

Python Cheat Sheet 3

Pandas

Series: estructuras en una dimension

Crear series
`serie = pd.Series()` crear serie vacía
`serie = pd.Series(array)` crear serie a partir de un array con el índice por defecto
`serie = pd.Series(array, index = ['a', 'b', 'c'...])` crear una serie con índice definida; debe ser lista de la misma longitud del array
`serie = pd.Series(lista)` crear una seria a partir de una lista
`serie = pd.Series(número, indice)` crear una serie a partir de un escalor con la longitud igual al número de índices
`serie = pd.Series(diccionario)` crear una serie a partir de un diccionario

Acceder a informacion de una serie
`serie.index` devuelve los índices
`serie.values` devuelve los valores
`serie.shape` devuelve la forma (no. filas)
`serie.size` devuelve el tamaño
`serie.dtypes` devuelve el tipo de dato

`serie[i]` devuelve el valor del elemento en índice i
`serie[[i,j]]` devuelve el valor de los dos elementos
`serie[i:m]` devuelve el valor de un rango

`serie[“etiqueta”]` devuelve el valor de los elementos en índices i y j

Operaciones con series
`serie1 +-*/ serie2` suma/resta/multiplica/divide las filas con índices comunes entre las dos series
`serie1.add(serie2, fill_value = número)` suma las filas con índices comunes, y suma el fill value a los valores sin índice comun
`serie1.sub(serie2, fill_value = número)` restan las filas de la serie2 de la serie1 cuando tienen índices comunes, y resta el fill value de las otras índices de serie1
`serie1.mul(serie2, fill_value = número)` multiplica las filas con índices comunes y multiplica el fill value con las otras *usar 1 para conservar el valor*
`serie1.mul(serie2, fill_value = número)` divida las filas de la serie1 entre las de la serie2 cuando tienen índices comunes, y divide las otras por el fill value
`serie1.mod(serie2, fill_value = número)` devuelve el modulo (division sin resta)
`serie1.pow(serie2, fill_value = número)` calcula el exponencial
`serie1.ge(serie2)` compara si serie1 es mayor que serie2 y devuelve True o False
`serie1.le(serie2)` compara si serie1 es menor que serie2 y devuelve True o False

Filtrado booleanos
`serie < > >= <= == valor` devuelve True o False segun si cada condición cumple la condición
`serie1[serie1 < > >= <= == valor]` devuelve solo los 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)
`serie.notnull()` devuelve True o False segun si los valores existen o son nulos ("" no cuenta como nulo)

DataFrames: estructuras en dos dimensiones

Crear DataFrames
`df = pd.DataFrame(data, index, columns)`
data: NumPy Array, diccionario, lista de diccionarios
index: índice 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

`df = pd.DataFrame(array)` crear un dataframe a partir de un array con índices y columnas por defecto
`df = pd.DataFrame(diccionario)` crear un dataframe a partir de un diccionario - los keys son los nombres de las columnas

Acceder a informacion de un DataFrame
`df.loc[“etiqueta_fila”, “etiqueta_columna”]` devuelve 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
`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
`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
`df.loc[[lista_etiquetas_filas], [lista_etiquetas_columnas]]` devuelve el contenido de varias filas / varias columnas
`df.loc[[lista_indices_filas], [lista_indices_columnas]]` devuelve el contenido de varias filas / varias columnas - se puede usar los índices/rangos de las listas [start:stop:step] dentro de los loc/iloc
`df.loc[df.etiqueta > x]` seleccionar datos basado en una condición usando operadores comparativos
`df.loc[(df.etiqueta > x) & (df.etiqueta == y)]` seleccionar datos que tienen que cumplir las dos condiciones (and)
`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 Serie booleana; hay que convertirla en lista
`variable_df.head(n)` devuelve las n primeras filas del df, o 5 por defecto

Crear columnas
`df[“nueva_columna”] = (df[“etiqueta_columna”] + x)` crea una nueva columna basada en otra
`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 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 índice indicada
`allow_duplicates = True` parametro cuando queremos permitir columnas duplicadas (por defecto es False)

Eliminar columnas
`df = df.drop(columns = [“column1”, “column2”])` eliminar columnas

DataFrames: carga de datos

Carga de datos
`df = pd.read_csv(“ruta/nombre_archivo.csv”)` crear un dataframe de un archivo de Comma Separated Values
`df = pd.read_csv(“ruta/nombre_archivo”, sep= “;”)` crear un dataframe de un csv si el separador es ;
`df = pd.read_csv(“ruta/nombre_archivo”, index_col= 0)` crear un dataframe de un csv si el archivo ya tiene una columna índice

`df = pd.read_excel(“ruta/nombre_archivo.xlsx”)` crear un dataframe de un archivo de Excel
- si sale `“ImportError:... openpyxl...”`, en el terminal: `pip3 install openpyxl` o `pip install openpyxl`

`df = pd.read_json(“ruta/nombre_archivo.json”)` crear un dataframe de un archivo de JavaScript Object Notation (formato crudo)
`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 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
`with open(‘ruta/nombre_archivo.pkl’, ‘wb’) as f:`
`pickle.dump(df,f)` pone los datos de un dataframe en el archivo.pkl

`pd.read_pickle(‘ruta/nombre_archivo.csv’).head(n)` leer n filas y 5 columnas del archivo pickle

`pd.read_parquet(‘ruta/nombre_archivo.parquet’)` leer un archivo parquet

`pd.read_sas(‘ruta/nombre_archivo.sas7bdatt’, format = ‘sas7bdatt’)` leer un archivo SAS de formato SAS7BDAT

`pd.read_spss(‘ruta/nombre_archivo.sav’)` leer un archivo SAS de formato SAS7BDAT

Guardado de datos
`df.to_csv(‘ruta/nombre_archivo.csv’)` guardar dataframe como archivo csv
`df.to_excel(‘ruta/nombre_archivo.xlsx’)` guardar dataframe como archivo de Excel
`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
`df.to_pickle(‘ruta/nombre_archivo.pkl’)` guardar dataframe como archivo de pickle

Librería PyDataset
`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
`df = data(‘nombre_dataset’)` guardar un dataset en un dataframe

Metodos para explorar un dataframe
`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
`df.sample(n)` devuelve n filas aleatorias de nuestro dataframe, o uno por defecto

Metodos de DataFrames

Metodos para explorar un dataframe
`df.shape` devuelve el número de filas y columnas
`df.dtypes` devuelve el tipo de datos que hay en cada columna
`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
`df.describe(include = object)` devuelve un dataframe con un resumen de los principales estadísticos, incluyendo columnas con variables tipo string
`df.info` devuelve un resumen sobre el no. de columnas, 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
`df[“nombre_columna”.value_counts()` o `df.nombre_columna.value_counts()` devuelve una serie con el recuento de valores únicos en orden descendente
`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
`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
`inplace = True` los cambios sobrescriben sobre el df
* cuando una columna se cambia a índice ya no es columna *

`df.reset_index(inplace = True)` quitar una columna como índice para que vuelva a ser columna
`df.rename(columns = {“nombre_columna”: “nombre_nueva”}, inplace = True)` cambia los nombres de una o mas columnas
ejemplo de dict comprehension para crear diccionario sobre las columnas existentes de un dataframe:
`diccionario = {col : col.upper() for col in df.columns}`
`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
`axis = 1` columnas
`axis = 0` filas
`df.rename(columns = diccionario, inplace = True)` cambia los nombres de las columnas según el diccionario
`df[“columna_nueva”] = pd.cut(x=df[“nombre_columna”], bins=[n,m,l...])` 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
`df.replace(to_replace = valor, value = valor_nuevo, inplace = True)` reemplaza cierto valor por otro que especificamos
`df[“nombre_columna”].replace(to_replace = valor, value = valor_nuevo, inplace = True)` reemplaza cierto valor en una columna por otro que especificamos
`df[“nombre_columna”] = df[“nombre_columna”] + x` reemplaza los valores de la columna por el valor + x (o otro valor que indicamos)

Python Cheat Sheet 4

NumPy (Numerical Python)

Crear arrays

Crear arrays con valores aleatorios
`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
`array = np.random.rand(z,y,x)` crea un array de floats aleatorias con la forma que le especifiquemos; 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 especifiquemos; por defecto genera números aleatorios entre 0-0.9999999...
`array = np.random.z,y,x=None` devuelve un número aleatorio en 0 y 0.999999999999...
`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

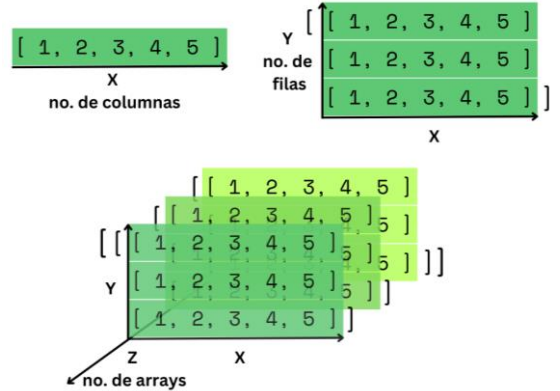
Crear otros tipos de arrays
`array = np.arange(valor_inicio, valor_final, saltos)` crea un array usando el formato [start:stop:step]
`array = np.ones(z,y,x)` crea un array de todo unos de la forma especificada
`array2 = np.ones_like(array1)` crea un array de todo unos de la forma basada en otra array
`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 vacío con datos por defecto tipo float
`array2 = np.empty_like(array1)` crea un array vacía 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 posición 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 array1
`np.multiply(array1, array2)` multiplica dos arrays
`np.divide(array1, array2)` 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
`array[i]` devuelve la índice i; las índices de los arrays unidimensionales funcionan igual que las listas
`array[i, j]` o `array[i][j]` 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 posición 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[(array > n) & (array < m)]` 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
`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 transposición según lo que especifiquemos 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 la forma
`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 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:
`axis = 0` columnas
`axis = 1` filas
- si especificamos el axis, la operación devuelve el resultado por cada fila o columna.
Por ejemplo:
`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 ejemplo:
`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 array
`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 array

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 array
`np.sin(array)` devuelve un array con la tangente de cada elemento del array

Operaciones de comparación en arrays bidimensionales
`np.any(array > n)` devuelve True o False según si cualquier valor del array cumple con la condición
`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 cumple con la condición
`np.all(array > n)` devuelve True o False según si todos los valores del array cumple con la condición
`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