Trabajo Práctico Integrador

Gestión de Datos de países en Python

Integrantes del Grupo:

1. **Cesar Luciano Angeleri - Comisión 1 –** lcnang@gmail.com
2. **Ronar Salazar Suzeta - Comisión 3 –** ronar76@gmail.com

Materia: Programación 1

Docente Titular Comisión 1 y 3: Cinthia Rigoni

Docente Tutor Comisión 1: Martin Garcia

Docente Tutor Comisión 3: Brian Lara

Fecha de Entrega: 11 de Noviembre de 2025

---

1. Objetivos

El objetivo principal de este Trabajo Práctico Integrador fue **desarrollar una aplicación de consola en Python que gestione datos geográficos de países**, sirviendo como plataforma para la aplicación y consolidación de los siguientes conocimientos adquiridos en Programación 1:

1. **Afianzar Estructuras de Datos:** Utilizar de **manera eficiente Listas como colecciones principales** y **Diccionarios como el modelo de registro** para cada país.
2. **Dominar la Modularización: Diseñar y construir un código claro y mantenible** a través de funciones que sigan el principio de **"una función, una responsabilidad"**.
3. I**mplementar Funcionalidades Clave:** Desarrollar las **operaciones de CRUD (Crear, Leer, Actualizar, Borrar) y las funcionalidades de filtrado, ordenamiento y cálculo de estadísticas** a partir de un **dataset** de datos **cargado** desde un **archivo CSV**.
4. **Garantizar la Robustez:** Incorporar **validaciones y control de flujo** adecuados para manejar errores de entrada del usuario y de formato del archivo.

2. Marco Teórico y Fundamentos Aplicados

El desarrollo del presente proyecto se fundamenta en conceptos esenciales de la programación estructurada y la gestión de datos en Python, los cuales aseguran la robustez, modularidad y eficiencia del sistema.

2.1. Estructuras de Datos Fundamentales

Listas

En Python, una **lista** es una **colección ordenada y mutable** que se utiliza para **almacenar colecciones de elementos**. En el TPI, **la lista es la estructura central** (denominada `paises`) que contiene todos los datos cargados desde el archivo CSV.

"Python tiene varios tipos de datos compuestos, utilizados para agrupar otros valores. El más versátil es la lista, la cual puede ser escrita como una lista de valores separados por coma (ítems) entre corchetes. Las listas pueden contener ítems de diferentes tipos, pero usualmente los ítems son del mismo tipo." *(Documentación Oficial de Python , 3. Una introducción informal a Python - 3.1.3. Listas)*.

Diccionarios de datos

Un **diccionario de datos,** es una **colección no ordenada, mutable e indexada por claves (pares clave-valor)**. **Cada país** en el sistema se modela como **un diccionario**, lo que permite un **acceso** semántico a sus **atributos** (`"nombre"`, `"poblacion"`, etc.).

"Es mejor pensar en un diccionario como un conjunto de pares clave:valor con el requerimiento de que las claves sean únicas (dentro de un diccionario)...Colocar una lista de pares clave:valor separada por comas dentro de las llaves añade pares clave:valor iniciales al diccionario;" ***(Documentación Oficial de Python , 5. Estructuras de datos - 5.5. Diccionarios)*****.**

2.2. Funciones y Modularización

La modularización es la **práctica** de **dividir** un **programa** en **módulos** o **funciones independientes, cada uno encargado de** una **tarea específica**, lo que facilita el desarrollo, la lectura y el mantenimiento.

"Una función es un bloque de código que solo se ejecuta cuando se la llama.Una función puede devolver datos como resultado.Una función ayuda a evitar la repetición de código." *(W3Schools, 2025, Python Functions).*

En `gestion_paises.py`, observamos una clara **modularización**, **con funciones** como:

- **Gestión de Datos:** `cargar_datos()`, `guardar_datos()`.
- **Lógica de Negocio:** `agregar_pais()`, `actualizar_pais()`, `buscar_pais()`.
- **Auxiliares:** `sin_acentos()`, `normalizar_continente()`, y las funciones `clave_nombre()`, `clave_poblacion()`, etc., utilizadas como `keys` en el algoritmo de ordenamiento `sorted()`.

2.3. Control de Flujo y Validación

El control de flujo **determina** el **orden** en que **se ejecutan las instrucciones** en un programa.

- **Estructuras Repetitivas y Condicionales:** Se emplean bucles `while True` para **mantener el menú principal activo** y estructuras `if/else` para la **validación y toma de decisiones**.

- **Sentencia** `match/case`**:** Esta característica moderna de Python (desde la versión 3.10) p**ermite una estructura de menú más legible y concisa** que el uso anidado de `elif`.
- **Validación y Robustez:** El código implementa **validaciones** como `isdigit()` para asegurar que las **entradas numéricas** (`poblacion` y `superficie`) **sean válidas antes de la conversión de tipos** (`int()),` garantizando la **robustez** del sistema y **evitando fallos** no controlados (`exceptions`). La **normalización de texto** (`sin_acentos`) se usa para que las **búsquedas sean insensibles** a mayúsculas y acentos.

2.4. Persistencia de Datos (CSV)

La persistencia de datos se refiere a la **capacidad** de un sistema para m**antener su información a lo largo del tiempo**, más allá de la ejecución del programa. Esto se logró mediante la **gestión de archivos CSV**.

"El módulo csv implementa clases para leer y escribir datos tabulares en formato CSV… Los programadores también pueden describir los formatos CSV entendidos por otras aplicaciones o definir sus propios formatos CSV para fines particulares." *(Documentación Oficial de Python, csv — CSV File Reading and Writing, sección 10.1.1).*

Las funciones `cargar_datos()` y `guardar_datos()` utilizan `csv.DictReader` y `csv.DictWriter`, lo que permite **mapear directamente las filas del archivo CSV a la estructura de datos interna** (Lista de Diccionarios).

3. Diseño del Caso Práctico

El diseño del caso práctico se centró en un ciclo de **desarrollo iterativo** que se prioriza la **modularidad** y la **usabilidad** en la **consola**.

3.1. Modelado de Datos

Cada país se representa internamente como un diccionario con claves y tipos de datos definidos:

```
{
    "nombre": str,
    "poblacion": int,
    "superficie": int,
    "continente": str
}
```

La colección completa de datos se almacena en la lista `paises`.

3.2. Flujo de Operaciones (Lógica General)

El sistema está diseñado para **ser dirigido por un menú principal iterativo**:

1. **Inicio**: El programa **comienza** ejecutando `main()`, que **llama inmediatamente** a `cargar_datos()` para **poblar** la lista **paises desde** el **archivo** `paises.csv`.
2. **Menú Principal**: El usuario **interactúa** mediante un **menú** con **opciones numéricas**.
3. **Operaciones CRUD**:
    - **Agregar País**: **Solicita** todos los **datos**, **normaliza entradas**, **verifica duplicados** y, si es exitoso, **agrega el diccionario** a `paises` y **llama** a `guardar_datos()`.
    - **Actualizar País**: Permite buscar un país existente y modificar su `población` y `superficie`.
    - **Buscar País**: Utiliza `prompt_toolkit` para **autocompletado y realiza una búsqueda por coincidencia parcial, insensible a acentos y mayúsculas**.
4. **Consultas/Estadísticas**: Las opciones de **Filtro, Ordenamiento y Estadística** operan sobre la lista `paises` y muestran los resultados mediante la **función formateada** `listar_paises()` (que utiliza la librería `tabulate`).
5. **Cierre**: La **opción** "7. Salir" **rompe** el **bucle principal**.

3.3. Librerías de Terceros

Se utilizaron **librerías externas** para **mejorar** la **experiencia** de **usuario** (**UX**) y la **presentación**:

- `tabulate:` Para **mostrar** los **resultados** de **listas**, **filtros** y **ordenamientos** en **formato de tabla de consola,** mejorando la legibilidad.
  (https://pypi.org/project/tabulate/)
- `colorama:` Para **añadir color a la salida de la consol**a (nombres de países por continente, rangos de población/superficie), lo que hace la **interfaz más atractiva y funcional**.
  (https://pypi.org/project/colorama/)
- `prompt_toolkit:` Utilizada en la **función** `buscar_pais` para ofrecer una **función de autocompletado**, mejorando significativamente la eficiencia de la búsqueda.
  (https://pypi.org/project/prompt-toolkit/)

4. Metodología Utilizada

La metodología de trabajo fue **iterativa y modular**, nos enfocamos en la **división de tareas y la validación constante**:

1. **Definición de Módulos (Separación de Responsabilidades):** Se estableció un diseño **modular con archivos** (aunque concentrado en `gestion_paises`.py por la restricción de un solo archivo, se mantienen las responsabilidades lógicas separadas dentro de funciones):
   - **Módulo de Persistencia**: `cargar_datos`, `guardar_datos`.
   - **Módulo de Normalización** y **Auxiliares**: `sin_acentos`, `normalizar_continente`, etc..
   - **Módulo Principal/Menú**: `main`.
   - **Bloques de Funcionalidad**: `agregar_pais`, `actualizar_pais`, `buscar_pais`, `menu_filtrar`, `menu_ordenar`, `mostrar_estadisticas`.
2. **Implementación y Testeo Progresivo:** Las funcionalidades se implementaron **secuencialmente**, empezando por la **carga de datos**. Cada **función** fue **probada inmediatamente después de su desarrollo**, asegurando que el código fuera **funcional antes de pasar al siguiente requisito**.
3. **Trabajo Colaborativo:** La comunicación fue constante para **dividir el trabajo en bloques de funciones**. La gestión del código se realizó con un **repositorio GitHub**, utilizando **ramas** para **evitar conflictos** y generando **pull requests** antes de la consolidación (**merge**), también se trabajó con un tablero de **Trello para la división de las tareas y colaboración.**

5. Resultados Obtenidos

El proyecto ha resultado en una **aplicación de consola completamente funcional** que cumple con los requisitos y restricciones del TPI.

5.1. Funcionalidad Clave y Robustez

- **Manejo de Archivos:** La **carga** de **datos** es robusta, **saltándose registros** con **mal formato,** sin generar `exceptions` y **normalizando nombres** y **continentes**.
- **Operaciones CRUD:** Las funciones `agregar_pais` y `actualizar_pais` funcionan correctamente y **actualizan el archivo CSV en tiempo real**, garantizando la **persistencia**. Se **manejan** correctamente los **casos de países ya existentes**.
- **Búsqueda Avanzada:** La función `buscar_pais` con `prompt_toolkit` es eficiente y **user-friendly**, permitiendo **búsquedas rápidas con autocompletado e insensibilidad a acentos/mayúsculas**.
- **Filtros y Ordenamiento:** Todas las opciones de **filtrado y ordenamiento operan correctamente**, incluyendo **filtros** automáticos por **rangos de población y superficie**.
- **Estadísticas:** Se calculan c**orrectamente los promedios y se identifican los extremos (mayor/menor población)** y el **conteo** por **continente**.

5.2. Uso de Estructuras y Librerías

- La elección de la **Lista de Diccionarios** se confirmó como la **estructura óptima** para la **manipulación** y **consulta eficiente** del **dataset**.
- La inclusión de `tabulate` y `colorama` **mejora la presentación en la consola**, haciendo la experiencia más profesional y fácil de interpretar.
- La **implementación** del **ordenamiento** mediante **funciones auxiliares** demostró el dominio de las funciones.

5.3. Desafíos y Soluciones

El mayor desafío fue la lógica de normalización de datos. Asegurar que **búsquedas** como **"méxico"** o **"MEXICO" coincidieran** con **"México"** en el **registro**, y que entradas como **"america"** o **"sudamerica"** se **unificaran** a **"América"**. La función `sin_acentos` **resolvió** la **insensibilidad** a **acentos** y **mayúsculas**, mientras que `normalizar_continente` permitió **agrupar términos comunes**, lo que añadió un tratamiento de datos al sistema que mejora la calidad.

6. Conclusiones

El desarrollo de este TPI ha sido un trabajo muy importante de la materia Programación 1, permitiéndonos i**ntegrar la teoría en un sistema completo y funcional**.

1. La principal lección fue el uso del principio de **modularidad**. **Dividir** el **proyecto** en **funciones** con una **única responsabilidad** hizo que **el desarrollo fuera más predecible**. **Cada función** podía ser **probada** y **resuelta** de **forma aislada**, lo que redujo drásticamente el tiempo de resolución de errores en comparación con un código monolítico.
2. Las **herramientas** y **estructuras de datos** aprendidas (**Listas, Diccionarios, Funciones y Persistencia en CSV**) son la **base** en un proyecto para la **gestión de datos**. Nos parecen fundamentales porque las listas modelan los elementos, los diccionarios modelan objetos más complejos y las funciones agrupan la lógica, permitiendo escalar el proyecto sin aumentar la complejidad de su lectura.
3. El **trabajo en equipo fué vital. Utilizar herramientas como GIthub y Trello, asi como distintos canales de comunicación es importante para desarrollar cualquier proyecto.** Tomamos como base la **modularidad para dividir las tareas** de acuerdo a las **funciones y requisitos del programa**: un integrante se centró en la **estructura del menú, la persistencia de datos y** también de las **funciones** de **búsqueda, filtros, ordenamiento y estadísticas.** El otro integrante desarrolló las **funcionalidades para añadir, actualizar países y validaciones necesarias, como normalizaciones**.

7. Repositorio GitHub

Se ha creado un repositorio público en GitHub para alojar todos los entregables.

- **Link al Repositorio:** https://github.com/Equipo-115-de-Programacion-1-TUP/TPI_Equipo115_Programacion1.git
- **Video:** Link al Video

El repositorio contiene:

1. **Código Fuente:** El archivo `gestion_paises.py`.
2. **Dataset Base:** El archivo `paises.csv`
3. **Documentación:** El presente informe en formato PDF.
4. `README.md`

8. Bibliografía

- *"Python Dictionaries" (https://www.w3schools.com/python/python_dictionaries.asp)*
- *"Documentación Oficial de Python - 5. Estructuras de datos - 5.5. Diccionarios" (https://docs.python.org/es/3/tutorial/datastructures.html#dictionaries)*
- *"Documentación Oficial de Python - 3. Una introducción informal a Python - 3.1.3. Listas" (https://docs.python.org/es/3/tutorial/introduction.html#lists)*
- *"Python Lists" (https://www.w3schools.com/python/python_lists.asp)*
- *"csv — CSV File Reading and Writing" (https://docs.python.org/es/3/library/csv.html)*
- *"Python Functions" (https://www.w3schools.com/python/python_functions.asp)*
- *"Prompt Toolkit" (https://pypi.org/project/prompt-toolkit/)*
- *"Tabulate" (https://pypi.org/project/tabulate/)*
- *"Colorama" (https://pypi.org/project/colorama/)*