INTRO A PANDAS

Librerías:
import pandas as pd

SERIES:
Estructura de datos unidimensional.
Contiene datos de **un solo tipo.**
Cada elemento en una Serie tiene una **etiqueta de índice** asociada. Los índices pueden ser **etiquetas personalizadas o valores numéricos** generados automáticamente.

Creación de Series:
Serie **vacía**: serie_vacia = pd.Series()
Serie a partir de **lista**: serie = pd.Series(lista)
Serie a partir de **diccionario**: serie = pd.Series(diccionario)
Serie con **índice personalizado**: serie = pd.Series(lista, index = [‘a’, ‘b’, ‘c’, ‘d’])

Propiedades de las series:
serie.**values**: devuelve los valores
serie.**index**: devuelve los índices
serie.**dtype**: tipo de datos int64, float64, object, datetime64
serie.**size**: número de elementos de la serie
serie.**shape**: forma de la serie → (n,) n es el número de elementos

Indexación en las series:
Por **posición**: serie[0] accede al primer elemento
Por la **etiqueta del índice**: serie[‘etiqueta’]
Por **rango**: serie [0:3] star:stop, devuelve del primer al tercer elemento
Por **lista de índices**: serie [[0, 2, 3]] devuelve el primer, tercer y cuarto elemento

DATAFRAMES:
Estructura de datos bidimensional.
Pueden contener **diferentes tipos de datos.**

Creación de DataFrames:
A partir de un **diccionario de listas**: cada key es una columna y cada elemento de su lista de values es un valor de esa columna. df = pd.DataFrame(diccionario)
A partir de una **lista de diccionarios**: las keys son las columnas, los values los elementos de esa columna. df = pd.DataFrame(lista_diccs)

Apertura de ficheros:
CSV:
pd.read_csv("../ruta/nombre_archivo.csv", sep = “;”, delimiter=None, header='infer', names=None, index_col=0, dtype=None)

- sep: si no devuelve DF especificar que están separadas por ;
- delimiter: lo mismo que sep
- header: encabezado de columna. None: sin nombres de columna, infer: nombres de columna del archivo, Número: indica el número de fila que será columna header=0.
- names: nombres de las columnas
- index_col: crear una columna índice
- dtype: para especificar el tipo de datos

Otros parámetros.

EXCEL:
pd.read_excel("ruta/archivo.xlsx", sheet_name=0, header=0, names=None, index_col=None, dtype=None)

- sheet_name: hoja del Excel que quieres leer. Leer varias hojas por su nombre =['Sheet1', 'Sheet2'].
- header: None: sin nombres de columna, 0: número de fila que será columna
- names: nombres de las columnas.
- index_col: qué columna será el índice. Un integer, nombre de columna o lista de columnas,
- dtype: para especificar el tipo de datos

JSON:
df = pd.read_json("nombre_archivo.json", orient=None, typ='frame', dtype=True, convert_axes=True, convert_dates=True, keep_default_dates=True, numpy=False, precise_float=False, date_unit=None, encoding=None, lines=False)

- orient: colums: por defecto, formato columna. Index: formato índice. Records: formato de registros. Split: formato dividido.. values: valores, sin etiqueta de columna o índice.
- Typ: frame: por defecto DataFrame. Series: crear un objeto Series
- dtype: para especificar el tipo de datos
- convert_axes: True indica si las etiquetas de los ejes deben convertirse en índices o nombres de columna.
- convert_dates: True indica si se deben convertir las cadenas de fecha y hora en objetos de fecha y hora.
- keep_default_dates: True mantener las fechas predeterminadas.
- numpy: indica si los datos deben devolverse como una matriz NumPy en lugar de un objeto DataFrame. Por defecto, es False.
- precise_float: indica si se deben utilizar números de punto float precisos en lugar de valores de punto float nativos de Python. Por defecto, es False.
- date_unit: especifica la unidad de fecha y hora si se deben convertir las cadenas de fecha y hora. Puede ser 's' para segundos o 'ms' para milisegundos.
- encoding: permite especificar la codificación del archivo JSON si no se puede inferir automáticamente.
- lines: indica si el archivo JSON contiene múltiples objetos JSON en líneas separadas en lugar de un solo objeto JSON.

PICKLE

Archivo binario que se usa para serializar y deserializar objetos. Cuando guardas objetos en un archivo pickle, se puede almacenar o enviar.

Pkl = pd.read_pickle("ruta/archivo.pkl", compression='infer')

- compression: por defecto infer, la biblioteca intentará inferir automáticamente el tipo de compresión.
  Puedes especificar un tipo de compresión explícitamente, como 'gzip' o 'bz2'

INDEXAXIÓN:

LOC, ILOC

Métodos para **acceder y manipular los datos** en un **DataFrame**.

LOC: df.loc[filas, columnas]

Se puede indicar una **etiqueta o** una **lista** de etiquetas. Ej: df.loc['Tues', 'Humidity']
Filas por **nombre del index, columna** por su **nombre**
Para ver **todos los valores** de filas o columnas sustituir por : → df.loc['Tues', :]
Para dar una **lista de valores** de filas o columnas → df.loc[['Mond', 'Tues'], :]
También se puede usar **start:stop:step**

ILOC: df.iloc[filas, columnas]
Para acceder utilizando **integers de fila o columna**, empiezan en 0. Ej: df.iloc[1, 3]
Para ver **todos los valores de filas o columnas** sustituir por : → df.iloc[1, :]
Para dar una **lista de valores de filas o columnas** → df.iloc[[1, 2, 3], 2]
También se puede usar **start:stop:step**

POR CONDICIÓN
Se puede indicar el **nombre de la columna** df['nombre_columna'] o lista columnas df['columna1', 'columna2']

Especificando una condición a la columna para que solo devuelva los valores que cumplen esa condición.
df1 = df.loc[df.Temperatura < 10, :] filtra para la columna Temperatura solo los valores menor de 10 y todas las demás columnas pero solo da las filas que coincidan que temperatura es menor de 10.
Con iloc se sustituye el nombre de la columna por su índice pero es necesario que sea en formato lista df.iloc[list(df[1] < 10), :]

Varias condiciones
LOC
df_dos_condiciones_loc = df.loc[(df.Wind > 20) & (df.Weather == 'Sunny'), ['Temperature', 'Wind']]

ILOC
df_dos_condiciones_iloc = df.iloc[list((df.Wind > 20) & (df.Weather == 'Sunny')), [1,2]]

CREAR COLUMNAS

Asignación directa:

df1 = df[‘nueva_columna’] = (df[‘columa_operacion’] * 12) Devuelve una nueva_columna cuyos valores son los de una columna ya existente *12

Método .assign()

df1 = df.assign(nueva_columna=df[‘columa_operacion’] * 12)

Método .insert()

df1 = df.insert(loc, column, value, allow_duplicates=False)

Ej: df.insert(0, "indice", range(1,8))

Inserta la columna índice en la posición 0, con un rango de números del 1 al 7.