Python Cheat Sheet 3	DataFrames	DataFrames: carga de datos	Metodos de DataFrames	Filtrados de datos
Pandas	Crear DataFrames df = pd.DataFrame(data, index, columns) data: NumPy Array, diccionario, lista de diccionarios	Carga de datos df = pd.read_csv("ruta/nombre_archivo.csv") crear un dataframe de un archivo de Comma Separated Values	Metodos para explorar un dataframe df.shape devuelve el número de filas y columnas	<pre>pd.options.display.max_columns = None df.head() para poder ver todas las columnas</pre>
Series: estructuras en una dimension	index: indice que por defecto se asigna como 0-(n-1), n siendo el número de filas;	<pre>df = pd.read_csv("ruta/nombre_archivo", sep= ";") un dataframe de un csv si el separador es;</pre>	<pre>df.dtypes devuelve el tipo de datos que hay en cada columna</pre>	Filtrado por una columna con operadores de comparación variable filtro = df[df["nombre columna"] == valor] extrae
<pre>Crear series serie = pd.Series() crear serie vacía serie = pd.Series(array) crear serie a partir de un array con el indice por defecto</pre>	<pre>index = [lista] para asignar "etiquetas" (nombres de filas) column: nombre de las columnas; por defecto 0-(n-1); columns =[lista] para poner mas nombres</pre>	<pre>df = pd.read_csv("ruta/nombre_archivo", index_col= 0) crear un dataframe de un csv si el archivo ya tiene una columna indice df = pd.read excel("ruta/nombre archivo.xlsx") crear un</pre>	df.columns devuelve los nombres de las columnas df.describe devuelve un dataframe con un resumen de los principales estadísticos (media, mediana, desviación estándar etc.) de las columnas numéricas	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
<pre>serie = pd.Series(array, index = ['a', 'b', 'c']) crear una serie con indice definida; debe ser lista de la misma longitude del array serie = pd.Series(lista) crear una seria a partir de</pre>	<pre>df = pd.DataFrame(array) de un array con indices y columnas por defecto df = pd.DataFrame(diccionario) partir de un diccionario - los keys son los nombres</pre>	dataframe de un archivo de Excel - si sale "ImportError: openpyxl", en el terminal: pip3 install openpyxl o pip install openpyxl	<pre>df.describe(include = object) devuelve un dataframe con un resumen de los principales estadísticosde las columnas con variables tipo string df.info() devuelve un resumen sobre el no. de columnas,</pre>	& and ┃ or ~ not
una lista serie = pd.Series(número, indice) crear una serie a partir de un escalar con la longitude igual al número de indices	de las columnas Acceder a informacion de un DataFrame df.loc["etiqueta fila", "etiqueta columna"] devuelve	<pre>df = pd.read_json("ruta/nombre_archivo.json") crear un dataframe de un archivo de JavaScript Object Notation (formato crudo)</pre>	nombres de columnas, numero de valores no nulos y los tipos de datos de las columnas df["nombre_columna"].unique() o df.nombre_columna.unique() devuelve un array con los valores únicos de la columna	<pre>variable_filtro = df[(df["columna1"] == valor) & (df["columna2"] == valor) & (df["columna3"] > n valor)] extrae las filas donde los valores de las columnas cumplan las condiciónes en parentesis</pre>
serie = pd.Series(diccionario) crear una serie a partir de un diccionario Acceder a informacion de una serie	el contenido de un campo en una columna de una fila df.loc["etiqueta_fila",:] devuelve los valores de todas las columnas de una fila	<pre>df = df['data'].apply(pd.Series) convertir el dataframe de json en un formato legible df = pd.read_clipboard(sep='\t') crear un dataframe de datas an forma de dataframe en el climboard; el</pre>	<pre>df["nombre_columna"].value_counts() o df.nombre_columna.value_counts() devuelve una serie con el recuento de valores únicos en orden descendente</pre>	<pre>variable_filtro = df[(df["columna1"] == valor) (df["columna1"] == valor) extrae las filas donde los valores de las columnas cumplan con una condición u otra</pre>
<pre>serie.index devuelve los indices serie.values devuelve los valores serie.shape devuelve la forma (no. filas) serie.size devuelve el tamaño serie.dtypes devuelve el tipo de dato</pre>	<pre>df.loc[:,"etiqueta_columna"] devuelve los valores de todas las filas de una columna df.iloc[indice_fila, indice_columna] devuelve el contenido de un campo en una columna de una fila</pre>	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	<pre>df.isnull() o df.isna() devuelve True o False según si cada valor es nulo o no df.isnull().sum() o df.isna().sum() devuelve el número de valores nulos por columnas</pre>	<pre>variable_filtro = ~(df[df["columna1"] == valor]) extrae las filas donde los valores de las columnas NO cumplan con la condición</pre>
<pre>serie[i] devuelve el valor del elemento en indice i serie[[i,j]] devuelve el valor de los dos elementos serie[i:m] devuelve el valor de un rango</pre>	<pre>df.iloc[indice_fila, :] devuelve los valores de todas las columnas de una fila df.iloc[:,indice_columna] devuelve el contenido de un campo en una columna de una fila</pre>	formato binario) para guardarlos en un archivo with open('ruta/nombre_archivo.pkl', 'wb') as f: pickle.dump(df,f) pone los datos de un dataframe en el archivo pkl	<pre>df.corr() devuelve la correlación por pares de columnas, excluyendo valores NA/nulos df.set_index(["nombre_columna"], inplace = True) establece el índice utilizando uno o mas columnas; puede sustituir o ampliar un índice existente</pre>	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)
<pre>serie["etiqueta"] devuelve el valor de los elementos en indices i y j</pre>	<pre>df.loc[[lista_etiquetas_filas], [lista_etiquetas_columnas]] devuelve el contenido de varias filas / varias columnas df.loc[[lista indices filas],</pre>	<pre>pd.read_pickle('ruta/nombre_archivo.csv').head(n) leer n filas y 5 columnas del archivo pickle</pre>	inplace = True los cambios sobreescriben sobre el df * cuando una columna se cambia a índice ya no es columna *	<pre>variable_filtro = df[df["nombre_columna"].str.contains (patron, regex = True)] extrae las filas cuyas valores de la columna nombrada contenienen el patron de regex</pre>
Operaciones con series serie1 +-*/ serie2 suma/resta/multiplica/divide las	<pre>[lista_indices_columnas]] devuelve el contenido de varias filas / varias columnas</pre>	<pre>pd.read_parquet('ruta/nombre_archivo.parquet') leer un archivo parquet</pre>	<pre>df.reset_index(inplace = True) quitar una columna como indice para que vuelva a ser columna</pre>	<pre>variable_filtro = df[df["nombre_columna"].str.contains ("substring", case = False, regex = False)] extrae las filas</pre>
<pre>filas con indices comunes entre las dos series serie1.add(serie2, fill_value = número) suma las filas con indices comunes, y suma el fill value a los</pre>	<pre>- se puede usar los indices/rangos de las listas [start:stop:step] dentro de los loc/iloc df.loc[df.etiqueta > x] seleccionar datos basado en</pre>	<pre>pd.read_sas('ruta/nombre_archivo.sas7bdat', format = 'sas7bdat') leer un archivo SAS de formato SAS7BDAT</pre>	<pre>df.rename(columns = {"nombre_columna": "nombre_nueva"}, inplace = True) cambia los nombres de una o mas columnas</pre>	cuyas valores de la columna nombrada contienen el substring, no siendo case sensitive
<pre>valores sin indice comun serie1.sub(serie2, fill_value = número) restan las filas de la seria2 de la serie1 cuando tienen indices comunes, y resta el fill value de las otras indices de serie1</pre>	una condición usando operadores comparativos df.loc[(df.etiqueta > x) & (df.etiqueta == y)] seleccionar datos que tienen que cumplir las dos condiciónes (and)	<pre>pd.read_spss('ruta/nombre_archivo.sav' leer un archivo SAS de formato SAS7BDAT Guardado de datos</pre>	ejemplo de dict comprehension para crear diccionario sobre las columnas existentes de un dataframe: diccionario = {col : col.upper() for col in df.columns}	<pre>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</pre>
<pre>serie1.mul(serie2, fill_value = número) multiplica las filas con indices comunes y multiplica el fill value con las otras *usar 1 para conservar el valor* serie1.mul(serie2, fill value = número) divida las</pre>	<pre>df.loc[(df.etiqueta > x) (df.etiqueta == y)] seleccionar datos que tienen que deben cumplir una de las dos condiciones (or) df.iloc[list(df.etiqueta > x), :] iloc no acepta una</pre>	<pre>df.to_csv('ruta/nombre_archivo.csv') como archivo csv df.to_excel('ruta/nombre_archivo.xlsx') dataframe como archivo de Excel</pre>	<pre>df.rename(columns = diccionario, inplace = True) cambia los nombres de las columnas según el diccionario df.drop(["columna1", "columna2"], axis = b) eliminar una o mas columnas o filas segun lo que especificamos</pre>	<pre>df[pd.notnull(df["nombre_columna"])] devuelve las filas que no tiene valores nulos en la columna especificada</pre>
filas de la serie1 entre las de la serie2 cuando tienen indices comunes, y divide las otras por el fill value serie1.mod(serie2, fill_value = número) devuelve el	Serie booleana; hay que convertirla en lista variable_df.head(n) devuelve las n primeras filas del df, o 5 por defecto	<pre>df.to_json('ruta/nombre_archivo.json') guardar dataframe como archivo de JSON df.to_parquet('ruta/nombre_archivo.parquet') guardar dataframe como archivo de parquet</pre>	<pre>axis = 1 columnas axis = 0 filas df.rename(columns = diccionario, inplace = True) cambia los nombres de las columnas según el diccionario</pre>	Reemplazar valores basados en indices y condiciones: indices_filtrados = df.index[df["columna"] == "valor"] for indice in indices_filtrados:
<pre>modulo (division sin resta) serie1.pow(serie2, fill_value = número) exponencial serie1.ge(serie2) compara si serie1 es mayor que</pre>	Crear columnas df["nueva_columna"] = (df["etiqueta_columna"] + x) crea una nueva columna basada en otra	df.to_pickle('ruta/nombre_archivo.pkl') guardar dataframe como archivo de pickle Librería PyDataset	<pre>df["columna_nueva"] = pd.cut(x=df["nombre_columna"], bins=[n,m,1]) 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</pre>	<pre>df["nombre_columna"].iloc[indice] = "valor_nuevo" Reemplazar valores basados en metodos NumPy:</pre>
<pre>serie2 y devuelve True o False serie1.le(serie2) serie2 y devuelve True o False</pre>	<pre>df = df.assign(nueva_columna= df["etiqueta_columna] + x) crea una nueva basada en otra df = df.assign(nueva_columna= [lista_valores]) crea una nueva columna de una lista de valores *tiene que</pre>	<pre>pip install pydataset o pip3 install pydataset from pydataset import data data() para ver los datasets listados en un dataframe por su id y título</pre>	<pre>intervalo cae el valor df.replace(to_replace = valor, value = valor_nuevo, inplace = True) reemplaza cierto valor por otro que</pre>	<pre>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</pre>
<pre>Filtrado booleanos serie < > >= <= == valor si cada condición cumple la condición serie1[serie1 < > >= <= == valor] devuelve solo los</pre>	ser de la misma longitud como el número de filas del dataframe* df.insert(indice_nueva_columna, "nombre_columna", valores) crea una nueva columna en la indice indicada	<pre>df = data('nombre_dataset') guardar un dataset en un dataframe Metodos para explorar un dataframe</pre>	especificamos df["nombre_columna"].replace(to_replace = valor, value = valor_nuevo, inplace = True) reemplaza cierto valor en una columna por otro que especificamos	<pre>df["nueva_columna"] = np.select(lista_de_condiciones, lista_de_opciones) crea una nueva columna con los valores basados en multiples condiciones</pre>
valores que cumplen la condición np.nan crear valor nulo (NaN) serie.isnull() devuelve True o False segun si los valores existen o son nulos ("" no cuenta como nulo)	allow_duplicates = True parametro cuando queremos permitir columnas duplicadas (por defecto es False) Eliminar columnas	<pre>df.head(n) devuelve las primeras n lineas del dataframe, o por defecto 5 df.tail(n) devuelve las últimas n lineas del dataframe, o por defecto 5</pre>	<pre>df["nombre_columna"] = df["nombre_columna"] + x reemplaza los valores de la columna por el valor + x (o otro valor que indicamos)</pre>	
<pre>serie.notnull() devuelve True o False segun si los valores existen o son nulos ("" no cuenta como nulo)</pre>	<pre>df = df.drop(columns = ["column1", "column2"]) eliminar columnas</pre>	<pre>df.sample(n) devuelve n filas aleatorias de nuestro dataframe, o uno por defecto</pre>		

Python Cheat Sheet 4

Indices, Subsets, Metodos de Arrays

NumPv (Numerical Pvthon)

Crear arrays con valores aleatorios

Crear arrays

array = np.random.randint(inicio, final, forma matriz) crea un array de números aleatorios entre dos valores; forma matriz: (z,y,x) z: número de arrays y: número de filas x: número de columnas array = np.random.randint(inicio, final) devuelve un número aleatorio en el rango $\frac{1}{2}$ = $\frac{1}{2}$ np.random.rand(z,y,x) crea un array de floats aleatorias con la forma que le especificemos;

por defecto genera números aleatorios entre 0-1 array = np.random.random sample((z,y,x)) crea un array de floats aleatorias con la forma que le especificemos; por defecto genera números aleatorios entre 0-0.9999999... array = np.random.z,y,x=None) devuelve un número

np.round(np.random.rand(z,y,x), n) crear array con floats de n decimales

Crear arrays de listas

array = np.array(lista, dtype= tipo) crea un array unidimensional de una lista array = np.array([lista1, lista2]) crea un array bidimensional de dos listas array = np.array([listadelistas1, listadelistas2]) crea un array bidimensional de dos listas

Crear otros tipos de arrays

array = np.arange(valor_inicio, valor_final, saltos) crea un array usando el formato [start:stop:step] array = np.ones(z,y,x) crea un array de todo unos de la forma especificada

array2 = np.ones_like(array1) crea un array de todo unos de la forma basada en otra array array = np.zeros(z,y,x) crea un array de todo zeros

de la forma especificada array2 = np.zeros like(array1) crea un array de todo zeros de la forma basada en otra array array = np.empty((z,y,x), tipo) crea un array vacio

con datos por defecto tipo float array2 = np.empty like(array1) crea un array vacia con la forma basada en otra array

array = np.eye(z,y,x, k = n) crea un array con unos en diagonal empezando en la posicion k

array = np.identity(x) crea una matriz de identidad con ceros en filas y unos en la diagonal, de forma cuadrada

Operaciones con arrays

np.add(array1, array2) suma dos arrays np.subtract(array1, array2) resta el array2 del np.multiply(array1, array2) multiplica dos arrays

np.divide(array1, array2) divide el array1 por el

Operaciones con escalares (un número)

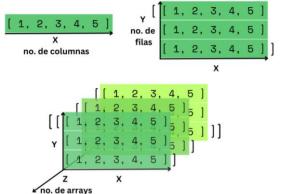
array + n n * array etc. - con cualquier operador algebraico

Indices de arrays

array[i] devuelve la indice i; las indices de los arrays unidimensionales funcionan igual que las listas arrav[i, i] o arrav[i][i] devuelve el elemento de la columna j de la fila i

array[:,:n] seleccionar todas las filas y las columnas hasta n-1

array[h, i, j] o array[h][i][j] devuelve el elemento de la columna j de la fila i del array h array[h][i][j] = n cambiar el valor del elemento en esta posicion al valor n



Subsets

array > n devuelve la forma del array con True o False según si el elemento cumple con la condición o no array[array > n] devuelve un subset: todos los valores que cumplen la condición en una lista dentro de un

array[(array > n) & (array < m)] devuelve un subset:</pre> todos los valores que cumplen las condiciones en una lista dentro de un array; se puede usar | para "or"

Metodos de arrays

nuevo array = array.copy() crea un a copia del array np.transpose(array bidimensional) cambia los filas del array a columnas y las columnas a filas

np.transpose(array multidimensional) cambia el número de columnas al número de arrays y viceversa; el número de filas no cambia

np.transpose(array multidimensional, (z,y,x)) hace la transposicion segun lo que especificemos usando las posiciones de la tupla (0,1,2) de la forma original array = np.arange(n).reshape((y,x)) crea un array usando reshape para definir la forma

array = np.reshape(array, (z,y,x)) crea un array con los valores de otro array usando reshape para definir

array = np.swapaxes(array, posicion, posicion) intercambia dos ejes de una matriz usando las posiciones (z=0,y=1,x=2) de la forma original

Otras operaciones

np.sort(array) devuelve un array con los valores de cada fila ordenados en orden ascendente por defecto np.sort(array, axis = 0) devuelve un array con los valores de cada columna ordenados en orden ascendente np.sort(-array) devuelve un array con los valores de cada fila ordenados en orden descendente np.round(array, decimals = x) devuelve un array con los valores del array redondeados a x decimales np.round(array, decimals = x) devuelve un array con los valores del array redondeados a x decimales np.where(array > x) devuelve los indices de los valores que cumplan la condición, por fila y columna

Operaciones estadísticas y matemáticas

El parametro axis en arrays bidimensionales:

axis = 0 columnas

axis = 1 filas - si especificamos el axis, la operación devuelve el resultado por cada fila o columna.

Operaciones estadísticas y matemáticas

np.sum(array, axis = 0) devuelve un array con la suma de cada fila

El parametro axis en arrays multidimensionales:

axis = 0 dimensión axis = 1 columnas axis = 2 filas

- si especificamos el axis, la operación devuelve el resultado por cada dimensión, fila o columna. Por eiemplo:

np.sum(array 3D, axis = 0) devuelve un array de una matriz con la suma de todas las matrices np.sum(array 3D, axis = 1) devuelve un array donde las filas contienen las sumas de las columnas de cada matriz

Operaciones con parámetro del axis:

np.sum(array 3D) devuelve la suma de todos los elementos de los matrices np.mean(array) devuelve la media de todo el array np.std(array) devuelve la desviación estándar de todo np.var(array) devuelve la varianza de valores de todo np.min(array) devuelve el valor mínimo del array np.max(array) devuelve el valor máximo del array

np.sum(array) devuelve la suma de los elementos del

np.cumsum(array) devuelve un array con la suma acumulada de los elementos a lo largo del array np.cumprod(array) devuelve un array con la multiplicación acumulada de los elementos a lo largo del arrav

Operaciones sin parámetro del axis:

np.sqrt(array) devuelve un array con la raíz cuadrada no negativa de cada elemento del array np.exp(array) devuelve un array con el exponencial de cada elemento del array np.mod(array1, array2) devuelve un array con el resto de la división entre dos arrays np.mod(array1, n) devuelve un array con el resto de la división entre el array y el valor de n np.cos(array) devuelve un array con el coseno de cada elemento del array np.sin(array) devuelve un array con el seno de cada elemento del arrav

np.sin(array) devuelve un array con la tangente de

Operaciones de comparación en arrays bidimensionales

cada elemento del array

np.any(array > n) devuelve True o False segun si cualquier valor del array cumpla con la condicion np.any(array > n, axis = b) devuelve un array con True o False por cada columna o fila según si algún valor de la fila o columna cumpla con la condición np.all(array > n) devuelve True o False segun si todos los valores del array cumpla con la condicion np.all(array > n, axis = b) devuelve un array con True o False por cada columna o fila según si todos los valores de la fila o columna cumplan con la condición

np.unique(array) devuelve un array con los valores

Funciones de conjuntos

únicos del array ordenados np.unique(array, return index=True) devuelve un array con los valores únicos del array ordenados y un array con la posición de la primera instancia de cada valor

np.unique(array, return inverse=True) devuelve un array con los valores únicos del array ordenados y un array con las posiciones de cada elemento de cada

np.unique(array, return counts=True) devuelve un array con los valores únicos del array ordenados v un array con el número de veces que aparece cada

np.unique(array, axis = b) devuelve un array con los valores únicos ordenados de las filas o columnas

Funciones para arrays unidimensionales

np.intersect1d(array1, array2) devuelve un array con los valores únicos de los elementos en común de dos np.intersect1d(array1, array2, return indices=True)

devuelve un array con los valores únicos de los elementos en común de dos arrays y arrays con los índices de cada valor, por arrav np.union1d(array1, array2) devuelve un array ordenado con los elementos resultantes de unir dos

arravs (valores únicos) np.in1d(array1, array2) devuelve un array con True o False por cada elemento de arrav1 según si aparece el mismo valor en array2

np.setdiff1d(array1, array2) devuelve un array ordenado con los valores únicos que están en arrav1 pero no en arrav2

np.setxor1d(array1, array2) devuelve un array ordenado con los valores únicos que NO están en común de los dos arrays

Guardar y salvar arrays en .txt

np.savetxt('ruta/nombre fichero.txt', array) guardar un array de uno o dos dimensiones como .txt variable = np.loadtxt('ruta/nombre fichero.txt', dtype = tipo) cargar datos de un archivo txt que tiene el mismo número de valores en cada fila

NumPy Random

np.random.seed(x) establece la semilla aleatoria del generador de números aleatorios, para que las funciones random que van después siempre cogerán los mismos valores "aleatorios"

np.random.uniform(n,m, size = (z,y,x)) genera muestras aleatorias de una distribución uniforme en el intervalo entre n y m np.random.binomial(n,m, size = (z,y,x)) genera muestras con una distribución binomial; n es el numero total de pruebas; m es la probabilidad de éxito np.random.normal(loc = n, scale = m, size =

(z,v,x)) genera números aleatorios de una distribución normal (curva de campana); loc es la media; scale es la desviación estándar np.random.permutation(array) devuelve un array con los mismos valores mezclados aleatoriamente