UTN TUPaD – Programación 1

Trabajo Integrador

“Gestor Informático de Países”

Docentes:

- Ana Mutti (Comisión 4)
- Martina Zabala (Comisión 10)

Alumnos:

- Franco Detarsio (Comisión 4)
- Agustín Pieve (Comisión 10)

1. Marco teórico

A lo largo del desarrollo del trabajo integrador recurrimos a diferentes estructuras y herramientas que nos brinda Python, a continuación, mencionaremos las más importantes junto con teoría de cada una de estas.

- **Listas**: una lista es una secuencia de valores que pueden ser de distintos tipos, inclusive si pertenecen a la misma lista, estos estarán entre corchetes y separados por coma [“valor”, 25, “v”], los valores dentro de la lista son llamados elementos, también podremos usa una lista dentro de otra lista, a esto le llamamos lista anidada.
  Una característica principal de las listas es que estas son mutables, nos permiten agregar varios valores a una lista vacía, o agregar nuevos a una lista con valores preexistentes. Otra característica fundamental es que podemos llamar a un elemento o modificarlo utilizando su índice, que será una expresión entera, cada elemento tiene un índice propio yendo de izquierda a derecha comenzando por el 0 y sumando +1 a este por cada posición avanzada.
  Para recorrer una lista podemos recurrir a un ciclo for, permitiéndonos realizar distintas funciones, como, por ejemplo, utilizar el iterador del ciclo como índice, pudiendo recorrer la lista o actuar sobre estos.
  Ejemplo:
  Lista = [1, 2, 3]

  ```
  for i in lista:
          print(lista[i])

  # esto devolvera
  1
  2
  3
  ```

- **Diccionarios**: Un diccionario es similar a una lista, la gran diferencia es que en un diccionario los índices pueden ser (casi) de cualquier tipo, a diferencia de la lista que solo admite enteros. Dentro del diccionario contamos con una colección de índices los cuales se denominan claves, y una colección de valores, cada clave está asociada a un valor único, generando una asociación que se denomina “clave-Valor” (“Key-Value”).

En general el orden de los valores en un diccionario es imprescindible, ya que estos no se indexan con índices enteros, en cambio se utiliza la clave para buscar el valor correspondiente.
Ejemplo:

```
diccionario = {“key1”: “value1”, “key2”: “value2”}
print(diccionario[“key1”]) # muestra “value1”
```

- **Funciones:** Una función es una secuencia de sentencias que realiza una computación y posee un nombre. Cuando defines una función, especificas el nombre y la secuencia de sentencias que se ejecutarán al “llamar” la función por su nombre. Puede recibir cero o más argumentos, que pueden ser utilizados durante la ejecución del cuerpo.
  Utilizar funciones nos permite dividir un programa largo en partes pequeñas y manejables, haciéndolo más fácil de leer, interpretar, buscar errores, etc. Logrando un programa modular, y con código reutilizable.
  Tenemos 2 tipos de funciones:

  - Funciones integradas y definidas: ya vienen construidas dentro del lenguaje o de algún modulo externo a mismo, solo hace falta llamarlas
  - Funciones definidas por el usuario: son construidas por el usuario ya que no vienen incorporadas en el lenguaje, estas son las que permiten una personalización del código y una modularizacion según se crea conveniente, permitiendo un bloque principal del programa más “limpio”, al evitar códigos largos o repetidos, cambiándolos por solo una línea de código.

    Ejemplo: Este ejemplo muestra como ingresando un numero como argumento, la función devolvería en este caso el valor duplicado.

    ```
    numero = 2
    def duplicar_num(numero):
        numero = numero * 2
        return: numero
    print(duplicar_num(numero)) # muestra 4
    ```

- **Condicionales:** Para que un programa sea útil necesitamos poder evaluar condiciones y así cambiar el comportamiento de este según corresponda, para estas situaciones recurrimos a condicionales. El condicional evaluará una expresión booleana (condición), si esta resulta verdadera, el programa ejecutará el bloque de código contenido dentro de esta con un correcto indentado, de ser falsa, saltea este bloque y actuara según tras estructuras que componen al mismo condicional, permitiendo mas de una rama posible, y a su vez evaluar otras condiciones si así se necesitara, las estructuras que permiten esto son:
    - if: condicional simple
    - elif: evalúa otra condición que asignemos
    - else: se ejecutará si ninguna condición se cumplió
    - vacío: si luego de if, elif no hay else, en caso de condición falsa, el código saltara el condicional y seguirá ejecutando lo que este debajo de este

- **Ordenamientos:** En nuestro caso utilizamos ordenamiento burbuja para ordenar países dentro del archivo.csv por nombre, población y superficie. Este es un método de ordenamiento simple que compara elementos de a pares y los intercambia si están en el orden incorrecto.
  En nuestro caso utilizamos bucles for anidados para comparar el elemento actual con el siguiente y definir si está en una posición correcta o hay que intercambiarlos.
  Ejemplo:

```
for i in range(len(lista) - 1):
        for j in range(i + 1, len(lista)):
            nombre_i = lista[i]["nombre"]
            nombre_j = lista[j]["nombre"]
```

  Una vez que ya tenemos ambos valores los comparamos y definimos si hacemos un intercambio entre ellos o no, mediante un condicional

```
if opcion == '1':  # ascendente
                if nombre_i > nombre_j:
                    lista[i], lista[j] = lista[j], lista[i]
```

- **Estadísticas básicas:** cuando hablamos de estadísticas básicas, nos referimos a medidas fundamentales que permiten describir y resumir un conjunto de datos numéricos**.** En este trabajo recurrimos a la media aritmética, que es la suma de los valores dividida entre el número de observaciones, comúnmente denominada promedio, es una medida de tendencia central de los datos.

- **Archivos CSV:** Sus iniciales hacen alusión a “comma separated values”, es un tipo de archivo de texto plano que almacena datos en forma tabular, donde: cada línea representa una fila y cada valor separado por comas, representa una columna. Es el formato más común de importación y exportación de hojas de cálculo y bases de datos, también permite manipular los datos contenidos fácilmente desde el código, sea agregando información al final, sobrescribiendo el archivo y extrayendo la información a través de la lectura de este.

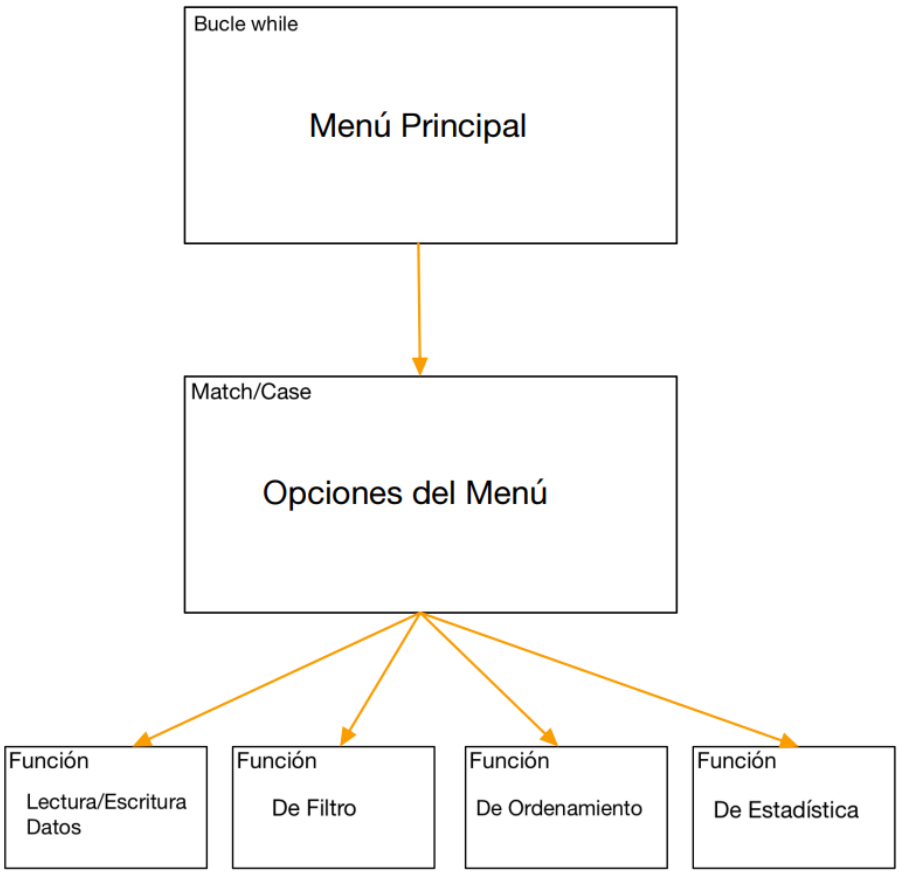
  En el caso de este trabajo, se utiliza el archivo “Base_de_Datos.csv” del cual obtenemos la información necesaria para que el programa pueda mostrar resultados, obtenemos esta información mediante lectura y asignación a una lista de diccionarios, lo que nos deja una estructura que facilita la manipulación de la información para distintos usos. También se da la opción de cargar nuevas filas o actualizar datos de una preexistente, esto permite mantener los datos ingresados por el usuario guardados en un archivo externo a la ejecución del programa, lo que nos deja con información permanente, bien almacenada y reutilizable.

2. Objetivo del trabajo

El objetivo de este trabajo es poner en practica todo lo aprendido a lo largo del cursado de la materia Programación – 1, incorporando el manejo de archivos utilizando un archivo .csv como base de datos y sostén principal de nuestro trabajo, así como aplicar todas las estructuras y herramientas aprendidas para poder dar forma al programa de una forma funcional, con código “limpio”, bien estructurado, con buena documentación, comentarios y modularización a través de funciones. A su vez busca fomentar el trabajo colaborativo y el desarrollo de habilidades blandas, como la exposición, la creación de contenido extra y la comunicación entre equipo.

3. Diseño del caso práctico

Flujo de operaciones principales en diagrama de flujo:



4. Metodología utilizada

Se decidió contar con un menú principal por el cual se usa un bucle while, el cual le da al usuario la posibilidad de decidir, que desea realizar con el programa de una forma interactiva y fácil de utilizar, se seguirá mostrando por pantalla hasta que el usuario decida salir utilizando la opción correspondiente. Según la opción numérica elegida por el usuario en el menú, por intermedio de una estructura match/case procedemos a ejecutar las funciones correspondientes a la opción elegida o solicitando al usuario que elija otra opción (por ejemplo: en el caso de los filtros si se desea que sea ascendente o descendente), si así se requiere y/o ingresar el nombre de un país/continente en cuanto a filtros o actualizaciones de información. El programa luego de validar la opción ingresada devolverá la información solicitada si se encuentra dentro del archivo.csv consumido, si el usuario ingreso mal la solicitud o no se encuentra lo que busca en la base de datos, se le notificará según el caso, así mismo cuando la información que desea actualizar sea correcta, se le confirmará el éxito de la operación.

5. Resultados obtenidos

Se logro una interfaz de usuario interactiva, de fácil uso y clara visualmente, el programa cuenta con entradas verificadas y notificación de error, lo que nos dio una correcta transición entre las entradas el procesamiento, la salida de los datos y una correcta persistencia de la información dentro del archivo.csv con técnicas de lectura y escrituras adecuadas.

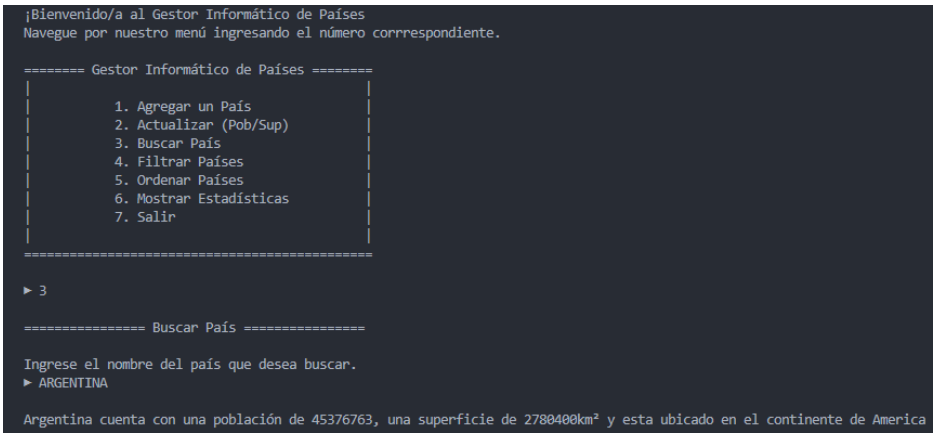
Nos presentó un mayor desafío la correcta lectura del Archivo.csv, y como pasarlo de manera correcta a una estructura que sea fácil de manipular para dar un correcto funcionamiento a las funciones, en este caso utilizamos una lista de diccionarios, lo que nos permitió acceder a los valores gracias a las claves del diccionario.

En un principio inicializábamos la lista con la cual tomaríamos los valores del archivo.csv al comienzo de la ejecución del programa, lo cual nos generaba un error de duplicación de datos ya que se asignaba nuevamente la función de lectura de datos luego de cada iteración, pudimos solucionarlo optando por inicializar la lista al comienzo de cada iteración del bucle while principal, lo que evita duplicación de datos, y nos permite trabajar siempre en base al archivo.csv actualizado.

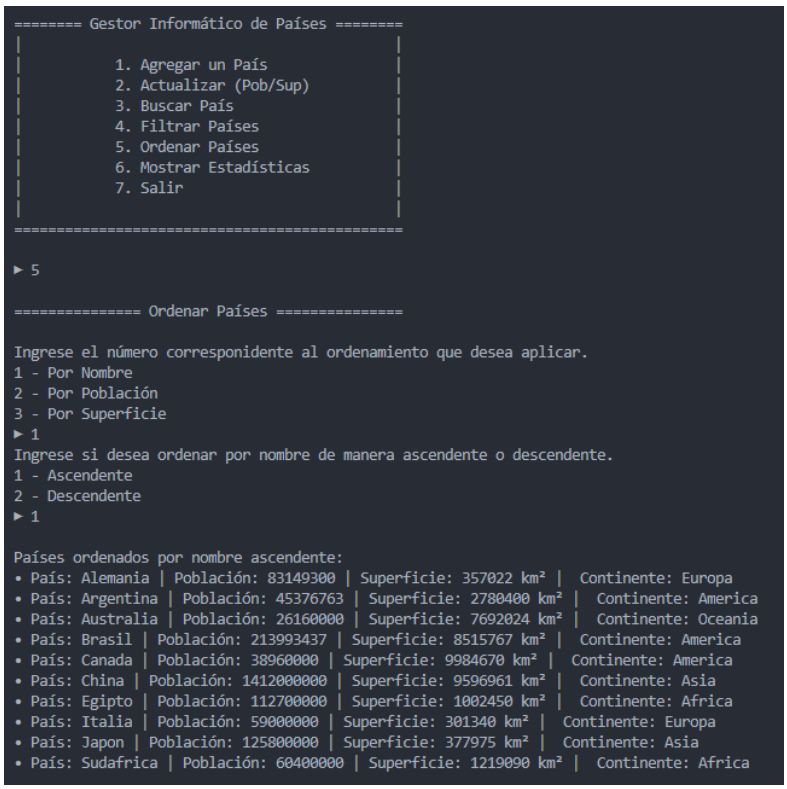
La comunicación fue clara desde un principio, nos pudimos distribuir correctamente las tareas, así como lograr un buen trabajo colaborativo a través de reuniones virtuales.

6. Capturas de pantalla de ejecución de ejemplos

Buscar país:



Ordenar países – Ascendente:



Mostrar estadísticas:

```
======== Gestor Informático de Países ========
|                                            |
|          1. Agregar un País                |
|          2. Actualizar (Pob/Sup)           |
|          3. Buscar País                    |
|          4. Filtrar Países                 |
|          5. Ordenar Países                 |
|          6. Mostrar Estadísticas           |
|          7. Salir                          |
|                                            |
==============================================

▶ 6

============ Mostrar Estadísticas ============

Ingrese el número correspondiente a la estadística que desea aplicar.
1 - País con mayor y menor población
2 - Promedio de población
3 - Promedio de superficie
4 - Cantidad de países por continente
▶ 1

Los países con Mayor y Menor Población son:

• Menor → País: Australia | Población: 26160000 | Superficie: 7692024km² | Continente: Oceania

• Mayor → País: China | Población: 1412000000 | Superficie: 9596961km² | Continente: Asia
```

Agregar un país:

```
======== Gestor Informático de Países ========
|                                            |
|          1. Agregar un País                |
|          2. Actualizar (Pob/Sup)           |
|          3. Buscar País                    |
|          4. Filtrar Países                 |
|          5. Ordenar Países                 |
|          6. Mostrar Estadísticas           |
|          7. Salir                          |
|                                            |
==============================================

▶ 1

============ 1. Agregar un País =============

Ingrese el nombre del país que desea agregar.
▶ NOrueGA
Ingrese la población correspondiente a Noruega
▶ 0
El valor ingresado para población no puede ser '0'
Ingrese el valor nuevamente
▶ 250000
Ingrese la superficie correspondiente a Noruega (km²)
▶ 500000
Ingrese el continente correspondiente a Noruega
▶ europa
País guardado Exitosamente
```

Base_de_Datos.csv

You, 4 minutes ago | 1 author (You)

```
1  nombre,poblacion,superficie,continente
2  Argentina,45376763,2780400,America
3  Japon,125800000,377975,Asia
4  Brasil,213993437,8515767,America
5  Alemania,83149300,357022,Europa
6  Canada,38960000,9984670,America
7  China,1412000000,9596961,Asia
8  Australia,26160000,7692024,Oceania
9  Egipto,112700000,1002450,Africa
10 Italia,59000000,301340,Europa
11 Sudafrica,60400000,1219090,Africa
12 Noruega,250000,500000,europa        You,
13
```

Actualizar (Pob/Sup):

```
======== Gestor Informático de Países ========
|                                            |
|           1. Agregar un País               |
|           2. Actualizar (Pob/Sup)          |
|           3. Buscar País                   |
|           4. Filtrar Países                |
|           5. Ordenar Países                |
|           6. Mostrar Estadísticas          |
|           7. Salir                         |
|                                            |
==============================================

▶ 2

========== 2. Actualizar (Pob/Sup) ==========

Ingrese el país que desearia actualizar su población y superficie.
▶           sudaFRiCa
Ingrese la población actualizada para Sudafrica
▶ -15
El valor ingresado no corresponde con un número entero
Ingrese nuevamente el valor deseado.
▶ 60055551
Ingrese la superficie actualizada para Sudafrica (km²)
▶ 1219999
Datos actualizados exitosamente.
```

Base_de_Datos.csv

You, 28 seconds ago | 1 author (You)

```
1  nombre,poblacion,superficie,continente
2  Argentina,45376763,2780400,America
3  Japon,125800000,377975,Asia
4  Brasil,213993437,8515767,America
5  Alemania,83149300,357022,Europa
6  Canada,38960000,9984670,America
7  China,1412000000,9596961,Asia
8  Australia,26160000,7692024,Oceania
9  Egipto,112700000,1002450,Africa
10 Italia,59000000,301340,Europa
11 Sudafrica,60055551,1219999,Africa
12 Noruega,250000,500000,europa
13
```

7. Conclusiones

Aprendimos a aplicar correctamente casi todos los conceptos aprendidos durante el cursado en un trabajo a escala media, logramos un código limpio, claro y modular con funciones declaradas manualmente, consumiendo y actualizando la información desde un archivo externo.

Estas herramientas nos parecen de suma importancia porque son la base de todo proyecto ya que combinan la entrada de datos, el correcto procesamiento y validación de ellos, la entrega de información al usuario, así como una correcta introducción al manejo de bases de datos.

A través de reuniones virtuales se analizo el proyecto y se definieron los puntos clave de este, por los cuales se fue avanzando de manera conjunta en un principio para luego dividir los puntos 5 y 6 pertenecientes al menú principal para un desarrollo mas veloz, para luego volver a validarlos en conjunto y hacer las correcciones finales.

Fuentes:

► Allen Downey, (2015) Pensar en Python – “Listas”

► Allen Downey, (2015) Pensar en Python – “Diccionarios”

► Python Software Foundation, (2025) Python docs. (Glossary: Function) https://docs.python.org/3/glossary.html#term-function

► Allen Downey, (2015) Pensar en Python – “Funciones”

► Cátedra Programación-1 U6 “notebook – Introducción a las funciones en Python”

► Allen Downey, (2015) Pensar en Python – “Condicionales y recursividad”

► Python Software Foundation, (2025) Python docs. https://docs.python.org/es/3.14/library/statistics.html#module-statistics

► Cátedra Programación-1 U8 “notebook – Enfoque.csv”

► Python Software Foundation, (2025) Python docs. https://docs.python.org/es/3.14/library/csv.html