TRABAJO PRÁCTICO INTEGRADOR

PROGRAMACIÓN - COMISIÓN 4

INTEGRANTES: FACUNDO RAMÍREZ Y SANTINO GODOY – Grupo: 9

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

Marco Teórico

El presente trabajo se enmarca en el desarrollo de una aplicación en el lenguaje Python, orientada a la gestión y análisis de información de distintos países mediante el uso de estructuras de datos, funciones y técnicas de filtrado y ordenamiento.

Los conceptos fundamentales que se ocupan en nuestro código son:

- **Listas:** estructuras de datos ordenadas y dinámicas que permiten almacenar colecciones de elementos. En este proyecto se utilizan para guardar los registros de países obtenidos desde la API.
- **Diccionarios:** estructuras clave–valor que permiten un acceso rápido a los datos. Cada país se representa como un diccionario con las claves "nombre", "población", "superficie" y "continente".
- **Funciones:** se utilizan para modularizar el programa, dividiendo el código en tareas independientes, en diferentes carpetas y dejando el código limpio.
- **Condicionales y bucles:** permiten controlar el flujo del programa y ejecutar acciones repetitivas (por ejemplo, recorrer listas o validar entradas).
- **Ordenamientos:** a través de las funciones sorted(), min() y max() y funciones lambda para ordenar los países según nombre, población o superficie.
- **Estadísticas básicas:** permiten obtener indicadores como país con mayor o menor población, promedios y cantidades por continente.
- **Archivos CSV:** se utilizan para almacenar los datos descargados de la API y facilitar su posterior lectura desde el programa.

Fuentes

Conocimientos básicos aprendidos dentro del Campus UTN. Plataforma virtual 1er año, Programación I.

https://docs.python.org/es/3/ | Python Software Foundation (2024). The Python Tutorial.

Página Oficial de API pública (REST Countries).

Objetivos

Desarrollar una aplicación en Python que permita gestionar información sobre países, aplicando listas, diccionarios, funciones, estructuras condicionales y repetitivas, ordenamientos y estadísticas. El sistema debe ser capaz de leer datos desde un archivo CSV, realizar consultas y generar indicadores clave a partir del dataset. El objetivo principal es afianzar el uso de estructuras de datos, modularización con funciones y técnicas de filtrado/ordenamiento, aplicando los conceptos aprendidos en Programación 1.

Diseño

Informe teórico de los conceptos aplicados:

1. Listas

Una lista (o array en otros lenguajes) es una colección ordenada de elementos que pueden ser accedidos por índice, iterados, modificados, etc. En Python, una lista se define por ejemplo así: mi_lista = [1, 2, 3]. Lo aplicamos de la siguiente manera:

En el módulo api.py, tras obtener datos de la API de REST Countries API se puede almacenar una lista de países para luego procesar.

En el módulo estadisticas.py u ordenamiento.py, se manejan listas de registros (cada país) para filtrado, orden, etc. Ventajas: permiten iterar fácilmente, aplicar funciones sobre cada elemento, conservar orden. Cuestiones técnicas: acceso por índice lista[i], añadir elementos lista.append(x), ordenar lista.sort(), etc.

2. Diccionarios

Un diccionario es una estructura de datos de pares clave-valor.

En Python: mi_dic = {"clave": valor, ...}

Aplicación en el proyecto:

Cada país puede representarse como un diccionario con claves como "nombre", "población", "continente", etc.

Usar diccionarios permite acceder de forma directa por clave en lugar de por índice, lo cual mejora la legibilidad (ej: pais["poblacion"]). Ventajas: búsqueda rápida por clave, buena para representar entidades. Cuestiones técnicas: iterar sobre dic.items(), dic.keys(), comprobaciones con in, etc.

3. Funciones

Una función es un bloque de código que realiza una tarea específica, puede recibir parámetros y devolver resultados.

En Python:

def mi_funcion(param1, param2) Aplicación en el proyecto:

En busquedas.py, habrá una función para buscar país por nombre o parcial.

En ordenamiento.py, funciones que ordenan listas según un criterio (población, superficie, etc). Ventajas: modularidad, reutilización, aislamiento de lógica, mejor mantenimiento. Cuestiones técnicas: definir parámetros, valores de retorno, manejo de errores (ej: país no encontrado), documentación (docstrings).

4. Condicionales

Una condicional es una estructura que permite ejecutar un bloque u otro según una condición (if/else). En Python:

if condición: # hacer algo

else: # hacer otra cosa

Aplicación en el proyecto:

En el menú de main.py, decidir qué opción elige el usuario (buscar, filtrar, ordenar, estadísticas, salir).

En validaciones (validaciones.py), chequear que la entrada del usuario sea válida (por ejemplo, que el número de opción esté en el rango). Ventajas: permite flujo de control dinámico, respuesta a diferentes escenarios. Cuestiones técnicas: incluir elif para múltiples opciones, manejo de error con try/except combinado, asegurar cobertura de casos.

5. Ordenamientos

El ordenamiento se refiere a reorganizar los elementos de una colección (lista) de acuerdo con un criterio (por ejemplo, población ascendente, superficie descendente). Aplicación en el proyecto:

En ordenamiento.py se implementa la lógica para ordenar los países según distintos campos.

Puede utilizarse el método list.sort(key=…) o sorted(lista, key=…) en Python. Ventajas: facilita la presentación estética y lógica de los datos al usuario, mejora usabilidad. Cuestiones técnicas: decidir criterio (clave de diccionario), dirección (ascendente/descendente), estabilidad del ordenamiento, efectos de mutabilidad.

6. Estadísticas básicas

Se refiere al cálculo de métricas simples sobre un conjunto de datos: promedio, contar, máximo, mínimo, distribución, etc. Aplicación en el proyecto:

En estadisticas.py se calcula, por ejemplo, el promedio de población, cantidad de países por continente.

Puede usarse funciones de Python como sum(), len(), max(), min() o librerías como statistics. Ventajas: entrega valor agregado al usuario, permite interpretar datos en vez de solo mostrarlos. Cuestiones técnicas: manejar listas vacías (evitar división por cero), datos nulos, elegir tipo adecuado de métrica (media vs mediana), formateo de resultados.

7. Archivos CSV

CSV (“Comma-Separated Values”) es un formato de texto plano para representar tablas: cada línea es un registro, campos separados por comas o algún otro delimitador. Aplicación en el proyecto:

En api.py el programa descarga datos desde la API y los almacena en paises.csv para persistencia.

Luego los módulos de búsqueda, filtros, ordenamiento y estadísticas leen ese archivo CSV para trabajar con los datos sin depender de la API cada vez. Ventajas: persistencia simple, compatibilidad con hojas de cálculo, facilidad de implementación. Cuestiones técnicas: usar el módulo csv en Python, manejar separador, codificación, saltos de línea, encabezados, conversión de tipos (strings = ints/floats), validaciones de datos.

Metodología usada

Se generó un archivo CSV antes de las operaciones activas: en Datos/api.py se conecta a la API de REST Countries API, descarga datos y los guarda en paises.csv.

Luego en src/ se implementan módulos separados: búsqueda (busquedas.py), filtros (filtros.py), ordenamiento (ordenamiento.py), estadísticas (estadisticas.py), validaciones (validaciones.py), menú principal (main.py).

También se describe el trabajo en equipo: uso de ramas de Git (“main”, “rama_santino”, “rama_ramirez”) para desarrollo colaborativo, fusiones (merges) al finalizar.

División modular del código: cada funcionalidad (API, lectura CSV, búsqueda, filtro, ordenamiento, estadísticas, validaciones) está separada en su propio archivo/módulo por lo cual esto facilita el mantenimiento, la reutilización y la claridad del flujo.

Persistencia intermedia: al usar un CSV como capa de persistencia entre la fase de adquirir datos (API) y las operaciones posteriores, se reduce la dependencia de llamadas externas repetidas, y se opera sobre un conjunto de datos local.

Control de versiones y colaboración: uso de ramas por integrante para trabajar independientes sin afectar la rama principal, y luego merges para integración final.

Flujo estructurado de interacción con el usuario: un menú principal en main.py que dirige a diferentes operaciones por opción (búsqueda, filtro, ordenamiento, estadísticas, salir) lo que refleja una metodología de diseño centrada en el usuario que interactúa.

Validación del input del usuario: existe módulo validaciones.py para asegurarse que entradas sean correctas antes de ejecutar operaciones, lo que incrementa robustez y calidad del software.

Uso de estructuras de datos adecuadas: listas, diccionarios, funciones, condicionales, etc. como se explicó anteriormente.

Trabajo iterativo: Aunque no se describe explícitamente “iteraciones” de desarrollo, la estructura con ramas sugiere un ciclo de desarrollo: programar - probar - merge - refinamiento.

Resultados

¿Qué cosas funcionaron bien en el proyecto?

Lo que más resaltaba en el proyecto, fue la comunicación y disposición al realizar tareas dadas y desafíos por cumplir entre ambos durante el proceso de creación del código.

¿Qué fue difícil o que presentó un desafío mayor?

Creemos que no hubo ningún desafío mayor o que complicase la situación, pero lo que no queríamos era generar errores durante la realización del código ya que podíamos sobrescribir cosas del compañero de trabajo. Como la comunicación fue buena, eso no se llevó a cabo por lo que pudo realizarse de manera claro y organizada.

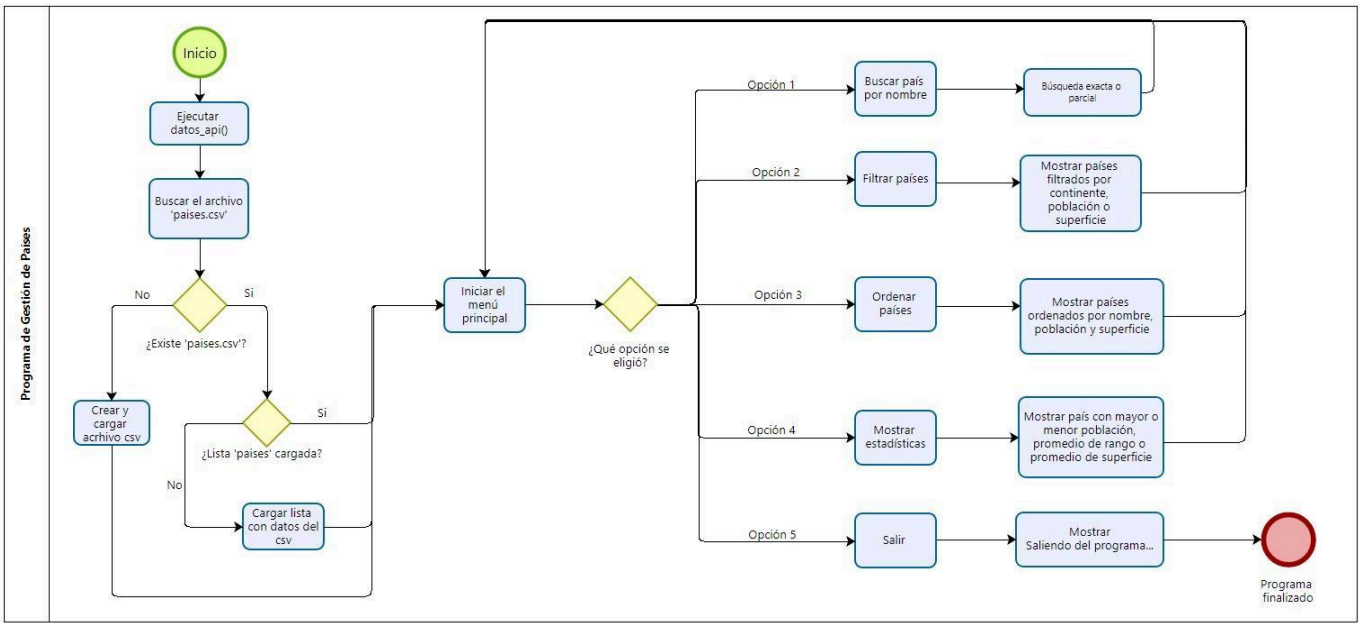
¿Hubo errores o decisiones que cambiaron durante el desarrollo?

Si bien no hubo errores críticos, a medida que fuimos desarrollando el proyecto, vimos la posibilidad de mejorar ciertas partes para mejorar el resultado.

¿Como fue la comunicación del equipo?

Ambos estuvimos comunicados constantemente para el desarrollo del proyecto, nos hacíamos consultas mediante meetings o llamadas, llegábamos a una elección para el resultado que creíamos conveniente.

IMÁGEN DE DIAGRAMA DE FLUJO DEL PROYECTO:



LINK DEL REPOSITORIO GITHUB:
https://github.com/SantiiGodoy/TPI_PRO4_GODOY_RAMIREZ.git

ENLACE VIDEO:
https://drive.google.com/file/d/1EnoxqcbZwKQwL_q816uenf_2ugBUJXPw/view?usp=sharing