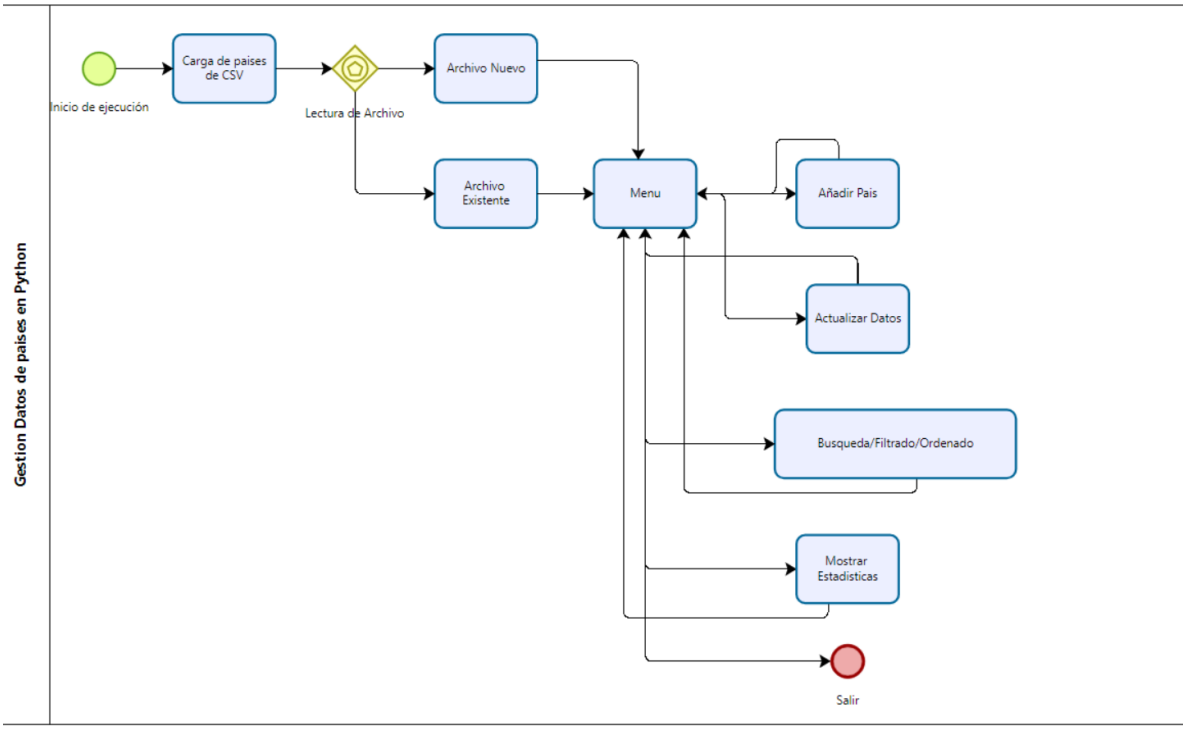
Por último, trabajamos con archivos csv para acceder a la base de datos donde se incluyen los distintos valores de información como, por ejemplo, lista: Argentina, 45376763, 2780400, America y aplicamos diferentes métodos para la lectura y escritura que aseguran el correcto funcionamiento de la aplicación y la persistencia en el guardado de datos manteniendo siempre un archivo confiable y sincronizado con las últimas modificaciones por parte del usuario.

Para lectura y escritura utilizamos el módulo de librería pathlib incluido en path que abre y cierra el archivo para luego escribir y leer los datos incluidos utilizando DictReader() junto con Dictwriter() que es la herramienta más eficaz dentro de Python para usar en diccionarios. Con ello nos aseguramos del formato del archivo respetando el formato de encabezado en los países con los campos (nombre, población, superficie, continente) para luego leer y escribir en las filas y columnas del archivo respetando el mismo formato para una interpretación correcta y exacta de cada dato que necesitamos incluir dentro del archivo csv. Se utilizan las funciones específicas escritor.writeheader que escribe los encabezados y escritor writerow. que escribe los datos en la misma fila respetando las columnas para mantener la consistencia de las entradas de datos en el archivo.

Diseño del caso práctico

A partir del análisis de la consigna y en consideración con las funcionalidades principales realizamos el siguiente diagrama para visualizar con claridad y simplicidad las distintas tareas que debemos desarrollar en el código.

Diagrama Principal de Aplicación



A partir de ello dividimos el código para modularizar y contemplar los distintos bloques de ejecución que tendrá la aplicación, finalmente decidiéndonos por la siguiente estructura: -

Utilidades generales (validaciones, entrada y salida de archivo csv; - núcleo de operaciones (donde aplicamos funciones de búsqueda y agregamos países a la base de datos); - Filtros y ordenamientos ; - Interfaz de consola; - claves de orden (donde agrupamos para ordenamiento de datos); - estadísticas.

Respecto de las estructuras de datos y los distintos módulos aplicados en el presente trabajo los describimos en el marco teórico abordado anteriormente donde desarrollamos la teórica y práctica aplicada en la aplicación tomando siempre en consideración la óptima ejecución por parte del usuario de una aplicación eficiente e intuitiva para un uso adecuado con la base de datos países.csv .

Resultados y Evaluación

El presente proyecto nos permitió afianzar y poner en práctica los contenidos aprendidos durante el cursado de programación I utilizando como base sólida los conceptos y teorías abordados durante el primer cuatrimestre además de investigar y ampliar nuestras herramientas y entendimiento de distintas funciones propias o de librerías de terceros implementadas en Python lo que nos permitió incorporar características y optimizaciones como el caso del uso de la función re full match para el uso de frases y palabras regulares garantizando entradas válidas y seguras en los nombres de países al impedir el ingreso de otros caracteres o símbolos que no forman parte de los nombres en español de los distintos países. Otra investigación por nuestra parte incluye el uso de pathlib que nos permite un manejo más seguro y eficiente de la ruta del archivo csv en los distintos sistemas operativos contrastando con la forma anterior de trabajarlo utilizando el modulo os también de Python donde se procesa de la ruta del archivo como un objeto, permitiendo cadenas de operaciones mas legibles y aplicando las diferentes sintaxis necesarias para los sistemas operativos sin complicaciones.

Conclusiones

Como conclusión del presente trabajo podemos decir que además de aplicar y fortalecer nuestro entendimiento de los distintos temas de la asignatura programación I que desarrollamos en el cursado además aprendimos que el trabajo en equipo y la comunicación entre compañeros afianza y fortalece la productividad, la crítica académica y profesional, y el crecimiento personal de habilidades blandas que complementan y mejoran el desarrollo de proyectos de mayor extensión y complejidad donde se requiere de aplicar con constancia elaboración, ejecución, prueba y testeo del código.

Bibliografía

- **Downey A. (2015)** Pensar en Python, Aprende a pensar como un informático **(2nd. Ed. Ver.2.4.0). Green Tea Press**
- **Python Software Foundation. (s.f.).** *Modelo de datos* (Versión 3.x). Documentación de Python. Recuperado de https://docs.python.org/es/3/reference/datamodel.html#modules
- **Python Software Foundation. (s.f.).** * csv — CSV File Reading and Writing(Versión 3.x). Documentación de Python. Recuperado de https://docs.python.org/es/3/library/csv.html
- **Python Software Foundation. (s.f.).** *re — Operaciones con expresiones regulares* (Versión 3.x). Documentación de Python. Recuperado de https://docs.python.org/es/3/library/re.html#re.Pattern.match