INFORME FINAL - INTEGRADOR TPI
Gestión de Países

Trabajo Práctico Integrador - Programación 1 – PROG5

Alumnos: Mendez Joaquín, Elias Juan Cruz.

Profesor/a: Fernandez Sofia.

Índice

1 Marco Teórico.

El proyecto fue desarrollado en el lenguaje de programación **Python**, aplicando los principios de la programación estructurada, el manejo de archivos y las buenas prácticas de validación de datos.
A lo largo del desarrollo se utilizaron distintos elementos del lenguaje, desde variables y estructuras de control hasta funciones modulares, con el objetivo de obtener un código estable, legible y libre de errores.

1.1 Variables y tipos de datos

Las **variables** representan espacios en memoria que almacenan valores y pueden modificarse durante la ejecución del programa.
En Python, las variables no requieren declaración de tipo, ya que el lenguaje realiza la asignación dinámica según el valor que se le otorgue.
En este proyecto se emplearon diferentes tipos de variables:

- **Cadenas de texto (str)**: utilizadas para almacenar nombres de países, continentes y rutas de archivo.
  Ejemplo: nombre = "Argentina".

- **Enteros (int)**: empleados para valores numéricos como la población o superficie.
  Ejemplo: población = 45376763.

- **Listas (list)**: estructuras de datos que pueden contener múltiples elementos ordenados. Se utilizaron para mantener la colección completa de países cargados desde el archivo CSV.
  Ejemplo:

países = [

{"nombre": "Argentina", "población": 45376763, "superficie": 2780400, "continente": "América"},

{"nombre": "Brasil", "población": 213993437, "superficie": 8515767, "continente": "América"}

]

- **Diccionarios (dict)**: estructuras formadas por pares clave-valor, ideales para representar entidades con varios atributos. Cada país se almacenó como un diccionario dentro de una lista.

Ejemplo:

país = {

"nombre": "Chile",

"población": 19700000,

"superficie": 756096,

"continente": "América"

}

1.2 Funciones

Las **funciones** son bloques de código reutilizables que permiten agrupar instrucciones bajo un nombre, facilitando la organización del programa y evitando la repetición de código.
Se definen en Python mediante la palabra clave def y pueden recibir parámetros y devolver valores.

En el integrador se utilizaron múltiples funciones con propósitos bien definidos:

- def cargar_csv(ruta): → lee el archivo CSV y devuelve la lista de países cargados.
- def guardar_csv(ruta, paises): → guarda la información actualizada en un archivo.
- def buscar_por_nombre(paises, termino): → devuelve una lista de países cuyo nombre contiene el término ingresado.
- def filtrar_por_rango(paises, campo, minimo, maximo): → selecciona los países dentro de un rango de población o superficie.
- def ordenar_paises(paises, campo, descendente=False): → ordena los países alfabéticamente o por valores numéricos.
- def estadisticas(paises): → calcula estadísticas generales como promedio, país más poblado y país menos poblado.

1.3 Estructuras de control

El programa emplea las tres estructuras de control fundamentales:

- **Secuencia:** ejecución ordenada de instrucciones en el orden en que aparecen.
- **Selección (condicionales):** permiten decidir qué acción tomar en función de una condición lógica.

Ejemplo:

```
if not paises:
    print("No hay países cargados.")
else:
    mostrar_paises(paises)
```

- **Repetición (bucles):** permiten repetir bloques de código. Se usaron:
  - for para recorrer listas de países.
  - while para mantener activo el menú hasta que el usuario elija salir.

  Ejemplo:

```
while True:
    mostrar_menu()
    opcion = input("Seleccione una opción: ")
    if opcion == "0":
        break
```

1.4 Validaciones

Las validaciones son procedimientos destinados a asegurar que los datos introducidos sean correctos antes de procesarlos.
En este proyecto, en lugar de usar excepciones (try/except), se optó por **verificaciones preventivas** mediante condiciones if.

Ejemplos:

- Verificar que el archivo exista antes de abrirlo:

```
if not os.path.isfile(ruta):
    print("Archivo no encontrado.")
```

- Confirmar que la entrada sea numérica antes de convertirla:

```
if s.lstrip("-").isdigit():
    return int(s)
else:
    return 0
```

Este método evita que el programa se detenga por errores de usuario, asegurando una ejecución continua y estable.

1.5 Manejo de archivos CSV

El archivo CSV (Comma-Separated Values) se utiliza para almacenar y recuperar información de forma tabular.
En Python, el módulo estándar csv permite manipular este tipo de archivos de manera sencilla.

Funciones clave utilizadas:

- cargar_csv() → lee el archivo y genera una lista de diccionarios con los países.
- guardar_csv() → escribe los datos actuales en un nuevo archivo.
- crear_csv_ejemplo() → genera un CSV de muestra si el original no existe.

Estas funciones garantizan la persistencia de los datos y evitan pérdidas de información.

1.6 Estructuras de datos complejas

Cada país se representa como un **diccionario**, y todos los países se almacenan dentro de una **lista**, formando una estructura tipo "lista de diccionarios".
Esta forma de organizar los datos permite acceder rápidamente a cualquier campo mediante su clave, por ejemplo:

```
paises[0]["nombre"]       # Devuelve 'Argentina'

paises[0]["poblacion"]    # Devuelve 45376763
```

La estructura es flexible y compatible con los módulos de lectura y escritura de CSV, facilitando la manipulación de la información.

1.7 Modularización

La modularización consiste en dividir un programa en partes independientes llamadas módulos o bloques.
En el integrador, el código se organiza en cuatro bloques principales:

1.7.1 Bloque 1: Validaciones y utilidades.

Contiene funciones auxiliares para limpiar texto y validar entradas.

1.7.2 Bloque 2: Persistencia.

Se encarga de la lectura y escritura de archivos CSV.

1.7.3 Bloque 3: Funcionalidad.

Incluye las operaciones de búsqueda, filtrado, ordenamiento y cálculo estadístico.

1.7.4 Bloque 4: Menú principal.

Gestiona la interacción con el usuario a través de un menú en consola.

Esta organización mejora la legibilidad, facilita las pruebas y simplifica futuras modificaciones o ampliaciones del programa.

1.8 Funciones integradas de Python

El código aprovecha diversas funciones internas del lenguaje para simplificar tareas:

- sorted() → ordena listas.
- max() y min() → identifican el país con mayor o menor población.
- sum() y len() → permiten calcular promedios.
- input() → recibe datos desde la consola.
- print() → muestra información al usuario.

Estas funciones garantizan eficiencia y reducen la cantidad de código necesario.

1.9 Recursividad

En este proyecto **no se implementó recursividad**.
La recursividad es una técnica mediante la cual una función se llama a sí misma para resolver un problema dividiéndolo en partes más pequeñas.
Aunque se trata de un concepto importante en programación (usado, por ejemplo, en factoriales, búsqueda binaria o procesamiento jerárquico), no fue necesaria en este caso, ya que las operaciones sobre listas se resolvieron eficientemente mediante estructuras iterativas como for y while.

De ser requerido, podría incorporarse en futuras versiones para recorrer estructuras anidadas (por ejemplo, países → regiones → ciudades), aplicando el mismo enfoque modular actual.

2 Análisis de Errores Posibles.

Durante el análisis del código original, se identificaron diversos puntos donde podían producirse errores:

2.1 Errores de archivo:

- Archivo CSV no encontrado o inaccesible.
- Falta de encabezados requeridos.
- Codificación incorrecta (solucionado forzando UTF-8).

2.2 Errores de datos:

- Celdas vacías o datos no numéricos en población o superficie.
- Valores con separadores (puntos, comas o espacios) que impedían la conversión.
- Filas con nombre vacío o nulo.

2.3 Errores de entrada del usuario:

- Ingreso de texto cuando se espera un número.
- Opciones fuera del rango válido del menú.
- Entradas vacías en campos obligatorios.

2.4 Errores lógicos:

- División por cero al calcular promedios si la lista estaba vacía.
- Filtrado o búsqueda sin resultados que generaban mensajes erróneos.

3 Cambios Aplicados al Código.

Cada error potencial fue eliminado mediante condiciones preventivas:

• En lugar de capturar excepciones, se verifican las condiciones antes de que fallen.

• Se implementaron funciones seguras como 'archivo_existe()', 'input_int()', 'input_opcion()', y 'parse_int_flexible()'.

• Se normalizaron todos los textos para evitar fallos por mayúsculas o espacios.

• Los cálculos estadísticos verifican previamente la existencia de datos válidos.

• Se agregó creación automática del archivo base 'paises_base.csv' si no existe.

4 Explicación de Cada Bloque del Código.

4.1 BLOQUE 1 - VALIDACIONES / UTILIDADES:

Incluye funciones encargadas de limpiar entradas, normalizar texto y validar números. Aquí se asegura que ningún dato vacío o incorrecto ingrese al sistema.

4.2 BLOQUE 2 - PERSISTENCIA (CSV):

Contiene la lógica para cargar y guardar los datos de países en archivos CSV. Se incluyen verificaciones de encabezados y formatos, y se genera automáticamente un archivo de ejemplo si no existe.

4.3 BLOQUE 3 - FUNCIONALIDAD PRINCIPAL:

Reúne todas las operaciones del sistema: búsqueda, filtrado, ordenamiento y estadísticas. Todas las funciones son deterministas y no generan excepciones. Cada operación trabaja sobre listas de diccionarios para mantener la consistencia.

4.4 BLOQUE 4 - MENÚ PRINCIPAL:

Implementa la interacción con el usuario mediante un menú por consola. Se verifica cada entrada antes de continuar y se asegura que la opción ingresada sea válida. Los mensajes de error son descriptivos y el flujo nunca se interrumpe abruptamente.

5 Mejoras de Validaciones.

El nuevo código evita fallos incluso en situaciones extremas. Cada función devuelve valores seguros en lugar de lanzar excepciones. Además, se mejoró la usabilidad mediante mensajes de advertencia claros. El manejo de datos numéricos se realiza con conversión manual, eliminando símbolos comunes sin depender de expresiones regulares.

Ejemplo: la función parse_int_flexible('45.376.763') devuelve 45376763.
Si se ingresa un valor no numérico ('abc'), devuelve 0 sin interrumpir la ejecución.

6 Ejemplo de Ejecución.

Ejemplo de flujo del programa:

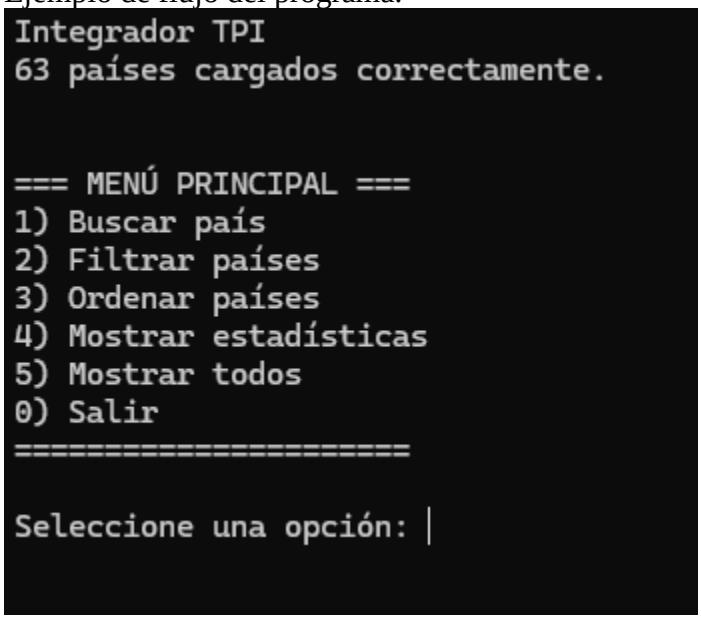

Figura 1: Menú.

Figura 2: 1- Buscar país

```
=== MENU PRINCIPAL ===
1) Buscar país
2) Filtrar países
3) Ordenar países
4) Mostrar estadísticas
5) Mostrar todos
0) Salir
=====================

Seleccione una opción: 2


Filtrar por (a) continente, (b) población, (c) superficie: continente
Opción inválida. Opciones válidas: a, b, c
Filtrar por (a) continente, (b) población, (c) superficie: a

Ingrese el continente: asia


-----------------------------------------------------------------
Nombre               |    Población | Superficie | Continente
-----------------------------------------------------------------
Turquía              |     86400000 |     783562 | Asia
Rusia                |    145000000 |   17098242 | Europa/Asia
China                |   1412000000 |    9596961 | Asia
India                |   1440000000 |    3287263 | Asia
Japón                |    125000000 |     377975 | Asia
Corea del Sur        |     52200000 |     100210 | Asia
Corea del Norte      |     26000000 |     120538 | Asia
Indonesia            |    281000000 |    1904569 | Asia
Filipinas            |    118000000 |     300000 | Asia
Tailandia            |     70200000 |     513120 | Asia
Vietnam              |    100000000 |     331212 | Asia
Malasia              |     34900000 |     330803 | Asia
Singapur             |      6000000 |        728 | Asia
Arabia Saudita       |     37000000 |    2149690 | Asia
Irán                 |     88000000 |    1648195 | Asia
Israel               |      9700000 |      22072 | Asia
```

Figura 3: 2- Filtrar Países

```
=== MENU PRINCIPAL ===
1) Buscar país
2) Filtrar países
3) Ordenar países
4) Mostrar estadísticas
5) Mostrar todos
0) Salir
======================

Seleccione una opción: 3
Ordenar por (nombre/poblacion/superficie): nombre
Sentido (asc/desc): asc
-----------------------------------------------------------------
Nombre               |    Población | Superficie | Continente
-----------------------------------------------------------------
Alemania             |     84000000 |     357022 | Europa
Arabia Saudita       |     37000000 |    2149690 | Asia
Argelia              |     46000000 |    2381741 | África
Argentina            |     46600000 |    2780400 | América
Australia            |     26700000 |    7692024 | Oceanía
Bélgica              |     11700000 |      30528 | Europa
Bolivia              |     12400000 |    1098581 | América
Brasil               |    214000000 |    8515767 | América
Camerún              |     29000000 |     475442 | África
Canadá               |     39000000 |    9984670 | América
Chile                |     19700000 |     756096 | América
China                |   1412000000 |    9596961 | Asia
Colombia             |     52300000 |    1141748 | América
Corea del Norte      |     26000000 |     120538 | Asia
Corea del Sur        |     52200000 |     100210 | Asia
Dinamarca            |      5900000 |      42933 | Europa
Ecuador              |     18500000 |     283561 | América
Egipto               |    112000000 |    1002450 | África
Emiratos Árabes Unidos |      9900000 |      83600 | Asia
España               |     47800000 |     505990 | Europa
Estados Unidos       |    334000000 |    9833517 | América
Etiopía              |    125000000 |    1104300 | África
Filipinas            |    118000000 |     300000 | Asia
```

Figura 4: 3- Ordenar países

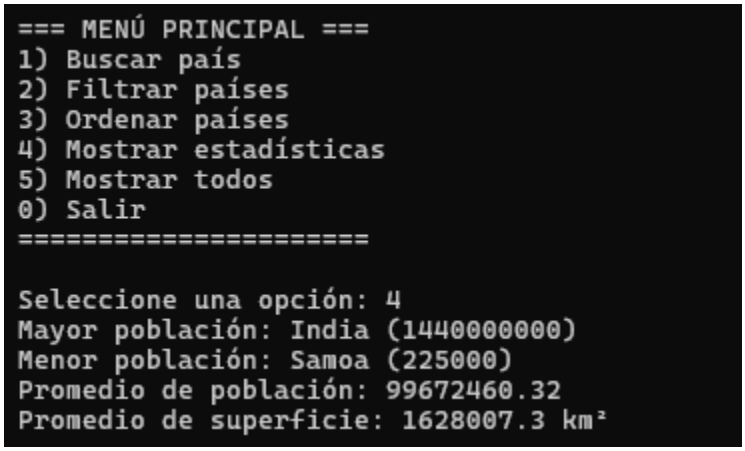

Figura 5: 4- Mostrar estadísticas

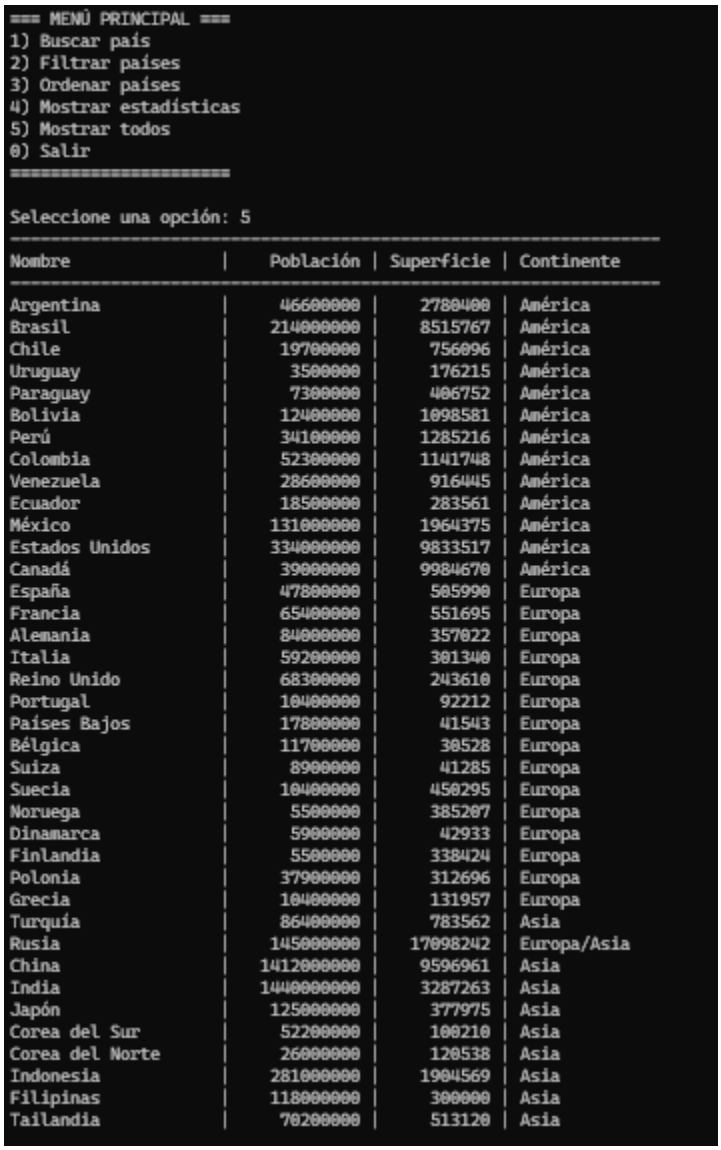

Figura 6: 5- Mostrar todos

7 Conclusiones.

El proyecto final logró alcanzar una versión completamente estable y funcional del integrador, sin necesidad de estructuras de manejo de excepciones. Todas las validaciones se realizan antes de ejecutar operaciones críticas, lo que garantiza que el programa no se detenga por errores en los datos o entradas del usuario.

8 Bibliografía.

Material aportado por la docente Fernandez Sofia.

Material aportado por el aula virtual: https://campus.frm.utn.edu.ar

Videos explicativos en el aula virtual: https://campus.frm.utn.edu.ar

Documentación en la biblioteca oficial de Python: https://www.python.org/doc