

1) Resolución del menú. Es decir lo creo y diseño.

1.1) para poder mostrar el menú, primero defino la función "mostrar_menu()"

```
README.md tpi_gestion_datos_paises.py aux_tpiipy tpi_paises.py
C:\Users\111\Downloads > aux_tpiipy > mostrar_menu
1 # Pasos para resolver el TPI Programación I
2
3 # (1) Resuelvo el rendí. Es decir lo creo y diseño.
4 # (2) para poder mostrar el menú, primero defino la función "mostrar_menu()"
5
6 def mostrar_menu():
7     # Para mostrar constantemente el menú:
8     while True:
9         print("**** Menú Principal ****")
10        print("1. Buscar país por nombre (coincidencia parcial o exacta)")
11        print("2. Filtrar por continente")
12        print("3. Filtrar por rango de población")
13        print("4. Filtrar por rango de superficie")
14        print("5. Ordenar países")
15        print("6. Mostrar estadísticas")
16        print("7. Ver todos los registros")
17        print("8. Salir")
18
19        opcion = input("Ingrese opción: ").strip()
20        match opcion:
21            case '1':
22                print("Buscar por países (coincidencia parcial o exacta)")
23            case '2':
24                print("Filtrar por continente")
25            case '3':
26                print("Filtrar por rango de población")
27            case '4':
28                print("Filtrar por rango de superficie")
29            case '5':
30                print("Ordenar países")
31            case '6':
32                print("Mostrar estadísticas")
33            case '7':
34                print("Ver todos los registros")
35            case '8':
36                print("Gracias por utilizar nuestra aplicación")
37                break
38            case _:
39                print("Gracias por utilizar nuestra aplicación")
```

```
# Invoco la función mostrar_menu()
```

```
> Unidad 7. Estructura de Datos Complejas 34         print("Ver todos los registros")
> Unidad 8. Manejo de Archivos 35         case '8':
36             print("Gracias por utilizar nuestra aplicación")
37             break
38         case _:
39             print("Gracias por utilizar nuestra aplicación")
40     # Invoca la función para poder mostrarla
41     mostrar_menu()
42
43
PROBLEMAS SALIDA CONSOILA DE DEBUGURACIÓN TERMINAL PUERTOS
*** 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
Ingresé opción: |
```

2) Resolución de las funciones asociadas a cada opción del menú comenzando con la primera.

“1. Buscar países por nombre (coincidencia parcial o exacta)”

```
7 # 2) Resuelvo las funciones asociadas al menú principal
8     # Opción 1
9 def buscar_nombre_paises():
10    paises = obtener_datos_paises() # Esta función se encargará de devolver todos los datos asociados
11                                # a cada país (nombre, población, continente, etc.)
12
13
14 def mostrar_menu():
15    # Para mostrar constantemente el menú:
16    while True:
17        print("**** Menú Principal ****")
18        print("1. Buscar país por nombre (coincidencia parcial o exacta)")
```

Para que la función “`buscar_nombre_paises()`” pueda funcionar, primero se debe obtener los nombres desde el archivo csv. Entonces creamos una función que nos devolverá el diccionario de los datos relacionados a los países “`paises = obtener_datos_paises()`”.

3) Importo el módulo csv

3.1) designo nombre de archivo

3.2) Defino la función “obtener_datos_paises()”

3.3) abrir el archivo “dataset_paises.csv”

3.4) creamos una lista “países = []”



```
5
6
7
8     # 3) Importo el módulo csv
9
10    # 3.1) Defino la función "obtener_datos_paises".
11
12    import csv
13
14    nombre_archivo = "dataset_paises.csv"
15
16    países = []
17
18    def obtener_datos_paises():
19        with open(nombre_archivo, newline="", encoding="utf-8") as archivo:
20            lector = csv.DictReader(archivo) # La función DictReader(), devuelve un iterador devolviendo un
21                                         # diccionario clave/valor a partir de la lectura de las líneas
22                                         # del archivo csv.
23
24
25    # 2) Resuelvo las funciones asociadas al menú principal
26    # Opción 1
27    def buscar_nombre_paises():
28
29        países = obtener_datos_paises() # Esta función se encargará de devolver todos los datos asociados
30        print(países)                 # a cada país (nombre, población, continente, etc.)
31
32
33    def mostrar_menu():
34        # Para mostrar constantemente el menú:
35        while True:
36            print("*** Menú Principal ***")
37            print("1. Buscar país por nombre (coincidencia parcial o exacta)")
38            print("2. Ver estadísticas de población")
39            print("3. Salir")
```

La clave del diccionario es el encabezado y el valor el contenido de la fila

3.5) Recorrer el lector fila por fila y construir un **nuevo diccionario**, convirtiendo los campos numéricos (que vienen como *strings*) a float.

```
13     leer_datos_paises():
14         h = open(nombre_archivo, newline="", encoding="utf-8") as archivo:
15             lector = csv.DictReader(archivo) # La función DictReader(), devuelve un iterador devolviendo un
16                                         # diccionario clave/valor a partir de la lectura de las líneas
17                                         # del archivo csv.
18         for fila in lector:
19             paises.append({"nombre": fila["nombre"], "poblacion": float(fila["poblacion"]), "superficie": float(fila["superficie"])
20
21
22         , "poblacion": float(fila["poblacion"]), "superficie": float(fila["superficie"]), "continente": fila["continente"]})
```

Pruebas realizadas

- No aparece la lista cuando selecciono la opción 1

4) Resolución la opción 1: “Buscar por países (nombre completo o parcial)”

```
31 |     )
32 |     # Opción 1
33 | def buscar_nombre_paises():
34 |
35 |     paises = obtener_datos_paises() # Esta función se encargará de devolver todos los datos asociados
36 |                                         # a cada país (nombre, población, continente, etc.)
37 |
38 |     termino = input("Ingrese el nombre del país o parte del nombre: ").strip()
39 |     if termino == "":
40 |         print("No ingresó ningún nombre en la búsqueda.")
41 |         return
42 |
43 |     # validación mayúsculas y minúsculas
44 |     termino_normalizado = termino.casefold()
45 |
46 |     # lista para guardar los países que coinciden:
47 |     coincidencia = []
48 |
49 |
50 |     for pais in paises:
51 |         nombre_normalizado = pais["nombre"].casefold()
52 |         # coincidencia normal o exacta
53 |         if termino_normalizado in nombre_normalizado:
54 |             coincidencia.append(pais)
55 |
56 |     if len(coincidencia) == 0:
57 |         print("No se encontraron países que coincidan con la búsqueda.")
58 |     else:
59 |         print(f"Se encontraron {len(coincidencia)} país(es): ")
60 |         for pais in coincidencia:
61 |             mostrar_pais(pais)
62 |         print()
```

Prueba realizada:

The screenshot shows the PyCharm IDE interface. The top navigation bar includes 'Archivo', 'Editor', 'Selección', 'Ver', 'Ir', and '...'. The title bar says 'Q. UTN-TUPaD-P1'. The left sidebar has sections for 'EXPLORADOR' (with 'EDTORES ABIERTOS' containing 'tpi_gestion_datos_paises.py'), 'UTN-TUPaD-P1' (with 'Trabajo_Práctico_Integrador' expanded to show 'dataset_paises.csv', 'README.md', and 'tpi_gestion_datos_paises.py'), and 'Unidad 1. Estructuras Secuenciales', 'Unidad 2. Trabajo Colaborativo', 'Unidad 3. Estructuras Condicionales', 'Unidad 4. Estructuras Repetitivas', 'Unidad 5. Listas', 'Unidad 6. Funciones', 'Unidad 7. Estructura de Datos Complejas', 'Unidad 8. Manejo de Archivos', and 'README.md'. The main editor area displays the code for 'tpi_gestion_datos_paises.py'. The terminal window below shows the execution of the script, prompting for input ('Ingrese opción:'), displaying search results ('Se encontraron 1 país(es): Argentina | Población: 45376763 | Superficie: 2780400 | Continente: América'), and listing a menu of options from 1 to 8.

```
tpi_gestion_datos_paises.py M x
Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > buscar_nombre_paises
33     def buscar_nombre_paises():
34         coincidencia.append(país)
35
36         if len(coincidencia) == 0:
37             print("No se encontraron países que coincidan con la búsqueda.")
38         else:
39             print(f"Se encontraron {len(coincidencia)} país(es): ")
40             for país in coincidencia:
41
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
42
43     7. Ver todos los registros
44     8. Salir
45
46 Ingrese opción: 1
47
48 Ingrese el nombre del país o parte del nombre: arg
49 Se encontraron 1 país(es):
50 Argentina | Población: 45376763 | Superficie: 2780400 | Continente: América
51
52 *** Menú Principal ***
53 1. Buscar país por nombre (coincidencia parcial o exacta)
54 2. Filtran por continente
55 3. Filtran por rango de población
56 4. Filtran por rango de superficie
57 5. Ordenan países
58 6. Mostrar estadísticas
59 7. Ver todos los registros
60 8. Salir
```

Solución de errores: carga los países dos veces

Antes:

- Usaba la misma lista global / cada llamada volvía a agregar países.

Luego:

- La función crea una lista nueva cada vez

```
Explorador: tpi_gestion_datos_paises.py M
Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > obtener_datos_paises
5
6
7 # 3) Importo el módulo csv
8 # 3.1) Defino la función "obtener_datos_paises".
9 import csv
10 nombre_archivo = "dataset_paises.csv"
11
12 def obtener_datos_paises():
13     paises = []
14     with open(nombre_archivo, newline="", encoding="utf-8") as archivo:
15         lector = csv.DictReader(archivo) # La función DictReader(), devuelve un iterador devolviendo un
16                                         # diccionario clave/valor a partir de la lectura de las líneas
17                                         # del archivo csv.
```

5) Resolución la opción 2:

```
Explorador: tpi_gestion_datos_paises.py M
Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > ...
62
63     # Opción 2
64     def filtrar_por_continente():
65         # cargar todos los países desde el archivo csv
66         paises = obtener_datos_paises()
67
68         # obtener desde el dataset el conjunto de continentes disponibles
69         continentes_disponibles = sorted({pais["continente"] for pais in paises})
70
71         print("Continentes disponibles en el dataset: ")
72         for cont in continentes_disponibles:
73             print(f" - {cont}")
74
75         # Pedir a usuario el continente (o parte del nombre)
76         termino = input("Ingrese el continente o parte del nombre: ").strip()
77
78         if termino == "":
79             print("No ingresó ningún continente en la búsqueda.")
80             return
81
82         # validación mayúsculas y minúsculas
83         termino_normalizado = termino.casefold()
84
85         # Buscar coincidencias
86         coincidencia = []
87         for pais in paises:
88             continente_normalizado = pais["continente"].casefold()
89             if termino_normalizado in continente_normalizado:
90                 coincidencia.append(pais)
91
92         #Mostrar resultados
93         if len(coincidencia)==0:
94             print("No se encontraron países para ese continente.")
95         else:
96             print(f"Se encontraron {len(coincidencia)} país(es): ")
97             mostrar_pais
98
99         print()
```

Prueba realizada:

```
tpi_gestion_datos_paises.py M
100
111     print(" 0. Salir ")
112
113     opcion = input("Ingrese opción: ").strip()
114     match opcion:
115         case '1':
116             buscar_nombre_paises()
117         case '2':
118             print("Filtrar por continente")
119         case '3':
120
121             print(" 8. Salir ")
122             Ingrese opción:
123             Filtrar por continente
124             *** Menú Principal ***
125             Filtrar por continente
126             *** Menú Principal ***
```

Error:

- Olvidé de cambiar el match case 2

Solución:

- Cambio el nombre del match case 2
- Error: no aparecen los nombres de los países del continente seleccionado porque no estaba llamando la función correctamente:

```
tpi_gestion_datos_paises.py M
Trabajo_Práctico_Integrador > tp_gestion_datos_paises.py > mostrar_menu()
99     def mostrar_menu():
100         print("6. Mostrar estadísticas")
101         print("7. Ver todos los registros")
102         print("8. Salir")
103
104         opcion = input("Ingrese opción: ").strip()
105         match opcion:
106             case '1':
107                 buscar_nombre_paises()
108             case '2':
109                 filtrar_por_continente()
110             case '3':
111                 print("Filtrar por rango de población")
112             case '4':
113                 print("Filtrar por rango de superficie")
114
115         print("Continentes disponibles en el dataset:")
116         print("- América")
117         print("- Asia")
118         print("- Europa")
119         print("- Oceanía")
120         print("- África")
121
122         Ingresa el continente o parte del nombre: América
123         Se encontraron 3 país(es):
124
125         *** Menú Principal ***
126
127
128     #Mostrar resultados
129     if len(coincidencia)==0:
130         print("No se encontraron países para ese continente.")
131     else:
132         print(f"Se encontraron {len(coincidencia)} país(es): ")
133         for país in coincidencia:
134             mostrar_pais(pais)
135
136         print()
```

Solución:

- Llamar a la función correctamente:

```
tpi_gestion_datos_paises.py M
Trabajo_Práctico_Integrador > tp_gestion_datos_paises.py > filtrar_por_continente()
64     def filtrar_por_continente():
65         for país in países:
66             continente_normalizado = país["continente"].casefold()
67             if termino_normalizado in continente_normalizado:
68                 coincidencia.append(país)
69
70         #Mostrar resultados
71         if len(coincidencia)==0:
72             print("No se encontraron países para ese continente.")
73         else:
74             print(f"Se encontraron {len(coincidencia)} país(es): ")
75             for país in coincidencia:
76                 mostrar_pais(país)
77
78         print()
79
80     #Mostrar estadísticas
81     if len(coincidencia)==0:
82         print("6. Mostrar estadísticas")
83         print("7. Ver todos los registros")
84         print("8. Salir")
85
86         Ingresa opción: 2
87         Continentes disponibles en el dataset:
88             - América
89             - Asia
90             - Europa
91             - Oceanía
92             - África
93
94         Ingresa el continente o parte del nombre: América
95         Se encontraron 3 país(es):
96         Argentina | Población: 45376769 | Superficie: 2780490 | Continente: América
97         Brasil | Población: 213999437 | Superficie: 8515767 | Continente: América
98         Canadá | Población: 38000005 | Superficie: 9984670 | Continente: América
99
100        *** Menú Principal ***
```

6) Resolución opción 3:

```
tpi_gestion_datos_paises.py M x
Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > ...
    print()

    # Opción 3
def filtrar_por_rango_poblacion():
    # cargar todos los países
    países = obtener_datos_países()
    print("****Filtrar por rango de población****")
    print("Los valores se ingresan en cantidad de habitantes (ej: 100000000)")

    # Pedir población mínima y máxima con validación
    mínimo = solicitar_entero_no_negativo("Ingrese población mínima: ")
    máximo = solicitar_entero_no_negativo("Ingrese población máxima: ")

    # Validar coherencia del rango
    if máximo < mínimo:
        print("La población máxima no puede ser menor que la mínima.")
        print("Intercambiando valores: mínimo = {máximo}, máximo = {mínimo}")
        mínimo, máximo = máximo, mínimo

    #Filtrar países dentro del rango [mínimo, máximo]
    coincidencia = []
    for país in países:
        población = país["población"]
        if mínimo <= población <= máximo:
            coincidencia.append(país)
    # Mostrar resultados
    print(f"\n{len(coincidencia)} países fueron encontrados en ese rango de población.")


def mostrar_menu():
    # Para mostrar constantemente el menú:
    while True:
        print("**** Menú Principal ****")
        print("1. Buscar país por nombre (coincidencia parcial o exacta)")
        print("2. Filtrar por continente")
        print("3. Filtrar por rango de población")
        print("4. Filtrar por rango de superficie")
        print("5. Ordenar países")
        print("6. Mostrar estadísticas")
        print("7. Ver todos los registros")
        print("8. Salir")
```

```
trabajo_Práctico_Integrador.py M x
    print()

    # Opción 3
def filtrar_por_rango_poblacion():
    # cargar todos los países
    países = obtener_datos_países()
    print("****Filtrar por rango de población****")
    print("Los valores se ingresan en cantidad de habitantes (ej: 100000000)")

    # Pedir población mínima y máxima con validación
    mínimo = solicitar_entero_no_negativo("Ingrese población mínima: ")
    máximo = solicitar_entero_no_negativo("Ingrese población máxima: ")

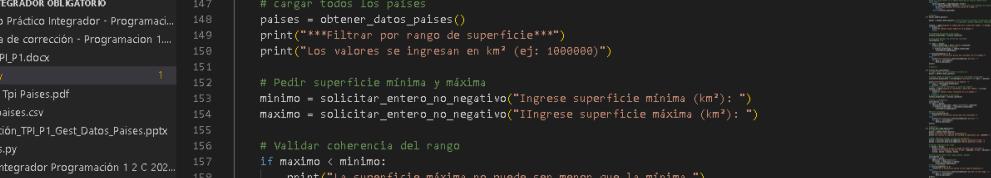
    # Validar coherencia del rango
    if máximo < mínimo:
        print("La población máxima no puede ser menor que la mínima.")
        print("Intercambiando valores: mínimo = {máximo}, máximo = {mínimo}")
        mínimo, máximo = máximo, mínimo

    #Filtrar países dentro del rango [mínimo, máximo]
    coincidencia = []
    for país in países:
        población = país["población"]
        if mínimo <= población <= máximo:
            coincidencia.append(país)
    # Mostrar resultados
    print(f"\n{len(coincidencia)} países fueron encontrados en ese rango de población.")


def mostrar_menu():
    # Para mostrar constantemente el menú:
    while True:
        print("**** Menú Principal ****")
        print("1. Buscar país por nombre (coincidencia parcial o exacta)")
        print("2. Filtrar por continente")
        print("3. Filtrar por rango de población")
        print("4. Filtrar por rango de superficie")
        print("5. Ordenar países")
        print("6. Mostrar estadísticas")
        print("7. Ver todos los registros")
        print("8. Salir")
```

Prueba realizada: ok

7) Resolución opción 4:



```
dataset_paises.csv          144     print()
145     # Opción 4
146 def filtrar_por_superficie():
147     # cargar todos los países
148     paises = obtener_datos_paises()
149     print("****Filtrar por rango de superficie****")
150     print("Los valores se ingresan en km2 (ej: 1000000)")
151
152     # Pedir superficie mínima y máxima
153     minimo = solicitar_entero_no_negativo("Ingrese superficie mínima (km2): ")
154     maximo = solicitar_entero_no_negativo("Ingrese superficie máxima (km2): ")
155
156     # Validar coherencia del rango
157     if maximo < minimo:
158         print("La superficie máxima no puede ser menor que la mínima.")
159         print("Intercambiando valores: mínima = {maximo}, máxima = {minimo}")
160         minimo, maximo = maximo, minimo
161
162     # Filtrar países dentro del rango
163     filtrados = []
164     for pais in paises:
165         sup = pais["superficie"]
166         if minimo <= sup <= maximo:
167             filtrados.append(pais)
168     if len(filtrados) == 0:
169         print("No se encontraron países en ese rango de superficie.")
```

```
print()
return

# Preguntar orden: ascendente o descendente
print("¿Cómo desea ordenar los resultados?")
print("A- Ascendente (de menor a mayor superficie)")
print("D- Descendente (de mayor a menor superficie)")
orden = input("Ingrese A ó D").strip().casefold()

if orden == "d":
    descendente = True
else:
    descendente = False

# Ordenar por superficie
filtrados_ordenados = sorted(filtrados, key=lambda p:p["superficie"], reverse=descendente)

# Mostrar resultados
print(f"Se encontraron {len(filtrados_ordenados)} país(es) en ese rango:")
for pais in filtrados_ordenados:
    mostrar_pais()
print()
```

Corrección:

- Reemplazar definición de la función “def filtrar_por_superficie()” por “def filtrar_por_rango_superficie()”

```
2. Rubrica de corrección - Programación 1.... 144
Apunte_TPI_P1.docx 145
aux_tpi.py 146 print()
# Opción 4
def filtrar_por_rango_de_superficie():
Borrador_Tpi_Paises.pdf 147 # cargar todos los países
dataset_paises.csv 148 países = obtener_datos_paises()
E_Presentación_TPI_P1_Gest_Datos_Paises.pptx 149 print("****Filtrar por rango de superficie****")
tpi_paises.py 150 print("Los valores se ingresan en km2 (ej: 1000000)")
```

Prueba realizada:

- Error: olvidé de llamar correctamente la función “mostrar_pais(?)”. Es decir, no agregué el argumento.

Solución:

- Llamé correctamente a la función “mostrar_pais(país)”. Se prueba y funciona bien. Salvo que aparece una falta de espacio, que puede generar confusión a la hora de elegir cómo deseamos ver el orden de los resultados (de forma ascendente o descendente respecto a la superficie). Se subsana “espacio” dentro del código.

```
aux_tp1.py x dataset_paises.csv tpi_paises.py
aux_tp1.py > filtrar_por_rango_superficie
146     def filtrar_por_rango_superficie():
147         print("No se encontraron países en ese rango de superficie.")
148         print()
149         return
150
151     # Preguntar orden: ascendente o descendente
152     print("¿Cómo desea ordenar los resultados?")
153     print("A- Ascendente (de menor a mayor superficie)")
154     print("D- Descendente (de mayor a menor superficie)")
155     orden = input("Ingrese A ó D: ").strip().casefold()
156
157     if orden == "d":
158         descendente = True
159     else:
160         descendente = False
161
162     # Ordenar por superficie
163     filtrados_ordenados = sorted(filtrados, key=lambda p:p["superficie"], reverse=descendente)
164
165     # Mostrar resultados
166     print(f"Se encontraron {len(filtrados_ordenados)} país(es) en ese rango: ")
167     for pais in filtrados_ordenados:
168         print(pais)
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
```

País	Población	Superficie	Continente
Francia	67413000	551695	Europa
Egipto	104258000	1002450	Africa
Egipto	104258000	1002450	Africa
Sudáfrica	59308600	1210990	Africa
Argentina	45376763	2780400	América
India	1393400000	3287263	Asia
Sudáfrica	59308600	1210990	Africa

Solución final: ok

The screenshot shows the PyCharm IDE interface. The left sidebar lists files: aux_tp1.py, dataset_paises.csv, tp1_paises.py, and aux_tp1.ipynb. The main editor window displays a Python script named aux_tp1.py with the following code:

```
aux_tp1.py x dataset_paises.csv tp1_paises.py
aux_tp1.py > Filtrar por rango superficie
146 def filtrar_por_rango_superficie():
147     print()
148     return
149
150 # Preguntar orden: ascendente o descendente
151 print("¿Cómo desea ordenar los resultados?")
152 print("A- Ascendente (de menor a mayor superficie)")
153 print("D- Descendente (de mayor a menor superficie)")
154 orden = input("Ingrese A ó D >> : ").strip().casefold()
155
156 if orden == "a":
157     print("Se encontraron 7 país(es) en ese rango:")
158     for pais in paises:
159         if pais['Superficie'] <= orden:
160             print(pais)
161
162 else:
163     print("Se encontraron 7 país(es) en ese rango:")
164     for pais in paises:
165         if pais['Superficie'] >= orden:
166             print(pais)
167
168 print("Menú Principal")
169 1. Buscar país por nombre (coincidencia parcial o exacta)
170 2. Filtrar por continente
171 3. Filtrar por rango de población
172 4. Filtrar por rango de superficie
173 5. Ordenar países
174 6. Mostrar estadísticas
175 7. Ver todos los registros
176 8. Salir
177
```

The terminal tab shows the output of the script:

```
¿Cómo desea ordenar los resultados?
A- Ascendente (de menor a mayor superficie)
D- Descendente (de mayor a menor superficie)
Ingrese A ó D >> : a
Se encontraron 7 país(es) en ese rango:
Francia | Población: 674138000 | Superficie: 551695 | Continente: Europa
Egipto | Población: 104258000 | Superficie: 1002458 | Continente: África
Sudáfrica | Población: 593086900 | Superficie: 1219990 | Continente: África
Argentina | Población: 45376763 | Superficie: 2780400 | Continente: América
India | Población: 1393490938 | Superficie: 3287263 | Continente: Asia
Australia | Población: 259256000 | Superficie: 7692024 | Continente: Oceanía
Brasil | Población: 213993437 | Superficie: 8515767 | 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
```

8) Resolución opción 5:

- Prueba realizada:

```

193     def ordenar_paises():
214         # Ordenar según criterio elegido
215         if criterio == "1":
216             paises_ordenados = sorted(paises, key=lambda p:p["nombre"].casefold(), reverse=descendente)
217         elif criterio == "2":
218             paises_ordenados = sorted(paises, key=lambda p:p["poblacion"], reverse=descendente)
219         else:
220             paises_ordenados = sorted(paises, key=lambda p:p["superficie"], reverse=descendente)
221
222
223
D- Descendente (de mayor a menor superficie)
Ingres A ó D > : d
Traceback (most recent call last):
  File "C:\Users\lili\OneDrive\UTN\3. TUPaD\2. Ago-Dic 2025 (cuatrimestre 1)\3. Materiales\1. Programación I\Trabajo Integrador Obligatorio\aux_tpi.py", line 269, in <module>
    mostrar_menu()
...
  File "C:\Users\lili\OneDrive\UTN\3. TUPaD\2. Ago-Dic 2025 (cuatrimestre 1)\3. Materiales\1. Programación I\Trabajo Integrador Obligatorio\aux_tpi.py", line 258, in mostrar_menu
    ordenar_paises()
...
  File "C:\Users\lili\OneDrive\UTN\3. TUPaD\2. Ago-Dic 2025 (cuatrimestre 1)\3. Materiales\1. Programación I\Trabajo Integrador Obligatorio\aux_tpi.py", line 218, in ordenar_paises
    paises_ordenados = sorted(paises, key=lambda p:p["nombre"].casefold(), reverse=descendente)
TypeError: 'function' object is not iterable
PS C:\Users\lili\OneDrive\UTN\3. TUPaD\2. Ago-Dic 2025 (cuatrimestre 1)\3. Materiales\1. Programación I\Trabajo Integrador Obligatorio> []

```

- Error: falta de () en la función "obtener_datos_paises()?"

```

190     print()
191
192     # Opción 5
193     def ordenar_paises():
194         paises = obtener_datos_paises()
195         print("\n***Ordenar Países***")
196         print("1. Por nombre")
197         print("2. Por población")

```

- Solución: se agregan los paréntesis a la función "obtener_datos_paises()" y funciona ok

```

Ejecuta criterio (1, 2 ó 3): 1
Orden:
A- Ascendente (de menor a mayor superficie)
D- Descendente (de mayor a menor superficie)
Ingres A ó D > : d
Países Ordenados:
Sudáfrica | Población: 59308690 | Superficie: 1219090 | Continente: África
Japón | Población: 125800000 | Superficie: 377975 | Continente: Asia
India | Población: 1393409938 | Superficie: 3287263 | Continente: Asia
Francia | Población: 67413000 | Superficie: 551695 | Continente: Europa
España | Población: 184258000 | Superficie: 1002450 | Continente: África
Canadá | Población: 38000000 | Superficie: 9994670 | Continente: América
Brasil | Población: 213993437 | Superficie: 8515767 | Continente: América
Australia | Población: 25925600 | Superficie: 7692024 | Continente: Oceanía
Argentina | Población: 45376763 | Superficie: 2788400 | Continente: América
Alemania | Población: 83149300 | Superficie: 357022 | Continente: Europa

*** 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

Ingresar opción: 5
***ordenar Países***
1. Por nombre

```

The screenshot shows a code editor interface with two tabs open: `aux_tpi.py` and `tpi_paises.py`. The `aux_tpi.py` tab contains a function definition:`def ordenar_paises():
 print("1. Por nombre")
 print("2. Por población")
 print("3. Por superficie")`In the terminal window, the user has run the script and selected option 2. The output shows a list of countries ordered by population:`***Ordenar Paises***
1. Por nombre
2. Por población
3. Por superficie
Elija criterio (1, 2 ó 3): 2
Orden:
A- Ascendente (de menor a mayor superficie)
D- Descendente (de mayor a menor superficie)
Ingrese A ó D >> : a
Países Ordenados:
Australia | Población: 25925600 | Superficie: 7692024 | Continente: Oceanía
Canadá | Población: 38008005 | Superficie: 9984679 | Continente: América
Argentina | Población: 45376763 | Superficie: 2788400 | Continente: América
Sudáfrica | Población: 59308690 | Superficie: 12190990 | Continente: África
Francia | Población: 67413000 | Superficie: 551695 | Continente: Europa
Alemania | Población: 83149300 | Superficie: 357022 | Continente: Europa
Egipto | Población: 104258000 | Superficie: 1002450 | Continente: África
Japón | Población: 125800000 | Superficie: 377975 | Continente: Asia
Brasil | Población: 213993437 | Superficie: 8515767 | Continente: América
India | Población: 1393409038 | Superficie: 3287263 | Continente: Asia`

This screenshot is nearly identical to the one above, showing the same code editor interface and terminal output. The only difference is the line number in the terminal output, which has changed from 193 to 199.

9) Resolución opción 6:

- Prueba realizada: ok

```
tpi_gestion_datos_paises.py M x
Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > mostrar_menu
print()

# Opción 6
def mostrar_estadisticas():
    paises = obtener_datos_paises()

    if len(paises) == 0:
        print("No hay datos de países para calcular estadísticas.")
        print()
        return

    print("\n***Estadísticas del dataset de países***")
    # Cantidad total de países
    total_paises = len(paises)

    # Listas auxiliares de población y superficie
    poblaciones = [p["poblacion"] for p in paises]
    superficies = [p["superficie"] for p in paises]

    # Población total y promedio
    poblacion_total = sum(poblaciones)
    poblacion_promedio = poblacion_total / total_paises

    # Superficie total y promedio
    superficie_total = sum(superficies)
    superficie_promedio = superficie_total / total_paises

    # País con mayor y menor población
    pais_mayor_pob = max(paises, key=lambda p:p["poblacion"])
    pais_menor_pob = min(paises, key=lambda p:p["poblacion"])

    # País con mayor y menor superficie
    pais_mayor_sup = max(paises, key=lambda p:p["superficie"])
    pais_menor_sup = min(paises, key=lambda p:p["superficie"])

Lin. 319, col. 38 Espacios:4 UTF-8 CRLF () Python Prettier
```

```
tpi_gestion_datos_paises.py M x
Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > mostrar_menu
227 def mostrar_estadisticas():
    # País con mayor y menor superficie
    pais_mayor_sup = max(paises, key=lambda p:p["superficie"])
    pais_menor_sup = min(paises, key=lambda p:p["superficie"])

    # Cantidad de países por continente
    conteo_continentes = {}
    for p in paises:
        cont = p["continente"]
        if cont in conteo_continentes:
            conteo_continentes[cont] += 1
        else:
            conteo_continentes[cont] = 1

    #Mostrar resultados
    print(f"\nCantidad total países: {total_paises}")
    print(f"Población total: {poblacion_total:.0f} habitantes")
    print(f"Población promedio: {poblacion_promedio:.0f} habitantes")
    print()
    print(f"Superficie total: {superficie_total:.0f} km²")
    print(f"Superficie promedio: {superficie_promedio:.0f} km²")
    print()
    print("País con mayor población:")
    mostrar_pais(pais_mayor_pob)
    print()
    print("País con menor población:")
    mostrar_pais(pais_menor_pob)
    print()
    print("País con mayor superficie:")
    mostrar_pais(pais_mayor_sup)
    print()
    print("País con menor superficie:")
    mostrar_pais(pais_menor_sup)

Lin. 319, col. 38 Espacios:4 UTF-8 CRLF () Python Prettier
```

```
tpi_gestion_datos_paises.py M ●
Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > mostrar_menu
227 def mostrar_estadisticas():
    print()
    print("País con menor superficie:")
    mostrar_pais(pais_menor_sup)
    print()
    print("Cantidad de países por continente: ")
    for continente, cantidad in conteo_continentes.items():
        print(f"- {continente}: {cantidad}")
    print()

def mostrar_menu():
    # Para mostrar constantemente el menú
    while True:
        print("**** Menú Principal ****")
        print("1. Buscar país por nombre (coincidencia parcial o exacta)")
        print("2. Filtrar por continente")
        print("3. Filtrar por rango de población")
        print("4. Filtrar por rango de superficie")
        print("5. Ordenar países")
        print("6. Mostrar estadísticas")
        print("7. Ver todos los registros")
        print("8. Salir")

        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':
                break
            case _:
                print("Opción no válida. Intente nuevamente.")

Fernando-Picco (Hace 19 horas) Lin. 294, col. 1 Espacios:4 UTF-8 CRLF () Python Prettier
```

```

    294 def mostrar_menu():
    295     print("1. Filtrar por rango de superficie")
    296     print("2. Ordenar países")
    297     print("3. Mostrar estadísticas")
    298     print("4. Ver todos los registros")
    299     print("5. Salir")
    300
    301     opcion = input("Ingrese opción: ").strip()
    302     match opcion:
    303         case '1':
    304             buscar_nombre_paises()
    305         case '2':
    306             filtrar_por_continente()
    307         case '3':
    308             filtrar_por_rango_poblacion()
    309         case '4':
    310             filtrar_por_rango_superficie()
    311         case '5':
    312             ordenar_paises()
    313         case '6':
    314             mostrar_estadisticas()
    315         case '7':
    316             print("Ver todos los registros")
    317         case '8':
    318             print("¡Gracias por utilizar nuestra aplicación! Hasta pronto!")
    319             break
    320         case _:
    321             print("La opción seleccionada no es válida para nuestra aplicación")
    322
    323     # Invoca la función para poder mostrarla
    324     mostrar_menu()
    325
    326
    327
    328
    329

```

- Funcionamiento: ok

```

    246     ...#Opción 6
    247     Resolución Segundo Parcial_TUPaD_F_Picco_C10.py
    248     def mostrar_estadisticas():
    249
    250         print("****Estadísticas del dataset de países***")
    251
    252         print("Cantidad total países: 10")
    253         print("Población total: 215661833 habitantes")
    254         print("Población promedio: 215664183 habitantes")
    255
    256         print("Superficie total: 35768356 km²")
    257         print("Superficie promedio: 3576836 km²")
    258
    259         print("País con mayor población:")
    260         print("India | Población: 1393409030 | Superficie: 3287263 | Continente: Asia")
    261
    262         print("País con menor población:")
    263         print("Australia | Población: 25925600 | Superficie: 7692024 | Continente: Oceania")
    264
    265         print("País con mayor superficie:")
    266
    267         print("País con mayor superficie:")
    268         print("Canadá | Población: 38008005 | Superficie: 9984670 | Continente: América")
    269
    270         print("País con menor superficie:")
    271         print("Alemania | Población: 83149300 | Superficie: 357022 | Continente: Europa")
    272
    273         print("Cantidad de países por continente:")
    274         print("- América: 3")
    275         print("- África: 2")
    276         print("- Oceania: 1")
    277         print("- Europa: 4")
    278         print("- Asia: 2")
    279
    280

```

- Verificación de funcionamiento: calculo realizado en excel

Dataset (dataset_paises) contents:

nombre	poblacion	superficie	continente	Columna1	Columna2
Argentina	45376763	2780400	América		
Japón	125900000	377975	Asia	población promedio	2156641833
Brasil	21393437	8515767	América		
Alemania	83149300	357022	Europa		
Australia	25525900	7692024	Oceania		
Egipto	104259000	1002450	Africa		
Canadá	33009005	9984670	América		
India	133046903	3287263	Asia		
Sudáfrica	5303690	1219090	Africa		
Francia	67413900	551695	Europa		
				poblacion_total	
					2156641833

10) Resolución opción 7:

- Prueba realizada: Ok

```

# Opción 7
def ver.todos.los_registros():
    países = obtener_datos_paises()

    if len(países) == 0:
        print("No hay datos para mostrar.")
        print()
        return
    print("****Lista completa de países en el dataset****")
    for país in países:
        mostrar_pais(país)
    print()

def mostrar_menu():
    # Para mostrar constantemente el menú:
    while True:
        print("**** Menú Principal ****")
        print("1. Buscar país por nombre (coincidencia parcial o exacta)")
        print("2. Filtrar por continente")
        print("3. Filtrar por rango de población")
        print("4. Filtrar por rango de superficie")
        print("5. Ordenar países")
        print("6. Mostrar estadísticas")
        print("7. Ver todos los registros")
        print("8. Salir")

        opción = input("Ingrese opción: ").strip()
        match opción:
            case '1':
                buscar_nombre_país()

```

```

    310     print("1. Buscar país por nombre (coincidencia parcial o exacta)")
    311     print("2. Filtrar por continente")
    312     print("3. Filtrar por rango de población")
    313     print("4. Filtrar por rango de superficie")
    314     print("5. Ordenar países")
    315     print("6. Mostrar estadísticas")
    316     print("7. Ver todos los registros")
    317     print("8. Salir")
    318
    319     opcion = input("Ingrese opción: ").strip()
    320
    321     match opcion:
    322         case '1':
    323             buscar_nombre_paises()
    324         case '2':
    325             filtrar_por_continente()
    326         case '3':
    327             filtrar_por_rango_poblacion()
    328         case '4':
    329             filtrar_por_rango_superficie()
    330         case '5':
    331             ordenar_paises()
    332         case '6':
    333             mostrar_estadisticas()
    334         case '7':
    335             ver.todos.los.registros()
    336         case '8':
    337             print("¡Gracias por utilizar nuestra aplicación! ¡Hasta pronto!")
    338             break
    339         case _:
    340             print("La opción seleccionada no es válida para nuestra aplicación")

```

País	Población	Superficie	Continente
Argentina	45376763	2780400	América
Japón	125800000	377975	Asia
Brasil	213993437	8515767	América
Alemania	83149380	357022	Europa
Australia	25925600	7692024	Oceania
Egipto	104258000	1002450	África
Canadá	38088000	9984679	América
India	1393409938	3287263	Asia
Sudáfrica	59308600	1219090	África
Francia	67413000	551695	Europa
Sudáfrica	59308600	1219090	África
Francia	67413000	551695	Europa
Francia	67413000	551695	Europa

*** Menú Principal ***
*** 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 población

11) Prueba final

Al realizar la prueba nos encontramos con las siguientes oportunidades de mejora del funcionamiento del programa para darle mayor fluidez:

- Mejora en la funcionalidad del programa al momento de utilizar la **opción 1** “Buscar país por nombre (coincidencia parcial o exacta)”. Por ejemplo, si ingreso en la búsqueda por nombre o continente, al colocar -Argentiña- ó –america- el resultado el “No se encontraron países que coincidan con la búsqueda”.

Explorador

EDTORES ABIERTOS

- tpi_gestion_datos_paises.py Trabajo_Práctico_Integrador
- Resolución Segundo Parcial_TUPaD_E.P...

UTN-TUPaD-P1

Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > mostrar_menu

```
304     print()
305
306     def mostrar_menu():
307         # Para mostrar constantemente el menú:
308         while True:
309             print("**** Menú Principal ****")
310             print("1. Buscar país por nombre (coincidencia parcial o exacta)")
311             print("2. Filtrar por continente")
312             print("3. Filtrar por rango de población")
313             print("4. Filtrar por rango de superficie")
314             print("5. Ordenar países")
315             print("6. Mostrar estadísticas")
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

8. Salir

Ingresse opción: 1

Ingresse el nombre del país o parte del nombre: argentina

No se encontraron países que coincidan con la búsqueda.

*** 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

Ingresse opción: 2

Continentes disponibles en el dataset:

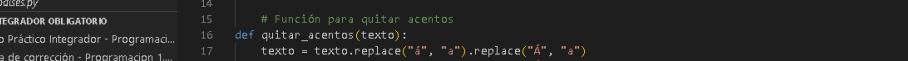
- América
- Asia
- Europa
- Oceania

Ingresse el continente o parte del nombre: america

No se encontraron países para ese continente.

Fernando Picco (Hace 19 horas) Lin. 31Q, Col. 73 (54 seleccionada) Espacios: 4 UTF-8 CRLF Python 3.12 (64-bit) Python 3.13 (64-bit) 27°C 12:20 26/11/2025

- Solución: se agregaron dos funciones auxiliares para quitar acentos y normalizar texto



```
# 3.2) Funciones auxiliares para quitar acentos y normalizar texto

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

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

```
Trabajo_Práctico_Integrador > tp1_gestion_datos_paises.py > buscar_nombre_paises
 61     def buscar_nombre_paises():
 62         paises = obtener_datos_paises() # Esta función se encargará de devolver todos los datos asociados
 63                                         # a cada país (nombre, población, continente, etc.)
 64
 65         termino = input("Ingrese el nombre del país o parte del nombre: ").strip()
 66         if termino == "":
 67             print("No ingresó ningún nombre en la búsqueda.")
 68             return
 69
 70         # validación mayúsculas y minúsculas
 71         termino_normalizado = normalizar(termino)
 72
 73         # lista para guardar los países que coinciden:
 74         coincidencia = []
 75
 76         for pais in paises:
 77             nombre_normalizado = normalizar([pais["nombre"]])
 78             # coincidencia normal o exacta
 79             if termino_normalizado in nombre_normalizado:
 80                 coincidencia.append(pais)
 81
 82
 83             if len(coincidencia) == 0:
 84                 print("No se encontraron países que coincidan con la búsqueda.")
 85             else:
 86                 print(f"Se encontraron {len(coincidencia)} país(es): ")
 87                 for pais in coincidencia:
 88                     mostrar_pais(pais)
 89             print()
 90
 91         # Opción 2
 92     def filtrar_por_continente():
 93         # cargar todos los países desde el archivo csv
 94         paises = obtener_datos_paises()
 95
 96         # obtener desde el dataset el conjunto de continentes disponibles
 97         continentes_disponibles = sorted({pais["continente"] for pais in paises})
 98
 99         print("Continentes disponibles en el dataset: ")
100         for cont in continentes_disponibles:
101             print("- " + {cont})
102         # Pedir a usuario el continente (o parte del nombre)
103         termino = input("Ingrese el continente o parte del nombre: ").strip()
104         if termino == "":
105             print("No ingresó ningún continente en la búsqueda.")
106             return
107
108         # validación mayúsculas y minúsculas
109         termino_normalizado = normalizar(termino)
110
111         # Buscar coincidencias
112         coincidencia = []
113         for pais in paises:
114             continente_normalizado = normalizar(pais["continente"])
115             if termino_normalizado in continente_normalizado:
116                 coincidencia.append(pais)
```

- Se realiza prueba de funcionamiento: ok

```

TRABAJO INTEGRADOR OBLIGATORIO
1. Trabajo Práctico Integrador - Programaci...
2. Rúbrica de corrección - Programación 1...
Apunte_TPI_P1.docx
aux_tpip.py
Borrador_Tpi_Paises.pdf
dataset_paises.csv
Presentación_TPI_P1_Gest_Datos_Paises.pptx
tpipaises.py
TutorialIntegrador Programación 1 2 C 202...

8. Salir
Ingrese opción: 1
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: 1
Ingrese el nombre del país o parte del nombre: Brasil
Se encontraron 1 país(es):
Brasil | Población: 213993437 | Superficie: 8515767 | 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: 1

```

- Mejorar el funcionamiento de la opción 5 “Ordenar países”. Al seleccionar alguno de los 3 criterios de ordenamiento y cómo (ascendente o descendente – A ó D-) al agregar otra letra distinta igual corre el código y toma orden ascendente.

```

TRABAJO INTEGRADOR OBLIGATORIO
1. Trabajo Práctico Integrador - Programaci...
2. Rúbrica de corrección - Programación 1...
Apunte_TPI_P1.docx
aux_tpip.py
Borrador_Tpi_Paises.pdf
dataset_paises.csv
Presentación_TPI_P1_Gest_Datos_Paises.pptx
tpipaises.py
TutorialIntegrador Programación 1 2 C 202...

*** Ordenar Países ***
1. Por nombre
2. Por población
3. Por superficie
Elija criterio (1, 2 ó 3): 1
Orden:
A- Ascendente (de menor a mayor superficie)
D- Descendente (de mayor a menor superficie)
Ingrese A ó D > : g
Países Ordenados:
Alemania | Población: 83149300 | Superficie: 357022 | Continente: Europa
Argentina | Población: 45376763 | Superficie: 2780400 | Continente: América
Australia | Población: 25925600 | Superficie: 7692024 | Continente: Oceanía
Brasil | Población: 213993437 | Superficie: 8515767 | Continente: América
Canadá | Población: 38808005 | Superficie: 9984679 | Continente: América
Egipto | Población: 104258000 | Superficie: 1002450 | Continente: África
Francia | Población: 67413000 | Superficie: 551695 | Continente: Europa
India | Población: 1393409938 | Superficie: 3287263 | Continente: Asia
Japón | Población: 125800000 | Superficie: 377975 | Continente: Asia
Sudáfrica | Población: 59308690 | Superficie: 1219090 | Continente: África

*** 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: 1

```

- Solución: validación de criterio y validación de A/D

```

294     # Opción 5
295     def ordenar_paises():
296         paises = obtener_datos_paises()
297         print("\n***Ordenar Paises***")
298         print("1. Por nombre")
299         print("2. Por población")
300         print("3. Por superficie")
301
302         # Validar criterio (1, 2, y 3)
303         criterio = input("Elija criterio (1, 2 ó 3): ").strip()
304         if criterio not in ("1", "2", "3"):
305             print("Criterio inválido. Volviendo al menú principal")
306             print()
307             return
308
309         # Validar orden (ascendente o descendente)
310         print("Orden: ")
311         print("A - Ascendente (de menor a mayor superficie)")
312         print("D - Descendente (de mayor a menor superficie)")
313
314         while True:
315             orden = input("Ingrese A ó D > : ").strip().casefold()
316             if orden in ("a", "d"):
317                 break
318             print("Opción inválida. Debe ingresar 'A' o 'D'.")
319
320         descendente = (orden == "d")
321
322         # Ordenar según criterio elegido
323         if criterio == "1":
324             paises_ordenados = sorted(paises, key=lambda p:p["nombre"].casefold(), reverse=descendente)
325         elif criterio == "2":
326             paises_ordenados = sorted(paises, key=lambda p:p["poblacion"], reverse=descendente)
327         else:
328             paises_ordenados = sorted(paises, key=lambda p:p["superficie"], reverse=descendente)
329
330         # Mostrar resultados
331         print("Paises Ordenados: ")
332         for pais in paises_ordenados:
333             mostrar_pais(pais)
334
335         print()

```

```

286     def ordenar_paises():
287         print("Opción inválida. Debe ingresar 'A' o 'D'.")
288
289         descendente = (orden == "d")
290
291         # Ordenar según criterio elegido
292         if criterio == "1":
293             paises_ordenados = sorted(paises, key=lambda p:p["nombre"].casefold(), reverse=descendente)
294         elif criterio == "2":
295             paises_ordenados = sorted(paises, key=lambda p:p["poblacion"], reverse=descendente)
296         else:
297             paises_ordenados = sorted(paises, key=lambda p:p["superficie"], reverse=descendente)
298
299         # Mostrar resultados
300         print("Paises Ordenados: ")
301         for pais in paises_ordenados:
302             mostrar_pais(pais)
303
304         print()

```

- Prueba: ok

```

286     def ordenar_paises():
287         print("Opción inválida. Debe ingresar 'A' o 'D'.")
288
289         descendente = (orden == "d")
290
291         # Ordenar según criterio elegido
292         if criterio == "1":
293             paises_ordenados = sorted(paises, key=lambda p:p["nombre"].casefold(), reverse=descendente)
294         elif criterio == "2":
295             paises_ordenados = sorted(paises, key=lambda p:p["poblacion"], reverse=descendente)
296         else:
297             paises_ordenados = sorted(paises, key=lambda p:p["superficie"], reverse=descendente)
298
299         # Mostrar resultados
300         print("Paises Ordenados: ")
301         for pais in paises_ordenados:
302             mostrar_pais(pais)
303
304         print()

```

Ingrese opción: 5

Ordenar Paises

1. Por nombre
2. Por población
3. Por superficie

Elija criterio (1, 2 ó 3): 2

Orden:

A - Ascendente (de menor a mayor superficie)
D - Descendente (de mayor a menor superficie)

Ingrese A ó D > : g

Opción inválida. Debe ingresar 'A' o 'D'.

Ingrese A ó D > : a

Paises Ordenados:

	Población	Superficie	Continente
Australia	25925000	7692024	Oceania
Canadá	38000005	9984678	América
Argentina	45376763	2780400	América
Sudáfrica	59308690	1219090	Africa
Francia	67413000	551695	Europa
Alemania	83149300	357022	Europa
Egipto	104258000	1002450	Africa
Japón	125800000	377975	Asia
Brasil	213993437	8515767	América
India	1393409038	3287263	Asia

*** Menú Principal ***

1. Buscar país por nombre (coincidencia parcial o exacta)
2. Filtrar por continente

*** Menú Principal ***

1. Buscar país por nombre (coincidencia parcial o exacta)
2. Filtrar por continente
1. Buscar país por nombre (coincidencia parcial o exacta)

12) Validación de archivo CSV

- Reemplazo:

```

    Archivo Editar Selección Ver Ir ...
    Explorador EDITORES ABIERTOS Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py M ...
    Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > obtener_datos_paises
    28     def quitar_acentos(texto):
    29         return texto
    30
    31     def normalizar(texto):
    32         # quita acentos y convierte a minúsculas
    33         return quitar_acentos(texto.casefold())
    34
    35     def obtener_datos_paises():
    36         paises = []
    37         # with open(nombre_archivo, newline="", encoding="utf-8") as archivo:
    38         #     lector = csv.DictReader(archivo) # La función DictReader(), devuelve un iterador devolviendo un
    39         #         # diccionario clave/valor a partir de la lectura de las líneas
    40         #         # del archivo csv.
    41         #         for fila in lector:
    42         #             paises.append({"nombre": fila["nombre"], "poblacion": float(fila["poblacion"]), "superficie": float(fila["superficie"])}
    43
    44     return paises
  
```

- Por:

```

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

13) validación de la función correspondiente a la opción 4 (“Filtrar por rango de superficie”)

```

    Explorador EDITORES ABIERTOS Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py M ...
    Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > ordenar_paises
    224
    225     return
    226     # Preguntar orden: ascendente o descendente
    227     print("¿Cómo desea ordenar los resultados?")
    228     print("A- Ascendente (de menor a mayor superficie)")
    229     print("D- Descendente (de mayor a menor superficie)")
    230     orden = input("Ingrese A ó D > ").strip().casefold()
    231
    232     if orden == "d":
    233         descendente = True
    234     else:
    235         descendente = False
  
```

- Error: seleccionando cualquier otra letra que no sea ni “A” ó “D”:

The screenshot shows a Python code editor with the following details:

- File Explorer:** Shows files like `tpi_gestion_datos_paises.py`, `dataset_paises.csv`, and `README.md`.
- Code Editor:** Displays the script `tpi_gestion_datos_paises.py` with a function `filtrar_por_rango_superficie()` that filters a list of countries based on their surface area.
- Terminal:** Shows the output of running the script, which includes:
 - Input: "Ingrese superficie mínima (km²): 8000000"
 - Input: "Ingrese superficie máxima (km²): 4000000"
 - Output: "La superficie máxima no puede ser menor que la mínima."
 - Output: "Intercambiando valores: mínima = 4000000, máxima = 8000000"
 - Output: "¿Cómo desea ordenar los resultados?"
 - Options: "A- Ascendente (de menor a mayor superficie)" and "D- Descendente (de mayor a menor superficie)"
 - Input: "Ingrese A ó D > : o"
 - Output: "Se encontraron 6 país(es) en ese rango:"
 - Table of results:

País	Población	Superficie	Continente
Francia	674138000	551695	Europa
Egipto	104258000	1002458	Africa
Sudáfrica	593086998	1219998	Africa
Argentina	45376763	2788400	América
India	13934490938	3287263	Asia
Australia	25925600	7692024	Oceania
 - Output: "Se encontraron 6 país(es) en ese rango:"
 - Table of results:

País	Población	Superficie	Continente
Francia	674138000	551695	Europa
Egipto	104258000	1002458	Africa
Sudáfrica	593086998	1219998	Africa
Argentina	45376763	2788400	América
India	13934490938	3287263	Asia
Australia	25925600	7692024	Oceania
- PowerShell/Terminal:** Shows the command `powershell` and `python`.

- Solución:

The screenshot shows a Python code editor with the following details:

- File Explorer:** Shows files like `tp1_gestion_datos_paises.py`, `dataset_paises.csv`, and `README.md`.
- Code Editor:** Displays a script for filtering countries based on their surface area.

```
tp1_gestion_datos_paises.py M x
Trabajo_Práctico_Integrador > tp1_gestion_datos_paises.py > filtrar_por_rango_superficie
199     def filtrar_por_rango_superficie():
200         # ...
201         if minimo <= superficie < maximo:
202             filtrados.append(pais)
203     if len(filtrados) == 0:
204         print("No se encontraron países en ese rango de superficie.")
205     print()
206
207     # Preguntar orden: ascendente o descendente
208     print("¿Cómo desea ordenar los resultados?")
209     print("A- Ascendente (de menor a mayor superficie)")
210     print("D- Descendente (de mayor a menor superficie)")
211
212     # Validación del orden (A / D)
213     while True:
214         orden = input("Ingrese A ó D >> ").strip().casefold()
215         if orden in ("a", "d"):
216             break
217         print("Opción inválida. Debe ingresar 'A' o 'D'.")
218
219     descendente = [orden == "d"]
220
221     # Ordenar por superficie
222     filtrados_ordenados = sorted(filtrados, key=lambda p:p["superficie"], reverse=descendente)
223
224     # Mostrar resultados
```

- Funcionamiento: ok

```
tp1_gestion_datos_paises.py M x
Trabajo_Práctico_Integrador > tp1_gestion_datos_paises.py > filtrar_por_rango_superficie
199     def filtrar_por_rango_superficie():
200         filtrados.append(pais)
201     if len(filtrados) == 0:
202         print("No se encontraron países en ese rango de superficie.")
203     print()

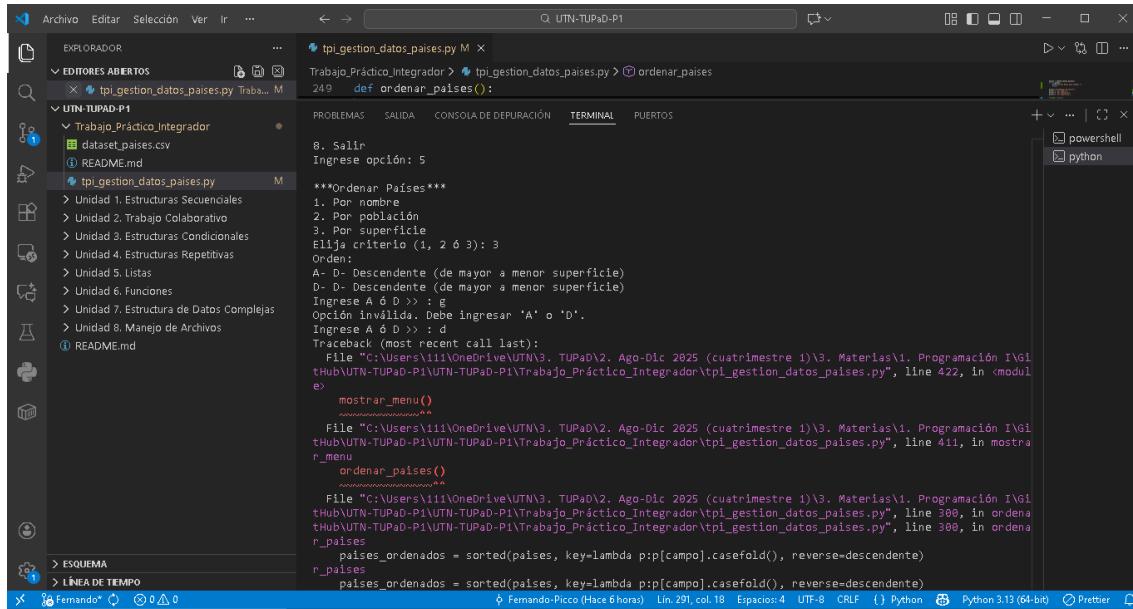
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
7. Ver todos los registros
8. Salir
Ingresé opción: 4

***Filtrar por rango de superficie***
Los valores se ingresan en km² (ej: 1000000)
Ingrese superficie mínima (km²): 4000000
Ingrese superficie máxima (km²): 4000000
La superficie máxima no puede ser menor que la mínima.
Intercambiando valores: mínima = 4000000, máxima = 4000000
¿Cómo desea ordenar los resultados?
A- Ascendente (de menor a mayor superficie)
D- Descendente (de mayor a menor superficie)
Ingrese A ó D > : i
Opción Inválida. Debe ingresar 'A' o 'D'.
Ingrese A ó D > : d
Se encontraron 5 países( ) en ese rango:
India | Población: 1399409938 | Superficie: 3287263 | Continente: Asia
Argentina | Población: 45376769 | Superficie: 2786488 | Continente: América
Sudáfrica | Población: 59386699 | Superficie: 1219099 | Continente: África
Egipto | Población: 184258866 | Superficie: 1002458 | Continente: África
Francia | Población: 67413000 | Superficie: 551695 | Continente: Europa

*** Menú Principal ***
1. Buscar país por nombre (coincidencia parcial o exacta)
2. Filtran por continente
3. Filtran por rango de población
4. Filtran por rango de superficie
```

14) validación de la función correspondiente a la opción 5 (“ordenar_paises”)

- Error:

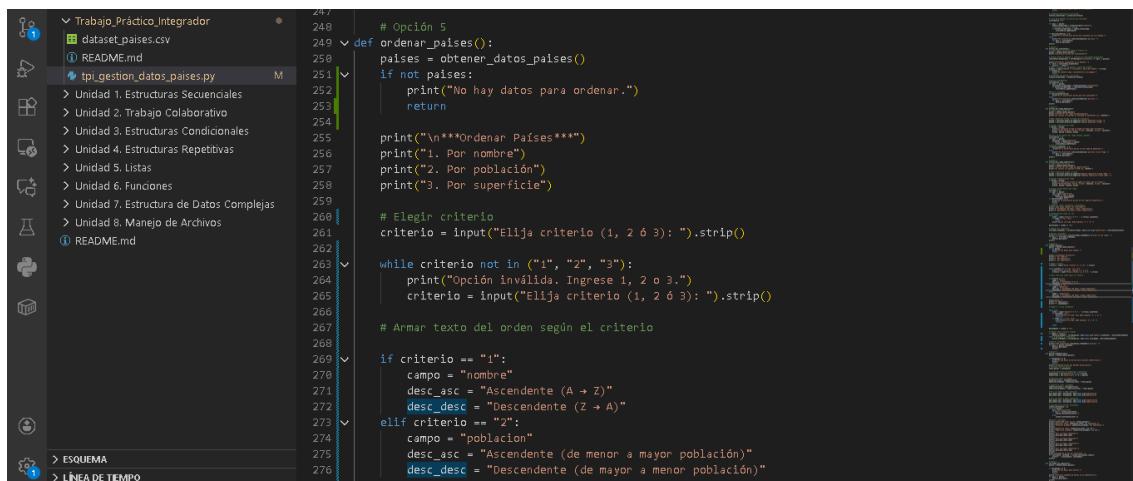


The screenshot shows the PyCharm IDE interface. The terminal window displays the following error message:

```
8. Salir
Ingresé opción: 5
***Ordenar Paises***
1. Por nombre
2. Por población
3. Por superficie
Elija criterio (1, 2 ó 3): 3
Orden:
A-D- Descendente (de mayor a menor superficie)
D-D- Descendente (de mayor a menor superficie)
Ingresé A ó D >: g
Opción inválida. Debe ingresar 'A' o 'D'.
Ingresé A ó D >: d
Ingresé opción: 5
Traceback (most recent call last):
  File "C:/Users/villii/OneDrive/UTN\3. TUPaD\2. Ago-Dic 2025 (cuatrimestre 1)\3. Materiales\1. Programación I\GitHub\UTN-TUPaD-P1\UTN-TUPaD-P1\Trabajo_Práctico_Integrador\tpi_gestion_datos_paises.py", line 422, in module
    mostrar_menu()
    ~~~~~~^~~~~~
  File "C:/Users/villii/OneDrive/UTN\3. TUPaD\2. Ago-Dic 2025 (cuatrimestre 1)\3. Materiales\1. Programación I\GitHub\UTN-TUPaD-P1\UTN-TUPaD-P1\Trabajo_Práctico_Integrador\tpi_gestion_datos_paises.py", line 411, in mostrar_menu
    ordenar_paises()
    ~~~~~~^~~~~~
  File "C:/Users/villii/OneDrive/UTN\3. TUPaD\2. Ago-Dic 2025 (cuatrimestre 1)\3. Materiales\1. Programación I\GitHub\UTN-TUPaD-P1\UTN-TUPaD-P1\Trabajo_Práctico_Integrador\tpi_gestion_datos_paises.py", line 380, in ordenar_paises
    paises_ordenados = sorted(paises, key=lambda p:[campo].casefold(), reverse=descendente)
    ~~~~~~^~~~~~
  File "C:/Users/villii/OneDrive/UTN\3. TUPaD\2. Ago-Dic 2025 (cuatrimestre 1)\3. Materiales\1. Programación I\GitHub\UTN-TUPaD-P1\UTN-TUPaD-P1\Trabajo_Práctico_Integrador\tpi_gestion_datos_paises.py", line 380, in ordenar_paises
    paises_ordenados = sorted(paises, key=lambda p:[campo].casefold(), reverse=descendente)
    ~~~~~~^~~~~~
```

The error occurs at line 422 of the file `tpi_gestion_datos_paises.py`, where the function `mostrar_menu()` is called. The error message indicates that the input 'g' is invalid because it must be either 'A' or 'D'. The stack trace shows that the error originates from the `ordenar_paises()` function, which is called from `mostrar_menu()`.

- Solución: corrección opción 5



The screenshot shows the PyCharm IDE interface with the corrected code for the `ordenar_paises()` function. The code now includes a validation step for the user input:

```
247 # Opción 5
248 def ordenar_paises():
249     paises = obtener_datos_paises()
250     if not paises:
251         print("No hay datos para ordenar.")
252         return
253
254     print("\n***Ordenar Paises***")
255     print("1. Por nombre")
256     print("2. Por población")
257     print("3. Por superficie")
258
259     # Elegir criterio
260     criterio = input("Elija criterio (1, 2 ó 3): ").strip()
261
262     while criterio not in ("1", "2", "3"):
263         print("Opción inválida. Ingrese 1, 2 ó 3.")
264         criterio = input("Elija criterio (1, 2 ó 3): ").strip()
265
266     # Armar texto del orden según el criterio
267
268     if criterio == "1":
269         campo = "nombre"
270         desc_asc = "Ascendente (A → Z)"
271         desc_desc = "Descendente (Z → A)"
272     elif criterio == "2":
273         campo = "poblacion"
274         desc_asc = "Ascendente (de menor a mayor población)"
275         desc_desc = "Descendente (de mayor a menor población)"
```

The changes made to the code include adding a validation loop for the user input, ensuring that only '1', '2', or '3' are accepted. This prevents the program from crashing when an invalid character like 'g' is entered.

The screenshot shows a Python code editor with the following details:

- Title Bar:** Archivo, Editar, Selección, Ver, Ir, ..., Q UTN-TUPa0-P1
- Left Sidebar:** Includes icons for file operations (New, Open, Save, Find, Replace, etc.), a search bar, and sections for "EDTORES ABIERTOS" (UTN-TUPa0-P1, tp_gestion_datos_paises.py), "ESQUEMA" (Línea de TIEMPO), and "Fernando" (with a profile icon).
- Code Editor Area:** The main window displays the content of `tpi_gestion_datos_paises.py`. The code defines a function `ordenar_paises()` that takes an order parameter ('A' or 'D') and a criterion ('1' for population or '2' for area). It prints the order and criterion, then enters a loop to validate the order input. If valid, it sets `descendente` based on the order and begins sorting the dataset. The dataset is defined as a list of dictionaries with keys 'nombre', 'poblacion', and 'superficie'.

```
def ordenar_paises():
    criterio = None
    campo = None
    desc_orden = None
    desc_desc = None

    if criterio == "1":
        campo = "poblacion"
        desc_orden = "Ascendente (de menor a mayor población)"
        desc_desc = "Descendente (de mayor a menor población)"
    else:
        campo = "superficie"
        desc_orden = "Ascendente (de menor a mayor superficie)"
        desc_desc = "Descendente (de mayor a menor superficie)"

    print("Orden: ")
    print(f"A: {desc_orden}")
    print(f"D: {desc_desc}")

    while True:
        orden = input("Ingrese A ó D >> : ").strip().casefold()
        if len(orden) != 1:
            print("Opción inválida. Solo debe ingresar 'A' o 'D'.")
            continue
        if orden not in ("a", "d"):
            print("Opción inválida. Debe ingresar 'A' o 'D'.")
            continue
        break

    descendente = (orden == "d")

    # Ordenar según criterio elegido
    if campo == "nombre":
        paises_ordenados = sorted(paises, key=lambda x: x["nombre"].casefold(), reverse=descendente)
```

```
tpi_gestion_datos_paises.py M x
Trabajo_Práctico_Integrador > tpi_gestion_datos_paises.py > ordenar_paises
249     def ordenar_paises():
250         ...
251         break
252
253         descendente = (orden == "d")
254
255         # Ordenar según criterio elegido
256         if campo == "nombre":
257             paises_ordenados = sorted(paises, key=lambda p:p["nombre"].casefold(), reverse=descendente)
258         else: # población y superficie
259             paises_ordenados = sorted(paises, key=lambda p:p[campo], reverse=descendente)
260
261         # Mostrar resultados
262         print(f"\nSe encontraron {len(paises_ordenados)} país(es): ")
263         for pais in paises_ordenados:
264             mostrar_pais(pais)
265             print()
```

- Prueba: ok

The screenshot shows a Windows desktop environment. In the center, there is a terminal window titled 'tpi_gestion_datos_paises.py' which displays a Python script for managing country data. The script includes functions for displaying all countries and showing a menu. The user has selected option 5, which lists countries by population. The terminal also shows an error message about invalid input for sorting criteria. Below the terminal, a powershell window is open with the command 'python'. On the left, there is a file explorer showing a project structure for 'UTN-TUPAD-P1' containing files like 'dataset_paises.csv', 'README.md', and several Python scripts. The taskbar at the bottom shows the application's name and some system icons.

Bibliografía

DataScientest. (s.f.). *Python If, Else: todo sobre las sentencias condicionales*. Obtenido de <https://datascientest.com/es/python-if-else>

El libro de Python. (s.f.). *Listas en Python*. Obtenido de <https://ellibrodepython.com/listas-en-python>