Cordoba Ezequiel y Arana Joaquín

**📘 Marco Teórico**

**Gestión de Datos de Países en Python**

**1. Listas**

Una lista en Python es una colección ordenada y mutable de elementos. Se utiliza para almacenar múltiples registros, como los países en este proyecto

Ejemplo:

paises = [

{'nombre': 'Argentina', 'poblacion': 45851378, 'superficie': 2736690, 'continente': 'america'},

...

]

📚 Fuente: Python Docs — Listas

**2. Diccionarios**

Los diccionarios son estructuras que almacenan pares clave-valor. Cada país se representa como un diccionario con claves como “nombre”, “población”, “continente”

Ejemplo:

pais = {'nombre': 'Chile', 'poblacion': 19859921, 'superficie': 743532, 'continente': 'america'}

📚 Fuente: Python Docs — Diccionarios

**3. Funciones**

Las funciones permiten encapsular lógica en bloques reutilizables. Cada función del sistema tiene una única responsabilidad, como buscar, filtrar, ordenar o calcular estadísticas

Ejemplo:

def buscar\_pais(paises, nombre):

return [p for p in paises if nombre.lower() in p['nombre'].lower()]

📚 Fuente: Programiz — Python Functions

**4. Condicionales**

Las estructuras condicionales (if, else, elif) permiten ejecutar diferentes bloques de código según condiciones lógicas. Se usan para validar opciones del menú y controlar el flujo del programa

Ejemplo:

if opcion == "1":

nombre = input("Ingrese país: ").strip()

resultado = buscar\_pais(paises, nombre)

📚 Fuente: W3Schools — Python Conditions

**5. Bucles**

Los bucles “for” y “while” permiten repetir acciones. Se usan para recorrer listas y mantener el menú activo

Ejemplo:

while True:

mostrar\_menu()

📚 Fuente: W3Schools — Python Loops

**6. Ordenamientos**

Se utiliza sorted() para ordenar listas por claves específicas. El parámetro “Key” define el criterio, y “reverse” permite invertir el orden

Ejemplo:

def ordenar\_paises(paises, clave, descendente):

return sorted(paises, key=lambda p: p[clave], reverse=descendente)

📚 Fuente: Real Python — Sorting Lists

**7. Estadísticas básicas**

Se aplican funciones como sum(), len(), max(), min() para calcular promedios, extremos y conteos

Ejemplo:

promedio = sum(p['poblacion'] for p in paises) / len(paises)

📚 Fuente: Python Docs — Built-in Functions

**8. Archivos CSV**

El sistema lee y escribe archivos CSV para almacenar los datos. Se usa csv.DictReader para leer y csv.Writer para escribir

Ejemplo:

with open("paises.csv", newline='', encoding='utf-8') as archivo:

lector = csv.DictReader(archivo)

paises = [dict(p) for p in lector]

📚 Fuente: Python Docs — CSV Module

**9. Escritura de archivos**

Se usa open() en modo “w” para crear o sobrescribir archivos. Esto permite generar el CSV desde cero si no existe

Ejemplo:

with open("paises.csv", "w", newline='', encoding='utf-8') as archivo:

escritor = csv.writer(archivo)

escritor.writerow(["nombre", "poblacion", "superficie", "continente"])

📚 Fuente: Python Docs — CSV Writing

**10. Validación de entradas**

Se aplican técnicas para evitar errores por entradas vacías o mal formateadas. Se usa .strip() para limpiar y try/except para capturar errores

Ejemplo:

try:

min\_p = int(input("Población mínima: ").strip())

except ValueError:

print("❌ Error: ingrese un número válido.")

📚 Fuente: Python Docs — Errors and Exceptions

**11. Manejo de errores**

El uso de permite capturar excepciones como “ValueError” o “FileNotFoundError”, evitando que el programa se detenga abruptamente

Ejemplo:

try:

with open("paises.csv") as archivo:

...

except FileNotFoundError:

print("Archivo no encontrado. Generando desde API...")

📚 Fuente: Python Docs — Exception Handling

**12. Modularización**

El sistema está dividido en módulos (main.py, api.py, menú.py, listas.py), lo que mejora la organización, facilita el mantenimiento y permite reutilizar funciones

📚 Fuente: Real Python — Python Modules

**13. Consumo de API externa**

Se utiliza “requests” para obtener datos desde una fuente externa. Se procesa el JSON recibido y se genera el CSV automáticamente

Ejemplo:

response = requests.get("https://restcountries.com/v3.1/all")

datos = response.json()

📚 Fuente: Requests Library

**14. Normalización de datos**

Se aplican transformaciones como .lower() y .strip() para asegurar que las búsquedas y filtros funcionen correctamente sin depender del formato original

Ejemplo:

if nombre.lower() in pais['nombre'].lower():

📚 Fuente: Python Docs — String Methods

**15. Interacción por consola**

El sistema utiliza inpuy() y print() para interactuar con el usuario. Se presenta un menú y se reciben opciones para ejecutar funciones específicas

Ejemplo:

print("1. Buscar país")

opcion = input("Seleccione una opción: ").strip()

📚 Fuente: W3Schools — Python Input/Output

**Conclusión**

Con este trabajo aprendimos bastante, desde cómo usar funciones hasta cómo manejar datos con listas, diccionarios y archivos CSV. También entendimos cómo consumir datos desde una API y organizar el código en módulos. Nos sirvió para ver cómo se arma un programa completo que funciona bien y se puede presentar.