UTN
FACULTAD REGIONAL MENDOZA
UNIVERSIDAD TECNOLÓGICA NACIONAL

(TPI)

Trabajo Práctico Integrador

marco teórico

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

Integrantes: Acosta Cristina, Lucero Abril

Profesor: Ariel Enferrel

Carrera: TUP

Cátedra: Programación I

Introducción

El presente Trabajo Práctico Integrador (TPI) tiene como objetivo aplicar de manera integral los conocimientos adquiridos en la asignatura Programación I, desarrollando un sistema en Python orientado a la gestión, análisis y procesamiento de datos sobre países.

El proyecto propone una solución modular que permita obtener, organizar y analizar información real proveniente de la API REST Countries, haciendo uso de estructuras de datos, lectura y escritura de archivos CSV, y técnicas de filtrado, ordenamiento y estadísticas.

A partir de la interacción del usuario mediante un menú principal, el sistema posibilita realizar distintas operaciones sobre el conjunto de datos, tales como:

Búsqueda de países por nombre (completo o parcial).

Filtrado según continente o rangos poblacionales.

Ordenamiento alfabético o numérico (por población o superficie).

Estadísticas y promedios (cantidad de países por continente, promedios, máximos y mínimos).

La aplicación fue desarrollada de forma conjunta entre Cristina Acosta y Abril Lucero, aplicando los conocimientos adquiridos hasta el momento y enriqueciendo los mismos, en el proceso de investigación.

Trabajamos de manera colaborativa utilizando un enfoque estructurado y modular, en donde cada módulo cumple una responsabilidad específica. Esta organización nos facilita la mantenibilidad del código, la legibilidad y el trabajo en equipo.

El resultado final es una herramienta práctica y funcional que integra conceptos como la comprensión del uso de APIs REST y del formato JSON, reforzando la conexión entre la teoría de estructuras de datos y su aplicación.

En síntesis, el desarrollo permitió afianzar el uso de listas, diccionarios, estructuras condicionales y repetitivas, junto con la implementación de funciones para lograr un programa claro, eficiente y reutilizable, cumpliendo con los objetivos pedagógicos de la materia y evidenciando un resultado final completo y funcional.

Desarrollo del proceso

API.py

El script API.py constituye la base de nuestro proyecto para obtener y manipular datos de países a través de la API REST (URL= "https://restcountries.com/v3.1 /all?fields=name,population,area,continents"). En API.py el principal objetivo es extraer información que necesitamos para el trabajo (nombre,área,población y continente) de manera automatizada y transformarla en un formato manejable para posteriores análisis.

Para lograr esto, importamos y utilizamos los modulos requests y csv de Python:

requests — Python's Requests Library

Esta biblioteca nos permite realizar solicitudes HTTP de manera sencilla, lo que nos facilita interactuar con la API de Rest Countries y obtener información sobre todos los países (respuesta = requests.get(URL, timeout=5))

csv — CSV File Reading and Writingse

El módulo csv implementa clases para leer y escribir datos tabulares en formato CSV lo que nos permite almacenar los datos de manera estructurada en un archivo que puede ser fácilmente leído o procesado por otros programas.

El flujo del script se desarrolla en dos funciones principales:

paises_lista()

Realiza una solicitud HTTP (GET) a la URL de la API, filtrando únicamente los campos requeridos: nombre, población, área y continente.

Convierte la respuesta en formato JSON, que es más fácil de manipular en Python.

Retorna una lista de diccionarios, donde cada diccionario representa un país con sus atributos.

paises_csv(paises)

Recibe la lista de países obtenida por paises_lista().

Utiliza la librería csv para escribir esta información en el archivo paises.csv, creando un registro por país con sus datos correspondientes.

Utilizamos csv.DictWriter, .writeheader, .writerow.

De este modo, API.py cumple con el objetivo de extraer datos de forma automatizada, transformarlos y almacenarlos en un formato estructurado, lo que permite su posterior utilización en análisis, filtrado u ordenamiento en otros módulos del proyecto.

\MODULOS\

La carpeta Módulos contiene un conjunto de scripts con funciones específicas orientadas al procesamiento, filtrado y análisis de los datos obtenidos desde la API. Su propósito es organizar el código de manera modular, separando las funcionalidades en distintos archivos para mantener una estructura clara y reutilizable.

Cada módulo cumple una función puntual dentro del proyecto y, en conjunto, complementan el funcionamiento general al integrarse con el menú principal. De esta forma, el proyecto favorece a la comprensión y escalabilidad del código.

Buscar_por_nombre.py

Dentro de esta carpeta, el primer script desarrollado fue Buscar_por_nombre.py, encargado de implementar una función que permite buscar información sobre un país específico dentro de la lista de países obtenida desde la API.

El script define una única función:

buscar_pais(paises, nombre)

Recibe dos argumentos: la lista de países (paises), proveniente de la función paises_lista() del script API.py, y el nombre del país (nombre) que ingresa el usuario a través de una opción del menú.

Su tarea es recorrer la lista y comparar los nombres para encontrar coincidencias, devolviendo los datos del país solicitado.

En esta etapa del desarrollo no fue necesario importar librerías externas, ya que la función trabaja únicamente con estructuras básicas de Python y los datos ya cargados desde la API.

Antes de integrar el módulo con el menú principal, se incluyó un bloque de prueba con if __name__ == "__main__": para verificar la correcta funcionalidad de la función de forma independiente. Esto permitió comprobar que la búsqueda respondía correctamente a diferentes entradas.

funciones_filtrado.py

El script funciones_filtrado.py amplía las posibilidades de análisis del proyecto al incorporar herramientas de filtrado de datos, permitiendo al usuario obtener subconjuntos de países según distintos criterios.

En este módulo se desarrollaron dos funciones principales:

filtrar_continente(paises, continente)

Esta función permite seleccionar únicamente los países pertenecientes a un continente determinado.

Recibe como parámetros la lista de países (obtenida desde API.py) y el nombre del continente, que es ingresado por el usuario a través del menú principal.

Compara el valor recibido con el campo “continente” de cada diccionario dentro de la lista, generando un nuevo conjunto de datos que contiene únicamente los países del continente solicitado.

filtrar_valor(paises, valor, min_p, max_p)

Esta función permite realizar un filtrado numérico según un rango determinado, ya sea por población o por área.

Recibe como argumentos la lista de países, el tipo de valor a evaluar (población o área), y los límites del rango (min_p y max_p).

Su diseño es flexible, ya que la misma función se reutiliza en diferentes opciones del menú, adaptándose al tipo de dato que el usuario elija analizar.

funciones_ordenamiento.py

El script funciones_ordenamiento.py incorpora al proyecto la posibilidad de ordenar los países según distintos criterios, como población, área o nombre. Este tipo de procesamiento permite analizar los datos de forma más organizada y comparativa, contribuyendo al objetivo general del proyecto de facilitar la exploración y comprensión de la información proveniente de la API.

En este módulo se definió una única función:

ordenar_paises(lista_paises, key, orden)

Recibe como argumentos la lista de países, el criterio de ordenamiento (key), y el tipo de orden (ascendente o descendente).

Su función es aplicar un método de ordenamiento sobre la lista utilizando el valor del campo seleccionado por el usuario (por ejemplo, “population” o “area”).

Para implementar esta funcionalidad, se importó la herramienta itemgetter desde el módulo operator, la cual permite acceder de forma eficiente a los valores de una clave específica dentro de cada diccionario de la lista. Esta fue una incorporación clave, ya que simplifica el código y mejora la legibilidad.

Durante el desarrollo de esta parte del proyecto, fue necesario investigar y aplicar nuevos conceptos de Python, como:

Operator.itemgetter para acceder a las claves de diccionario.

try–except y KeyError, para manejar posibles errores que podrían surgir si alguna clave no existía en el diccionario.

El uso de sorted(), que permite generar una nueva lista ordenada sin modificar la original.

Si bien son temas que el grupo continuará reforzando, la implementación de esta función permitió un primer acercamiento práctico al manejo de errores y al ordenamiento programático de datos, habilidades esenciales en el trabajo con grandes volúmenes de información.

estadisticas.py

El script estadisticas.py se encarga de generar distintos análisis estadísticos sobre la información de los países, aprovechando las funciones de ordenamiento y filtrado previamente desarrolladas.

En este módulo se definieron tres funciones principales:

mayor_menor_poblacion(paises)

Recibe como parámetro la lista de países proveniente del menú principal.

Para determinar el país con mayor y menor población, reutiliza la función ordenar_paises() del módulo funciones_ordenamiento, ordenando la lista de manera ascendente según la clave "poblacion".

Posteriormente, imprime el primer y el último elemento de la lista, correspondientes al país con menor y mayor población respectivamente.

Promedio (paises, valor)

Permite calcular el promedio de un atributo seleccionado por el usuario, como población o área.

Acumula los valores del campo elegido dentro de un ciclo y divide la suma total entre la cantidad de países procesados.

cantidad_paises(paises)

Esta función contabiliza la cantidad de países por continente.

Para ello, se construye un diccionario cuyas claves representan los nombres de los continentes y los valores indican la cantidad de países pertenecientes a cada uno.

Durante el proceso, se optó por dividir América en dos categorías (“North America” y “South America”) para lograr una representación más precisa.

A medida que se recorren los países en el bucle for, se incrementa el contador correspondiente a cada continente, generando un resumen final.

En conjunto, estas tres funciones ponen en práctica conceptos como la reutilización de funciones, la organización de datos en estructuras dinámicas y el uso de cálculos agregados, consolidando el enfoque modular del proyecto y dotan al proyecto de una capacidad analítica más profunda, permitiendo transformar la información cruda obtenida de la API en resultados estadísticos significativos.

menú.py

El script menú.py constituye el punto de entrada principal del proyecto, integrando todas las funciones y módulos previamente desarrollados en un sistema interactivo para el usuario. Su objetivo es orquestar la ejecución de las distintas funcionalidades, permitiendo seleccionar qué operación realizar sobre los datos de países obtenidos desde la API.

El menú presenta cinco opciones principales:

Buscar país por nombre

Llama a la función buscar_pais() del módulo Buscar_por_nombre.py.

Permite al usuario ingresar el nombre de un país y recibir la información correspondiente de la lista de países.

Filtrar países

Llama a las funciones filtrar_continente() y filtrar_valor() del módulo funciones_filtrado.py.

El usuario puede seleccionar un continente o un rango de valores (población o área) y obtener un subconjunto de países que cumplan con los criterios elegidos.

Ordenar países

Utiliza la función ordenar_paises() del módulo funciones_ordenamiento.py.

Permite al usuario organizar los países según un campo específico (nombre, población o área) y en orden ascendente o descendente.

Estadísticas

Integra las funciones mayor_menor_poblacion() y funcion_promedio() del módulo estadisticas.py.

Brinda información analítica, como el país con mayor y menor población, promedios de población o área, y cantidad de países por continente.

Salir

Permite finalizar la ejecución del programa.

Para implementar la lógica del menú, se utilizó match-case, lo que facilita el manejo de múltiples opciones y permite ejecutar las funciones correspondientes según la elección del usuario. De esta manera, todas las funciones modulares del proyecto se conectan de forma coherente y dinámica, brindando una experiencia interactiva y estructurada, donde el usuario puede explorar, filtrar, ordenar y analizar los datos sin necesidad de manipular directamente los scripts o los archivos de datos.

Conclusión

El desarrollo de este proyecto nos permitió fortalecer nuestras bases técnicas en Python y programación modular, comprendiendo la importancia de construir primero una estructura sólida antes de abordar tecnologías más avanzadas.

Al inicio, nuestro interés se centró en aprender herramientas como Flask, Docker, HTML y CSS, con la intención de generar una aplicación completa desde el principio. Sin embargo, al no contar con conocimientos previos sólidos en programación y manejo de datos, gran parte del tiempo se destinó a intentar comprender conceptos avanzados sin tener aún la base necesaria, lo que resultó en frustración y retrasos.

Esta experiencia nos enseñó que es fundamental dominar los fundamentos antes de escalar hacia proyectos más complejos. Por ello, decidimos enfocarnos en lo esencial: extracción de datos desde una API, manipulación de listas y diccionarios, filtrado, ordenamiento y análisis estadístico, así como la construcción de un menú interactivo que integrara todas las funcionalidades de manera modular.

Al finalizar, logramos una estructura organizada, funcional y reutilizable, que no solo cumple con los objetivos planteados, sino que también sirve como base sólida para escalar el proyecto en el futuro. Con esta experiencia, adquirimos confianza en la programación estructurada y aprendimos a planificar el desarrollo paso a paso, priorizando primero los conceptos fundamentales y dejando la integración con tecnologías más avanzadas para un siguiente nivel de implementación.

WEBGRAFÍA

https://www.adobe.com/acrobat/resources/document-files/what-is-a-csv-file.html

https://requests.readthedocs.io/en/latest/

https://docs.python.org/es/3.8/library/operator.html

https://realpython.com/python-sort/

https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Status

https://docs.python.org/es/3/tutorial/modules.html

https://backend.turing.edu/module2/lessons/how_the_web_works_http

https://docs.python.org/es/3/library/re.html

https://www.freecodecamp.org/espanol/news/compresion-de-diccionario-en-python-explicado-con-ejemplos/