TELEDUC
Educación a Distancia

Herramientas avanzadas de programación en *Python* para procesamiento de datos

Resumen

Módulo 2

Importante: Para todos los códigos que se entregarán en este resumen, se asume que ya se ejecutó la siguiente línea de código:

```
import pandas as pd
```

Además, que existe un Data Frame creado de nombre df

En el módulo anterior, se estudiaron las operaciones básicas de un Data Frame. Específicamente, cómo crearlos. Además, vimos cómo operar con sus columnas (creación, edición, eliminación y visualización), y también cómo hacer filtros básicos en el Data Frame. Finalmente, vimos cómo guardar esto en un archivo CSV.

Características de las columnas:

- **Función *dtypes*:** Sirve para listar las columnas de un Data Frame, es decir, sus nombres y tipos. De forma general, se aplica la función de la siguiente manera:

```
df.dtypes
```

- **Renombrar una columna, función *rename*:** Permite cambiar el nombre de una columna, recibiendo como parámetro el nombre antiguo y el nuevo. Este cambio debe guardarse en la misma variable que representa al Data Frame, en caso contrario no quedará guardado. De forma general, uno puede aplicar la función de la siguiente manera:

```
df =
df.rename(columns={"nombre_antiguo_columna":"nombre
_nuevo_columna"})
```

Las columnas tienen distintos tipos, estos son:

Tipo de dato	Descripción	Comentario
Object	Es un string	Un texto
int64	Es un int	Un entero
float64	Es un float	Un decimal
Bool	*True* o *False*	Valor binario, base de operaciones lógicas
datetime64	Valores: fecha y tiempo	Día, hora, mes, etc.
timedelta[ns]	Diferencia entre valores de fecha y tiempo	En segundos
Category	Lista finita	Valores de texto

Se puede cambiar el tipo de dato de una columna mediante la función *astype*.

- **Cambiar el tipo de dato de una columna, función *astype*:** Permite cambiar el tipo de dato de una columna. Recibe como parámetro el nuevo tipo de dato al que se quiere cambiar como un *string*. Este cambio debe guardarse en la misma columna donde se está cambiando el tipo, en caso contrario no quedará guardado. Se aplica directamente sobre una columna.

```
df[nombre columna] = df[nombre columna].astype(tipo de
 dato al que se quiere cambiar el tipo de la columna)
```

Para caracterizar a un Data Frame:

- **Función *shape*:** Permite conocer el número de filas y columnas de un Data Frame. De forma general: df.shape
- **Función *size*:** Permite conocer la cantidad total de datos de un Data Frame. Es la multiplicación de la cantidad de filas y columnas. De forma general: df.size

Otra característica del Data Frame es el *index*. Al cargar un Data Frame e imprimirlo en consola, se observa que cada fila tiene asignado un número. Este es un identificador, y se denomina *index.*

- **Función set_index:** Permite definir una de las columnas como el index de un Data Frame. Recibe como parámetro el nombre del index. Este cambio debe guardarse en la misma variable que representa al Data Frame, en caso contrario no quedará guardado. De forma general:

```
df = df.set_index(nombre columna)
```

Importante: Si se cambia el *index* de un Data Frame (por ejemplo a una columna de tipo *object*), se puede aplicar la función de igual manera que como se enseñó en el módulo anterior. Es decir, es posible visualizar una fila individualmente, un conjunto de valores, y también hacer todo tipo de filtros.

Para visualizar información de un Data Frame, aprenderemos las siguientes funciones:

- **Función head:** Se ocupa para poder ver las primeras 10 filas de un Data Frame. De forma general:

```
df.head()
```

- **Función tail:** Se ocupa para poder ver las últimas 10 filas de un Data Frame. De forma general:

```
df.tail()
```

Es posible acceder a cualquier fila mediante su índice (posición dentro del Data Frame), independiente del *index* que este posea. Para eso ocuparemos la función *iloc*.

- **Función *iloc*:** Permite hacer un filtro en una fila o un intervalo de ellas (rango de índices) mediante su posición. De forma general:

```
df.iloc[índice de fila o rango de índices de fila]
```

Se pueden manejar datos faltantes (que no estén en el Data Frame), mediante la función *fillna*. En general *pandas* cargará estos datos con un NaN (*not a number*).

- **Función *fillna*:** Sirve para rellenar los valores NaN de un Data Frame. Este cambio debe guardarse en la misma variable que representa al Data Frame, en caso contrario no quedará guardado. De forma general:

```
df = df.fillna(valor con el que se quiere reemplazar los
                    valores NaN)
```

También se pueden agregar nuevas filas a un Data Frame.

- **Función *append*:** Sirve para agregar nuevas filas a un Data Frame. Estas filas deben tener las mismas columnas que el Data Frame original. Este cambio debe guardarse en la misma variable que representa al Data Frame, en caso contrario no quedará

guardado. De forma general, y para un Data Frame df2 que tiene las mismas columnas que df

```
df = df.append(df2)
```

Finalmente, es posible eliminar filas de un Data Frame.

- **Función *drop*:** Sirve para eliminar filas de un Data Frame. Se especifican estas filas mediante su *index*, accediendo a ellas mediante un intervalo especificado en la función *iloc*. Este cambio debe guardarse en la misma variable que representa al Data Frame, en caso contrario no quedará guardado. De forma general:

```
df= df.drop(df.iloc[filas o rango de filas].index)
```