

Trabajo Práctico Integrador

Alumnos – Grupo n° 77

Camilo Angeleri y Facundo Bocquet.

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Programacion I

Docente Titular

Martín Alejandro García

Docente Tutor

Martina Belen Zabala

11 de Noviembre de 2025

Tabla de contenido

Consignas	4
Desarrollo	5
Conclusión	10
Referencias	11

Trabajo Práctico Integrador

Introducción

En este trabajo práctico integrador tuvimos como objetivo desarrollar una aplicación en Python que permita gestionar información sobre países, almacenada en un archivo CSV.

El programa implementa estructuras de datos (listas y diccionarios), funciones, condicionales, ordenamientos y estadísticas básicas, cumpliendo con los criterios de modularización, validación de datos y claridad del código.

El sistema permite agregar, modificar, buscar, filtrar y ordenar países, así como obtener estadísticas generales a partir de los registros cargados.

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.

Consignas

1. Diseño (previo al código)

- Explicar en un informe teórico los conceptos aplicados:
 - o Listas
 - o Diccionarios
 - o Funciones
 - o Condicionales
 - o Ordenamientos
 - o Estadísticas básicas
 - o Archivos CSV
- Definir el flujo de operaciones principales en un diagrama o esquema.

2. Funcionalidades mínimas del sistema

El programa debe ofrecer un menú de opciones en consola que permita:

- Buscar un país por nombre (coincidencia parcial o exacta).
- Filtrar países por:
 - o Continente
 - o Rango de población
 - o Rango de superficie
- Ordenar países por:
 - o Nombre
 - o Población
 - o Superficie (ascendente o descendente)
- Mostrar estadísticas:
 - o País con mayor y menor población
 - o Promedio de población

- o Promedio de superficie
- o Cantidad de países por continente

3. Validaciones

- Controlar errores de formato en el CSV.
- Evitar fallos al ingresar filtros inválidos o búsquedas sin resultados.
- Mensajes claros de éxito/error.

Desarrollo

- **informe teórico de los conceptos aplicados**

- o **Listas**

Las listas son estructuras mutables ya que permiten modificar los valores de sus elementos, eliminar o añadir elementos de forma ordenada. En este proyecto, su uso es fundamental para manipular datos leídos del archivo CSV.

Ejemplos de aplicación:

- **lista_paises**: se genera en la función `verificar_paises()` para almacenar todos los países cargados en la base.
- **filas**: lista que guarda las filas completas del archivo antes de reescribirlo (en funciones de actualización).
- **coincidencias_continente**: lista temporal utilizada en los filtros.
- En `mostrar_estadisticas()`, la lista **paises** almacena tuplas con todos los datos de cada país: (nombre, población, superficie, continente).

Importancia:

Las listas permiten recorrer, filtrar y ordenar los datos fácilmente, lo que resulta esencial para operaciones como búsqueda y ordenamiento.

o Diccionarios

Los diccionarios se utilizan para almacenar pares clave–valor, facilitando el conteo y agrupamiento de información.

Ejemplo de uso:

```
continentes = {}  
for elemento in paises:  
    cont = elemento[3]  
    continentes[cont] = continentes.get(cont, 0) + 1
```

En `mostrar_estadisticas()`, este diccionario permite calcular cuántos países pertenecen a cada continente.

Importancia:

Permiten una búsqueda más rápida y un manejo organizado de información categórica (como los continentes).

o Funciones

El código se encuentra modularizado, cumpliendo con el principio de “una función = una responsabilidad”.

Cada función cumple una tarea específica, mejorando la legibilidad y el mantenimiento.

Ejemplos:

- `agregar_pais_a_base()` → escribe un nuevo país en el CSV.
- `verificar_paises()` → genera la lista actualizada de países.
- `cambiar_poblacion()` y `cambiar_superficie()` → modifican datos existentes.
- `filtrar_por_rango_poblacion()` → busca países según un rango de población.
- `mostrar_estadisticas()` → realiza cálculos y muestra resultados.

También se utilizan **funciones lambda** para ordenar listas o seleccionar máximos/mínimos:

```
filas.sort(key=lambda fila: int(fila[indice]), reverse=modo)
```

Importancia:

La modularización facilita el mantenimiento y la reutilización del código, promoviendo una estructura ordenada.

o Condicionales

Las estructuras condicionales (if, elif, else) controlan el flujo del programa y permiten validar los datos ingresados.

Ejemplos:

- Verificación de tipos de entrada:
`if poblacion_valor.isnumeric() and int(poblacion_valor) > 0:`
- Control de existencia de un país:
`if pais_valor not in lista_paises:`
- Filtrado de resultados:
`if rango_min <= int(fila[1]) <= rango_max:`

Importancia:

Los condicionales permiten tomar decisiones en tiempo de ejecución, garantizando que el programa sea robusto frente a errores o datos inválidos.

o Ordenamientos

El sistema ofrece la posibilidad de ordenar países por distintos criterios (nombre, población o superficie) de forma ascendente o descendente.

Ejemplo:

```
filas.sort(key=funcion_key, reverse=modo)
```

donde `funcion_key` es una función lambda que define el criterio de comparación.

Importancia:

El ordenamiento facilita la interpretación de los datos y demuestra el manejo de funciones de orden superior y parámetros de control en Python.

o Estadísticas básicas

La función `mostrar_estadisticas()` calcula indicadores globales sobre los países registrados:

- **Máximo y mínimo de población:**
`pais_mayor_pob = max(paises, key=lambda x: x[1])`
`pais_menor_pob = min(paises, key=lambda x: x[1])`
- **Promedios:**
`promedio_poblacion = sum(p[1] for p in paises) / len(paises)`
`promedio_superficie = sum(p[2] for p in paises) / len(paises)`
- **Conteo por continente:** mediante un diccionario.

Importancia:

Permiten analizar los datos y obtener conclusiones cuantitativas a partir del dataset, aplicando funciones nativas de Python (sum, max, min, len).

o Archivos CSV

El programa trabaja con un archivo CSV (base.csv) que almacena los datos persistentes del sistema.

Se utiliza el módulo csv para leer y escribir registros de forma ordenada.

Lectura:

```
with open("base.csv","r",newline="") as base:  
    lector = csv.reader(base)
```

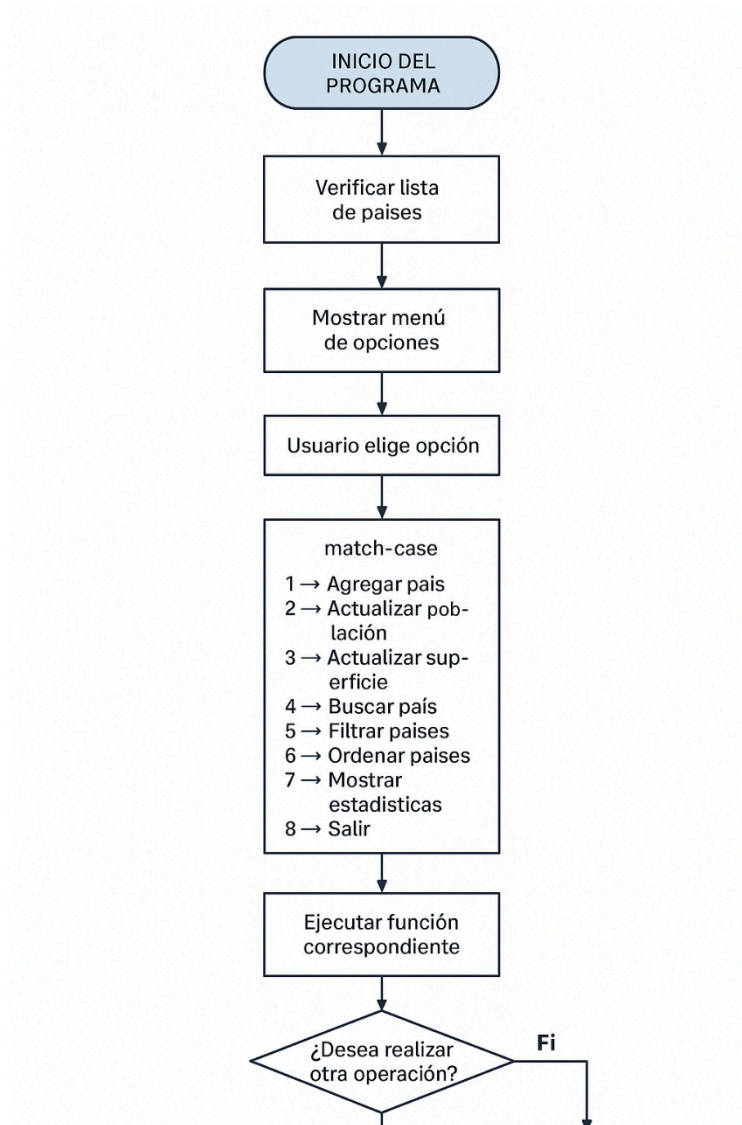
Escritura:

```
with open("base.csv","w",newline="") as base:  
    escritor = csv.writer(base)  
    escritor.writerow(encabezado)  
    escritor.writerows(filas)
```

Importancia:

El manejo de archivos CSV permite la persistencia de datos y la interoperabilidad con otras aplicaciones o bases de datos.

- Diagrama del flujo de operaciones



Conclusión

La culminación de este Trabajo Práctico Integrador (TPI) representa una **consolidación fundamental de nuestro aprendizaje** como grupo. El desafío de desarrollar una aplicación funcional en Python nos obligó a trasladar los conceptos teóricos de **Programación I** a un entorno de desarrollo práctico.

El principal logro del grupo fue la adquisición de **fluidez en la aplicación de las estructuras de datos esenciales**. Trabajamos de manera eficiente con **listas anidadas** para manejar el *dataset* principal y utilizamos **diccionarios** estratégicamente, por ejemplo, para el conteo de países por continente, lo que simplificó los cálculos estadísticos.

Otro aprendizaje clave fue la **disciplina en la modularización**. Al crear **funciones** específicas para cada operación del menú (CRUD, ordenamiento, estadísticas), no solo cumplimos con el requisito de **código reutilizable y claro**, sino que experimentamos de primera mano cómo un diseño modular facilita la **división del trabajo** y la **depuración colaborativa**.

Finalmente, la integración del módulo **csv** para la **persistencia de datos** fue un hito. Logramos que el sistema funcionara de manera autónoma, almacenando y recuperando registros de forma robusta. En conjunto, el TPI sirvió como un **valioso ejercicio de integración**, fortaleciendo nuestras habilidades técnicas y sentando las bases necesarias para abordar proyectos de *software* de mayor escala en el futuro.

Referencias

González Duque, R. (s.f.). *Python para todos*. Mundogeeek. [Disponible en:

<http://mundogeeek.net/tutorial-python/>]

Python Software Foundation. (última versión). *El tutorial de Python*. Documentación oficial de

Python. [Disponible en: <https://docs.python.org/es/3/tutorial/>]

Python Software Foundation. (última versión). *csv — Lectura y escritura de archivos CSV*.

Documentación oficial de Python. [Disponible en:

<https://docs.python.org/es/3.9/library/csv.html>]