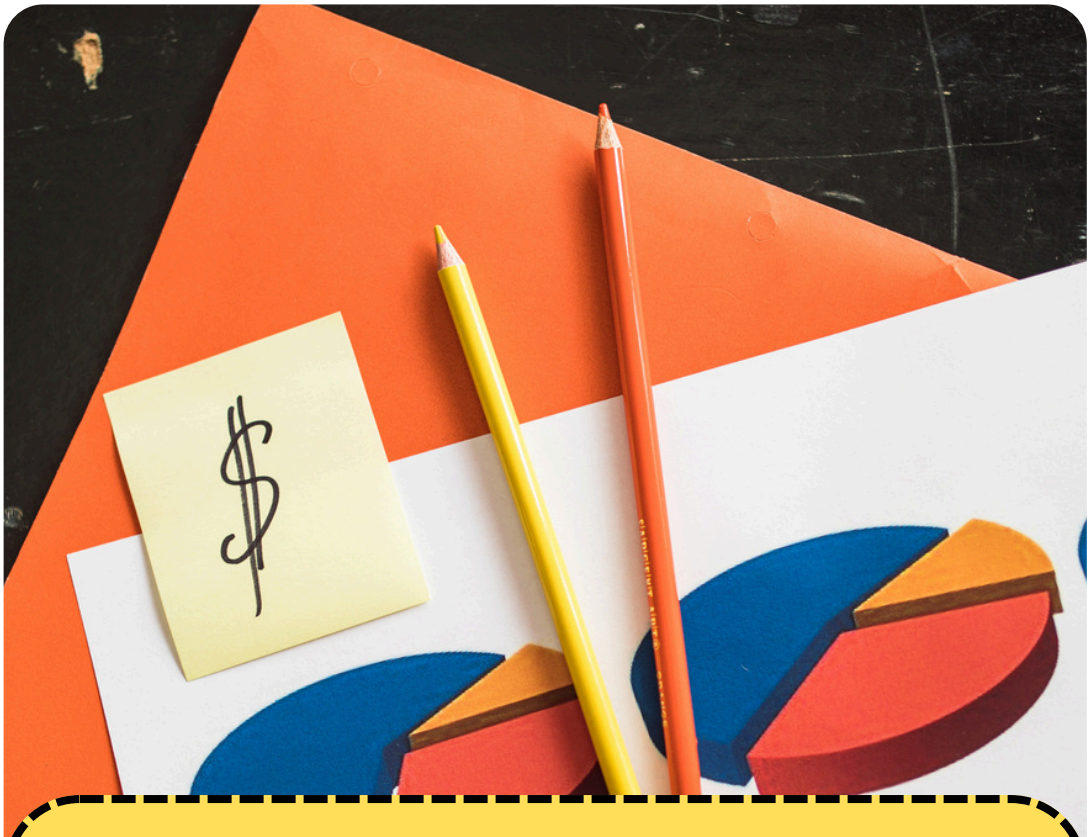
TIC

Lección 1: Herramientas de manipulación de datos tabulares.



Los datos tabulares coleccionan información en forma de tablas. Si bien se han venido empleando herramientas como Google sheets o Excel para visualizarlas y manipularlas, cuando se trata de analizar y procesar colecciones de información más grandes y de forma automática, se deben emplear herramientas más especializadas que permitan la gestión de bases de datos, el cálculo y la creación de información a partir de muchos archivos.

Una de las principales herramientas para Python que permite la manipulación de datos es Pandas. Esta biblioteca se basa en tres estructuras de datos que son:

Series: Estructuras de una dimensión. **DataFrame:** Estructura que modela las tablas de datos, en escencia es un arreglo en donde, cada columna es del tipo series. **Panel:** Son estructuras de tres dimensiones.

La biblioteca Pandas está basada en los tipos de datos y estructuras que emplean bibliotecas como numpy y como Matplotlib que permiten manipular arreglos numéricos y graficar. Por lo que muchas funcionalidades son las mismas, facilitando trabajar con la biblioteca.

Las series se asemejan a los arreglos (listas). Son elementos homogéneos, es decir, de un mismo tipo de dato y su tamaño es inmutable. Es decir, no se puede alterar. Sin embargo, su contenido si puede cambiar. Toda serie de pandas tiene un índice que permite acceder cada uno de los elementos de la serie y tiene un valor, que es el contenido como tal de la serie, por ejemplo, una serie que contenga las asignaturas de un curso puede verse como en la tabla 1.

Indice:	*A1*	*A2*	*A3*	*A4*
Valores:	Matemática	Ciencias	Estadística	Inglés

Las series se pueden crear a partir de listas o de diccionarios. En el caso de los diccionarios, si no se indican los índices, se tomarán las llaves del diccionario como los índices. Las series tienen algunos atributos útiles, como son:

Size: permite conocer el número de elementos de la serie
Index: entrega una lista con los nombres de las filas
Dtype: indica el tipo de datos de la serie.

Para acceder a un elemento de la serie se puede emplear la posición, por ejemplo, serie[i] o el ínice serie[indice]. Las series pueden entregar subconjuntos, por ejemplo, si en lugar de un único número o un único índice se envía una lista en el subíndice, la serie retornará los valores solicitados como una subserie.

Resumen descriptivo
de una serie
Las series incluyen métodos que permiten trabajar directamente con los valores. Por ejemplo:

s.Count() permite contar el número de elementos que no son nulos ni NaN en la serie s **s.sum()** : Devuelve la suma de los datos de la serie s cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str. s.cumsum() : Devuelve una serie con la suma acumulada de los datos de la serie s cuando los datos son de un tipo numérico. s.value_counts() : Devuelve una serie con la frecuencia (número de repeticiones) de cada valor de la serie s. s.min() : Devuelve el menor de los datos de la serie s. s.max() : Devuelve el mayor de los datos de la serie s. s.mean() : Devuelve la media de los datos de la serie s cuando los datos son de un tipo numérico. s.std() : Devuelve la desviación típica de los datos de la serie s cuando los datos son de un tipo numérico. s.describe(): Devuelve una serie con un resumen descriptivo que incluye el número de datos, su suma, el mínimo, el máximo, la media, la desviación típica y los cuartiles.

Las series también pueden sufrir cambios con operaciones binarios. Por ejemplo, si a una serie se le quiere sumar, restar, multiplicar o dividir por un número, se puede hacer con los signos **+, –, *, /.** Las series pueden ser alteradas con funciones. Para eso se puede emplear el método apply, el cual devuelve una serie como resultado de aplicar la función a cada uno de los elementos de la serie s. En este punto es donde se convierte en algo útil las funciones anónimas. Por ejemplo, para cambiar todos los textos de una serie de mayúsculas a minúsculas se puede aplicar como: **s.apply(lambda x: x.lower())**
en donde lambda permite definir una función anónima. Las series también cuentan con funciones como dropna que permiten eliminar los nulos de una serie, sort que permite ordenar los elementos de la serie según su valor y sort_index que permite ordenar por el índice.

df.info() : Devuelve información (número de filas, número de columnas, índices, tipo de las columnas y memoria usado) sobre el DataFrame df. **df.shape :** Devuelve una tupla con el número de filas y columnas del DataFrame df. **df.size :** Devuelve el número de elementos del DataFrame. **df.columns :** Devuelve una lista con los nombres de las columnas del DataFrame df. **df.index :** Devuelve una lista con los nombres de las filas del DataFrame df. **df.dtypes :** Devuelve una serie con los tipos de datos de las columnas del DataFrame df. **df.head(n)** : Devuelve las n primeras filas del DataFrame df. **df.tail(n)** : Devuelve las n últimas filas del DataFrame df. Para cambiar el nombre de las filas y las columnas de un DataFrame se utiliza el siguiente método: **df.rename(columns=columnas, index=filas):** Devuelve el DataFrame que resulta de renombrar las columnas indicadas en las claves del diccionario columnas con sus valores y las filas indicadas en las claves del diccionario filas con sus valores en el DataFrame df. El acceso a los datos de un DataFrame se puede hacer a través de posiciones o través de los nombres de las filas y columnas. **df.iloc[i, j] :** Devuelve el elemento que se encuentra en la fila i y la columna j del DataFrame df. Pueden indicarse secuencias de índices para obtener partes del DataFrame. **df.iloc[filas, columnas] :** Devuelve un DataFrame con los elementos de las filas de la lista filas y de las columnas de la lista columnas. **df.iloc[i] :** Devuelve una serie con los elementos de la fila i del DataFrame df. Acceso a los elementos mediante nombres: **df.loc[fila, columna] :** Devuelve el elemento que se encuentra en la fila con nombre fila y la columna de con nombre columna del DataFrame df. **df.loc[filas, columnas]** : Devuelve un DataFrame con los elemento que se encuentra en las filas con los nombres de la lista filas y las columnas con los nombres de la lista columnas del DataFrame df.

df[columna] : Devuelve una serie con los elementos de la columna de nombre columna del DataFrame df. df.columna : Devuelve una serie con los elementos de la columna de nombre columna del DataFrame df. Es similar al método anterior pero solo funciona cuando el nombre de la columna no tiene espacios en blanco. Al igual que la series, los dataframes pueden resumir la información contenida con métodos de descripción. Por ejemplo:

df.count() : Devuelve una serie número de elementos que no son nulos ni NaN en cada columna del DataFrame df. df.sum() : Devuelve una serie con la suma de los datos de las columnas del DataFrame df cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str. df.cumsum() : Devuelve un DataFrame con la suma acumulada de los datos de las columnas del DataFrame df cuando los datos son de un tipo numérico. df.min() : Devuelve una serie con los menores de los datos de las columnas del DataFrame df. df.max() : Devuelve una serie con los mayores de los datos de las columnas del DataFrame df. df.mean() : Devuelve una serie con las media de los datos de las columnas del DataFrame df cuando los datos son de un tipo numérico. df.std() : Devuelve una serie con las desviaciones típicas de los datos de las columnas del DataFrame df cuando los datos son de un tipo numérico. df.describe(include = tipo) : Devuelve un DataFrame con un resumen estadístico de las columnas del DataFrame df del tipo tipo. Para los datos numéricos (number) se calcula la media, la desviación típica, el mínimo, el máximo y los cuartiles de las columnas numéricas. Para los datos no numéricos (object) se calcula el número de valores, el número de valores distintos, la moda y su frecuencia. Si no se indica el tipo solo se consideran las columnas numéricas.