

Trabajo Práctico Integrador

Programación

Alumnos - Grupo n° 4

Sabrina Gimenez e Ismael Saleme

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Docente Titular

Cinthia Riggoni

31 de Octubre de 2025

Tabla de contenido

| | |
|----------------------------------|----|
| Introducción | 3 |
| Objetivos | 3 |
| Desarrollo | |
| Marco teórico | 4 |
| Flujo de operaciones principales | 7 |
| Funcionalidad | |
| Programa principal | 7 |
| Programa complementario | 13 |
| Validaciones | 19 |
| Programa en funcionamiento | 19 |
| Conclusión | 28 |
| Referencias | 29 |

Trabajo Práctico N°01

Introducción

El presente trabajo integrador tiene como objetivo desarrollar un código en Python que permita gestionar información sobre países, utilizando estructuras de datos como listas y diccionarios, junto con funciones, condicionales, ordenamientos y estadísticas básicas. El sistema debe ser capaz de leer datos desde un archivo CSV, realizar consultas dinámicas y generar indicadores clave a partir del dataset.

Este proyecto busca consolidar los conocimientos adquiridos en Programación 1, promoviendo la modularización del código, la validación de entradas y el uso de estructuras eficientes para el manejo de datos.

Objetivos

Que los participantes logren:

- Aplicar estructuras de datos como listas y diccionarios.
- Modularizar el código utilizando funciones claras.
- Implementar filtros por continente, población y superficie.
- Ordenar países por nombre, población y superficie.
- Calcular estadísticas básicas como promedios y extremos.
- Validar entradas y manejar errores de forma clara y segura.
- Documentar el proyecto y presentar una demostración funcional en video e imágenes.

Desarrollo

Marco teórico

Listas

Las listas son estructuras ordenadas y mutables ya que permiten modificar los valores de sus elementos, eliminar o añadir elementos.

En ellas se pueden almacenar tantos elementos como se desee, incluso pueden existir listas vacías (sin elementos). Para definirlas se utilizan corchetes y se separan los elementos mediante. Ejemplo: `nombre_lista = [valor1, valor2, ...]`.

En este trabajo las utilizamos para representar colecciones de países, poblaciones, superficies, etc.

Diccionarios

Un diccionario es una colección mutable y desordenada de elementos, donde cada uno está compuesto por un par único de clave-valor. Ejemplo: mi_diccionario = {"nombre": "Alice", "edad": 25, "profesión": "Ingeniera"}

En este trabajo fue la Estructura clave–valor nos permite acceder rápidamente a atributos de cada país y para representar registros con nombre, población, superficie y continente.

Funciones

Una función es un bloque de código que realiza una tarea específica. Al usar funciones es posible dividir un problema en partes pequeñas y manejables. Se compone de:

- Entrada (argumentos): datos que recibe la función.
- Proceso: instrucciones que realiza.
- Salida (retorno): resultado que devuelve.

Son muy útiles ya que permiten reutilizar una porción de código tantas veces como sea necesario.

La palabra reservada para definir una función es `def`. Para crearla debemos indicar un nombre para la función y establecer sus argumentos.

La estructura que se usa para definir funciones es la siguiente:

```
def nombre_funcion(nombre_argumento):
```

código a ejecutar

```
return valor_de_retorno
```

Ejemplo:

```
def saludar():  
    return "¡Hola! ¿Cómo estás?"  
  
saludo = saludar()  
  
print(saludo)
```

Nos permite dividir el programa en bloques reutilizables, cada uno con una tarea específica. Facilitan la organización, el mantenimiento y la legibilidad del código.

Condicionales

Los condicionales son estructuras de control que permiten ejecutar diferentes bloques de código dependiendo de si una condición es verdadera o falsa. La instrucción más básica para realizar una evaluación condicional es la sentencia `if`, que evalúa una expresión booleana (es decir, una expresión que puede ser verdadera o falsa) y ejecuta un bloque de código sólo si la condición se cumple. Además, se pueden usar otras estructuras como `elif` (abreviatura de "else if") y `else` para manejar diferentes casos o situaciones, ofreciendo una mayor flexibilidad en la lógica del programa. Las condicionales son fundamentales para la toma de decisiones en los programas, permitiendo que el flujo de ejecución se ajuste dinámicamente según las circunstancias.

Ejemplo:

```
if puntuación >= 90:  
    print("Excelente")  
  
elif puntuación >= 70:  
    print("Bueno")  
  
else:  
    print("Necesitas mejorar")
```

Nos permiten tomar decisiones en el programa según ciertas condiciones. Se usan para validar entradas, aplicar filtros y mostrar mensajes adecuados.

Ordenamientos

Los métodos de ordenamiento son algoritmos que realizan la operación de arreglar los registros de una tabla en algún orden secuencial de acuerdo a un criterio de ordenamiento. El ordenamiento se efectúa con base en el valor de algún campo en un grupo de datos. El ordenamiento puede estar dado de forma iterativa o recursiva según la naturaleza y forma de ejecución del mismo.

Ejemplos de métodos de ordenamiento: BubbleSort, SelectionSort, InsertionSort, etc.

Se aplican para organizar los países según criterios como nombre, población o superficie. Se utiliza sorted() con funciones key personalizadas.

Estadísticas básicas

Son funciones que calculan el promedio o el valor típico de una población o muestra.

Los aplicamos para obtener indicadores relevantes del dataset.

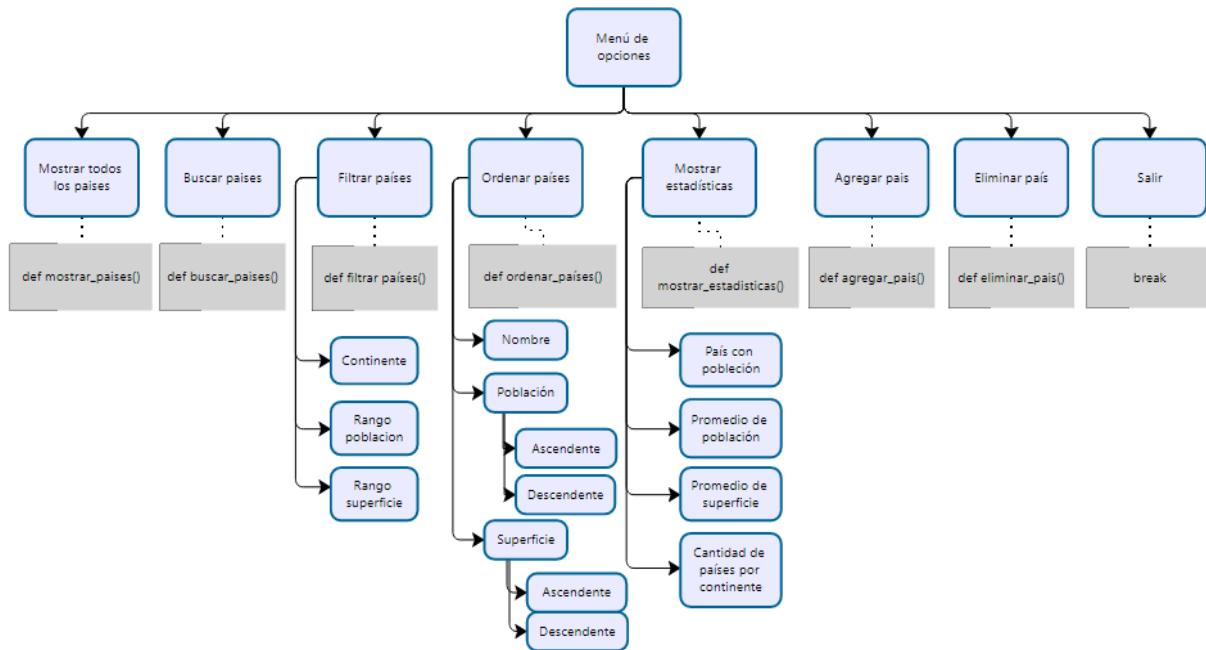
Archivos CSV

Los archivos CSV (Comma Separated Values) son similares a planillas: cada línea representa una fila, y los valores están separados por comas.

Ejemplo de archivo **productos.csv**

Formato de texto plano que permite almacenar datos estructurados. Utilizamos el módulo **csv** para leer y procesar los registros de países.

Flujo de operaciones principales



Funcionalidad

Programa Principal

- Importación de herramientas a utilizar:

```

2 import csv
3 import os
4 import unicodedata
5 import fcs_filtrado_paises
6
7 #-----
8 # RUTA ARCHIVO
9 #-----
10
11 RUTA_ARCHIVO = "ARCHIVO_PAISES\\paises.csv"
12
  
```

- Inicialización del programa:

```

16
17 #Funcion para crear archivo si no existe
18 def inicializar_archivo():
19     if not os.path.exists("paises.csv"):
20         with open("paises.csv", "w", encoding="utf-8",newline="") as archivo:
21
22             #nombre de las Claves/Encabezados
23             encabezados = ["nombre", "poblacion", "superficie", "continente"]
24
25             #Crea una lista de diccionarios y toma la primer linea/fila como encabezados como (key), y los demás como (value)
26             escritor = csv.DictWriter(archivo, fieldnames=encabezados)
27
28             #escribe los encabezados en el archivo
29             escritor.writeheader()
30
  
```

- Quita de tildes para evitar errores de sintaxis:

```

30
31     #Funcion que quita tildes para evitar errores en comparaciones y busquedas.
32     def quitar_tildes(texto):
33         return ''.join(
34             c for c in unicodedata.normalize('NFD', texto)
35             if unicodedata.category(c) != 'Mn'
36         )
37

```

- Muestra de países:

```

41
42     def mostrar_paises():
43         try:
44             with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
45                 #lee el contenido del archivo como lista de diccionarios
46                 contenido = csv.DictReader(archivo)
47                 contenido = list(contenido)
48                 if not contenido:
49                     print("No hay paises registrados.")
50                     return
51                 # Encabezado con ancho fijo
52                 print("-" * 70)
53                 print(f"{'Nombre':<20} {'Población':<17} {'Superficie':<10} {'Continente':>15}")
54                 print("-" * 70)
55                 contenido_valido = []
56                 #Países
57                 for fila in contenido:
58                     try:
59                         nombre = fila['nombre']
60                         poblacion = int(fila['poblacion'])
61                         superficie = float(fila['superficie'])
62                         continente = fila['continente']
63                         print(f"{'nombre':<20} | {poblacion:>12} | {superficie:>14} | {continente:>15} |")
64                         contenido_valido.append(fila)
65                     except ValueError:
66                         #elimina países con errores de formato
67                         print(f"- País {fila.get('nombre')}"' eliminado por error de formato.")
68                     print("-" * 70)
69
70             #Sobrescribo el csv original
71             if contenido_valido:
72                 with open(RUTA_ARCHIVO, "w", encoding="utf-8") as archivo:
73                     encabezado = ["nombre","poblacion","superficie","continente"]
74                     escritor = csv.DictWriter(archivo, fieldnames=encabezado)
75                     escritor.writeheader()
76                     escritor.writerows(contenido_valido)
77             except FileNotFoundError:
78                 print("El archivo no existe. Creando uno nuevo....")
79                 inicializar_archivo()

```

- Búsqueda de países:

```

81 def buscar_pais():
82
83     pais_buscar = input("\nIngrese el nombre del país que desea buscar: ").strip()
84
85     if not pais_buscar:
86         print("\n" + "=" * 30)
87         print("Entrada vacía. Intentelo de nuevo...")
88         return
89
90     if pais_buscar.isdigit():
91         print("\n" + "=" * 30)
92         print("No puede ingresar números. Intentelo de nuevo...")
93         return
94
95     #Quito las tildes al país ingresado para la comparación.
96     pais_sin_tildes = quitar_tildes(pais_buscar.lower())
97
98     encontrado = False
99
100    try:
101        with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
102            lector = csv.DictReader(archivo)
103
104            for fila in lector:
105
106                #Quito las tildes del país que está dentro del CSV
107                nombre_sin_tildes = quitar_tildes(fila["nombre"].lower())
108
109                #Comparación
110                if nombre_sin_tildes == pais_sin_tildes:
111
112                    print("\n" + "=" * 30)
113                    print("|País Encontrado: ")
114                    print(f" |Nombre: {fila['nombre']} | Población: {fila['poblacion']} | Superficie: {fila['superficie']} | Continente: {fila['continente']}")
115
116                    encontrado = True
117                    break
118
119                if not encontrado:
120                    print("El país ingresado no se encuentra en el registro.")
121
122    except FileNotFoundError:
123        print("El archivo no existe. Creando uno nuevo....")
124        inicializar_archivo()
125

```

- Filtrado de países:

```

126 def filtrar_paises():
127
128     print("\n" + "=" * 30)
129     print("|Tipos de filtro: \n")
130
131     print("|1. Filtrar por continente")
132     print("|2. Filtrar por rango de población")
133     print("|3. Filtrar por rango de superficie")
134
135     try:
136
137         opc = int(input("\n-Seleccione una opción de filtro: "))
138         print("\n" + "=" * 30)
139
140         match opc:
141             case 1:
142                 fcs_filtrado_paises.filtrado_continente()
143             case 2:
144                 fcs_filtrado_paises.filtrado_rango_poblacion()
145             case 3:
146                 fcs_filtrado_paises.filtrado_superficie()
147             case _:
148                 print("Valor inválido. Intentelo de nuevo..")
149                 return
150
151     except ValueError:
152         print("\n" + "=" * 30)
153         print("ERROR. No puede ingresar letras. Intentelo de nuevo...")

```

- Ordenamiento de países según opción seleccionada:

```

155 def ordenar_paises():
156
157     print("\n" + "=" * 30)
158     print("\n- Tipos de ordenamiento:\n ")
159
160     print("1. Por nombre")
161     print("2. Por poblacion (ascendente o descendente)")
162     print("3. Por superficie (ascendente o descendente)")
163
164     try:
165         opc = int(input("\n- Ingrese una opcion de ordenamiento: "))
166         print("\n" + "=" * 30)
167
168         match opc:
169
170             #Ordenar alfabeticamente
171             case 1:
172                 fcs_filtrado_paises.ordenamiento_nombre()
173
174
175             #Ordenar por poblacion (ascendente o descendente)
176             case 2:
177                 print("|Ascendente (a)")
178                 print("|Descendente (d)")
179
180             try:
181                 opcion = input("\n- Elige una opcion: ").lower()
182
183                 match opcion:
184                     case "a":
185                         fcs_filtrado_paises.ordenamiento_poblacion(opcion)
186
186                     case "d":
187                         fcs_filtrado_paises.ordenamiento_poblacion(opcion)
188                     case _:
189                         print("\n" + "=" * 30)
190                         print("Valor invalido. Intentelo de nuevo...")
191                         return
192
192             except ValueError:
193                 print("\n" + "=" * 30)
194                 print("ERROR. No puede ingresar numeros. Intentelo de nuevo...")
195
196             #Ordenar por superficie (ascendente o descendente)
197             case 3:
198                 print("|Ascendente (a)")
199                 print("|Descendente (d)")
200
201             try:
202                 opcion = input("\n- Elige una opcion: ").lower()
203
204                 match opcion:
205                     case "a":
206                         fcs_filtrado_paises.ordenamiento_superficie(opcion)
207                     case "d":
208                         fcs_filtrado_paises.ordenamiento_superficie(opcion)
209             except ValueError:
210                 print("\n" + "=" * 30)
211                 print("ERROR. No puede ingresar numeros. Intentelo de nuevo...")
212
213     except ValueError:
214         print("\n" + "=" * 30)
215         print("ERROR. No puede ingresar letras. Intentelo de nuevo...")

```

- Muestra de estadísticas según opción seleccionada:

```

217     def mostrar_estadisticas():
218         print("\n" + "=" * 30)
219         print("\n- Tipos de estadísticas:\n")
220
221         print("[1. País con menor y mayor población]")
222         print("[2. Promedio de población]")
223         print("[3. Promedio de superficie]")
224         print("[4. Cantidad de países por continente]")
225
226     try:
227
228         opc = int(input("\n- Ingrese una opción de estadística: "))
229         print("\n" + "=" * 30)
230
231         match opc:
232             case 1:
233                 fcs_filtrado_paises.pais_menor_mayor()
234             case 2:
235                 fcs_filtrado_paises.promedio_poblacion()
236             case 3:
237                 fcs_filtrado_paises.promedio_superficie()
238             case 4:
239                 fcs_filtrado_paises.paises_por_continente()
240             case _:
241                 print("No puede ingresar números negativos. Intentelo de nuevo...")
242                 return
243
244     except ValueError:
245         print("ERROR. No puede ingresar letras. Intentelo de nuevo...")

```

- Agregar país:

```

def agregar_pais():
    continentes_validos = ["américa", "europa", "asia", "africa", "oceania", "antartida"]

    #-----
    #Entradas del usuario y verificaciones
    #-----

    #Entrada y validación de nombre de país
    nuevo_nombre = str(input("\n-Ingrese el nombre del país a ingresar: ").strip())
    if not nuevo_nombre or nuevo_nombre.isdigit():

        print("=" * 30)
        print("Entrada vacía o valor incorrecto. Intentelo de nuevo...")
        return

    with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
        lector = csv.DictReader(archivo)
        paises = list(lector)

    #Verificación para saber si existe dentro del csv
    for país in paises:

        if quitar_tildes(país["nombre"].lower()) == quitar_tildes(nuevo_nombre.lower()):
            print("El país ingresado ya se encuentra dentro del sistema..")
            return

    #-----
    #Entrada y validación de población
    nuevo_poblacion = input("\n-Ingrese la población del país: ").strip()
    if not nuevo_poblacion.isdigit() or int(nuevo_poblacion) < 0:

        print("=" * 30)
        print("No puede ingresar letras o números negativos. Intentelo de nuevo...")
        return

    #-
    #Entrada y validación de superficie
    nuevo_superficie = input("\n-Ingrese la superficie del país: ").strip()

    try:
        if int(nuevo_superficie) < 0:

            print("=" * 30)
            print("No puede ingresar letras o números negativos. Intentelo de nuevo...")
            return
    except ValueError:
        print("=" * 30)
        print("La superficie debe ser un número válido. Intentelo de nuevo...")
        return

    #-----
    #Entrada y validación de continente
    nuevo_continente = str(input("\n-Ingrese el continente del país: ").strip())
    if nuevo_continente not in continentes_validos:

```

```

def agregar_pais():
    print("-" * 30)
    print("El continente ingresado no existe. Intentelo de nuevo")
    return

#Creacion de la fila del pais
nuevo_pais = {"nombre": nuevo_nombre.capitalize(), "poblacion": nuevo_poblacion, "superficie": nuevo_superficie, "continente": nuevo_continente.capitalize()}

try:
    #Escritura de la nueva fila
    with open(RUTA_ARCHIVO, "a", newline="") as archivo:
        encabezados = ["nombre", "poblacion", "superficie", "continente"]
        escritor = csv.DictWriter(archivo, fieldnames=encabezados)
        escritor.writerow(nuevo_pais)
        print("\n| País agregado con éxito.")

except FileNotFoundError:
    print(" El archivo no existe. Creando uno nuevo...")
    inicializar_archivo()

except csv.Error:
    print("Error al procesar el archivo CSV. Verifique el formato.")

```

- Eliminar pais:

```

320 def eliminar_pais():
321
322     pais_eliminar = str(input("\n-Ingrese el país que desea eliminar: ")).strip()
323
324     #Entrada vacía
325     if not pais_eliminar:
326         print("Entrada vacía. Intentelo de nuevo.")
327         return
328
329     #Existencia de archivo
330     if not os.path.exists(RUTA_ARCHIVO):
331         print("El archivo no existe. Creando uno nuevo...")
332         inicializar_archivo()
333         return
334
335     try:
336         with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
337
338             lector = csv.DictReader(archivo)
339             paises = list(lector)
340
341             encontrado = False
342             paises_modificado = []
343
344             #Busqueda del país ingresado por el usuario
345             for pais in paises:
346                 if quitar_tildes(pais["nombre"].lower()) == quitar_tildes(pais_eliminar.lower()):
347                     encontrado = True
348                 else:
349                     paises_modificado.append(pais)
350
351             #Escritura del nuevo archivo (sin el país eliminado)
352             if encontrado:
353
354                 with open(RUTA_ARCHIVO, "w", encoding="utf-8", newline="") as archivo:
355                     encabezado = ["nombre", "poblacion", "superficie", "continente"]
356                     escritor = csv.DictWriter(archivo, fieldnames=encabezado)
357                     escritor.writeheader()
358                     escritor.writerows(paises_modificado)
359
360                     print("-" * 30)
361                     print("\n| País eliminado con éxito -")
362
363             else:
364                 print("-" * 30)
365                 print("El país ingresado no se encuentra en el sistema. ")
366
367     except csv.Error:
368         print("Error al procesar el archivo CSV. Verifique el formato.")

```

- Menú de opciones

```

371     def menu():
372         while True:
373             try:
374                 print("=" * 30)
375
376                 print(" ===== Menú de opciones: =====\n")
377                 print("|1. Mostrar todos los países: ")
378                 print("|2. Buscar país (por nombre): ")
379                 print("|3. Filtrar países: ")
380                 print("|4. Ordenar países: ")
381                 print("|5. Mostrar estadísticas: ")
382                 print("|6. Agregar país: ")
383                 print("|7. Eliminar país: ")
384                 print("|8. Salir")
385
386                 print("=" * 30)
387
388                 opcion = int(input("\n- Seleccione una opción: "))
389
390                 match opcion:
391                     case 1:
392                         mostrar_paises()
393                     case 2:
394                         buscar_pais()
395
396                     case 3:
397                         filtrar_paises()
398                     case 4:
399                         ordenar_paises()
400                     case 5:
401                         mostrar_estadisticas()
402                     case 6:
403                         agregar_pais()
404                     case 7:
405                         eliminar_pais()
406                     case 8:
407                         print("- Saliendo del programa..... Hasta luego!")
408                         break
409                     case _:
410                         print("Opción no válida. Intente de nuevo.")
411
412             except ValueError:
413                 print("=" * 30)
414                 print("ERROR. No puede ingresar algo que no sea un número. Intentelo de nuevo... ")
415
416     #-----
417     #PRINCIPAL
418     #-----
419
420     menu()

```

Programa Complementario

- Importación de herramientas a utilizar:

```

❸ lcs_filtrado_paises.py > ...
1  import csv
2  import unicodedata
3
4  RUTA_ARCHIVO = "ARCHIVO_PAISES\\países.csv"
5
6  def quitar_tildes(texto):
7      return ''.join(
8          c for c in unicodedata.normalize('NFD', texto)
9          if unicodedata.category(c) != 'Mn'
10     )
11

```

- Filtrado rango superficie:

```

133     def filtrado_superficie():
134
135         try:
136
137             #Entrada de usuario y validaciones.
138             sup_min = int(input("Ingrese la superficie minima: "))
139
140             if not sup_min:
141                 print("\n" + "=" * 30)
142                 print("Entrada vacia. Intentelo de nuevo...")
143                 return
144
145             sup_max = int(input("Ingrese la superficie maxima: "))
146
147             if not sup_max:
148                 print("\n" + "=" * 30)
149                 print("Entrada vacia. Intentelo de nuevo...")
150                 return
151
152         except ValueError:
153             print("\n" + "=" * 30)
154             print("No puede ingresar letras. Intentelo de nuevo...")
155             return
156
157     try:
158
159         #Apertura de archivo
160         with open(RUTA_ARCHIVO, "r", encoding="utf-8") as f:
161             lector = csv.DictReader(f)
162             encontrados = []
163
164
165             #Busqueda de las superficies solicitadas
166             for fila in lector:
167                 try:
168                     superficie = float(fila["superficie"])
169
170                     if sup_min <= superficie <= sup_max:
171
172                         encontrados.append(fila)
173
174                 except ValueError:
175                     continue
176
177
178             #Escritura de las superficies solicitadas
179             if encontrados:
180
181                 print("\n|Países con superficie entre {sup_min} y {sup_max} |")
182
183                 print("-" * 70)
184                 print(F"{'Nombre':<20} {'Población':<17} {'Superficie':<10} {'Continente':>15}")
185                 print("-" * 70)
186
187                 for pais in encontrados:
188                     nombre = pais["nombre"]
189                     poblacion = int(pais["poblacion"])
190                     superficie = float(pais["superficie"])
191                     continente = pais["continente"]
192
193                     print(F"({nombre:<20} | {poblacion:<17} | {superficie:<10} | {continente:>15})")
194                     print("-" * 70)
195
196         except FileNotFoundError:
197             inicializar_archivo()

```

- Filtrado de continente:

```

16     def filtrado_continente():
17
18         continentes_validos = ["america","europa","asia","africa","oceania","antartida"]
19
20         continente = input("Ingrese el continente con el que desea filtrar: ").strip()
21
22         #Verificaciones de continente
23         if not continente:
24             print("\n" + "=" * 30)
25             print("Entrada vacia. Intentelo de nuevo...")
26             return
27
28         if continente.isdigit():
29             print("\n" + "=" * 30)
30             print("No puede ingresar numero. Intentelo de nuevo...")
31             return
32
33         if continente not in continentes_validos:
34             print("\n" + "=" * 30)
35             print("El continente ingresado no existe. Intentelo de nuevo...")
36             return
37
38
39         cont_sin_tildes = quitar_tildes(continente.lower())

```

```

40
41     try:
42         with open(RUTA_ARCHIVO, "r", encoding="utf-8") as f:
43             lector = csv.DictReader(f)
44             encontrados = []
45
46             for fila in lector:
47                 if guitar_tildes(fila["continente"].lower()) == cont_sin_tildes:
48                     encontrados.append(fila)
49
50             if encontrados:
51                 print(f"\nPaises del continente de {continente}: \n")
52
53                 print("-" * 70)
54                 print(f"\t|Nombre':<20} {'Población':<17} {'Superficie':<10} {'Continente':>15}")
55                 print("-" * 70)
56
57
58                 for pais in encontrados:
59                     nombre = pais["nombre"]
60                     poblacion = int(pais["poblacion"])
61                     superficie = float(pais["superficie"])
62                     continente = pais["continente"]
63
64                     print(f"\t|{nombre:<20} | {poblacion:>12} | {superficie:>14} | {continente:>15} |")
65
66     except FileNotFoundError:
67         print(" El archivo no existe. Creando uno nuevo...")
68         inicializar_archivo()

```

- Filtrado rango población:

```

70     def filtrado_rango_poblacion():
71         try:
72
73             #Entrada de usuario y validaciones
74             pobl_min = int(input("Ingrese la poblacion minima: "))
75
76             if not pobl_min:
77                 print("\n" + "=" * 30)
78                 print("Entrada vacia. Intentelo de nuevo...")
79                 return
80
81             pobl_max = int(input("Ingrese la poblacion maxima: "))
82
83             if not pobl_max:
84                 print("\n" + "=" * 30)
85                 print("Entrada vacia. Intentelo de nuevo...")
86                 return
87
88
89         except ValueError:
90             print("\n" + "=" * 30)
91             print("No puede ingresar letras. Intentelo de nuevo...")
92             return
93
94         try:
95
96             #Apertura de archivo
97             with open(RUTA_ARCHIVO, "r", encoding="utf-8") as f:
98                 lector = csv.DictReader(f)
99                 encontrados = []
100
101             #Busqueda de las poblacion solicitadas
102             for fila in lector:
103                 try:
104                     poblacion = int(fila["poblacion"])
105
106                     if pobl_min <= poblacion <= pobl_max:
107                         encontrados.append(fila)
108
109                 except ValueError:
110                     continue
111
112             #Escritura de la poblacion solicitadas
113             if encontrados:
114
115                 print("\n" + "=" * 30)
116                 print(f"\t|Paises con poblacion entre {pobl_min} y {pobl_max} de habitantes: \n")
117
118                 print("-" * 70)
119                 print(f"\t\t|Nombre':<20} {'Población':<17} {'Superficie':<10} {'Continente':>15}")
120                 print("-" * 70)
121
122                 for pais in encontrados:
123                     nombre = pais["nombre"]
124                     poblacion = int(pais["poblacion"])
125                     superficie = float(pais["superficie"])
126                     continente = pais["continente"]
127
128                     print(f"\t\t|{nombre:<20} | {poblacion:<17} | {superficie:<10} | {continente:>15}|")
129
130         except FileNotFoundError:
131             print(" El archivo no existe. Creando uno nuevo...")
132             inicializar_archivo()

```

- Buscar por nombre y ordenamiento por nombre:

```

198 #-----#
199 #ORDENAMIENTOS
200 #-----#
201
202 def buscar_nombre(pais):
203     return pais["nombre"].lower()
204 def ordenamiento_nombre():
205     try:
206
207         #Apertura de archivo
208         with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
209             lector = csv.DictReader(archivo)
210             paises = list(lector)
211
212         if not paises:
213             print("\n" + "=" * 30)
214             print("El archivo esta vacio. No hay paises para ordenar.")
215
216         paises_ordenados = sorted(paises, key=buscar_nombre)
217
218         #Se sobrecribe el CSV original por el nuevo ordenado
219         with open(RUTA_ARCHIVO, "w", encoding="utf-8") as archivo:
220             encabezado = ["nombre", "poblacion","superficie","continente"]
221             escritor = csv.DictWriter(archivo, fieldnames=encabezado)
222             escritor.writeheader()
223             escritor.writerows(paises_ordenados)
224             print("\n" + "=" * 30)
225             print("Paises ordenado correctamente.")
226
227     except FileNotFoundError:
228         print(" El archivo no existe. Creando uno nuevo...")
229         inicializar_archivo()

```

- Obtener población y ordenamiento por población:

```

231 def obtener_poblacion(pais):
232     return int(pais["poblacion"])
233 def ordenamiento_poblacion(opcion):
234     try:
235
236         with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
237             lector = csv.DictReader(archivo)
238             paises = list(lector)
239
240         if not paises:
241             print("\n" + "=" * 30)
242             print("El archivo esta vacio. No hay paises para ordenar.")
243
244         #Ordenamiento Ascendente y Descendente (a/d)
245         if opcion == "a":
246             ordenados_poblacion = sorted(paises, key=obtener_poblacion)
247             print("\n" + "=" * 30)
248             print("- Paises ordenados ascendentemente - .")
249
250         elif opcion == "d":
251             ordenados_poblacion = sorted(paises, key=obtener_poblacion, reverse=True)
252             print("\n" + "=" * 30)
253             print("- Paises ordenados descendentemente - .")
254
255         #Se sobrescribe el CSV original por el nuevo ordenado
256         with open(RUTA_ARCHIVO, "w", encoding="utf-8") as archivo:
257             encabezado = ["nombre","poblacion","superficie","continente"]
258             escritor = csv.DictWriter(archivo, fieldnames=encabezado)
259             escritor.writeheader()
260             escritor.writerows(ordenados_poblacion)
261
262     except FileNotFoundError:
263         print(" El archivo no existe. Creando uno nuevo...")
264         inicializar_archivo()

```

- Obtener superficie y ordenamiento por superficie:

```

266     def obtener_superficie(pais):
267         return float(pais["superficie"])
268     def ordenamiento_superficie(opcion):
269         try:
270             with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
271                 lector = csv.DictReader(archivo)
272                 paises = list(lector)
273
274                 if not paises:
275                     print("\n" + "=" * 30)
276                     print("El archivo está vacío. No hay países para ordenar.")
277
278                 # Ordenamiento Ascendente o Descendente (a/d)
279                 if opcion == "a":
280                     ordenados_superficie = sorted(paises, key=obtener_superficie)
281                     print("- Países ordenados ascendenteamente - .")
282
283                 elif opcion == "d":
284                     ordenados_superficie = sorted(paises, key=obtener_superficie, reverse=True)
285                     print("- Países ordenados descendenteamente - .")
286
287                 # Sobreescribo el CSV original por el nuevo ordenado
288                 with open(RUTA_ARCHIVO, "w", encoding="utf-8") as archivo:
289                     encabezado = ["nombre", "poblacion", "superficie", "continente"]
290                     escritor = csv.DictWriter(archivo, fieldnames=encabezado)
291                     escritor.writeheader()
292                     escritor.writerows(ordenados_superficie)
293
294
295         except FileNotFoundError:
296             print(" El archivo no existe. Creando uno nuevo...")
297             inicializar_archivo()

```

- País con mayor o menor población:

```

303     def pais_menor_mayor():
304         try:
305             with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
306                 lector = csv.DictReader(archivo)
307
308                 primera = True
309                 nombre_mayor = nombre_menor = None
310                 may_poblacion = None
311                 men_poblacion = None
312
313                 for fila in lector:
314                     try:
315                         pobl = int(fila["poblacion"])
316                     except (ValueError, KeyError):
317                         continue
318
319                     if primera:
320                         may_poblacion = men_poblacion = pobl
321                         nombre_mayor = nombre_menor = fila["nombre"]
322                         primera = False
323
324                     else:
325                         if pobl > may_poblacion:
326                             may_poblacion = pobl
327                             nombre_mayor = fila["nombre"]
328
329                         if pobl < men_poblacion:
330                             men_poblacion = pobl
331                             nombre_menor = fila["nombre"]
332
333                     if primera:
334                         print("No hay países con población válida.")
335                         return
336
337                     print(f"\nPais con mayor poblacion: {nombre_mayor} |Poblacion: {may_poblacion}")
338                     print(f"\nPais con menor poblacion: {nombre_menor} |Poblacion: {men_poblacion}")
339
340         except FileNotFoundError:
341             inicializar_archivo()

```

- Promedio población:

```

343 def promedio_poblacion():
344     try:
345         with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
346             lector = csv.DictReader(archivo)
347             paises = list(lector)
348
349             suma_poblacion = 0
350
351             #Suma de la poblacion de cada país.
352             for fila in paises:
353                 try:
354                     suma_poblacion += int(fila["poblacion"])
355                 except ValueError:
356                     continue
357
358             #Promedio total
359
360             try:
361                 prom_poblacion = suma_poblacion / len(paises)
362             except ZeroDivisionError:
363                 print("ERROR. Valor invalido para division..")
364
365             print("\n" + "=" * 30)
366             print(f"\n|El promedio de poblacion mundial es de: {round(prom_poblacion)}")
367
368     except FileNotFoundError:
369         print(" El archivo no existe. Creando uno nuevo...")
370         inicializar_archivo()

```

- Promedio superficie:

```

372 def promedio_superficie():
373     try:
374         with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
375             lector = csv.DictReader(archivo)
376             paises = list(lector)
377
378             suma_superficie = 0
379
380             #Suma de la superficie de cada país
381             for fila in paises:
382                 try:
383                     suma_superficie += float(fila["superficie"])
384                 except ValueError:
385                     continue
386
387             #Promedio total
388             prom_superficie = suma_superficie / len(paises)
389
390             print("\n" + "=" * 30)
391             print(f"\n|El promedio de la superficie mundial es de: {round(prom_superficie,2)}")
392
393     except FileNotFoundError:
394         print(" El archivo no existe. Creando uno nuevo...")
395         inicializar_archivo()

```

- Conteo de países por continente:

```

397 def paises_por_continente():
398     try:
399         with open(RUTA_ARCHIVO, "r", encoding="utf-8") as archivo:
400             lector = csv.DictReader(archivo)
401             paises = list(lector)
402
403             #Diccionario de continentes vacio
404             conteo_continentes = {}
405
406             for pais in paises:
407
408                 #Asignacion de continente
409                 continente = pais["continente"].strip().capitalize()
410
411                 #Verificacion si suma al contador del continente
412                 if continente in conteo_continentes:
413                     conteo_continentes[continente] += 1
414
415                 #Verificacion si ingresa un nuevo continente
416                 else:
417                     conteo_continentes[continente] = 1
418
419             for continente, cantidad in conteo_continentes.items():
420                 print(f"\n|Continente {continente} | Cantidad de países: {cantidad}")
421
422
423     except FileNotFoundError:
424         print(" El archivo no existe. Creando uno nuevo...")
425         inicializar_archivo()

```

Validaciones

- Control de errores en el formato del CSV.

El control de errores para el formato de csv lo controlamos anteponiendo la inicialización del archivo csv, ya que si no existe, lo crea.

- Manejo de filtros inválidos o búsquedas sin resultados.

Para manejar los posibles errores del programa utilizamos la funcionalidad de try-except:

Si el archivo existe, se abre y se muestra el mensaje, si no, se lanza una excepción **FileNotFoundException**, es decir, el programa no se rompe, muestra un mensaje de error .

Mientras que en el menú principal si la opción es correcta se ejecuta y si no lanza excepción **ValueError**,es decir, el programa no se rompe, muestra un mensaje de error .

- Mensajes claros de éxito, error o advertencia.

Para una mejor interpretación de nuestro código hicimos una diferenciación en cada uno de los mensajes de error para entender qué error se comete específicamente.

Programa en funcionamiento

- Menú de opciones

```
===== Menú de opciones: =====

|1. Mostrar todos los países:
|2. Buscar país (por nombre):
|3. Filtrar países:
|4. Ordenar países:
|5. Mostrar estadísticas:
|6. Agregar país:
|7. Eliminar país:
|8. Salir

- Seleccione una opción: |
```

- Muestra de países:

| Nombre | Población | Superficie | Continente |
|----------------------|------------|------------|-------------|
| Afganistán | 43844111 | 652860.0 | Asia |
| Albania | 2771508 | 27400.0 | Europa |
| Alemania | 83294000 | 357022.0 | Europa |
| Andorra | 82904 | 470.0 | Europa |
| Angola | 39040039 | 1246700.0 | Africa |
| Antigua y Barbuda | 94209 | 440.0 | América |
| Arabia Saudita | 37057000 | 2149690.0 | Asia |
| Argelia | 47435312 | 2381740.0 | Africa |
| Argentina | 45851378 | 2736690.0 | América |
| Armenia | 2952365 | 28470.0 | Asia |
| Australia | 26974026 | 7682300.0 | Oceania |
| Austria | 9113574 | 82499.0 | Europa |
| Azerbaiyán | 10397713 | 82658.0 | Asia |
| Bahamas | 403033 | 10010.0 | América |
| Bangladés | 175686899 | 130170.0 | Asia |
| Barbados | 282623 | 430.0 | América |
| Baréin | 1643332 | 760.0 | Asia |
| Belice | 422924 | 22810.0 | América |
| Benín | 14814460 | 112760.0 | Africa |
| Bielorrusia | 8997663 | 202910.0 | Europa |
| Birmania | 55800000 | 676578.0 | Asia |
| Bolivia | 12581843 | 1083300.0 | América |
| Bosnia y Herzegovina | 3140095 | 51197.0 | Europa |
| Botswana | 2672000 | 581730.0 | Africa |
| Brasil | 217403611 | 8515767.0 | América |
| Brunéi | 4660000 | 5765.0 | Asia |
| Bulgaria | 6820000 | 110879.0 | Europa |
| Burkina Faso | 22987000 | 272967.0 | Africa |
| Burundi | 12900000 | 27834.0 | Africa |
| Bután | 796682 | 38117.0 | Asia |
| Bélgica | 11758663 | 30280.0 | Europa |
| Cabo Verde | 561981 | 4633.0 | Africa |
| Combuya | 17900000 | 181035.0 | Asia |
| Camerún | 28900000 | 475442.0 | Africa |
| Canadá | 39547000 | 9984670.0 | América |
| Catar | 2930000 | 11571.0 | Asia |
| Chad | 19000000 | 1284000.0 | Africa |
| Chile | 19700000 | 756102.0 | América |
| China | 1412000000 | 9596961.0 | Asia |
| Chipre | 1250000 | 9251.0 | Europa/Asia |

- Búsqueda de países:

```
=====
- Seleccione una opción: 2
Ingrese el nombre del país que desea buscar: argentina
=====
| País Encontrado:
| Nombre: Argentina | Poblacion: 45851378 | Superficie: 2736690 | Continente: América
=====
```

- Filtrado de países:

```
=====
- Seleccione una opción: 3
=====
|Tipos de filtro:
 ①. Filtrar por continente
 ②. Filtrar por rango de población
 ③. Filtrar por rango de superficie

-Seleccione una opción de filtro: ■
=====
```

❖ Continente

| Seleccione una opción de filtro: 1 | | | |
|--|------------|------------|------------|
| ===== | | | |
| Ingrese el continente con el que desea filtrar: asia | | | |
| Paises del continente de asia: | | | |
| Nombre | Población | Superficie | Continente |
| Afganistán | 43844111 | 652860.0 | Asia |
| Arabia Saudita | 37057000 | 2149690.0 | Asia |
| Armenia | 2952365 | 28470.0 | Asia |
| Azerbaiyán | 10397713 | 82658.0 | Asia |
| Bangladés | 175686899 | 130170.0 | Asia |
| Barcín | 1643332 | 760.0 | Asia |
| Birmánia | 55800000 | 676578.0 | Asia |
| Brunei | 460000 | 5765.0 | Asia |
| Bután | 796682 | 38117.0 | Asia |
| Camboya | 17900000 | 181035.0 | Asia |
| Catar | 2930000 | 11571.0 | Asia |
| China | 1412000000 | 9596961.0 | Asia |
| Corea del Norte | 25900000 | 120538.0 | Asia |
| Corea del Sur | 51700000 | 100210.0 | Asia |
| Emiratos Árabes Unidos | 9400145 | 83600.0 | Asia |
| Filipinas | 116787254 | 342351.0 | Asia |
| India | 1463865525 | 3287263.0 | Asia |
| Indonesia | 285721236 | 1904569.0 | Asia |
| Irak | 44012345 | 438317.0 | Asia |
| Irán | 89172000 | 1648195.0 | Asia |
| Israel | 9601234 | 22072.0 | Asia |
| Japón | 122456000 | 377975.0 | Asia |
| Jordania | 11012345 | 89342.0 | Asia |
| Kazajistán | 19987000 | 2724900.0 | Asia |
| Kirguistán | 7001234 | 199951.0 | Asia |
| Kuwait | 4301234 | 17818.0 | Asia |
| Laos | 7648353 | 236800.0 | Asia |
| Líbano | 5613100 | 10452.0 | Asia |
| Malasia | 34628000 | 330803.0 | Asia |
| Maldivas | 521021 | 298.0 | Asia |
| Mongolia | 3458000 | 1564116.0 | Asia |
| Nepal | 31267000 | 147516.0 | Asia |

❖ Rango de población

| -Seleccione una opción de filtro: 2 | | | |
|--|-----------|------------|------------|
| ===== | | | |
| Ingrese la población mínima: 32400000 | | | |
| Ingrese la población máxima: 546000000 | | | |
| ===== | | | |
| Paises con población entre 32400000 y 546000000 de habitantes: | | | |
| Nombre | Población | Superficie | Continente |
| Afganistán | 43844111 | 652860.0 | Asia |
| Alemania | 83294000 | 357022.0 | Europa |
| Angola | 39040039 | 1246700.0 | Africa |
| Arabia Saudita | 37057000 | 2149690.0 | Asia |
| Argelia | 47435312 | 2381740.0 | Africa |
| Argentina | 45851378 | 2736690.0 | América |
| Bangladés | 175686899 | 130170.0 | Asia |
| Birmánia | 55800000 | 676578.0 | Asia |
| Brasil | 217403611 | 8515767.0 | América |
| Canadá | 39547000 | 9984670.0 | América |
| Colombia | 53200000 | 1141748.0 | América |
| Corea del Sur | 51700000 | 100210.0 | Asia |
| Egipto | 118366000 | 1002450.0 | África |
| España | 48032665 | 505990.0 | Europa |
| Estados Unidos | 347275807 | 9833517.0 | América |
| Etiopía | 135472954 | 1104300.0 | Africa |
| Filipinas | 116787254 | 342351.0 | Asia |
| Francia | 68304000 | 551695.0 | Europa |
| Ghana | 34210562 | 238533.0 | Africa |
| Indonesia | 285721236 | 1904569.0 | Asia |
| Irak | 44012345 | 438317.0 | Asia |
| Irán | 89172000 | 1648195.0 | Asia |
| Italia | 58871000 | 301340.0 | Europa |
| Japón | 122456000 | 377975.0 | Asia |
| Kenia | 58036700 | 580367.0 | Africa |
| Malasia | 34628000 | 330803.0 | Asia |
| Marruecos | 37840000 | 446550.0 | Africa |
| Mozambique | 33742000 | 801596.0 | Africa |
| Méjico | 132328035 | 1964375.0 | América |
| Nigeria | 229152217 | 923768.0 | Africa |
| Pakistán | 255219554 | 770880.0 | Asia |

❖ Rango de superficie

- Seleccione una opción de filtro: 3

=====

Ingrese la superficie mínima: 3500
 Ingrese la superficie máxima: 450000

|Paises con superficie entre 3500 y 450000

| Nombre | Población | Superficie | Continente |
|------------------------|-----------|------------|-------------|
| Albania | 2771508 | 27400.0 | Europa |
| Alemania | 83294000 | 357022.0 | Europa |
| Armenia | 2952365 | 28470.0 | Asia |
| Austria | 9113574 | 82409.0 | Europa |
| Azerbaiyán | 10397713 | 82658.0 | Asia |
| Bahamas | 403033 | 10010.0 | América |
| Bangladés | 175686899 | 130170.0 | Asia |
| Belice | 422924 | 22810.0 | América |
| Benín | 14814460 | 112760.0 | Africa |
| Bielorrusia | 8997603 | 202910.0 | Europa |
| Bosnia y Herzegovina | 3140095 | 51197.0 | Europa |
| Burundi | 4600000 | 5765.0 | Asia |
| Bulgaria | 6820000 | 110879.0 | Europa |
| Burkina Faso | 22987000 | 272967.0 | Africa |
| Burundi | 12900000 | 27834.0 | Africa |
| Bután | 796682 | 38117.0 | Asia |
| Bélgica | 11758603 | 30280.0 | Europa |
| Cabo Verde | 561901 | 4033.0 | Africa |
| Camboya | 17900000 | 181035.0 | Asia |
| Catar | 2930000 | 11571.0 | Asia |
| Chipre | 1250000 | 9251.0 | Europa/Asia |
| Corea del Norte | 25900000 | 120538.0 | Asia |
| Corea del Sur | 51700000 | 180210.0 | Asia |
| Costa de Marfil | 29000000 | 322463.0 | Africa |
| Costa Rica | 5200000 | 51100.0 | América |
| Croacia | 3900000 | 56594.0 | Europa |
| Cuba | 11300000 | 109884.0 | América |
| Dinamarca | 5934465 | 43094.0 | Europa |
| Ecuador | 18001800 | 276841.0 | América |
| El Salvador | 6501894 | 21041.0 | América |
| Emiratos Árabes Unidos | 9400145 | 83600.0 | Asia |
| Eritrea | 3500100 | 117600.0 | Africa |

- Ordenamiento de países según opción seleccionada:

- Seleccione una opción: 4

=====

- Tipos de ordenamiento:

[1. Por nombre
 [2. Por población (ascendente o descendente)
 [3. Por superficie (ascendente o descendente)]

- Ingrese una opción de ordenamiento: []

❖ Nombre

- Ingrese una opción de ordenamiento: 1

=====

=====

Paises ordenado correctamente.

=====

| Nombre | Población | Superficie | Continente |
|----------------------|------------|------------|-------------|
| Afganistán | 43844111 | 652860.0 | Asia |
| Albania | 2771508 | 27400.0 | Europa |
| Alemania | 83294000 | 357022.0 | Europa |
| Andorra | 82904 | 470.0 | Europa |
| Angola | 39040039 | 1246700.0 | Africa |
| Antigua y Barbuda | 94209 | 440.0 | América |
| Arabia Saudita | 32057000 | 2149690.0 | Asia |
| Argelia | 4743512 | 2381740.0 | Africa |
| Argentina | 45851378 | 2736690.0 | América |
| Armenia | 2952365 | 28470.0 | Asia |
| Australia | 26974026 | 7682300.0 | Oceania |
| Austria | 9113574 | 82409.0 | Europa |
| Azerbaiyán | 10397713 | 82658.0 | Asia |
| Bahamas | 403033 | 10010.0 | América |
| Bangladés | 175686899 | 130170.0 | Asia |
| Barbados | 282623 | 430.0 | América |
| Barcín | 1643332 | 760.0 | Asia |
| Belice | 422924 | 22810.0 | América |
| Benín | 14814460 | 112760.0 | Africa |
| Bielorrusia | 8997603 | 202910.0 | Europa |
| Esmania | 55800000 | 676578.0 | Asia |
| Bolivia | 12581843 | 1093300.0 | América |
| Bosnia y Herzegovina | 3140095 | 51197.0 | Europa |
| Botswana | 2672000 | 581730.0 | Africa |
| Brasil | 217403611 | 8515767.0 | América |
| Brunei | 460000 | 5765.0 | Asia |
| Bulgaria | 6820000 | 110879.0 | Europa |
| Burkina Faso | 22987000 | 272967.0 | Africa |
| Burundi | 12900000 | 27834.0 | Africa |
| Bután | 796682 | 38117.0 | Asia |
| Bélgica | 11758603 | 30280.0 | Europa |
| Cabo Verde | 561901 | 4033.0 | Africa |
| Cambaya | 17900000 | 181035.0 | Asia |
| Camerún | 28900000 | 475442.0 | Africa |
| Canadá | 39547000 | 9984670.0 | América |
| Catar | 2930000 | 11571.0 | Asia |
| Chad | 19000000 | 1284000.0 | Africa |
| Chile | 19700000 | 756102.0 | América |
| China | 1412000000 | 9596961.0 | Asia |
| Cipre | 1250000 | 9251.0 | Europa/Asia |
| Colombia | 53200000 | 1141748.0 | América |
| Comoras | 987000 | 2235.0 | Africa |

❖ Población

```
-----
- Tipos de ordenamiento:
|1. Por nombre
|2. Por población (ascendente o descendente)
|3. Por superficie (ascendente o descendente)

- Ingrese una opción de ordenamiento: 2

=====
|Ascendente (a)
|Descendente (d)

- Elija una opción: |
```

➤ Ascendente

| Nombre | Población | Superficie | Continente |
|-------------------|-----------|------------|-------------|
| Nauru | 10704 | 21.0 | Oceania |
| Palau | 18992 | 459.0 | Oceania |
| Mónaco | 3942 | 2.02 | Europa |
| Liechtenstein | 39244 | 160.0 | Europa |
| Islas Marshall | 42100 | 181.0 | Oceania |
| Dominica | 72669 | 751.0 | América |
| Madagascar | 78902 | 3212321.0 | Africa |
| Andorra | 82904 | 470.0 | Europa |
| Antigua y Barbuda | 94299 | 440.0 | América |
| Granada | 113949 | 344.0 | América |
| Micronesia | 116254 | 702.0 | Oceania |
| Kiribati | 130456 | 811.0 | Oceania |
| Curaçao | 155000 | 444.0 | América |
| Barbados | 282623 | 430.0 | América |
| Islandia | 380123 | 103000.0 | Europa |
| Bahamas | 403033 | 10010.0 | América |
| Belice | 422924 | 22810.0 | América |
| Brunei | 460000 | 5765.0 | Asia |
| Maldivas | 521021 | 298.0 | Asia |
| Malta | 535064 | 316.0 | Europa |
| Cabo Verde | 561901 | 4033.0 | Africa |
| Montenegro | 619211 | 13812.0 | Europa |
| Luxemburgo | 654768 | 2586.0 | Europa |
| Islas Salomón | 740456 | 28896.0 | Oceania |
| Bután | 796682 | 38117.0 | Asia |
| Guyana | 804567 | 214969.0 | América |
| Fiyi | 940446 | 18274.0 | Oceania |
| Comoras | 987000 | 2235.0 | Africa |
| Esuatini | 1204854 | 17364.0 | Africa |
| Chipre | 1250000 | 9251.0 | Europa/Asia |
| Mauricio | 1300944 | 2040.0 | Africa |
| Estonia | 1322769 | 45227.0 | Europa |
| Bahrain | 1643132 | 760.0 | Asia |
| Letonia | 1866843 | 64589.0 | Europa |
| Eslovenia | 2108977 | 20273.0 | Europa |
| Guinea-Bisáu | 2200456 | 36125.0 | Africa |
| Gabón | 23017264 | 267668.0 | Africa |
| Namibia | 2540900 | 825615.0 | Africa |
| Moldavia | 2600000 | 33843.0 | Europa |
| Lituania | 26556000 | 65300.0 | Europa |

➤ Descendente

| Nombre | Población | Superficie | Continente |
|-----------------|------------|------------|------------|
| India | 1463865525 | 3287263.0 | Asia |
| China | 1412000000 | 9596961.0 | Asia |
| Estados Unidos | 347175807 | 9833517.0 | América |
| Indonesia | 285721236 | 1904569.0 | Asia |
| Pakistán | 255219554 | 770880.0 | Asia |
| Nigeria | 229152217 | 923768.0 | Africa |
| Brasil | 217403611 | 8515767.0 | América |
| Bangladés | 175686899 | 130170.0 | Asia |
| Etiopía | 135472954 | 1104300.0 | Africa |
| México | 132328035 | 1964375.0 | América |
| Japón | 122456000 | 377975.0 | Asia |
| Egipto | 118366000 | 1002450.0 | Africa |
| Filipinas | 116787254 | 342353.0 | Asia |
| RD Congo | 102262000 | 2344858.0 | Africa |
| Irán | 89172000 | 1648195.0 | Asia |
| Alemania | 83294000 | 357022.0 | Europa |
| Francia | 68304000 | 551695.0 | Europa |
| Italia | 58871000 | 301340.0 | Europa |
| Kenia | 58036700 | 580367.0 | Africa |
| Birmania | 55800000 | 676578.0 | Asia |
| Colombia | 53200000 | 1141748.0 | América |
| Corea del Sur | 51700000 | 100210.0 | Asia |
| España | 48932665 | 505990.0 | Europa |
| Argelia | 47435312 | 2381740.0 | Africa |
| Argentina | 45851378 | 2736690.0 | América |
| Iraík | 44012345 | 438317.0 | Asia |
| Afganistán | 43844111 | 652860.0 | Asia |
| Canadá | 39547000 | 9984670.0 | América |
| Angola | 39040039 | 1246700.0 | Africa |
| Polonia | 37958000 | 312696.0 | Europa |
| Marruecos | 37840000 | 446550.0 | Africa |
| Arabia Saudita | 37057000 | 2149690.0 | Asia |
| Malasia | 34628000 | 330803.0 | Asia |
| Perú | 34523000 | 1285210.0 | América |
| Ghana | 34210562 | 238533.0 | Africa |
| Mozambique | 33742000 | 801590.0 | Africa |
| Nepal | 31267000 | 147516.0 | Asia |
| Madagascar | 30969500 | 587041.0 | Africa |
| Costa de Marfil | 29000000 | 322463.0 | Africa |
| Camerún | 28900000 | 475442.0 | Africa |
| Níger | 26987000 | 1267000.0 | Africa |

❖ Superficie

```

- Tipos de ordenamiento:
|1. Por nombre
|2. Por población (ascendente o descendente)
|3. Por superficie (ascendente o descendente)

- Ingrese una opción de ordenamiento: 3

=====
|Ascendente (a)
|Descendente (d)

- Elija una opción: 1
  
```

➤ Ascendente

| Nombre | Población | Superficie | Continente |
|-------------------|-----------|------------|-------------|
| Mónaco | 39242 | 2,02 | Europa |
| Nauru | 10704 | 21,0 | Oceania |
| Liechtenstein | 39244 | 160,0 | Europa |
| Islas Marshall | 42100 | 181,0 | Oceania |
| Maldivas | 521021 | 298,0 | Asia |
| Malta | 535064 | 316,0 | Europa |
| Granada | 113949 | 344,0 | América |
| Bahamas | 282623 | 430,0 | América |
| Antigua y Barbuda | 94209 | 440,0 | América |
| Curaçao | 155900 | 444,0 | América |
| Palau | 18992 | 459,0 | Oceania |
| Andorra | 82904 | 470,0 | Europa |
| Micronesia | 116254 | 702,0 | Oceania |
| Dominica | 72695 | 751,0 | América |
| Bacán | 164332 | 760,0 | Asia |
| Kiribati | 130456 | 811,0 | Oceania |
| Mauricio | 1300944 | 2040,0 | Africa |
| Comoras | 987000 | 2235,0 | Africa |
| Luxemburgo | 654768 | 2586,0 | Europa |
| Cabo Verde | 561981 | 4033,0 | Africa |
| Brunéi | 460090 | 5765,0 | Asia |
| Chipre | 1250000 | 9251,0 | Europa/Asia |
| Bahamas | 403033 | 10010,0 | América |
| Líbano | 5613108 | 10452,0 | Asia |
| Jamaica | 2804567 | 10991,0 | América |
| Gambia | 2706253 | 11295,0 | Africa |
| Catar | 2930000 | 11571,0 | Asia |
| Qatar | 2930000 | 11571,0 | Asia |
| Montenegro | 619211 | 13812,0 | Europa |
| Esuatini | 1204954 | 17364,0 | Africa |
| Kuwait | 4301234 | 17818,0 | Asia |
| Fiyi | 940446 | 18274,0 | Oceania |
| Eslavonia | 2108977 | 20273,0 | Europa |
| El Salvador | 6501894 | 21041,0 | América |
| Israel | 9601234 | 22072,0 | Asia |
| Belize | 422924 | 22810,0 | América |
| Ruanda | 13900000 | 26338,0 | Africa |
| Albania | 2771508 | 27400,0 | Europa |
| Haití | 11704567 | 27750,0 | América |
| Burundi | 12900000 | 27834,0 | Africa |
| Armenia | 2952365 | 28470,0 | Asia |
| Islas Salomón | 740456 | 28896,0 | Oceania |

➤ Descendente

| Nombre | Población | Superficie | Continente |
|--------------------|------------|------------|------------|
| Canadá | 39547900 | 9984679,0 | Ameríca |
| Estados Unidos | 347275807 | 9833517,0 | Ameríca |
| China | 1412000000 | 9596961,0 | Asia |
| Brasil | 217403611 | 8515767,0 | Ameríca |
| Australia | 26974026 | 7682300,0 | Oceania |
| India | 1463865525 | 3287263,0 | Asia |
| Madagascar | 78902 | 312321,0 | Africa |
| Argentina | 45851378 | 2736690,0 | America |
| Kazajistán | 19987000 | 2725900,0 | Asia |
| Angelia | 47435312 | 2381740,0 | Africa |
| RD Congo | 102262000 | 2344858,0 | Africa |
| Arabia Saudita | 37057000 | 2149690,0 | Asia |
| México | 132328035 | 1964375,0 | America |
| Indonesia | 285721236 | 1904569,0 | Asia |
| Cibia | 6931000 | 1759540,0 | Africa |
| Irán | 89172000 | 1648195,0 | Asia |
| Mongolia | 34580000 | 1564116,0 | Asia |
| Perú | 34523000 | 1285216,0 | America |
| Chad | 19000000 | 1284000,0 | Africa |
| Niger | 26987000 | 1267000,0 | Africa |
| Angola | 39040039 | 1246700,0 | Africa |
| Mali | 22979000 | 1240192,0 | Africa |
| Colombia | 53260000 | 1141748,0 | America |
| Etiopía | 135472954 | 1104380,0 | Africa |
| Bolivia | 12581843 | 1083380,0 | America |
| Mauritania | 4925000 | 1030700,0 | Africa |
| Egipto | 118366000 | 1002450,0 | Africa |
| Nigeria | 229152217 | 923768,0 | Africa |
| Namibia | 25409000 | 825615,0 | Africa |
| Mozambique | 33742000 | 801590,0 | Africa |
| Pakistán | 255219554 | 770880,0 | Asia |
| Chile | 19700000 | 756102,0 | America |
| Birmánia | 55800000 | 676578,0 | Asia |
| Afganistán | 43844111 | 652860,0 | Asia |
| Madagascar | 30969500 | 587041,0 | Africa |
| Botswana | 2672000 | 581730,0 | Africa |
| Kenia | 58036700 | 580367,0 | Africa |
| Francia | 68304000 | 551695,0 | Europa |
| España | 48032651 | 505990,0 | Europa |
| Camerún | 28900000 | 475442,0 | Africa |
| Papúa Nueva Guinea | 10180000 | 462840,0 | Oceania |
| Marruecos | 37840000 | 446550,0 | Africa |

- Muestra de estadísticas según opción seleccionada:

```
- Seleccione una opción: 5
=====
- Tipos de estadísticas:
|1. País con menor y mayor población
|2. Promedio de población
|3. Promedio de superficie
|4. Cantidad de países por continente
- Ingrese una opción de estadística: |
```

❖ País por población

```
- Ingrese una opción de estadística: 1
=====
|País con mayor población: India |Poblacion: 1463865525
|País con menor población: Nauru |Poblacion: 10704
=====
```

❖ Promedio población

```
- Ingrese una opción de estadística: 2
=====
|El promedio de población mundial es de: 47819154
=====
```

❖ Promedio Superficie

```
- Ingrese una opción de estadística: 3
=====
|El promedio de la superficie mundial es de: 716330.0
=====
```

❖ Cantidad de países por continente

```
- Ingrese una opción de estadística: 4
=====
|Continente Asia | Cantidad de paises: 35
|Continente Europa | Cantidad de paises: 35
|Continente África | Cantidad de paises: 35
|Continente América | Cantidad de paises: 29
|Continente Oceanía | Cantidad de paises: 10
|Continente Europa/asia | Cantidad de paises: 2
|Continente África | Cantidad de paises: 1
=====
```

● Agregar país:

```
=====
- Seleccione una opción: 6
-Ingrese el nombre del país a ingresar: rusia
-Ingrese la población del país: 45600000
-Ingrese la superficie del país: 32456
-Ingrese el continente del país: europa
|País agregado con éxito.
=====
```

| | | | |
|---------|-----------|----------|--------|
| Ruanda | 13900000 | 26338.0 | Africa |
| Rumania | 18987000 | 238397.0 | Europa |
| Rusia | 456000000 | 32456.0 | |

● Eliminar país:

```
10-2021
=====
- Seleccione una opción: 7
-Ingrese el país que desea eliminar: rusia
=====
|País eliminado con éxito -
=====
```

| | | | |
|---------|----------|----------|--------|
| Ruanda | 13900000 | 26338.0 | Africa |
| Rumania | 18987000 | 238397.0 | Europa |

● Salir:

```
⑧
=====
- Seleccione una opción: 8
- Saliendo del programa..... Hasta luego!
=====
```

Conclusión

El desarrollo de este trabajo nos permitió consolidar de manera práctica los conceptos fundamentales de la programación en Python, aplicando estructuras de datos, funciones, condicionales, ordenamientos y estadísticas sobre un conjunto de datos real. A través de la gestión de información sobre países, logramos integrar múltiples habilidades técnicas en un sistema funcional, modular y escalable.

Uno de los principales aprendizajes fue comprender la importancia de la modularización del código, dividiendo el programa en funciones específicas que cumplen una única responsabilidad. Esta práctica no solo mejora la legibilidad y el mantenimiento del código, sino que también facilita el trabajo colaborativo, permitiendo que cada integrante del grupo se enfoque en una parte del sistema.

El uso de listas y diccionarios fue clave para representar y manipular los datos de manera eficiente. Las listas nos permitieron almacenar colecciones de países, mientras que los diccionarios ofrecieron una forma flexible de acceder a las características de cada país, como nombre, población, superficie y continente. Esta estructura nos permitió implementar filtros y ordenamientos personalizados con facilidad.

La lectura desde un archivo CSV nos introdujo al manejo de archivos externos, una habilidad esencial en el desarrollo de software real. Aprendimos a validar el formato de los datos, controlar errores de lectura y asegurar que el sistema funcione correctamente incluso ante entradas inesperadas.

Además, la implementación de estadísticas básicas como promedios, máximos, mínimos y conteos por categoría nos permitió extraer información relevante del dataset, demostrando cómo la programación puede ser una herramienta para el análisis de datos.

Desde el punto de vista organizacional, este trabajo fomentó la colaboración en equipo, la planificación de tareas y la comunicación constante. La división de roles fue fundamental para avanzar de manera ordenada, y cada integrante aportó desde sus fortalezas técnicas y creativas. El uso de GitHub como repositorio colaborativo nos permitió versionar el código, documentar el proyecto y mantener un registro claro de los avances.

Finalmente, la elaboración del video tutorial y del informe teórico nos ayudó a reflexionar sobre el proceso de desarrollo, identificar los desafíos enfrentados y valorar los logros alcanzados. Esta instancia de presentación nos permitió comunicar de forma clara y profesional el funcionamiento del sistema, reforzando nuestras habilidades de documentación y exposición técnica.

En resumen, este trabajo integrador fue una experiencia que nos permitió aplicar la mayoría de los contenidos de Programación 1 en un contexto real, desarrollar un sistema completo y fortalecer tanto nuestras habilidades técnicas como nuestras habilidades colaborativas.

Referencias

- Apuntes de la cátedra
 - ❖ <https://colab.research.google.com/drive/1pWNQ-rHTRtU9URvxL0CBYjiHXtvYVEr8?usp=sharing#scrollTo=LA8b7NR73osNr8>
 - ❖ https://colab.research.google.com/drive/1iWUJaLgC9WrxLXjxDRynd_SK6v_snxQf?usp=sharing#scrollTo=Yq3pHHVwF7R
 - ❖ <https://campus.frm.utn.edu.ar/mod/resource/view.php?id=157158>
- Ordenamientos <https://github.com/gbaudino/MetodosDeOrdenamiento>
- Estadísticas <https://docs.python.org/es/3.8/library/statistics.html>