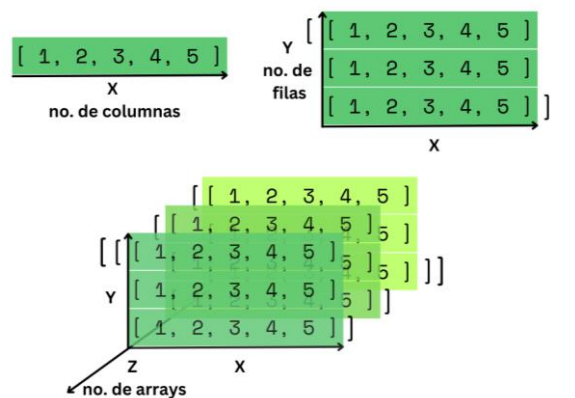


Python Cheat Sheet 3	DataFrames	DataFrames: carga de datos	Metodos de DataFrames	Filtrados de datos
Pandas	Crear DataFrames <code>df = pd.DataFrame(data, index, columns)</code> data : NumPy Array, diccionario, lista de diccionarios index : indice que por defecto se asigna como 0-(n-1), n siendo el número de filas; index = [lista] para asignar “etiquetas” (nombres de filas) column : nombre de las columnas; por defecto 0-(n-1); columns = [lista] para poner mas nombres	Carga de datos <code>df = pd.read_csv(“ruta/nombre_archivo.csv”)</code> crear un dataframe de un archivo de Comma Separated Values <code>df = pd.read_csv(“ruta/nombre_archivo”, sep= “;”) </code> crear un dataframe de un csv si el separador es ; <code>df = pd.read_csv(“ruta/nombre_archivo”, index_col= 0)</code> crear un dataframe de un csv si el archivo ya tiene una columna indice <code>df = pd.read_excel(“ruta/nombre_archivo.xlsx”)</code> crear un dataframe de un archivo de Excel - si sale “ ImportError:... openpyxl... ”, en el terminal: <code>pip3 install openpyxl</code> o <code>pip install openpyxl</code> <code>df = pd.read_json(“ruta/nombre_archivo.json”)</code> crear un dataframe de un archivo de JavaScript Object Notation (formato crudo) <code>df = df[‘data’].apply(pd.Series)</code> convertir el dataframe de json en un formato legible <code>df = pd.read_clipboard(sep=‘\t’)</code> crear un dataframe de datos en forma de dataframe en el clipboard; el separador podria ser \n ; , etc. Pickle : modulo que serializa objetos (convertir objetos complejos en una serie de bytes, en este caso en formato binario) para guardarlos en un archivo <code>with open(‘ruta/nombre_archivo.pkl’, ‘wb’) as f:</code> <code>pickle.dump(df,f)</code> pone los datos de un dataframe en el archivo.pkl <code>pd.read_pickle(‘ruta/nombre_archivo.csv’).head(n)</code> leer n filas y 5 columnas del archivo pickle <code>pd.read_parquet(‘ruta/nombre_archivo.parquet’)</code> leer un archivo parquet <code>pd.read_sas(‘ruta/nombre_archivo.sas7bdat’, format = ‘sas7bdat’)</code> leer un archivo SAS de formato SAS7BDAT <code>pd.read_spss(‘ruta/nombre_archivo.sav’)</code> leer un archivo SAS de formato SAS7BDAT Guardado de datos <code>df.to_csv(‘ruta/nombre_archivo.csv’)</code> guardar dataframe como archivo csv <code>df.to_excel(‘ruta/nombre_archivo.xlsx’)</code> guardar dataframe como archivo de Excel <code>df.to_json(‘ruta/nombre_archivo.json’)</code> guardar dataframe como archivo de JSON <code>df.to_parquet(‘ruta/nombre_archivo.parquet’)</code> guardar dataframe como archivo de parquet <code>df.to_pickle(‘ruta/nombre_archivo.pkl’)</code> guardar dataframe como archivo de pickle Librería PyDataset <code>pip install pydataset</code> o <code>pip3 install pydataset</code> from pydataset import data data() para ver los datasets listados en un dataframe por su id y titulo <code>df = data(‘nombre_dataset’)</code> guardar un dataset en un dataframe	Metodos para explorar un dataframe <code>df.shape</code> devuelve el número de filas y columnas <code>df.dtypes</code> devuelve el tipo de datos que hay en cada columna <code>df.columns</code> devuelve los nombres de las columnas <code>df.describe</code> devuelve un dataframe con un resumen de los principales estadísticos (media, mediana, desviación estándar etc.) de las columnas numéricas <code>df.describe(include = object)</code> devuelve un dataframe con un resumen de los principales estadísticosde las columnas con variables tipo string <code>df.info()</code> devuelve un resumen sobre el no. de columnas, nombres de columnas, numero de valores no nulos y los tipos de datos de las columnas <code>df[“nombre_columna”].unique()</code> o <code>df.nombre_columna.unique()</code> devuelve un array con los valores únicos de la columna <code>df[“nombre_columna”.value_counts()</code> o <code>df.nombre_columna.value_counts()</code> devuelve una serie con el recuento de valores únicos en orden descendente <code>df.isnull()</code> o <code>df.isna()</code> devuelve True o False según si cada valor es nulo o no <code>df.isnull().sum()</code> o <code>df.isna().sum()</code> devuelve el número de valores nulos por columnas <code>df.corr()</code> devuelve la correlación por pares de columnas, excluyendo valores NA/nulos <code>df.set_index([“nombre_columna”], inplace = True)</code> establece el indice utilizando uno o mas columnas; puede sustituir o ampliar un índice existente inplace = True los cambios sobreescriben sobre el df * cuando una columna se cambia a indice ya no es columna * <code>df.reset_index(inplace = True)</code> quitar una columna como indice para que vuelva a ser columna <code>df.rename(columns = {“nombre_columna”: “nombre_nueva”}, inplace = True)</code> cambia los nombres de una o mas columnas ejemplo de dict comprehension para crear diccionario sobre las columnas existentes de un dataframe: <code>diccionario = {col : col.upper() for col in df.columns}</code> <code>df.rename(columns = diccionario, inplace = True)</code> cambia los nombres de las columnas según el diccionario <code>df.drop([“columna1”, “columna2”], axis = b)</code> eliminar una o mas columnas o filas segun lo que especificamos axis = 1 columnas axis = 0 filas <code>df.rename(columns = diccionario, inplace = True)</code> cambia los nombres de las columnas según el diccionario <code>df[“columna_nueva”] = pd.cut(x=df[“nombre_columna”], bins=[n,m,l...])</code> separa los elementos de un dataframe en diferentes intervalos (n-m, m-l, etc); con este sintaxis se crea una columna nueva que indica en cual intervalo cae el valor <code>df.replace(to_replace = valor, value = valor_nuevo, inplace = True)</code> reemplaza cierto valor por otro que especificamos <code>df[“nombre_columna”.replace(to_replace = valor, value = valor_nuevo, inplace = True)</code> reemplaza cierto valor en una columna por otro que especificamos <code>df[“nombre_columna”] = df[“nombre_columna”] + x</code> reemplaza los valores de la columna por el valor + x (o otro valor que indicamos)	<code>pd.options.display.max_columns = None</code> ejecutar antes del df.head() para poder ver todas las columnas Filtrado por una columna con operadores de comparación variable_filtro = df[df[“nombre_columna”] == valor] extrae las filas donde el valor de la columna igual al valor dado * se puede usar con cualquier operador de comparación * Filtrado por multiples columnas con operadores logicos & and or ~ not variable_filtro = df[(df[“columna1”] == valor) & (df[“columna2”] == valor) & (df[“columna3”] > n valor)] extrae las filas donde los valores de las columnas cumplan las condiciones en parentesis variable_filtro = df[(df[“columna1”] == valor) (df[“columna1”] == valor) extrae las filas donde los valores de las columnas cumplan con una condición u otra variable_filtro = ~(df[df[“columna1”] == valor]) extrae las filas donde los valores de las columnas NO cumplan con la condición Metodos de pandas de filtrar variable_filtro = df[df[“nombre_columna”.isin(iterable)] extrae las filas cuyas valores de la columna nombrada están en el iterable (una lista, serie, dataframe o diccionario) variable_filtro = df[df[“nombre_columna”].str.contains (patron, regex = True)] extrae las filas cuyas valores de la columna nombrada contienen el patron de regex variable_filtro = df[df[“nombre_columna”.str.contains (“substring”, case = False, regex = False)] extrae las filas cuyas valores de la columna nombrada contienen el substring, no siendo case sensitive variable_filtro = df[df[“nombre_columna”.str.contains (“substring”, case = False, regex = False)] extrae las filas cuyas valores de la columna nombrada contienen el substring, no siendo case sensitive <code>df[pd.notnull(df[“nombre_columna”])]</code> devuelve las filas que no tiene valores nulos en la columna especificada Reemplazar valores basados en indices y condiciones: indices_filtrados = df.index[df[“columna”] == “valor”] for indice in indices_filtrados: df[“nombre_columna”.iloc[indice] = “valor_nuevo” Reemplazar valores basados en metodos NumPy: df[“nueva_columna”] = np.where(df[“nombre_columna”] > n, “categoria_if_true”, “categoria_if_false”) crea una nueva columna con los valores basados en una condición df[“nueva_columna”] = np.select(lista_de_condiciones, lista_de_opciones) crea una nueva columna con los valores basados en multiples condiciones
				

Python Cheat Sheet 4
NumPy (Numerical Python)
Crear arrays
Crear arrays con valores aleatorios <code>array = np.random.randint(inicio, final, forma_matriz)</code> crea un array de números aleatorios entre dos valores; <code>forma_matriz:</code> (z,y,x) z: número de arrays y: número de filas x: número de columnas <code>array = np.random.randint(inicio, final)</code> devuelve un número aleatorio en el rango <code>array = np.random.rand(z,y,x)</code> crea un array de floats aleatorias con la forma que le especificemos; por defecto genera números aleatorios entre 0-1 <code>array = np.random.random_sample((z,y,x))</code> crea un array de floats aleatorias con la forma que le especificemos; por defecto genera números aleatorios entre 0-0.9999999... <code>array = np.random.z,y,x=None</code> devuelve un número aleatorio en 0 y 0.9999999999999... <code>np.round(np.random.rand(z,y,x), n)</code> crear array con floats de n decimales
Crear arrays de listas <code>array = np.array(lista, dtype= tipo)</code> crea un array unidimensional de una lista <code>array = np.array([lista1, lista2])</code> crea un array bidimensional de dos listas <code>array = np.array([listadelistas1, listadelistas2])</code> crea un array bidimensional de dos listas
Crear otros tipos de arrays <code>array = np.arange(valor_inicio, valor_final, saltos)</code> crea un array usando el formato [start:stop:step] <code>array = np.ones(z,y,x)</code> crea un array de todo unos de la forma especificada <code>array2 = np.ones_like(array1)</code> crea un array de todo unos de la forma basada en otra array <code>array = np.zeros(z,y,x)</code> crea un array de todo zeros de la forma especificada <code>array2 = np.zeros_like(array1)</code> crea un array de todo zeros de la forma basada en otra array <code>array = np.empty((z,y,x), tipo)</code> crea un array vacío con datos por defecto tipo float <code>array2 = np.empty_like(array1)</code> crea un array vacía con la forma basada en otra array <code>array = np.eye(z,y,x, k = n)</code> crea un array con unos en diagonal empezando en la posición k <code>array = np.identity(x)</code> crea una matriz de identidad con ceros en filas y unos en la diagonal, de forma cuadrada
Operaciones con arrays <code>np.add(array1, array2)</code> suma dos arrays <code>np.subtract(array1, array2)</code> resta el array2 del array1 <code>np.multiply(array1, array2)</code> multiplica dos arrays <code>np.divide(array1, array2)</code> divide el array1 por el array2
Operaciones con escalares (un número) array + n n * array etc. - con cualquier operador algebraico

Indices, Subsets, Metodos de Arrays
Indices de arrays <code>array[i]</code> devuelve la índice i; las índices de los arrays unidimensionales funcionan igual que las listas <code>array[i, j]</code> o <code>array[i][j]</code> devuelve el elemento de la columna j de la fila i <code>array[:,n]</code> seleccionar todas las filas y las columnas hasta n-1 <code>array[h, i, j]</code> o <code>array[h][i][j]</code> devuelve el elemento de la columna j de la fila i del array h <code>array[h][i][j] = n</code> cambiar el valor del elemento en esta posición al valor n

Subsets <code>array > n</code> devuelve la forma del array con True o False según si el elemento cumple con la condición o no <code>array[array > n]</code> devuelve un subset: todos los valores que cumplen la condición en una lista dentro de un array <code>array[(array > n) & (array < m)]</code> devuelve un subset: todos los valores que cumplen las condiciones en una lista dentro de un array; se puede usar para “or”
Metodos de arrays <code>nuevo_array = array.copy()</code> crea una copia del array <code>np.transpose(array_bidimensional)</code> cambia las filas del array a columnas y las columnas a filas <code>np.transpose(array_multidimensional)</code> cambia el número de columnas al número de arrays y viceversa; el número de filas no cambia <code>np.transpose(array_multidimensional, (z,y,x))</code> hace la transposición según lo que especificemos usando las posiciones de la tupla (0,1,2) de la forma original <code>array = np.arange(n).reshape((y,x))</code> crea un array usando reshape para definir la forma <code>array = np.reshape(array, (z,y,x))</code> crea un array con los valores de otro array usando reshape para definir la forma <code>array = np.swapaxes(array, posicion, posicion)</code> intercambia dos ejes de una matriz usando las posiciones (z=0,y=1,x=2) de la forma original
Otras operaciones <code>np.sort(array)</code> devuelve un array con los valores de cada fila ordenados en orden ascendente por defecto <code>np.sort(array, axis = 0)</code> devuelve un array con los valores de cada columna ordenados en orden ascendente <code>np.sort(-array)</code> devuelve un array con los valores de cada fila ordenados en orden descendente <code>np.round(array, decimals = x)</code> devuelve un array con los valores del array redondeados a x decimales <code>np.round(array, decimals = x)</code> devuelve un array con los valores del array redondeados a x decimales <code>np.where(array > x)</code> devuelve los índices de los valores que cumplan la condición, por fila y columna

Operaciones estadísticas y matemáticas
Operaciones estadísticas y matemáticas El parametro axis en arrays bidimensionales: <code>axis = 0</code> columnas <code>axis = 1</code> filas - si especificamos el axis, la operación devuelve el resultado por cada fila o columna. Por ejemplo: <code>np.sum(array, axis = 0)</code> devuelve un array con la suma de cada fila El parametro axis en arrays multidimensionales: <code>axis = 0</code> dimensión <code>axis = 1</code> columnas <code>axis = 2</code> filas - si especificamos el axis, la operación devuelve el resultado por cada dimensión, fila o columna. Por ejemplo: <code>np.sum(array_3D, axis = 0)</code> devuelve un array de una matriz con la suma de todas las matrices <code>np.sum(array_3D, axis = 1)</code> devuelve un array donde las filas contienen las sumas de las columnas de cada matriz
Operaciones con parámetro del axis: <code>np.sum(array_3D)</code> devuelve la suma de todos los elementos de los matrices <code>np.mean(array)</code> devuelve la media de todo el array <code>np.std(array)</code> devuelve la desviación estándar de todo <code>np.var(array)</code> devuelve la varianza de valores de todo <code>np.min(array)</code> devuelve el valor mínimo del array <code>np.max(array)</code> devuelve el valor máximo del array <code>np.sum(array)</code> devuelve la suma de los elementos del array <code>np.cumsum(array)</code> devuelve un array con la suma acumulada de los elementos a lo largo del array <code>np.cumprod(array)</code> devuelve un array con la multiplicación acumulada de los elementos a lo largo del array
Operaciones sin parámetro del axis: <code>np.sqrt(array)</code> devuelve un array con la raíz cuadrada no negativa de cada elemento del array <code>np.exp(array)</code> devuelve un array con el exponencial de cada elemento del array <code>np.mod(array1, array2)</code> devuelve un array con el resto de la división entre dos arrays <code>np.mod(array1, n)</code> devuelve un array con el resto de la división entre el array y el valor de n <code>np.cos(array)</code> devuelve un array con el coseno de cada elemento del array <code>np.sin(array)</code> devuelve un array con el seno de cada elemento del array <code>np.sin(array)</code> devuelve un array con la tangente de cada elemento del array
Operaciones de comparación en arrays bidimensionales <code>np.any(array > n)</code> devuelve True o False según si cualquier valor del array cumple con la condición <code>np.any(array > n, axis = b)</code> devuelve un array con True o False por cada columna o fila según si algún valor de la fila o columna cumple con la condición <code>np.all(array > n)</code> devuelve True o False según si todos los valores del array cumplen con la condición <code>np.all(array > n, axis = b)</code> devuelve un array con True o False por cada columna o fila según si todos los valores de la fila o columna cumplen con la condición

Funciones de conjuntos
<code>np.unique(array)</code> devuelve un array con los valores únicos del array ordenados <code>np.unique(array, return_index=True)</code> devuelve un array con los valores únicos del array ordenados y un array con la posición de la primera instancia de cada valor <code>np.unique(array, return_inverse=True)</code> devuelve un array con los valores únicos del array ordenados y un array con las posiciones de cada elemento de cada valor <code>np.unique(array, return_counts=True)</code> devuelve un array con los valores únicos del array ordenados y un array con el número de veces que aparece cada valor <code>np.unique(array, axis = b)</code> devuelve un array con los valores únicos ordenados de las filas o columnas
Funciones para arrays unidimensionales <code>np.intersect1d(array1, array2)</code> devuelve un array con los valores únicos de los elementos en común de dos arrays <code>np.intersect1d(array1, array2, return_indices=True)</code> devuelve un array con los valores únicos de los elementos en común de dos arrays y arrays con los índices de cada valor, por array <code>np.union1d(array1, array2)</code> devuelve un array ordenado con los elementos resultantes de unir dos arrays (valores únicos) <code>np.in1d(array1, array2)</code> devuelve un array con True o False por cada elemento de array1 según si aparece el mismo valor en array2 <code>np.setdiff1d(array1, array2)</code> devuelve un array ordenado con los valores únicos que están en array1 pero no en array2 <code>np.setxor1d(array1, array2)</code> devuelve un array ordenado con los valores únicos que NO están en común de los dos arrays
Guardar y salvar arrays en .txt <code>np.savetxt('ruta/nombre_fichero.txt', array)</code> guardar un array de uno o dos dimensiones como .txt <code>variable = np.loadtxt('ruta/nombre_fichero.txt', dtype = tipo)</code> cargar datos de un archivo txt que tiene el mismo número de valores en cada fila
NumPy Random
<code>np.random.seed(x)</code> establece la semilla aleatoria del generador de números aleatorios, para que las funciones random que van después siempre cogerán los mismos valores “aleatorios” <code>np.random.uniform(n,m, size = (z,y,x))</code> genera muestras aleatorias de una distribución uniforme en el intervalo entre n y m <code>np.random.binomial(n,m, size = (z,y,x))</code> genera muestras con una distribución binomial; n es el número total de pruebas; m es la probabilidad de éxito <code>np.random.normal(loc = n, scale = m, size = (z,y,x))</code> genera números aleatorios de una distribución normal (curva de campana); loc es la media; scale es la desviación estándar <code>np.random.permutation(array)</code> devuelve un array con los mismos valores mezclados aleatoriamente