Trabajo Práctico Integrador

Grupo: 76

Alumnos

Carlos Chiavarini, comisión 3

Zerpa Alexis Cristian Boris, comisión 10

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Organización Empresarial

Docente Titular

Alberto Cortez

Docentes Tutores

Brian Lara y Martina Zabala

11 de Noviembre de 2025

Objetivos del Trabajo

El presente trabajo práctico integrador tiene como propósito aplicar los conocimientos adquiridos en la materia **Programación I** para el desarrollo de un sistema informático que gestione información de países utilizando el lenguaje **Python**.
A través de este proyecto, buscamos consolidar nuestras competencias en la programación estructurada, la manipulación de datos y la organización modular del código.

De manera específica, nos propusimos alcanzar los siguientes objetivos:

1. **Diseñar e implementar un programa funcional en Python** que permita registrar, modificar, consultar y analizar información relacionada con distintos países, empleando estructuras de datos adecuadas como listas y diccionarios.
2. **Aplicar los principios de la modularización**, dividiendo el código en bloques temáticos (datos, búsquedas, filtros, ordenamiento, estadísticas, validaciones y flujo principal), con el fin de mejorar la organización, la legibilidad y la mantenibilidad del sistema.
3. **Incorporar la persistencia de datos mediante archivos CSV**, asegurando que la información pueda almacenarse y recuperarse entre ejecuciones, simulando así un entorno de base de datos simplificada.
4. **Implementar un menú interactivo** que permita la comunicación con el usuario a través de la consola, utilizando la estructura match-case como mecanismo moderno de control de flujo, para facilitar la selección de opciones y el acceso a las distintas funcionalidades.
5. **Desarrollar funciones de validación** que garanticen la integridad de los datos ingresados, evitando valores vacíos, negativos o duplicados, y asegurando que la información registrada sea coherente y confiable.
6. **Generar herramientas de análisis y estadísticas**, tales como el cálculo de promedios, la identificación de países con mayor o menor población y superficie, y la cantidad de países por continente, fortaleciendo el manejo lógico y matemático de la información.
7. **Fomentar el trabajo colaborativo y el razonamiento lógico**, aplicando técnicas de resolución de problemas, depuración de errores y pruebas funcionales para alcanzar un producto final sólido, eficiente y bien documentado.

En conjunto, estos objetivos nos permitieron integrar los conocimientos teóricos con la práctica, logrando un proyecto completo que combina programación, estructura de datos, manejo de archivos y validación de información dentro de un entorno realista de gestión de datos.

Marco Teórico

En este trabajo práctico integrador aplicamos los principales conceptos de la **programación estructurada** utilizando el lenguaje **Python**, con el objetivo de desarrollar una aplicación funcional que gestione información sobre distintos países.
A continuación, desarrollamos los fundamentos teóricos de los elementos y técnicas empleadas.

Lenguaje Python

Python es un lenguaje de programación **interpretado**, **multiparadigma** y **de alto nivel**, ampliamente utilizado tanto en la educación como en la industria por su sintaxis simple y su gran versatilidad.
Entre sus principales características se destacan:

- Su **tipado dinámico**, que permite declarar variables sin definir explícitamente su tipo de dato.
- Su **administración automática de memoria**, lo que reduce errores y facilita la gestión de datos.
- Su amplio **ecosistema de librerías** que permiten realizar tareas complejas con pocas líneas de código.

En el presente trabajo utilizamos **Python 3**, cuya sintaxis moderna incluye herramientas como el **match-case**, la comprensión de listas y la manipulación sencilla de archivos CSV.

Funciones

En el paradigma estructurado, las **funciones** constituyen la base de la modularización del código.
Una función es un bloque de instrucciones que realiza una tarea específica y puede ser invocado desde cualquier parte del programa.
Su utilización nos permite:

- **Evitar la repetición de código**, al encapsular operaciones que se repiten en distintos puntos.
- **Facilitar la lectura y el mantenimiento**, ya que cada función tiene un propósito claro.
- **Organizar el flujo lógico del programa**, separando responsabilidades por tipo de tarea (por ejemplo: búsqueda, filtrado, estadísticas, etc.).

Cada función puede recibir **parámetros de entrada** y devolver un **valor de salida** mediante la instrucción return.
En nuestro proyecto, empleamos funciones para realizar tareas como **leer el archivo CSV**, **agregar países**, **filtrar resultados** o **calcular promedios**.

Estructuras de datos: listas y diccionarios

Para el almacenamiento y manipulación de la información utilizamos dos estructuras fundamentales de Python:

- **Listas (list)**: son colecciones ordenadas y mutables que nos permiten agrupar múltiples elementos. En este trabajo, usamos listas para almacenar el conjunto total de países.
- **Diccionarios (dict)**: son estructuras de datos que almacenan pares *clave: valor*. Cada país fue representado como un diccionario con los campos nombre, poblacion, superficie y continente.

La combinación de listas y diccionarios facilita la manipulación de información estructurada y la búsqueda de datos mediante claves.

Control de flujo: condicionales y bucles

El control del flujo del programa se realiza a través de **estructuras condicionales** y **estructuras de repetición**:

- Las sentencias **if, elif y else** permiten ejecutar distintas instrucciones dependiendo del cumplimiento de una condición lógica.
  Esto nos posibilita verificar casos específicos, por ejemplo, si un país existe antes de agregarlo o modificarlo.
- Los **bucles for y while** nos permiten recorrer estructuras de datos o repetir acciones hasta cumplir una condición.
  En este trabajo, utilizamos bucles para recorrer la lista de países, mostrar registros y calcular estadísticas.

Estructura match-case

A partir de **Python 3.10**, se incorpora la instrucción match-case, una alternativa moderna al uso de múltiples condicionales if-elif-else.

Esta estructura funciona de manera similar a un switch en otros lenguajes, permitiendo una sintaxis más clara y legible.

En nuestro programa, la aplicamos para manejar el **menú principal** del sistema, donde cada opción seleccionada por el usuario activa una función específica.

De esta manera, mejoramos la organización del código y la comprensión de las opciones disponibles.

Entrada y salida de datos

El programa interactúa con dos tipos de entradas y salidas:

1. **Entrada del usuario por consola**, mediante la función input().
   Esto permite que el usuario ingrese datos nuevos o seleccione opciones del menú.
2. **Entrada y salida de archivos CSV**, utilizando el módulo estándar csv.
   Este formato (Comma Separated Values) almacena datos tabulares en texto plano y facilita la persistencia de la información.

De esta forma logramos **persistir los datos** (mantenerlos disponibles entre ejecuciones del programa), lo que es esencial en un sistema de gestión.

Validaciones

Durante el desarrollo implementamos diferentes validaciones para garantizar la **integridad de los datos** y evitar errores.
Entre ellas:

- Comprobamos que los campos numéricos sean positivos.
- Evitamos el ingreso de nombres vacíos o duplicados.
- Estandarizamos los datos al almacenarlos (por ejemplo, usando .strip() para quitar espacios y .title() para capitalizar nombres).

Estas validaciones aseguran que la información contenida en el archivo CSV sea coherente y confiable.

Modularización

La **modularización** consiste en dividir el código en **bloques funcionales** que agrupan operaciones relacionadas.
Esto permite aislar responsabilidades, mejorar la legibilidad y facilitar futuras ampliaciones.

En nuestro trabajo organizamos el programa en los siguientes bloques principales:

- **Bloque Main:** contiene el flujo general y el menú de opciones.
- **Bloque Datos:** encargado de leer y guardar información en el archivo CSV.
- **Bloque Búsquedas:** incluye funciones que permiten localizar países según distintos criterios.
- **Bloque Ordenamiento:** implementa el ordenamiento de los registros según nombre, población o superficie.
- **Bloque Filtros:** agrupa las funciones que seleccionan países por continente o rango de población.
- **Bloque Estadísticas:** calcula datos agregados, como promedios o máximos/mínimos.
- **Bloque Validaciones:** centraliza las verificaciones de los datos ingresados.

Esta estructura modular facilita la comprensión del código y cumple con las buenas prácticas de desarrollo profesional.

Conclusión del Marco Teórico

En resumen, el marco teórico se sustenta en la aplicación de los principios fundamentales de la programación estructurada mediante Python.
A lo largo del desarrollo, empleamos funciones, estructuras de control, manejo de archivos y modularización para construir una herramienta práctica y coherente con los objetivos del trabajo.

Metodología Utilizada

Para llevar adelante el desarrollo del proyecto **“Gestión de Datos de Países en Python”**, adoptamos una metodología de trabajo estructurada, basada en las etapas del ciclo de vida del software.
Nuestro enfoque se centró en combinar la comprensión teórica de los conceptos de programación con la aplicación práctica en un entorno de desarrollo real.
A continuación, detallamos las principales fases y procedimientos que seguimos:

1. Análisis del problema

En una primera instancia, analizamos la consigna del trabajo con el objetivo de comprender los requerimientos funcionales y no funcionales del sistema.
Identificamos las principales necesidades: la carga, almacenamiento, modificación, filtrado y análisis de información sobre distintos países.
También definimos que el sistema debía ser interactivo, persistente y de fácil uso desde consola.

2. Diseño del sistema

Con los requerimientos claros, procedimos al diseño del sistema, definiendo la estructura lógica del programa.
Determinamos el uso de **listas** para almacenar múltiples registros y **diccionarios** para representar cada país con sus atributos (nombre, población, superficie y continente).
Además, planificamos la **modularización** del código en bloques temáticos, lo que nos permitió mantener una organización clara y coherente:

- **Bloque Main:** flujo general y menú principal.
- **Bloque Datos:** lectura y escritura del archivo CSV.
- **Bloque Búsquedas, Filtros y Ordenamientos:** operaciones sobre los datos.
- **Bloque Estadísticas:** generación de cálculos y resúmenes.
- **Bloque Validaciones:** control de consistencia y formato de la información.

El diseño incluyó la definición de funciones específicas para cada tarea, buscando favorecer la reutilización y reducir errores.

3. Desarrollo e implementación

Durante esta etapa, implementamos el programa en **Python 3.12**, aplicando buenas prácticas de programación estructurada.
Optamos por la estructura **match-case** para la navegación del menú, ya que mejora la legibilidad y reemplaza al uso tradicional de condicionales múltiples.
También incorporamos tres módulos estándar de Python para ampliar las capacidades del programa. Utilizamos **csv** para gestionar la lectura y escritura del archivo de datos en formato tabular, lo que nos permitió mantener la persistencia de la información. Empleamos **os** para interactuar con el sistema operativo, verificando la existencia del archivo y creando uno nuevo cuando fue necesario. Finalmente, incluimos **statistics** (específicamente la función **mean**) para calcular promedios de población y superficie, facilitando el análisis estadístico de los datos registrados.

Cada función fue documentada con comentarios explicativos, y se cuidó la elección de nombres de variables descriptivos y en español, siguiendo los lineamientos del trabajo.

4. Pruebas y validaciones

Una vez completada la implementación, realizamos **pruebas funcionales** con distintos escenarios de datos para garantizar el correcto funcionamiento del sistema.
Verificamos que las operaciones de alta, modificación, búsqueda y filtrado produjeran los resultados esperados.
Además, aplicamos **validaciones de entrada** para asegurar que los datos cargados por el usuario fueran válidos y que no se produjeran errores de tipo o formato durante la ejecución.

También se comprobó la correcta lectura del archivo CSV, controlando aspectos como la codificación de caracteres, los encabezados y el manejo de datos vacíos.

5. Documentación y presentación

Finalmente, elaboramos la documentación técnica y teórica del trabajo, describiendo los fundamentos del lenguaje utilizado, la estructura del código y los resultados obtenidos.
La presentación del informe busca reflejar tanto el proceso de razonamiento lógico como la correcta aplicación de los conceptos de programación estructurada.

Esta fase nos permitió reflexionar sobre el proceso de desarrollo y mejorar la claridad y legibilidad del código, reforzando la importancia de la documentación en los proyectos de software.

6. Evaluación del trabajo

Concluimos la metodología realizando una **evaluación del producto final**, verificando el cumplimiento de los objetivos planteados.
El sistema demostró ser estable, modular, fácil de mantener y adaptable a futuras mejoras, lo que evidencia la efectividad del proceso metodológico empleado.

En resumen, nuestra metodología se basó en una secuencia lógica de análisis, diseño, implementación, prueba y documentación, asegurando la calidad del programa y el aprendizaje integral de los conceptos teóricos aplicados en la práctica.

Resultados Obtenidos

Como resultado del proceso de desarrollo e implementación, logramos obtener un programa completamente funcional, capaz de gestionar información sobre distintos países de manera ordenada, eficiente y persistente.
Durante las pruebas realizadas, verificamos que el programa puede **leer y escribir correctamente los datos en el archivo CSV**, permitiendo que la información se mantenga disponible entre ejecuciones.
Asimismo, confirmamos el correcto funcionamiento de las **operaciones de alta y modificación**, las cuales actualizan los registros de forma inmediata y confiable.

El **menú interactivo**, implementado mediante la estructura match-case, se comportó de manera clara y fluida, guiando al usuario en todo momento y permitiendo acceder fácilmente a cada una de las funciones disponibles.
Las opciones de **búsqueda, filtrado y ordenamiento** ofrecieron resultados precisos, validando la correcta manipulación de las listas y diccionarios que conforman la estructura interna de datos.

En cuanto al análisis de información, las **funciones estadísticas** desarrolladas —basadas en la librería statistics— permitieron obtener con exactitud los **promedios de población y superficie**, así como identificar los países con valores extremos en cada categoría.
Esto confirmó la validez de los cálculos y la eficacia de las operaciones lógicas implementadas.

Durante la etapa de validación, comprobamos que el programa **maneja adecuadamente los errores de entrada**, impidiendo el registro de valores vacíos, negativos o no numéricos, lo cual garantiza la integridad del archivo de datos.
Además, verificamos la correcta lectura del archivo base paises.csv y su compatibilidad con distintos formatos de codificación de texto.

En términos generales, los resultados obtenidos evidencian un sistema robusto, modular y fácil de mantener.
La aplicación no solo cumple con las funciones de un gestor de datos básico, sino que también incorpora buenas prácticas de programación, como la separación lógica del código, la validación de entradas y la documentación de funciones.

Finalmente, observamos que el proyecto puede ser **expandido en el futuro** con nuevas funcionalidades, como la eliminación de registros, la exportación a otros formatos o incluso la implementación de una interfaz gráfica.
De esta manera, el trabajo alcanzó plenamente nuestros objetivos académicos y técnicos, consolidando el aprendizaje práctico de los conceptos fundamentales de la programación.

Conclusiones

A lo largo del desarrollo del trabajo práctico integrador **“Gestión de Datos de Países en Python”**, logramos aplicar de manera efectiva los principios fundamentales de la **programación estructurada**, consolidando tanto los aspectos teóricos como las habilidades prácticas adquiridas en la materia.
El proyecto nos permitió comprender cómo un conjunto de conceptos —funciones, estructuras de datos, validaciones, modularización y manejo de archivos— puede integrarse para construir un sistema funcional, claro y coherente.

Mediante el uso del lenguaje **Python**, conseguimos implementar un programa capaz de **gestionar información real** de forma ordenada y persistente.
El sistema demostró ser estable y flexible, ofreciendo operaciones de alta, modificación, búsqueda, filtrado, ordenamiento y análisis estadístico de los datos.
La aplicación de la estructura **match-case** simplificó el flujo del menú principal, mejorando la legibilidad del código, mientras que el uso del módulo **csv** permitió mantener la información almacenada entre ejecuciones, simulando un entorno de base de datos.
Asimismo, la incorporación del módulo **statistics** nos facilitó el cálculo de promedios y la obtención de resultados analíticos precisos.

Durante el proceso de desarrollo, confirmamos la importancia de las **validaciones de entrada** y de la correcta gestión de los errores, elementos esenciales para asegurar la integridad y confiabilidad de los datos.
También comprobamos que la **modularización** es una herramienta fundamental para organizar el código, facilitar su mantenimiento y permitir futuras ampliaciones sin alterar la estructura base del sistema.

En términos de aprendizaje, este trabajo fortaleció nuestra comprensión sobre el ciclo completo del desarrollo de software: desde el análisis y diseño, hasta la implementación, prueba y documentación.
El trabajo colaborativo nos permitió resolver desafíos de manera conjunta, reforzando la lógica, la planificación y la capacidad de adaptación frente a problemas técnicos.

En conclusión, consideramos que el proyecto cumplió con todos los objetivos propuestos, integrando teoría y práctica en una experiencia significativa de aprendizaje.
El programa desarrollado constituye un ejemplo concreto de cómo los principios de la programación estructurada pueden aplicarse para resolver problemas reales de gestión de datos, sentando una base sólida para avanzar hacia proyectos más complejos en el futuro.

Links

Repositorio en Github: https://github.com/cgchiavarini/UTN-TUPAD-TPI-P1

Video en Youtube: https://youtu.be/iql0bS72Jtw

Bibliografia

Python Software Foundation. (2024). *The Python Standard Library*. Python Documentation. https://docs.python.org/3/library/

Python Software Foundation. (2024). *Tutorial: Python 3 Documentation*. https://docs.python.org/3/tutorial/

W3Schools. (2024). *Python Tutorial*. https://www.w3schools.com/python/

Real Python. (2024). *Learn Python Programming*. https://realpython.com/

GeeksforGeeks. (2024). *Python Programming Language – Introduction*. https://www.geeksforgeeks.org/python-programming-language/

Material de la catedra Programación I, TUPaD, UTN (2025).