

# Trabajo Práctico Integrador

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

**Materia:** Programación I

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

**Alumnos / Comisión:** Fernando Picco (C10)

**Fecha de entrega:** 02 de diciembre de 2025.

Universidad Tecnológica Nacional



## 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

### Estructuras condicionales

#### 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

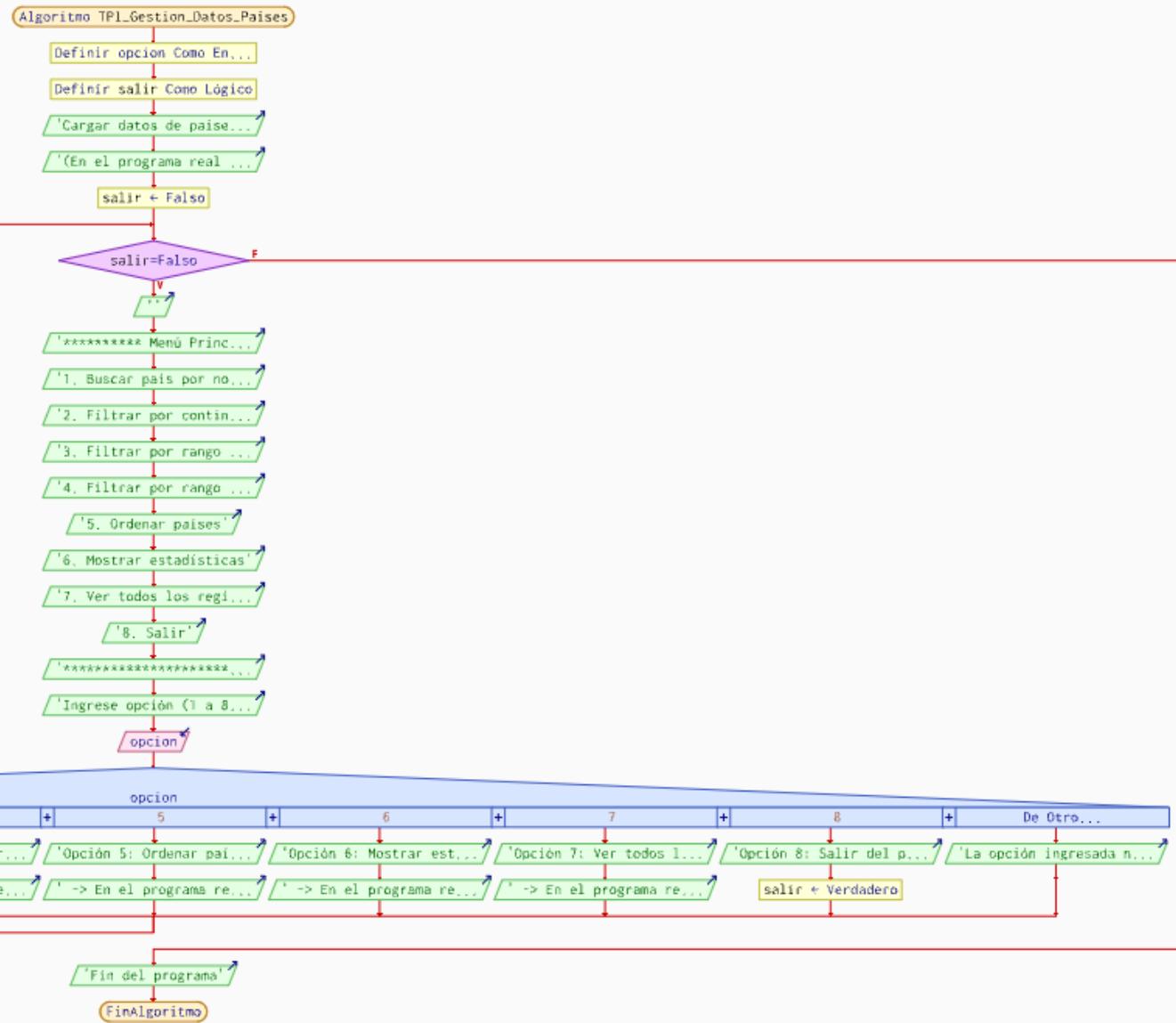
def obtener_datos_paises():
def filtrar_por_continente():
def filtrar_por_rango_poblacion():
def filtrar_por_rango_superficie():
def quitar_acentos(texto):
def normalizar(texto):
def mostrar_pais(pais):
```

```
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



## Flujo de operaciones

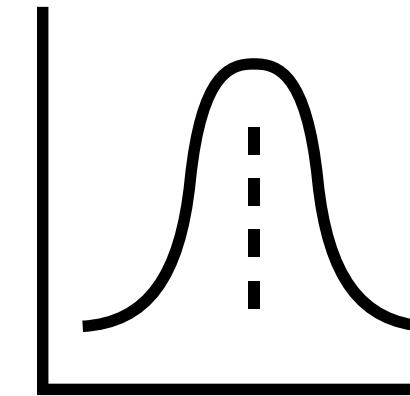


## Arquitectura del código: Organización

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



A	B	C	D	E	F	G	H
1	nombre,poblacion,superficie,continente						
2	Argentina,45376763,2780400,América						
3	Japón,125900000,377975,Asia						
4	Brasil,21393437,8515767,América						
5	Alemania,83149300,357022,Europa						
6	Australia,25925600,7692024,Oceania						
7	Egipto,104259000,1002450,África						
8	Canadá,39008005,9884670,América						
9	India,1333409003,3287263,Asia						
10	Sudáfrica,59308390,1219090,África						
11	Francia,67413000,551695,Europa						
12							



## Arquitectura del código. Ejemplo

### ➤ Módulo de persistencia: *obtener\_datos\_paises()*

```
33 def obtener_datos_paises():
34     paises = []
35     # with open(nombre_archivo, newline="", encoding="utf-8") as archivo:
36     #     lector = csv.DictReader(archivo) # La función DictReader(), devuelve un iterador devolviendo un
37     #                                     # diccionario clave/valor a partir de la lectura de las líneas
38     #                                     # del archivo csv.
39     #     for fila in lector:
40     #         paises.append({"nombre": fila["nombre"], "poblacion": float(fila["poblacion"]), "superficie": float(fila["superficie"]),
41     #                                     "continente": fila["continente"]})
42     #return paises
43     try:
44         # Intento abrir el archivo csv
45         with open(nombre_archivo, newline="", encoding="utf-8") as archivo:
46             lector = csv.DictReader(archivo)
47
48             for numero_fila, fila in enumerate(lector, start=2):
49                 # start=2 porque la línea 1 es el encabezado
50                 # Validar que existan todas las columnas necesarias
51                 columnas_obligatorias = ("nombre", "poblacion", "superficie", "continente")
52                 if not all(col in fila and fila[col] != "" for col in columnas_obligatorias):
53                     print(f"[AVISO] Línea {numero_fila} ignorada: columnas faltantes o vacías -> {fila}")
54                     continue
55
56                 # Validar que población y superficie sean numéricas
57                 try:
58                     poblacion = float(fila["poblacion"])
59                     superficie = float(fila["superficie"])
60                 except ValueError:
61                     print(f"[AVISO] Línea {numero_fila} ignorada: valores numéricos inválidos -> {fila}")
62                     continue
63
64                 # Si todo está bien, agrego el país a la lista
65                 paises.append({
66                     "nombre": fila["nombre"],
67                     "poblacion": poblacion,
68                     "superficie": superficie,
69                     "continente": fila["continente"]
70                 })
71             except FileNotFoundError:
72                 print(f"[ERROR] No se encontró el archivo '{nombre_archivo}'")
73             except UnicodeDecodeError:
74                 print(f"[ERROR] Problema de codificación al leer '{nombre_archivo}'. ¿Está guardado en UTF-8?")
75
76 return paises
```

## Funcionalidades principales

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

```
Ingrese opción: 1

***Búsqueda de país por nombre (coincidencia parcial o exacta)***

Ingrese el nombre del país o parte del nombre: argentina
Se encontraron 1 país(es):
Argentina | Población: 45376763 | Superficie: 2780400 | Continente: América

*** Menú Principal ***
1. Buscar país por nombre (coincidencia parcial o exacta)
2. Filtrar por continente
3. Filtrar por rango de población
4. Filtrar por rango de superficie
5. Ordenar países
6. Mostrar estadísticas
7. Ver todos los registros
8. Salir

Ingrese opción: █
```

## Demostración del programa

The screenshot shows a Windows desktop environment. On the left, there is a dark-themed file explorer window titled 'EXPLORADOR' showing various files and folders related to a 'Trabajo Integrador Obligatorio'. In the center, there is a code editor window titled 'tpi\_gestion\_datos\_paises.py'. The code is a Python script for a menu-based application. At the bottom, a terminal window is open, showing the execution of the script and its menu options. The terminal output includes:

```
Integrador Obligatorio & C:\Users\111\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/111/OneDrive/UTN/3. TUPaD/2. Ago-Dic 2025 (cuatrimestre 1)/3. Materias/1. Programación I/Trabajo Integrador Obligatorio/tpi_gestion_datos_paises.py"
*** Menú Principal ***
1. Buscar país por nombre (coincidencia parcial o exacta)
2. Filtrar por continente
3. Filtrar por rango de población
4. Filtrar por rango de superficie
5. Ordenar países
6. Mostrar estadísticas
7. Ver todos los registros
8. Salir
Ingrrese opción: []
```



## 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.

```
18 |     # Función para quitar acentos
19 | def quitar_acentos(texto):
20 |     texto = texto.replace("á", "a").replace("Á", "a")
21 |     texto = texto.replace("é", "e").replace("É", "e")
22 |     texto = texto.replace("í", "i").replace("Í", "i")
23 |     texto = texto.replace("ó", "o").replace("Ó", "o")
24 |     texto = texto.replace("ú", "u").replace("Ú", "u")
25 |     texto = texto.replace("ñ", "n").replace("Ñ", "n")
26 |
27 |     return texto
28 |
29 | def normalizar(texto):
30 |     # quita acentos y convierte a minúsculas
31 |     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!**