

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

Listas

```
python

países = []

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

Diccionarios

```
python

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

Funciones

```
python

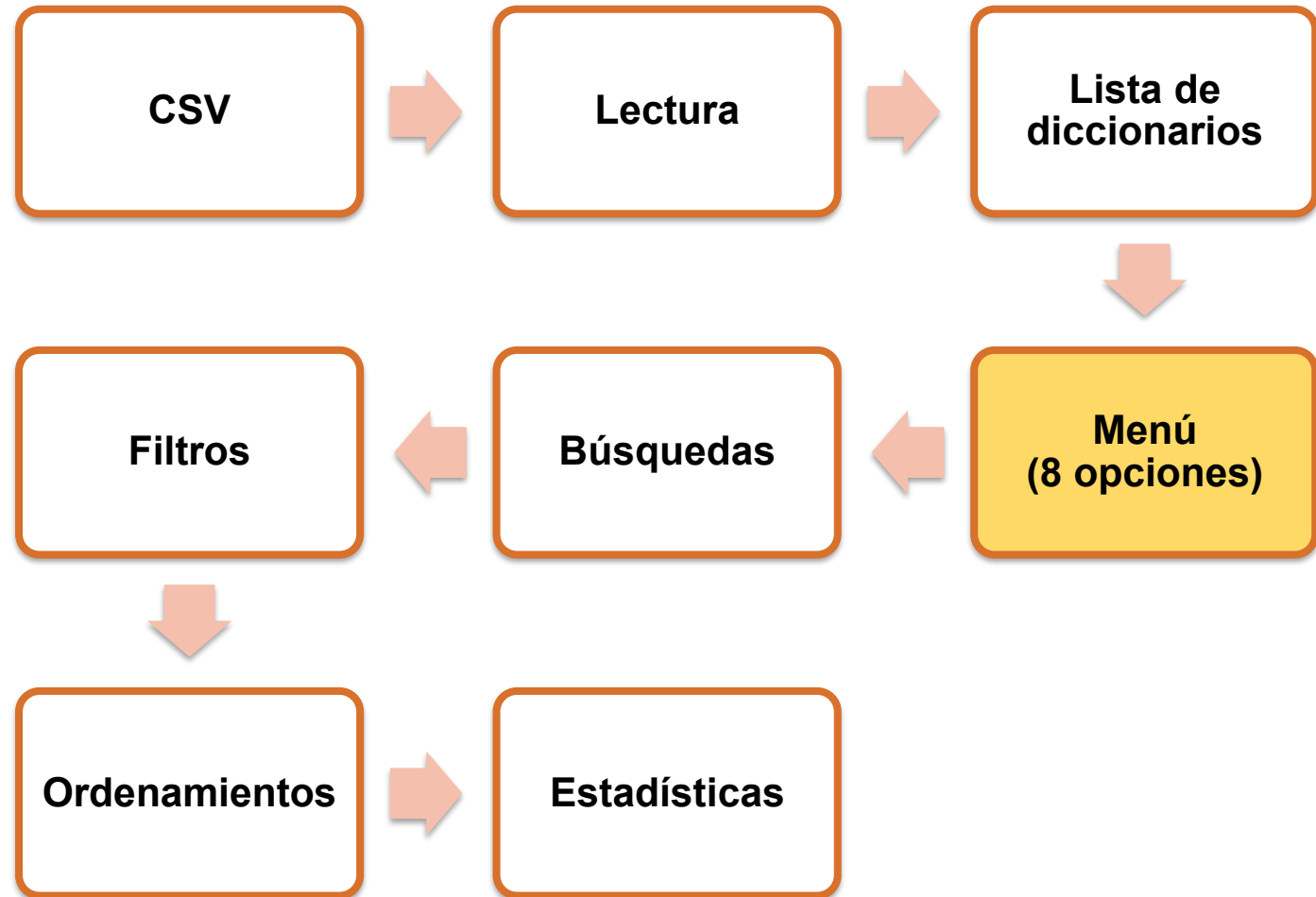
def obtener_datos_países():
def filtrar_por_continente():
def filtrar_por_rango_poblacion():
def filtrar_por_rango_superficie():
def quitar_acentos(texto):
def normalizar(texto):
def mostrar_pais(pais):
```

Estructuras condicionales

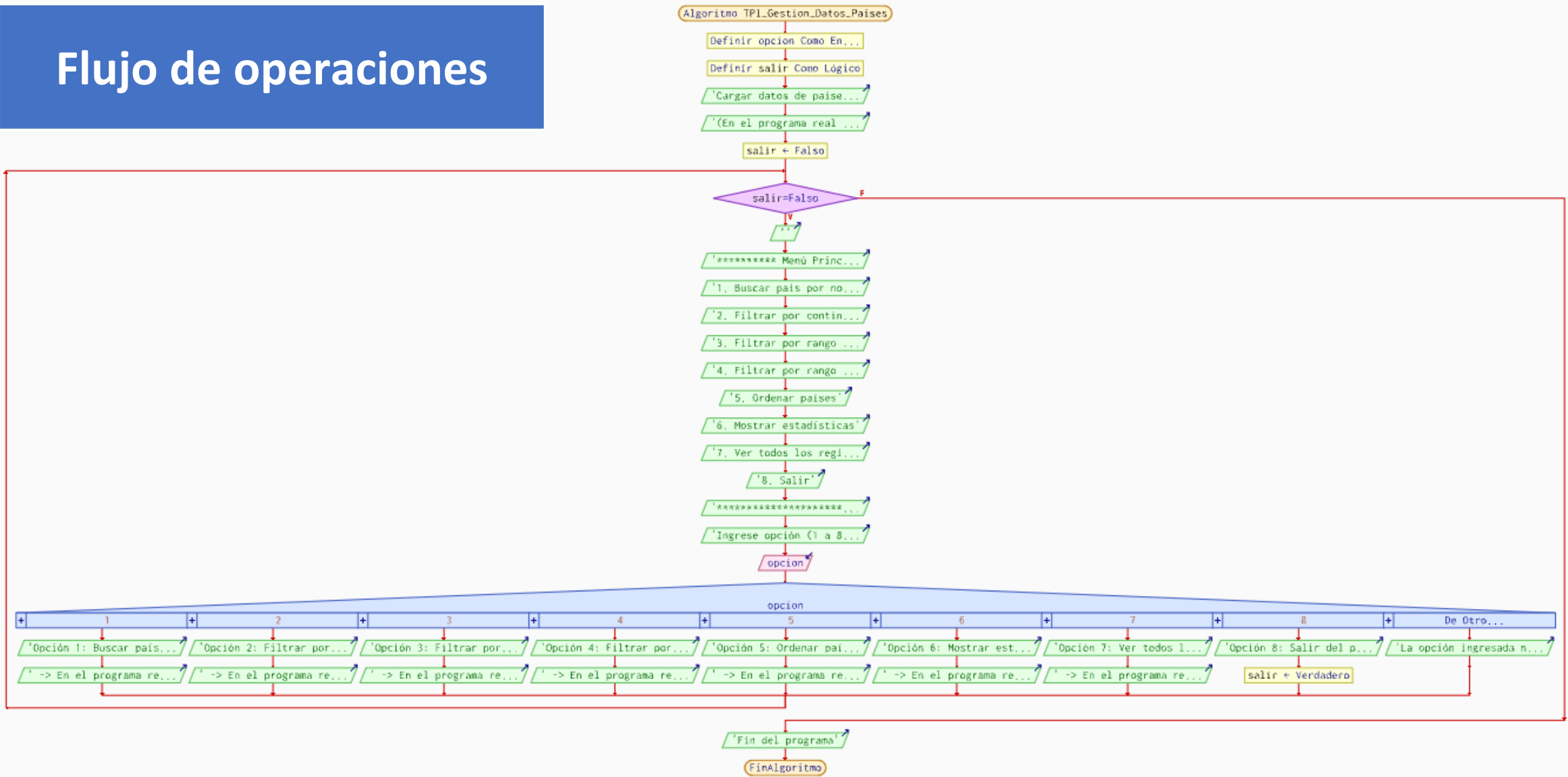
```
opcion = input("Ingrese opción: ").strip()
match opcion:
    case '1':
        buscar_nombre_países()
    case '2':
        filtrar_por_continente()
    case '3':
        filtrar_por_rango_poblacion()
    case '4':
        filtrar_por_rango_superficie()
    case '5':
        ordenar_países()
    case '6':
        mostrar_estadísticas()
    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



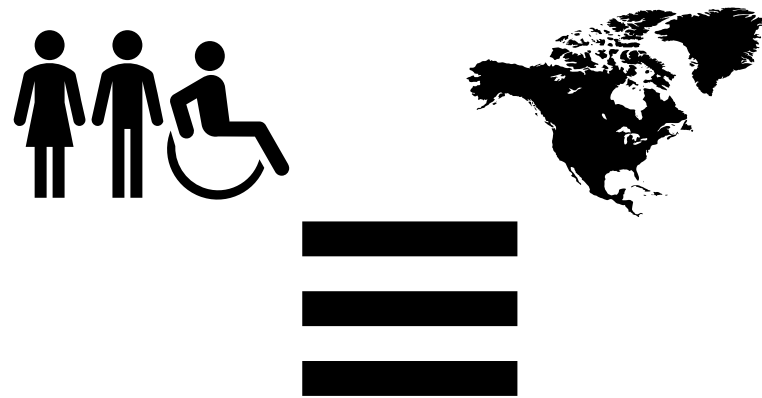
Flujo de operaciones



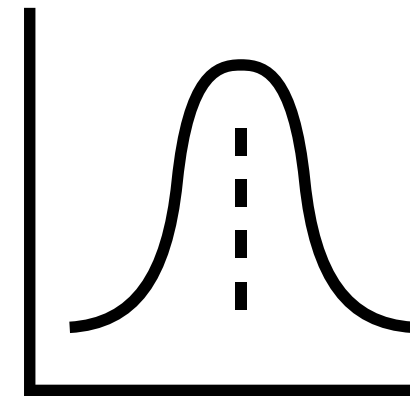
Tecnicatura Universitaria en Programación a Distancia

Arquitectura del código: Organización

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

A screenshot of a Microsoft Excel spreadsheet. The spreadsheet has a header row with the following data: "nombre,poblacion,superficie,continente". The rows below contain data for various countries: Argentina, Japón, Brasil, Alemania, Australia, Egipto, Canadá, India, Sudáfrica, and Francia. The columns are labeled A, B, C, D, E, F, G, H. The spreadsheet is titled "dataset_p..." and is saved in "Este PC".

	A	B	C	D	E	F	G	H
1	nombre,poblacion,superficie,continente							
2	Argentina,45375763,2790400,América							
3	Japón,125900000,377975,Asia							
4	Brasil,213993437,8515767,América							
5	Alemania,83145000,357022,Europa							
6	Australia,25925600,7692024,Oceanía							
7	Egipto,104250000,1002450,África							
8	Canadá,38009005,9984670,América							
9	India,1393409038,3287263,Asia							
10	Sudáfrica,59309690,1219090,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"]), "continente": fila["continente"]})
41     # return paises
42     try:
43         # Intento abrir el archivo csv
44         with open(nombre_archivo, newline="", encoding="utf-8") as archivo:
45             lector = csv.DictReader(archivo)
46
47             for numero_fila, fila in enumerate(lector, start=2):
48                 # start=2 porque la línea 1 es el encabezado
49                 # Validar que existan todas las columnas necesarias
50                 columnas_obligatorias = ("nombre", "poblacion", "superficie", "continente")
51                 if not all(col in fila and fila[col] != "" for col in columnas_obligatorias):
52                     print(f"[AVISO] Línea {numero_fila} ignorada: columnas faltantes o vacías -> {fila}")
53                     continue
54
55                 # Validar que población y superficie sean numéricas
56                 try:
57                     poblacion = float(fila["poblacion"])
58                     superficie = float(fila["superficie"])
59                 except ValueError:
60                     print(f"[AVISO] Línea {numero_fila} ignorada: valores numéricos inválidos -> {fila}")
61                     continue
62
63                 # Si todo está bien, agrego el país a la lista
64                 paises.append({
65                     "nombre": fila["nombre"],
66                     "poblacion": poblacion,
67                     "superficie": superficie,
68                     "continente": fila["continente"]
69                 })
70     except FileNotFoundError:
71         print(f"[ERROR] No se encontró el archivo '{nombre_archivo}'.")
72     except UnicodeDecodeError:
73         print(f"[ERROR] Problema de codificación al leer '{nombre_archivo}'. ¿Está guardado en UTF-8?")
74
75     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: █
```

Tecnicatura Universitaria en Programación a Distancia

Demostración del programa

The screenshot displays a Python IDE with a dark theme. The left sidebar shows a file explorer with a project named 'TRABAJO INTEGRADOR OBLIGATORIO'. The main editor window shows a file named 'tpi_gestion_datos_paises.py' with the following code:

```
390 print()  
391  
392 def mostrar_menu():  
393     # Para mostrar constantemente el menú:  
394     while True:  
395         print("*** Menú Principal ***")  
396         print("1. Buscar país por nombre (coincidencia parcial o exacta)")  
397         print("2. Filtrar por continente")  
398         print("3. Filtrar por rango de población")  
399         print("4. Filtrar por rango de superficie")  
400         print("5. Ordenar países")  
401         print("6. Mostrar estadísticas")  
402         print("7. Ver todos los registros")  
403         print("8. Salir")  
404  
405         opcion = input("Ingrese opción: ").strip()  
406         match opcion:  
407             case '1':
```

The bottom panel shows the 'TERMINAL' output, which displays the program's execution:

```
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  
Ingrese opción: 
```

The status bar at the bottom indicates the file is in UTF-8 encoding, uses CRLF line endings, and is running Python 3.13 (64-bit) with Prettier formatting.



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         return texto
27
28     def normalizar(texto):
29         # quita acentos y convierte a minúsculas
30         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!**