Tecnicatura Universitaria en Programación a Distancia

Trabajo Práctico Integrador

Gestión de Datos de Países en Python: Filtros, Ordenamientos y Estadísticas

Universidad Tecnológica Nacional

Materia: Programación I

Profesores: Ariel Enferrel (Titular) / Martina Zabala (Tutora)

Alumnos / Comisión: Fernando Picco (C10)

Fecha de entrega: 02 de diciembre de 2025.

Objetivos del Trabajo Práctico Integrador (TPI)

Objetivo general del TPI

Desarrollar una aplicación modular en Python, capaz de procesar y analizar un conjunto de datos realista proveniente de un archivo CSV.

Objetivos específicos

- Aplicar estructuras fundamentales de Python.
- Incorporar validaciones y lectura de archivos.
- Diseñar un programa modular.

Marco teórico
Conceptos clave

- Listas (colección ordenada y mutable)
- Diccionarios (clave – valor)
- Funciones (modularización)
- Condicionales (decisiones if / match)
- Ordenamientos (sorted() con key)
- Estadísticas básicas (totales, promedios, máximos / mínimos)
- Archivos CSV (lectura estructurada con DicReader)

Marco teórico Ejemplos: .Parte 2

Listas

```python
paises = []
```

```
paises.append({
    "nombre": fila["nombre"],
    "poblacion": poblacion,
    "superficie": superficie,
    "continente": fila["continente"]
})
```

Diccionarios

```python
{
    "nombre": "Argentina",
    "poblacion": 45376763,
    "superficie": 2780400,
    "continente": "América"
}
```

Funciones

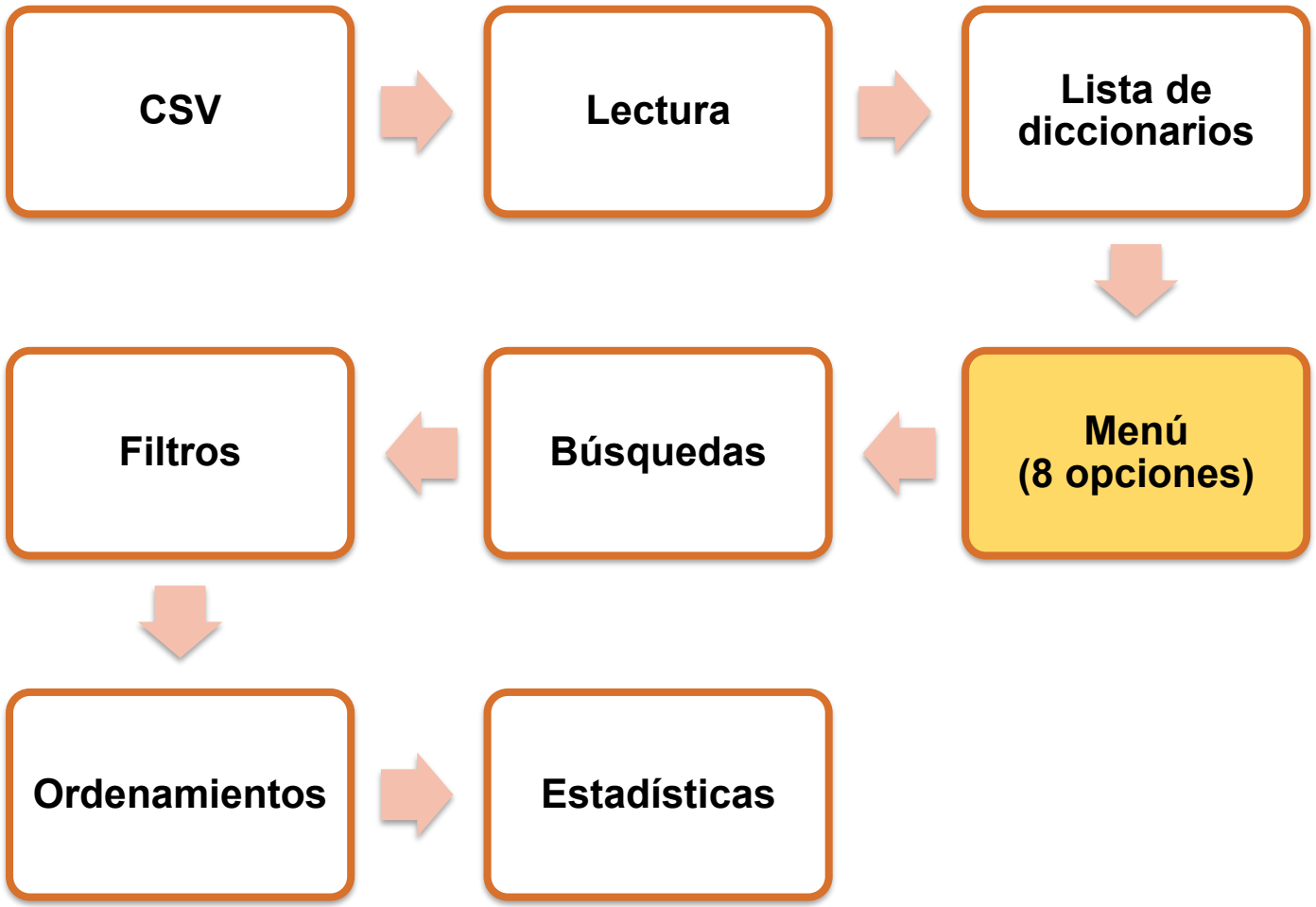
```python
def obtener_datos_paises():
def filtrar_por_continente():
def filtrar_por_rango_poblacion():
def filtrar_por_rango_superficie():
def quitar_acentos(texto):
def normalizar(texto):
def mostrar_pais(pais):
```

Estructuras condicionales

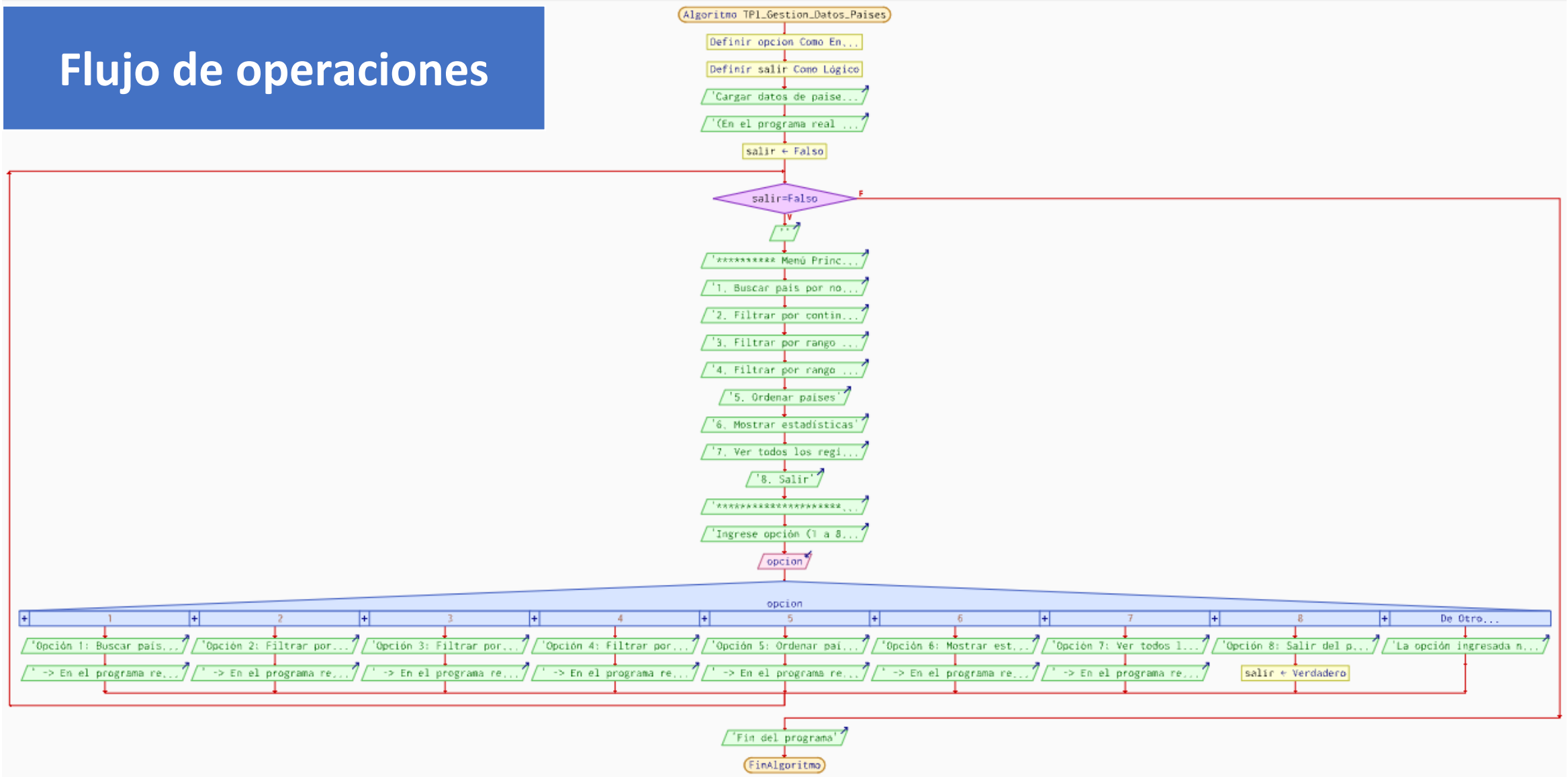
```python
opcion = input("Ingrese opción: ").strip()
match opcion:
    case '1':
        buscar_nombre_paises()
    case '2':
        filtrar_por_continente()
    case '3':
        filtrar_por_rango_poblacion()
    case '4':
        filtrar_por_rango_superficie()
    case '5':
        ordenar_paises()
    case '6':
        mostrar_estadisticas()
    case '7':
        ver_todos_los_registros()
    case '8':
        print("¡Gracias por utilizar nuestra aplicación!¡Hasta pronto!")
        break
    case _:
        print("La opción seleccionada no es válida para nuestra aplicación")
```

Diseño del caso práctico

Arquitectura conceptual del caso práctico

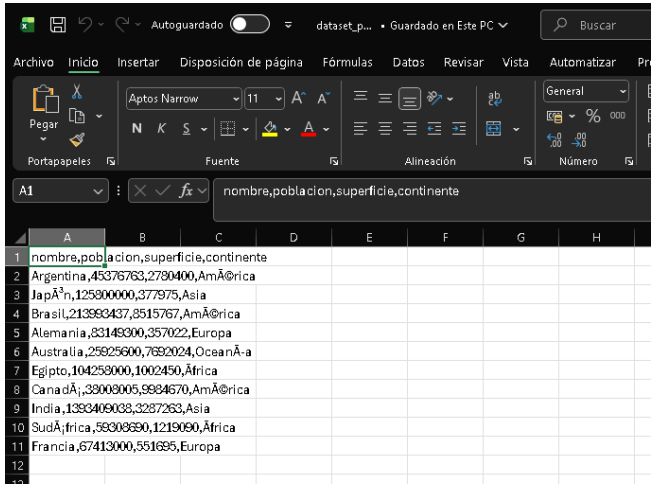


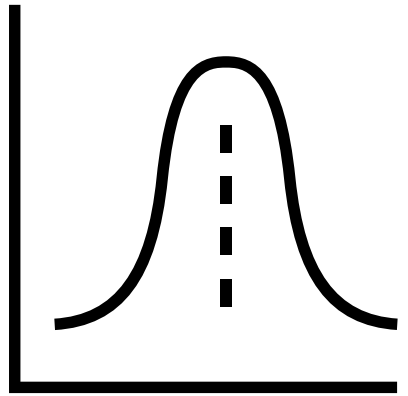
Flujo de operaciones



Arquitectura del código: Organización

- Módulo de persistencia
- Módulo de filtros
- Módulo de ordenamientos
- Módulo de estadísticas
- Menú principal





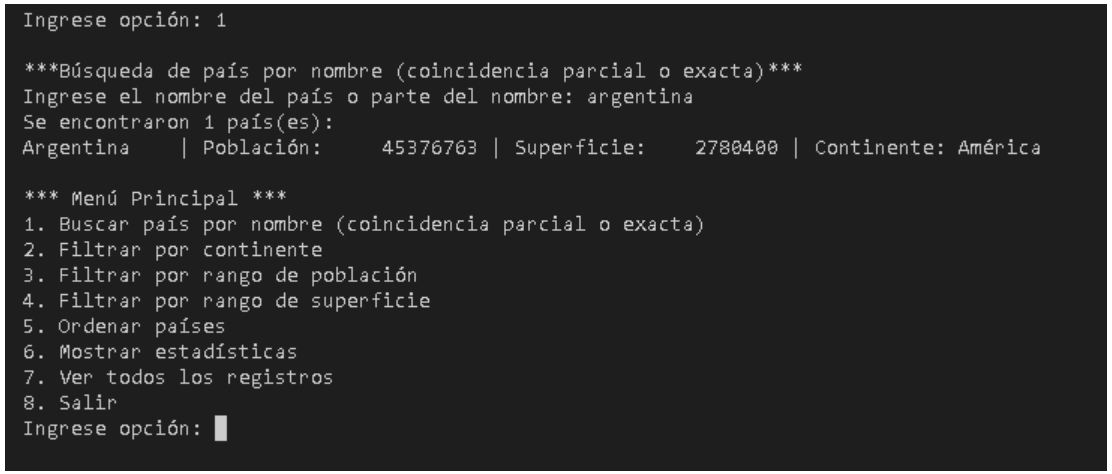
Arquitectura del código. Ejemplo

- Módulo de persistencia: *obtener_datos_paises()*

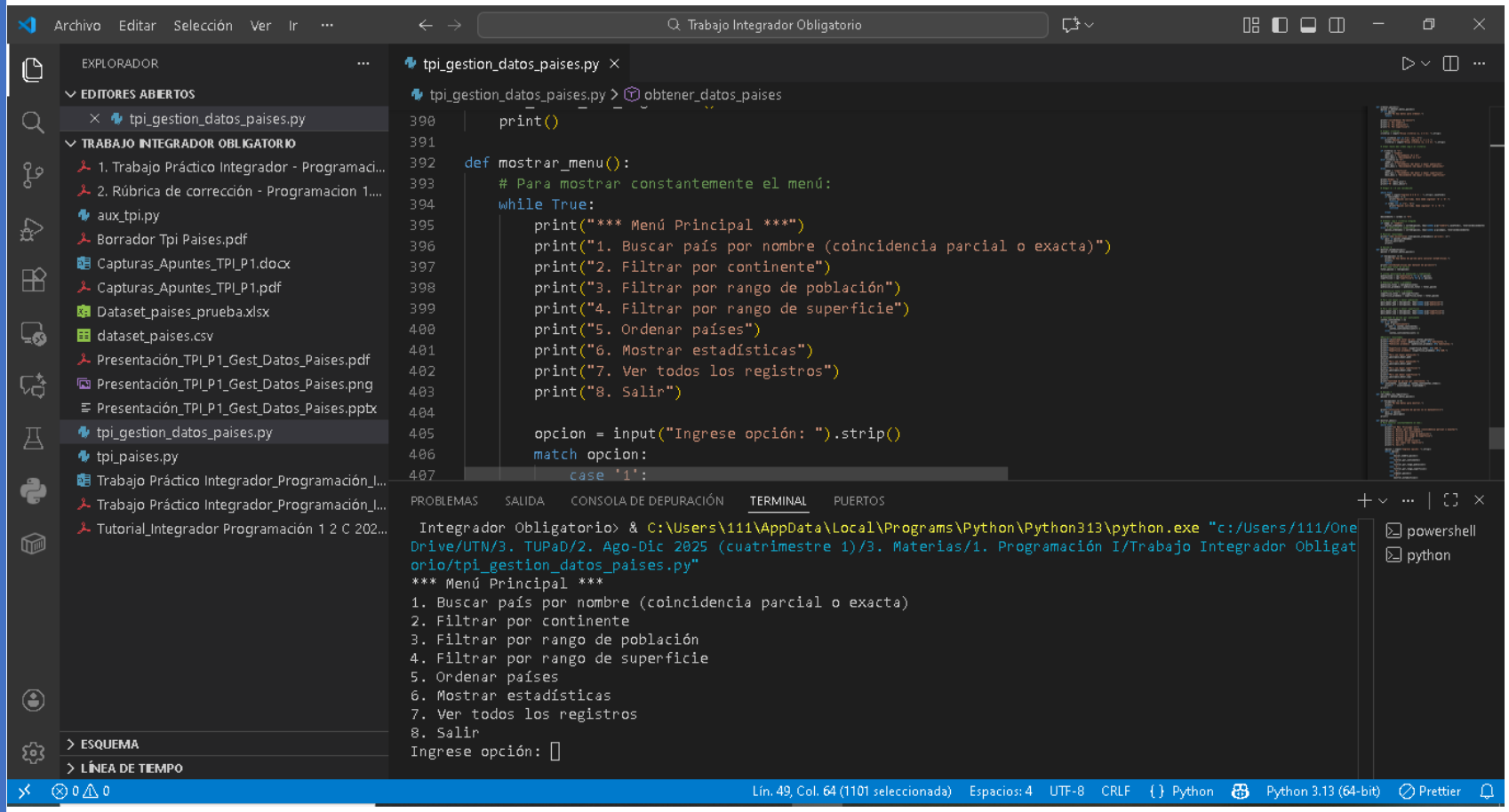
```python
def obtener_datos_paises():
    paises = []
    # with open(nombre_archivo, newline="", encoding="utf-8") as archivo:
    #     lector = csv.DictReader(archivo) # La función DictReader(), devuelve un iterador devolviendo un
                                           # diccionario clave/valor a partir de la lectura de las líneas
                                           # del archivo csv.
    #     for fila in lector:
    #         paises.append({"nombre": fila["nombre"], "poblacion": float(fila["poblacion"]), "superficie": float(fila["superficie"]), "continente": fila["continente"]})
    #return paises
    try:
        # Intento abrir el archivo csv
        with open(nombre_archivo, newline="", encoding="utf-8") as archivo:
            lector = csv.DictReader(archivo)

            for numero_fila, fila in enumerate(lector, start=2):
                # start=2 porque la línea 1 es el encabezado
            # Validar que existan todas las columnas necesarias
                columnas_obligatorias = ("nombre", "poblacion", "superficie", "continente")
                if not all(col in fila and fila[col] != "" for col in columnas_obligatorias):
                    print(f"[AVISO] Línea {numero_fila} ignorada: columnas faltantes o vacías -> {fila}")
                    continue

            # Validar que población y superficie sean numéricas
                try:
                    poblacion = float(fila["poblacion"])
                    superficie = float(fila["superficie"])
                except ValueError:
                    print(f"[AVISO] Línea {numero_fila} ignorada: valores numéricos inválidos -> {fila}")
                    continue

            # Si todo está bien, agrego el país a la lista
                paises.append({
                    "nombre": fila["nombre"],
                    "poblacion": poblacion,
                    "superficie": superficie,
                    "continente": fila["continente"]
                })
    except FileNotFoundError:
        print(f"[ERROR] No se encontró el archivo '{nombre_archivo}'.")
    except UnicodeDecodeError:
        print(f"[ERROR] Problema de codificación al leer '{nombre_archivo}'. ¿Está guardado en UTF-8?")

    return paises
```

Funcionalidades principales

- Buscar por nombre
- Filtrar por continente
- Filtrar por población / superficie
- Ordenar países
- Mostrar estadísticas
- Ver los registros



Demostración del programa



Mejoras de errores y validaciones

- Validación de acentos / mayúsculas
- Conversión numérica con try / except
- Validación de columnas
- Control de rangos
- Manejo de opciones no válidas.

```python
    # Función para quitar acentos
def quitar_acentos(texto):
    texto = texto.replace("á", "a").replace("Á", "a")
    texto = texto.replace("é", "e").replace("É", "e")
    texto = texto.replace("í", "i").replace("Í", "i")
    texto = texto.replace("ó", "o").replace("Ó", "o")
    texto = texto.replace("ú", "u").replace("Ú", "u")
    texto = texto.replace("ñ", "n").replace("Ñ", "n")
    return texto

def normalizar(texto):
    # quita acentos y convierte a minúsculas
    return quitar_acentos(texto.casefold())
```

Resultados obtenidos

Entre los resultados obtenidos destaco:

- El programa cumple con todos los requisitos del trabajo práctico.
- La lectura y validación del CSV funciona correctamente.
- Los filtros, ordenamientos y estadísticas se ejecutan sin errores.

Entre los principales desafíos:

- La validación del archivo CSV fue la parte más compleja.
- También fue desafiante normalizar textos para permitir búsquedas sin considerar acentos o mayúsculas.

Conclusiones

- Aplicación práctica de todos los conceptos.
- Mayor familiaridad con VS Code y GitHub.
- El trabajo ayudó a vencer el miedo inicial.
- Aun queda camino y mucha práctica por recorrer.
- Significó un avance real en términos de autonomía y principalmente confianza.
- Este trabajo representó un avance real en mi formación como programador.

Repositorio GitHub

(con README y código completo)

- Repositorio GitHub
- Video explicativo
- Video demostración de ejecución

¡Muchas gracias por la atención!