Trabajo Práctico Integrador - Programación I

Profesora Cinthia Rigoni
Integrantes: - *Joaquín Godoy*
- *Ares Martín Ocaña*

Explicación de Conceptos Aplicados

Listas

El uso primario que reciben las listas en este proyecto es el de almacenar temporalmente los datos del archivo de datos. Esto se debe a que no se puede trabajar con un archivo en modo de lectura y escritura al mismo tiempo, por ende se deben utilizar listas para almacenar la lectura del archivo en un dato que Python pueda manipular y luego usar esas listas para reescribir el archivo con la información actualizada.

Diccionarios

Similarmente a las listas; el uso de diccionarios se podría aplicar para la manipulación organizada de los datos del archivo. En este proyecto elegimos no utilizarlos y trabajar siempre con listas, aunque se podrían usar los dos tipos.

Funciones

El uso de funciones fue clave en el desarrollo del producto. Principalmente nos permitió organizar las utilidades del programa en bloques individuales por cada una, lo cual produce un código mucho más organizado. Además, usamos funciones para almacenar código que queríamos usar más de una vez (por ejemplo, para normalizar el texto ingresado por el usuario). Finalmente, dividir todo en funciones permitió compartimentalizar el código en distintos archivos: un archivo principal para mostrar el menú de opciones y ejecutarlas, un archivo que contiene las funciones principales, y un archivo de utilidades para las funciones simples que íbamos reutilizando y que no formaban parte de la lógica principal del programa.

Condicionales

Las estructuras condicionales se utilizaron para realizar validaciones, frenando la ejecución de una determinada secuencia si no se cumplían las condiciones correctas. También fueron clave al trabajar con listas cuando no se quería manipular todos los elementos de ella; por ejemplo: para mostrar solo los países que coincidieran entera o parcialmente con el texto ingresado por el usuario en *buscar_pais()*. Finalmente, también se utilizaron para la creación de menúes interactivos con distintas opciones.

Ordenamientos

Se utilizaron funciones nativas de Python para ordenar listas de países y mostrarlas al usuario, como el uso de *max()* y *min()* para mostrar los países de mayor y menor población en *mostrar_estadisticas()* o *.sort()* para ordenar los países según distintos datos en *ordenar_paises()*.

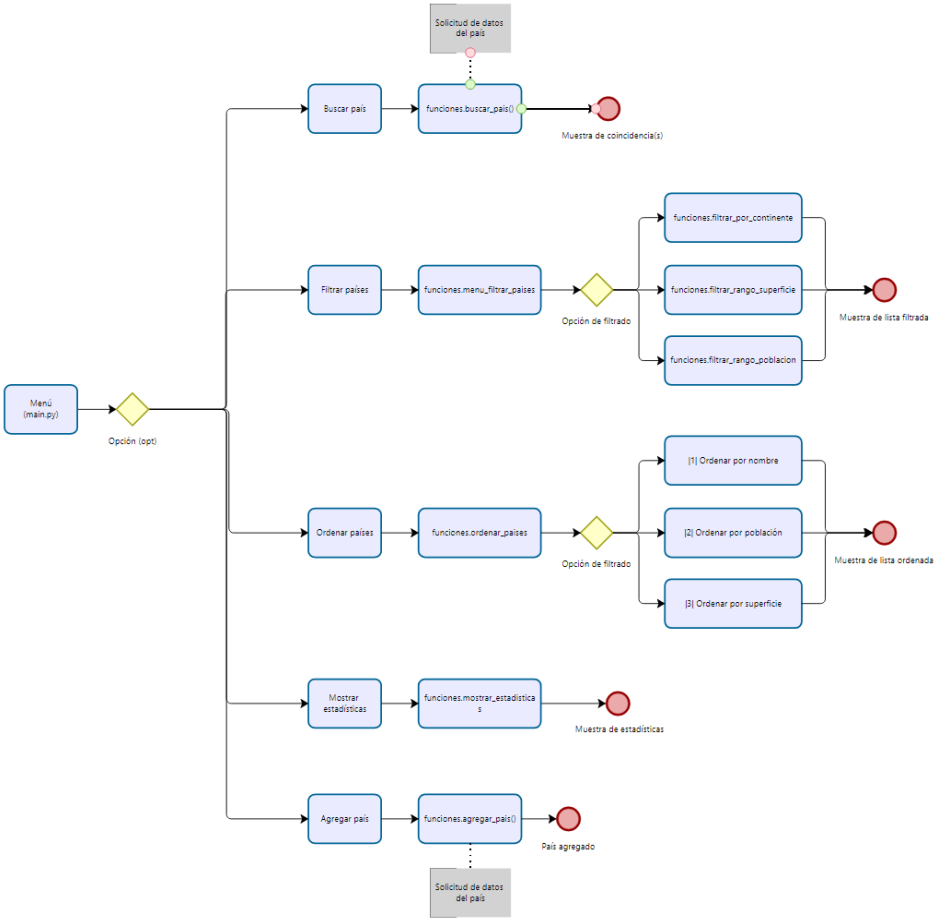
Estadísticas Básicas

Las estadísticas se aplicaron en el proyecto para obtener información general a partir de los datos del archivo. Su uso principal fue resumir los valores numéricos y mostrar resultados representativos sin necesidad de revisar cada registro individualmente. En la función *mostrar_estadisticas()*, se emplearon operaciones básicas como el cálculo del país con mayor y menor población mediante *max()* y *min()*, así como el promedio de población y superficie usando *sum()* y *len()*. Además, se contabilizaron los países por continente para mostrar una distribución de frecuencias.

Archivos CSV

El archivo CSV se utilizó como formato principal para almacenar los datos del proyecto. En Python, se trabajó con ellos mediante el módulo *csv*, que facilita tanto la lectura como la escritura. Se usó el archivo para registrar la información de los países, permitiendo leerla al iniciar el programa, manipularla en memoria y luego actualizar el archivo con los cambios realizados.

Diagrama de Flujo de Operaciones Principales



[Link a diagrama para verlo más grande]

Muestra del código

```python
import funciones

while True:
    print("=== MENÚ ===")
    print("    |1| Buscar un país")
    print("    |2| Filtrar países")
    print("    |3| Ordenar países")
    print("    |4| Mostrar estadísticas")
    print("    |5| Agregar país")
    print("    |6| Salir\n")

    opt = input("Opción: ")
    print()

    if opt == '1':
        funciones.buscar_pais()
    elif opt == '2':
        funciones.menu_filtrar_paises()
    elif opt == '3':
        funciones.ordenar_paises()
    elif opt == '4':
        funciones.mostrar_estadisticas()
    elif opt == '5':
        funciones.agregar_pais()
    elif opt == '6':
        print("Saliendo...")
        break
    else:
        print("Opción inválida.\n")
```

main.py importa *funciones.py*, mostrando el menú de opciones y ejecutandolas respectivamente.

```python
import unicodedata

def normalizar_caracteres(texto):
    texto_nfd = unicodedata.normalize("NFD", texto)
    return "".join(c for c in texto_nfd if unicodedata.category(c) != "Mn")
```

Función adicional en *utilidades.py*.

```python
import csv
import utilidades # Modulo local con funciones auxiliares (ej. normalizar)

def buscar_pais():
    # Solicita y normaliza el termino de busqueda del usuario
    pais_buscar = input("Nombre del país a buscar: ")
    pais_buscar = pais_buscar.strip() # Quita espacios en blanco
    pais_buscar = pais_buscar.lower() # Convierte a minusculas
    pais_buscar = utilidades.normalizar_caracteres(pais_buscar) # Quita tildes

    pais_encontrado = False
    print()

    # Abre el archivo CSV para lectura
    with open("paises.csv", "r", newline="", encoding="utf-8") as archivo:
        lector = csv.reader(archivo)
        next(lector) # Omite la fila de encabezado
        for fila in lector:
            if fila: # Asegura que la fila no este vacia
                # Comprueba coincidencia exacta (normalizada)
                if (fila[0]).lower() == pais_buscar:
                    pais_encontrado = True
                    print(fila[0])
                    print(f"Población: {int(fila[1]):,}")
                    print(f"Superficie: {float(fila[2]):,.0f} km^2\n")
                # Comprueba coincidencia parcial (si el termino esta contenido)
                elif ((fila[0]).lower()).startswith(pais_buscar):
                    pais_encontrado = True
                    print(fila[0])
                    print(f"Población: {int(fila[1]):,}")
                    print(f"Superficie: {float(fila[2]):,.0f} km^2\n")

    if not pais_encontrado:
        print("País no encontrado.")
```

Función *buscar.país()* en *funciones.py*, devolviendo coincidencias parciales o exactas. Vemos como al principio del archivo se importan los módulos a utilizar.

```python
def filtrar_por_continente():
    continente_filtrar = input("\nNombre del continente a filtrar: ")
    continente_filtrar = utilidades.normalizar_caracteres(continente_filtrar)
    continente_filtrar = continente_filtrar.strip().capitalize()
    print()

    # Lista de continentes válidos (normalizados)
    continentes_validos = ["America", "Europa", "Asia", "Africa", "Oceania"]

    # Validación
    if continente_filtrar not in continentes_validos:
        print("Continente inválido.\n")
        return

    continente_encontrado = False

    with open("paises.csv", "r", newline="", encoding="utf-8") as archivo:
        lector = csv.reader(archivo)
        next(lector)  # Omitir encabezado

        for fila in lector:
            if fila:
                # Normalizar el continente del CSV
                continente_csv = utilidades.normalizar_caracteres(fila[3]).capitalize()

                # Comparación ya normalizada
                if continente_csv == continente_filtrar:
                    continente_encontrado = True
                    print(fila[0])
                    print(f"Población: {int(fila[1]):,}")
                    print(f"Superficie: {float(fila[2]):,.0f} km^2\n")

    if not continente_encontrado:
        print("No se encontraron países en ese continente.\n")
```

La función *filtrar_por_continente()* funciona con el continente que ingrese el usuario.

```python
def filtrar_rango_poblacion():
    print("Filtrando por población:")

    # Validación del rango
    try:
        limite_inferior = int(input("Ingrese el límite inferior del rango: "))
        limite_superior = int(input("Ingrese el límite superior del rango: "))

        if limite_inferior < 0 or limite_superior < 0:
            print("Inválido - los límites no pueden ser menores a 0.\n")
            return

        if limite_superior < limite_inferior:
            print("Inválido - el límite superior no puede ser menor al límite inferior.\n")
            return

    except ValueError:
        print("Inválido - los valores deben ser un número entero.\n")
        return

    pais_encontrado = False
    print()

    with open("paises.csv", "r", newline="", encoding="utf-8") as archivo:
        lector = csv.reader(archivo)
        next(lector)  # Salta encabezado
        for fila in lector:
            if fila:
                poblacion = int(fila[1])
                if limite_inferior < poblacion < limite_superior:
                    pais_encontrado = True
                    print(fila[0])
                    print(f"Población: {poblacion:,}")
                    print(f"Superficie: {float(fila[2]):,.0f} km^2\n")

    if not pais_encontrado:
        print("No se encontró ningún país dentro del rango.\n")
```

La función *filtrar_rango_población()* permite filtrar por población ingresando un límite inferior y superior.

```python
def filtrar_rango_superficie():
    # Bucle de validacion de entrada para los limites (admite decimales)
    while True:
        try:
            print("Filtrando por superficie:")
            limite_inferior = float(input("Ingrese el límite inferior del rango: "))
            limite_superior = float(input("Ingrese el límite superior del rango: "))

            if limite_inferior < 0 or limite_superior < 0:
                print("Inválido - los límites no pueden ser menores a 0.")
                print()
            elif limite_superior < limite_inferior:
                print("Inválido - el límite superior no puede ser menor al límite inferior.")
                print()
        except ValueError:
            print("Inválido - los valores deben ser un número entero o decimal.")
            print()
        else:
            break

    pais_encontrado = False
    print()

    with open("paises.csv", "r", newline="", encoding="utf-8") as archivo:
        lector = csv.reader(archivo)
        next(lector)
        for fila in lector:
            if fila:
                # Comprueba si la superficie (columna 2) esta dentro del rango
                # Nota: convierte la superficie del CSV a int para comparar
                if limite_superior > int(fila[2]) > limite_inferior:
                    pais_encontrado = True
                    print(fila[0])
                    print(f"Población: {int(fila[1]):,}")
                    print(f"Superficie: {float(fila[2]):,.0f} km^2\n")

    if not pais_encontrado:
        print("No se encontró ningún país dentro del rango.")
```

La función *filtrar_rango_superficie()* funciona de manera similar.

```python
def menu_filtrar_paises():
    # Bucle principal del menu
    while True:
        print("===MENÚ===")
        print("Filtrar por:")
        print("    |1| Continente")
        print("    |2| Rango de población")
        print("    |3| Rango de superficie")
        print("    |4| Salir")

        opt = input("\nOpción: ")

        if opt == '1':
            filtrar_por_continente()
        elif opt == '2':
            filtrar_rango_poblacion()
        elif opt == '3':
            filtrar_rango_superficie()
        elif opt == '4':
            break # Regresa al menu principal
        else:
            print("Opción inválida.\n")
```

Esta función muestra un submenú con las distintas opciones de filtrado.

```python
def ordenar_paises():
    paises = []

    try:
        # Lee los datos del CSV a una lista de diccionarios
        with open("paises.csv", newline="", encoding="utf-8") as archivo:
            lector = csv.DictReader(archivo)
            for fila in lector:
                try:
                    # Procesa y limpia cada fila antes de agregarla
                    paises.append({
                        "nombre": fila.get("nombre", "").strip(),
                        "poblacion": int(float(fila.get("poblacion", 0))),
                        "superficie": float(fila.get("superficie", 0)),
                        "continente": fila.get("continente", "").strip().capitalize()
                    })
                except (ValueError, TypeError):
                    continue # Omite filas mal formadas
    except FileNotFoundError:
        print("No se encontró el archivo paises.csv.")
        return

    if not paises:
        print("No hay datos para ordenar.")
        return

    # Menu de criterios de ordenamiento
    print("=== ORDENAR PAÍSES ===")
    print("    |1| Ordenar por nombre")
    print("    |2| Ordenar por población")
    print("    |3| Ordenar por superficie\n")

    opt = input("Opción: ").strip()
    print()

    if opt not in ("1", "2", "3"):
        print("Opción inválida.")
        return
```

```
# Seleccion de direccion (Ascendente/Descendente)
orden = input("    (A) Ascendente / (D) Descendente: ").strip().upper()
print()

if orden not in ("A", "D"):
    print("Opción inválida.")
    return

descendente = (orden == "D") # Booleano para el parametro 'reverse'

# Aplica el ordenamiento usando 'sort' y una funcion lambda 'key'
if opt == '1':
    # Ordena alfabeticamente ignorando mayusculas
    paises.sort(key=lambda p: p["nombre"].lower() if p["nombre"] else "", reverse=descendente)
elif opt == '2':
    # Ordena numericamente por poblacion
    paises.sort(key=lambda p: p["poblacion"], reverse=descendente)
elif opt == '3':
    # Ordena numericamente por superficie
    paises.sort(key=lambda p: p["superficie"], reverse=descendente)

print()
print("=== RESULTADO ORDENADO ==")
print()

# Imprime la lista ordenada con formato
for pais in paises:
    # Usa .get() para manejar datos potencialmente faltantes
    nombre = pais.get("nombre") or "Desconocido"
    poblacion = pais.get("poblacion") or 0
    superficie = pais.get("superficie") or 0
    continente = pais.get("continente") or "Desconocido"

    print(nombre)
    print(f"    Población: {poblacion:,}")
    print(f"    Superficie: {superficie:,.0f} km^2")
    print(f"    Continente: {continente}\n")
```

Esta función muestra datos ordenados.

```python
def mostrar_estadisticas():
    paises = [] # Lista para almacenar los datos leidos

    try:
        # Lee el archivo usando DictReader para facilitar el acceso por nombre de columna
        with open("paises.csv", newline="", encoding="utf-8") as archivo:
            lector = csv.DictReader(archivo)
            for fila in lector:
                try:
                    # Convierte y almacena los datos de cada fila
                    paises.append({
                        "nombre": fila.get("nombre", "").strip(),
                        "poblacion": int(float(fila.get("poblacion", 0))),
                        "superficie": float(fila.get("superficie", 0)),
                        "continente": fila.get("continente", "").strip().capitalize()
                    })
                except (ValueError, TypeError):
                    # Omite filas con datos numericos invalidos
                    continue
    except FileNotFoundError:
        print("No se encontró el archivo paises.csv.")
        return

    # Comprueba si se cargaron datos
    if not paises:
        print("No hay datos para analizar.")
        return

    # Calculos estadisticos
    pais_mayor = max(paises, key=lambda p: p["poblacion"])
    pais_menor = min(paises, key=lambda p: p["poblacion"])

    prom_pob = sum(p["poblacion"] for p in paises) / len(paises)
    prom_sup = sum(p["superficie"] for p in paises) / len(paises)
```

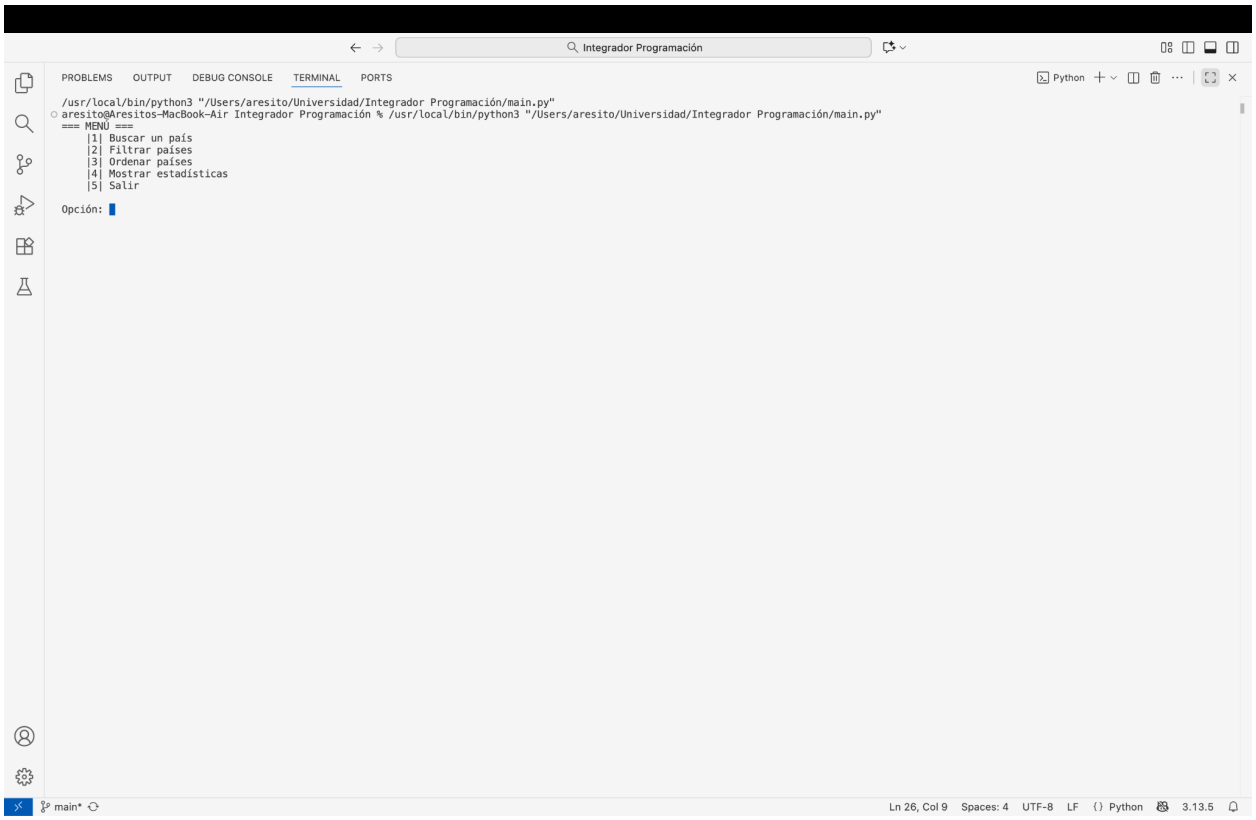
```python
    # Conteo de paises por continente
    continentes = {}
    for p in paises:
        c = p["continente"] or "Desconocido" # Agrupa continentes vacios
        continentes[c] = continentes.get(c, 0) + 1

    # Muestra los resultados
    print("=== ESTADÍSTICAS ===")
    print(f"    Mayor población: {pais_mayor['nombre']} ({pais_mayor['poblacion']:,} habitantes)")
    print(f"    Menor población: {pais_menor['nombre']} ({pais_menor['poblacion']:,} habitantes)")
    print(f"    Promedio de población: {prom_pob:,.0f} habitantes")
    print(f"    Promedio de superficie: {prom_sup:,.0f} km^2\n")

    print("Cantidad de países por continente:")
    for continente, cantidad in sorted(continentes.items()):
        print(f"    {continente}: {cantidad}\n")
```

Se muestran estadísticas de los datos.

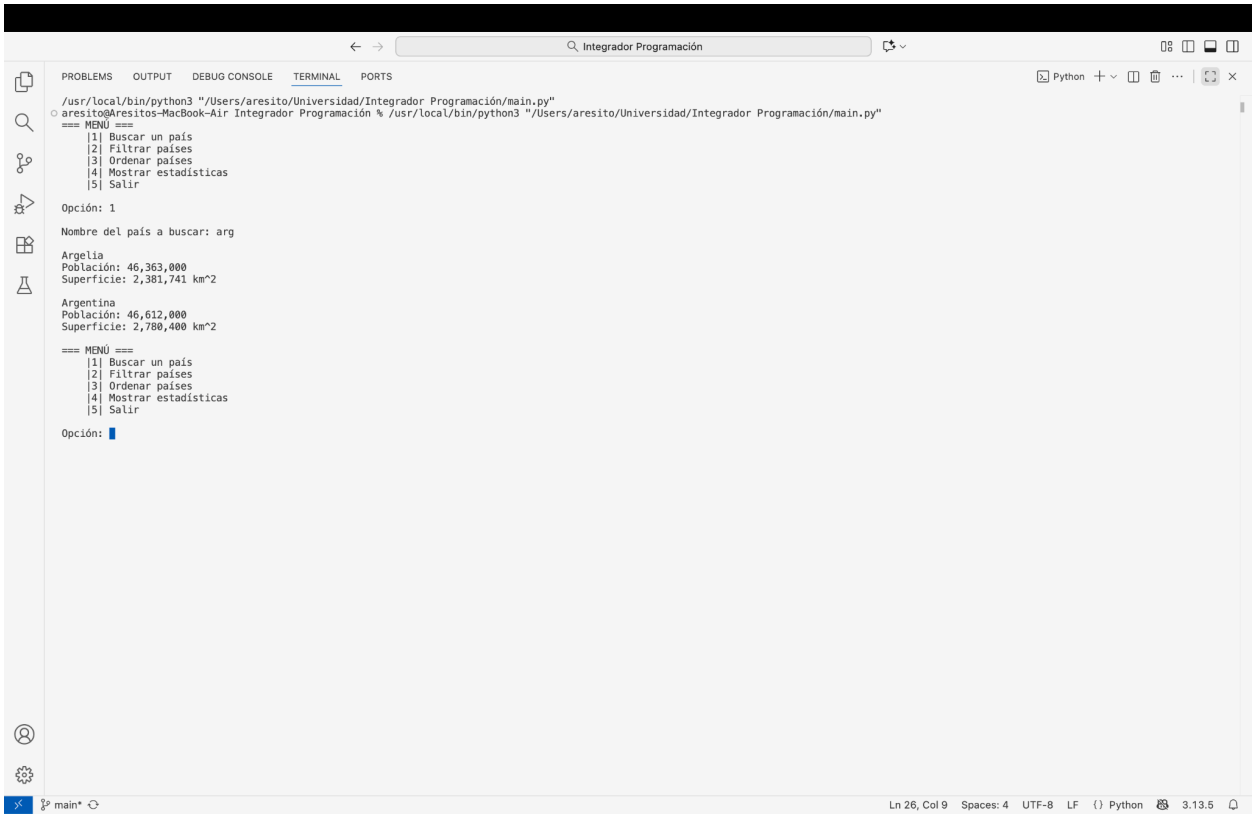
```python
def agregar_pais():
    print("=== AGREGAR NUEVO PAÍS ===")

    # Recopila los datos del nuevo pais
    nombre = input("Nombre del país: ").strip().capitalize()
    poblacion = input("Población: ").strip()
    superficie = input("Superficie (km^2): ").strip()
    continente = input("Continente: ").strip().capitalize()

    print()

    # ----- VALIDACIONES -----

    if not (nombre and poblacion and superficie and continente):
        print("Inválido - todos los campos son obligatorios.\n")
        return # Termina la funcion si faltan datos

    pais_invalido = False
    for c in nombre:
        if not (c.isalpha() or c.isspace()):
            pais_invalido = True

    if pais_invalido:
        print("Inválido - El nombre del país no puede contener números o caracteres especiales.\n")
        return

    continente_invalido = False
    for c in continente:
        if not (c.isalpha() or c.isspace()):
            continente_invalido = True

    if continente_invalido:
        print("Inválido - El nombre del continente no puede contener números o caracteres especiales.\n")
        return

    try:
        poblacion = int(float(poblacion)) # Permite decimales en input, pero guarda entero
        superficie = float(superficie)
    except ValueError:
        print("Población o superficie inválidas.\n")
        return

    # Abre el archivo en modo 'append' (agregar al final)
    with open("paises.csv", "a", newline="", encoding="utf-8") as archivo:
        escritor = csv.writer(archivo)
        # Escribe la nueva fila
        escritor.writerow([nombre, poblacion, superficie, continente])
        print("País añadido exitosamente.\n")
```

Finalmente, *agregar_pais()* permite agregar un país si sus datos son válidos.

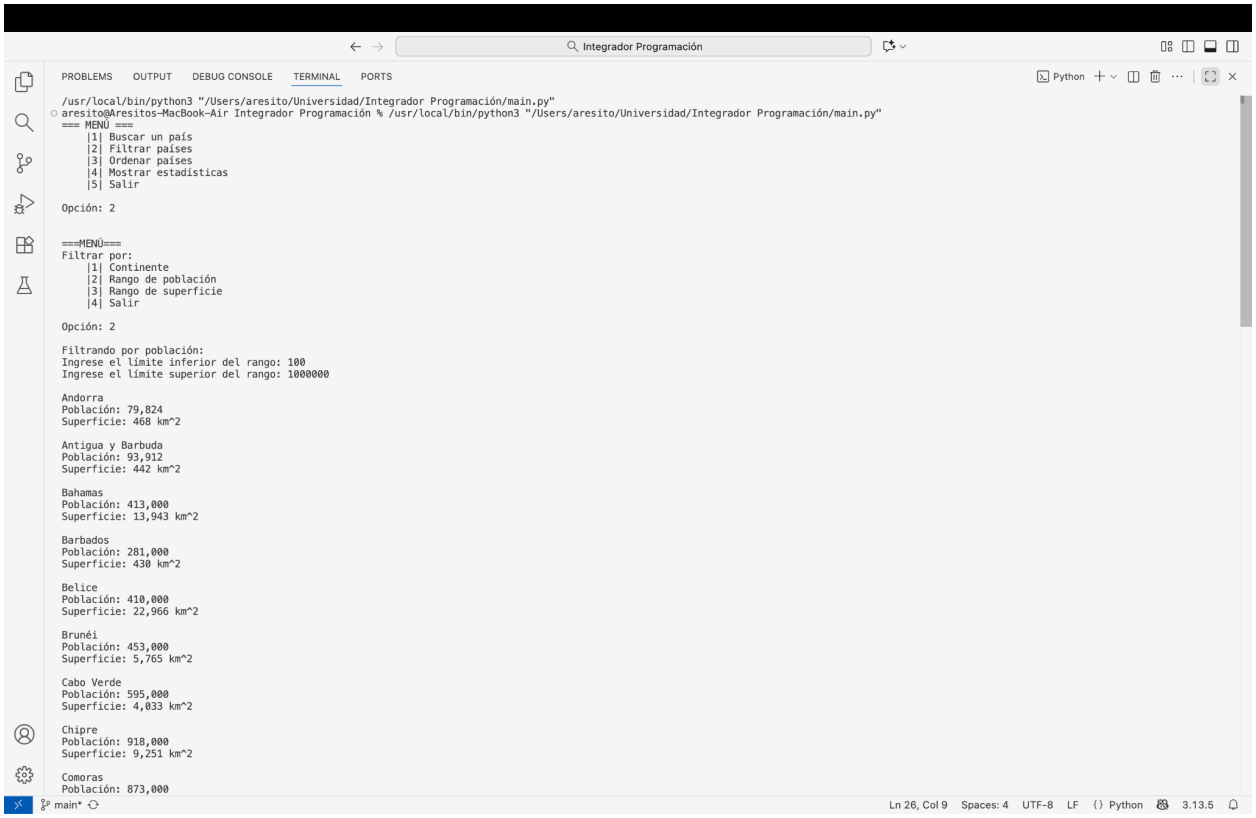
Capturas de Pantalla de Ejecución



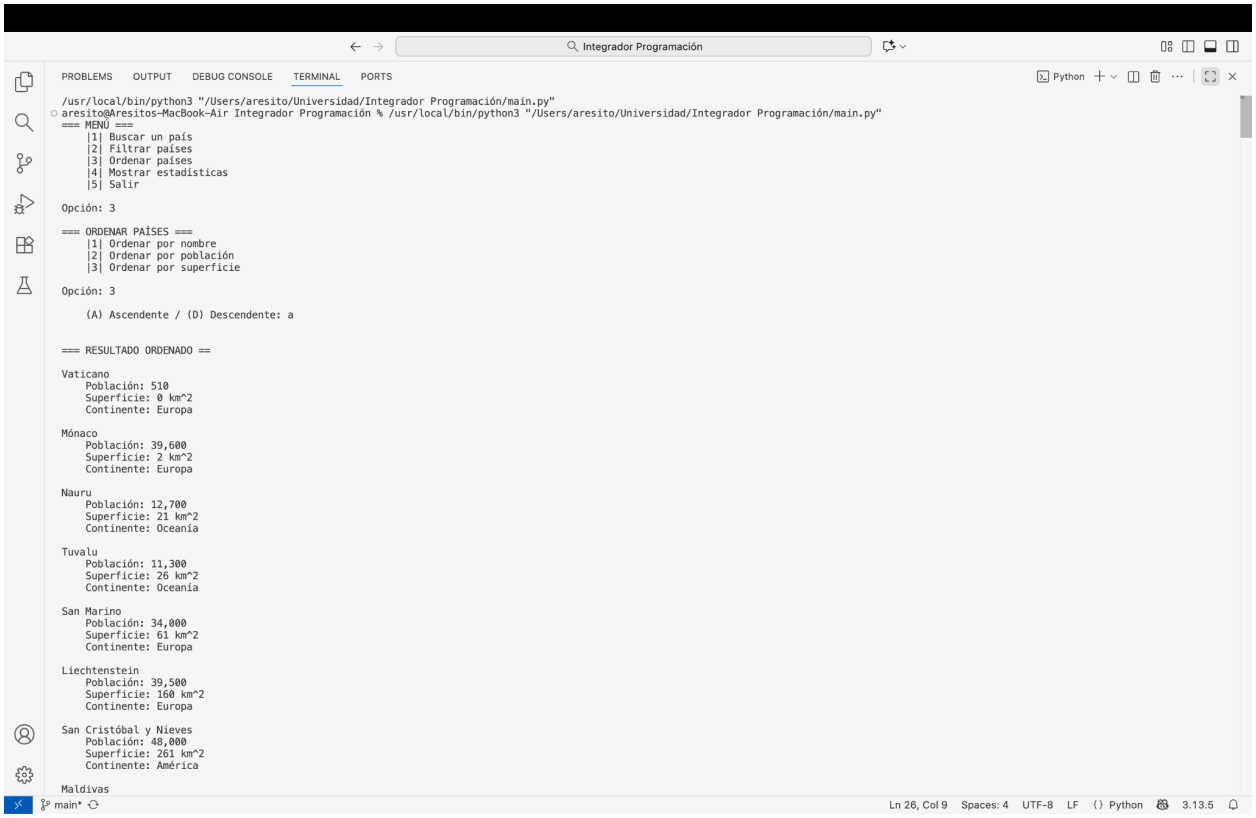
Se muestra el menú de opciones



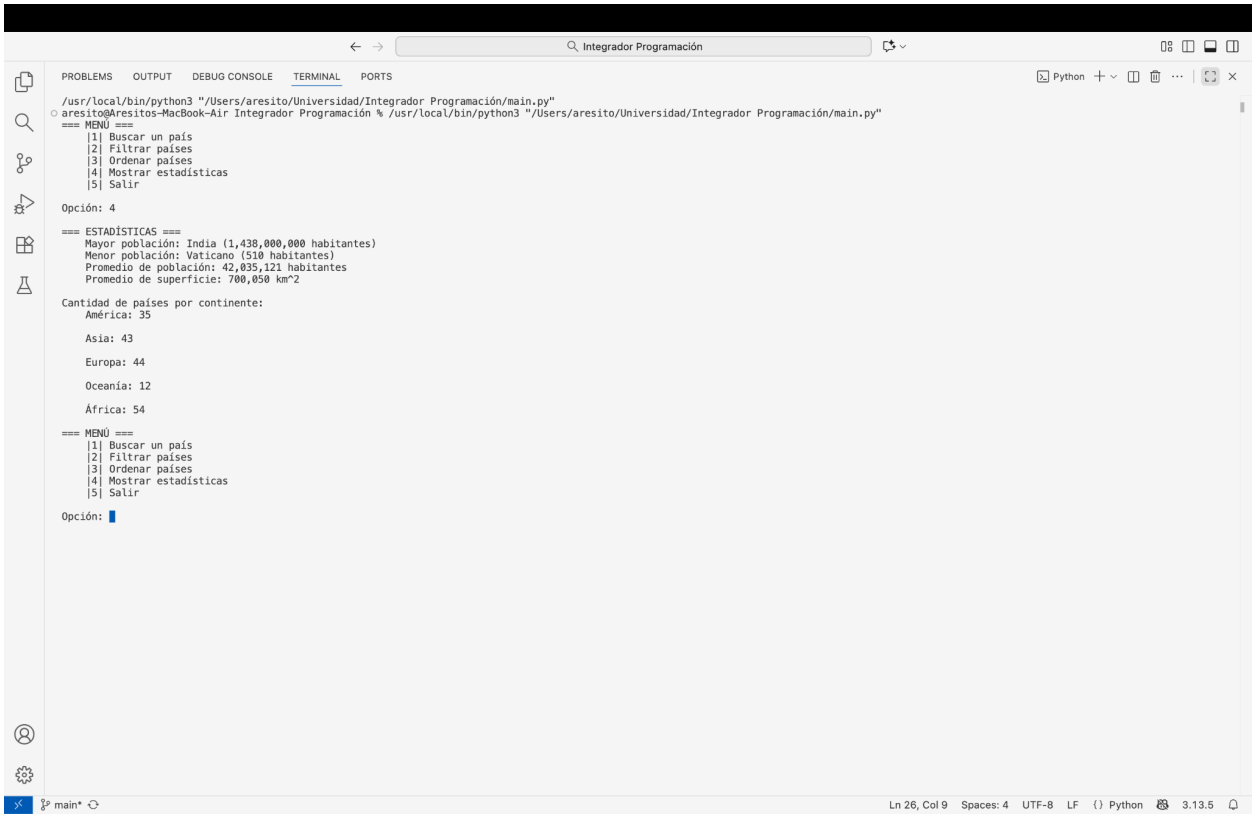
Ejemplo de la primera función: se ingresa una búsqueda (*“arg”*) y se devuelven los resultados parciales. Si hubiera una coincidencia exacta, se mostraría solo esa coincidencia.



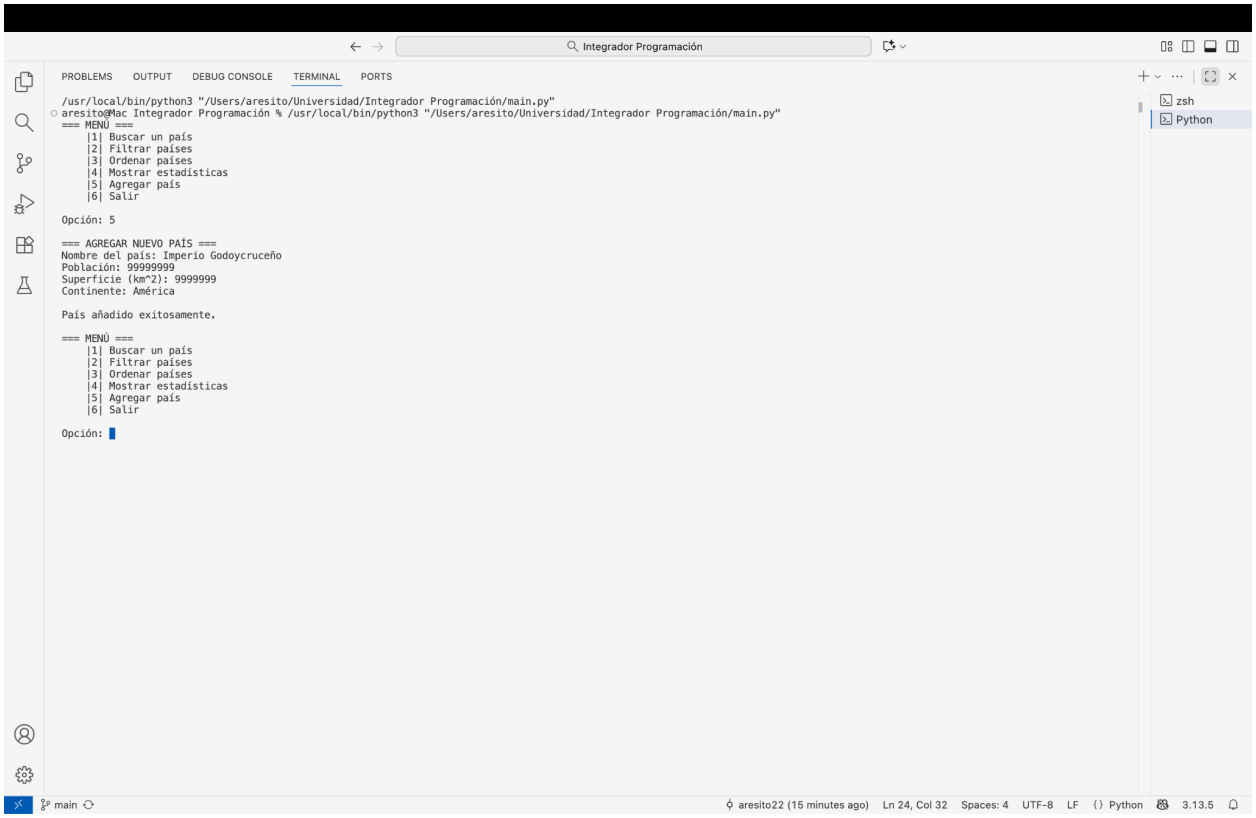
En la opción 2, se ofrece un submenú para elegir por que filtrar. En este ejemplo, se elige filtrar por población, colocando un límite inferior de 100 y uno superior de 1.000.000.



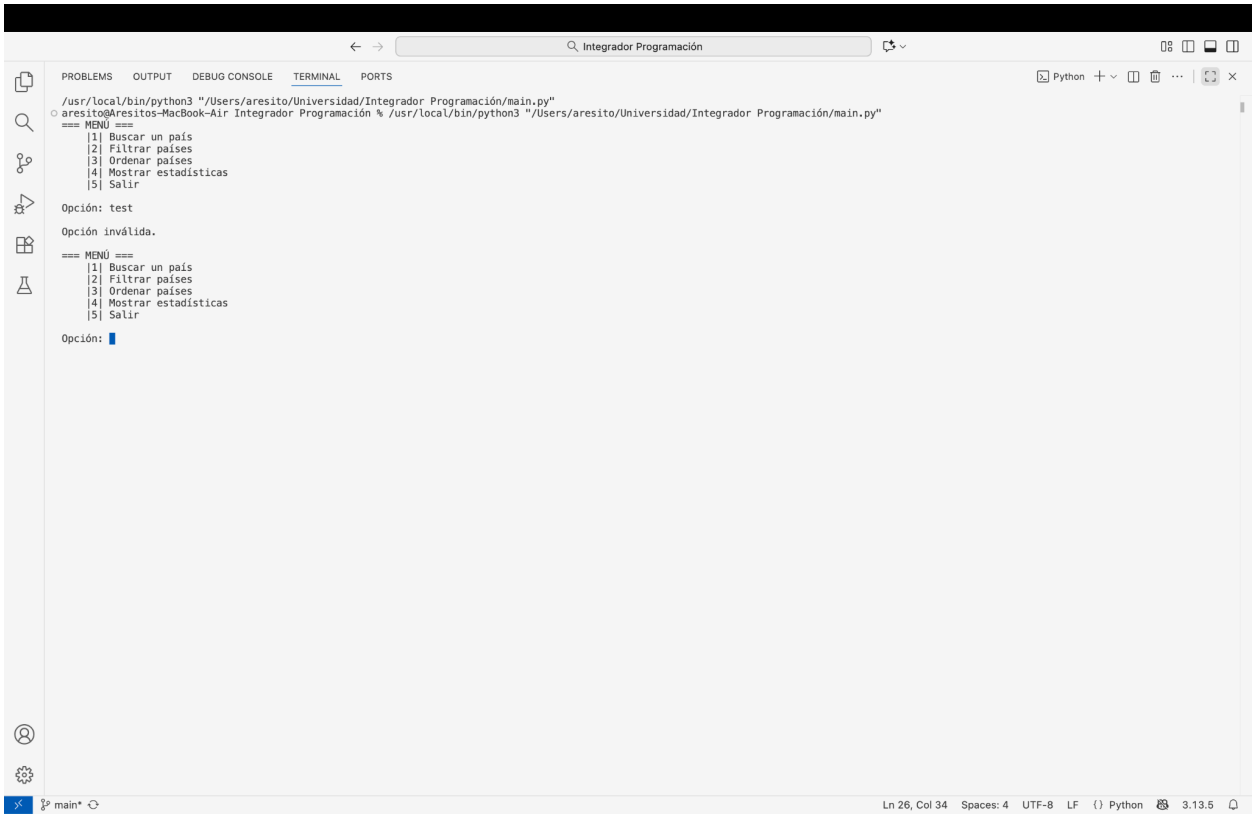
En la opción 3, se pueden filtrar países y también se ofrece un submenú de opciones, donde se ve cómo se elige la opción de ordenar por superficie. Luego, se pregunta si se debería mostrar de forma ascendiente o descendiente.



La opción 4 muestra estadísticas.



La opción 5, añadida por nosotros, permite agregar un país, llenando todos los campos de información necesarios. Si todos los campos son válidos, se agrega al CSV.



Si se elige una opción que no esté en el menú, se le notifica al usuario de que es una opción inválida y se vuelve a mostrar el menú.

Conclusión grupal

En este trabajo pudimos integrar de manera práctica varios de los contenidos vistos durante la materia, especialmente el uso de funciones, manejo de archivos CSV y organización modular del código. A lo largo del desarrollo fuimos ajustando y mejorando la estructura del programa hasta lograr una versión estable, clara y fácil de mantener. También aprendimos la importancia de normalizar datos, como en el caso de la eliminación de tildes para permitir búsquedas más flexibles, y de separar correctamente la lógica en archivos diferentes para evitar complicaciones a futuro.

El proyecto nos permitió ejercitar no solo la parte técnica sino también la comunicación y la toma de decisiones dentro del grupo: dividir tareas, revisar el código de los demás y buscar soluciones cuando aparecían problemas como *imports*, formatos o inconsistencias en el CSV. En conjunto, consideramos que el resultado final demuestra un dominio sólido de los conceptos trabajados y una buena capacidad para aplicarlos en un programa completo y funcional.