

Python Cheat Sheet 1
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Variables ampliadas por text (CONCATENATION)</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Para encadenar texto</div> <div>categoria1 = "verde" color_detalle = categoria1 + ' ' + 'oscuro' print(categoria1 + ' oscuro') print(categoria1, 'oscuro')</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>type() and isinstance()</div> <div>float/int/str(variable) cambia el tipo de data/type type(variable) devuelve: class 'float/int/str' isinstance(variable, float/int/str) comprobar el tipo de dato (devuelve True/False)</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Operaciones Algebraicas</div> <div>+ sumar / dividir - restar // divider y redondear (modulus) * multiplicar % resto de una division (floor division) ** elevar round(x) redondear número x</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Operaciones Binarias</div> <div>== comprobar si valores coinciden is comprobar si valores son exacamente igual != comprobar si valores son diferentes is not comprobar si valores no son exactamente iguales > (>=) mayor que (mayor o igual que) < (<=) menor que (menor o igual que) and ambas verdaderas or ambas o solo una verdadera in/not in comprobar si hay un valor en una lista etc.</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Metodos String</div> <div>string.upper()z MAYUSCULAS string.lower() minusculas string.capitalize() Primera letra de la frase en may. string.title() Primera Letra De Cada Palabra En May. string.swapcase() mINUSCULAS A mAYUSCULAS O vICEVERSA string.strip() quita espacios del principio y final string.split() divide string en lista - por espacios por defecto, o especifica otro divisor en () string.replace("frase", "frase") reemplaza la primera frase del string por el otro " ".join(string) une los elementos de una lista en una string con el separador espificado en " " list(string) convierte un variable string en una lista string.find("substring") encuentra el indice en que empiece el substring/'-1' si no existe el substring string[i] devuelve el elemento en la indice i string[i:j] devuelve un rango de caracteres</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div># metodos permanentes (cambia el variable, no devuelve nada)</div>

Listas [] Metodos no permanentes
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>lista = [] crea una lista vacia len(lista) devuelve el no. de elementos min(lista)/max(lista) saca el valor minimo y maximo lista.count() devuelve el no. de elementos que hay en la lista de un valor determinado en los() sorted(lista) ordenar una lista de menor a mayor lista.copy() hacer una copia de la lista</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Metodos con indices</div> <div>list.index(x) devuelve la indice de x en la lista lista[i] devuelve el elemento en la indice i [start:stop:step] lista[i:j:x] devuelve los elementos por el rango de i a j (incluye i pero no j) saltando por x lista[-i:-j] devuelve los elementos por los indices negativos (incluye -j pero no -i)</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Listas – Acciones Permanentes</div> <div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Ampliar una lista</div> <div>[lista1, lista2] junta listas pero se mantienen como listas separadas lista1 + lista2 hace una lista mas larga . .append() lista.append(x)# añade un solo elemento (lista, string, integer o tuple) a la lista .extend() lista.extend(lista2)# añade los elementos de una lista al final de la lista .insert() insert(i, x)# mete un elemento (x) en un índice(i)</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Ordenar una lista</div> <div>.sort() lista.sort()# ordena de menor a mayor, usar con (reverse=True) para ordenar de mayor a menor lista.reverse()# ordena los elementos al revés del orden guardado</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Quitar elementos de una lista</div> <div>.pop() lista.pop(i)# quita el elemento en indice i y devuelve su valor .remove() lista.remove(x)# quita el primer elemento de la lista con valor x lista.clear()# vacia la lista del lista# borra la lista del lista[i]# borra el elemento en indice i</div>

Diccionarios { key : value , }
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>diccionario = {x:y} compuestos por un key(x) unica y un valor(y) (cualquier tipo de datos) dict() variable = dict(x=y, m=n) crear un diccionario dicc.copy() crear una copia len(dicc) devuelve el no. de elementos (x:y) hay en el diccionario sorted(dicc) ordena los keys; usar con .items() para ordenar tuplas de los elementos o .values() para ordenar los valores solos</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Diccionarios – Metodos</div> <div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Obtener informacion de un diccionario</div> <div>dicc.keys() devuelve todas las keys dicc.values() devuelve todos los valores dicc.items() devuelve tuplas de los key:value in/not in comprobar si existe una clave dicc.get(x, y) devuelve el valor asociado al key x, o si no existe devuelve el output y dicc["key"] devuelve el valor del key (ver abajo que tiene mas usos)</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Ampliar un diccionario</div> <div>.update() dicc.update({x:y})# para insertar nuevos elementos dicc["key"] = valor# para insertar un nuevo key o valor, o cambiar el valor de un key dicc.setdefault(x, y)# devuelve el value del key x, o si no existe la key x, la crea y asigna el valor y por defecto</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Quitar elementos de un diccionario</div> <div>dicc.pop(x)# elimina la key x (y lo devuelve) dicc.popitem()# elimina el ultimo par de key:value dicc.clear()# vacia el diccionario</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Tuplas (,) inmutables, indexados</div> <div>tupla = (x,y) tuplas se definen con () y , o solo , tupla1 + tupla2 juntar tuplas tuple(lista) crear tuplas de una lista tuple(dicc) crear tuplas de los keys de un diccionario tuple(dicc.values()) crear tuplas de los valores tuple(dicc.items()) crear tuplas de los key:valores len(tupla) devuelve el no. de elementos in/not in comprobar si hay un elemento tupla.index(x) devuelve el indice de x tupla.count(x) devuelve el no. de elementos con valor x en la tupla *para cambiar el contenido de una tupla hay que convertirla en una lista y luego a tupla*</div>

zip()
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>zip(iterable1, iterable2) crea una lista de tuplas de parejas de los elementos de las dos listas (mientras se puede) listzip.sort() ordena las tuplas del zip por el primer elemento</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>Sets {}</div> <div>no permiten duplicados, no tienen orden</div> <div>set = {x,y} set(iterable) solo permite un argumento iterable; elimina duplicados in/not in comprobar si hay un elemento len(set) devuelve el no. de elementos Ampliar un set set.add(x)# añadir un elemento set.update(set o lista)# añadir uno o mas elementos con [] o {} o un variable tipo lista o set Quitar elementos de un set set.pop()# elimina un elemento al azar set.remove(x)# elimina el elemento x set.discard(x)# elimina el elemento x (y no devuelve error si no existe) set.clear()# vacia el set Operaciones con dos Sets set1.union(set2) devuelve la union de los dos sets: todos los elementos menos dupl. set1.intersection(set2) devuelve los elementos comunes de los dos sets set1.difference(set2) devuelve los sets que estan en set1 pero no en set2 (restar) set1.symmetric_difference(set2) devuelve todos los elementos que no estan en ambos set1.isdisjoint(set2) comprobar si todos los elementos de dos sets son diferentes set1.issubset(set2) comprobar si todos los elementos de set1 estan en set2 set1.superset(set2) comprobar si todos los elementos de set2 estan en set1</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>input()</div> <div>permite obtener texto escrito por teclado del usuario input("el texto que quieres mostrar al usuario") se puede guardar en un variable por defecto se guarda como un string x = int(input("escribe un número")) para usar el variable como integer o float se puede convertir en el variable</div>

Sentencias de control
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>if ... elif ... else if estableca una condición para que se ejecute el código que esta debajo del if. *tiene que estar indentado* elif para chequear mas condiciones después de un if else agrupa las condiciones que no se han cumplido; no puede llevar condiciones nuevas if x > y: print("x es mayor que y") elif x == y: print("x es igual que y") else: print("x e y son iguales") while repite el código mientras la condición sea True, o sea se parará cuando la condición sea False se pueden incluir condiciones con if... elif... else *pueden ser infinitos* (si la condición no llega a ser False) while x < 5: print("x es mayor que 5")</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>For loops</div> <div>sirven para iterar por todos los elementos de un variable que tiene que ser un iterable (lista, diccionario, tupla, set, or string) se pueden combinar con if ... elif ... else, while, u otro for loop en diccionarios por defecto itera por las keys; podemos usar dicc.values() para acceder a los valores for i in lista: print("hola mundo")</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>List comprehension</div> <div>su principal uso es para crear una lista nueva de un un for loop en una sola línea de código [lo que queremos obtener iterable condición (opcional)]</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>try ... except</div> <div>Se usan para evitar que nuestro código se pare debido a un error en el código. Se puede imprimir un mensaje que avisa del error. try: print("2.split()) except: print("no funciona")</div>
<div><div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div><div><div><div></div></div><div><div></div></div></div></div></div> <div>range()</div> <div>nos devuelve una lista de números que por defecto se aumentan de uno en uno empezando por 0 range(start:stop:step) se puede especificar por donde empieza y el limite (que debe ser +1 por que se para uno antes del limite que ponemos como stop) tambien se puede especificar saltos</div>

Python Cheat Sheet 2
Funciones
Definir una funcion: def nombre_funcion(parametro1, parametro2, ...): return valor_del_return
Lllamar una funcion: nombre_funcion (argumento1, argumento2, ...)
return: es opcional, pero sin return devuelve None parametros por defecto: – siempre deben ser lo ultimo
*args: una tupla de argumentos sin limite **kwargs: diccionarios cuyas keys se convierten en parámetros y sus valores en los argumentos de los parámetros
def nombre_funcion(parametros, *args, **kwargs, parametro_por_defecto = valor) arg/kwarg: sin */** dentro de la funcion arg[0]
Lllamar una funcion con *args: nombre_funcion (argumento, argumento, argumento, ...) o nombre_funcion (*[lista_o_tupla_de_args])
Lllamar una funcion con **kwargs: nombre_funcion (**diccionario)
Clases
Definir una clase: class NombreClase:
<pre>def __init__(self, atributo1, atributo2): self.atributo1 = atributo1 self.atributo2 = atributo2 self.atributo_por_defecto = 'valor'</pre>
<pre>def nombre_funcion1(self, parametros) self.atributo += 1 return f"el nuevo valor es {self.atributo}"</pre>
Definir una clase hija: class NombreClaseHija(NombreClaseMadre): def __init__(self, atributo1, atributo2): super ().__init__(atributo_hereditado1, ...)
<pre>def nombre_funcion_hija (self, parametros):</pre>
Crear un objeto de la clase: variable_objeto = NombreClase(valor_atributo1, valor_atributo2) instanciar (crear) un objeto variable_objeto.atributo devuelve el valor del atributo guardado para ese objeto variable_objeto.atributo = nuevo_valor para cambiar el valor del atributo variable_objeto.nombre_funcion() llamar una funcion
print (help(NombreClase) imprime informacion sobre la clase

Regex
- una abreviatura de `expresión regular`, `regex` es una cadena de texto que permite crear patrones que ayudnan a emparejar, localizar y gestionar strings import re para poder trabajar con regex
Operadores communes de regex + coincide con el carácter precedente una o más veces * coincide con el carácter precedente cero o más veces u opcional ? indica cero o una ocurrencia del elemento precedente . coincide con cualquier carácter individual ^ coincide con la posición inicial de cualquier string \$ coincide con la posición final de cualquier string
Sintaxis básica de regex \w cualquier caracter de tipo alfabético \d cualquier caracter de tipo númeroico \s espacios \n saltos de línea \W cualquier caracter que no sea una letra \D cualquier caracter que no sea un dígitos \S cualquier elemento que no sea un espacio () aísla sólo una parte de nuestro patrón de búsqueda que queremos devolver [] incluye todos los caracteres que queremos que coincidan e incluso incluye rangos como este: a-z y 0-9 es como el operador ‘or’ \ señala una secuencia especial (escapar caracteres especiales) { } Exactamente el número especificado de ocurrencias {n} Exactamente n veces {n,} Al menos n veces {n,m} Entre n y m veces
Métodos Regex re.findall() busca en todo el string y devuelve una lista con todas las coincidencias en nuestro string re.search() busca en todo el string y devuelve un objeto con la primera coincidencia en nuestro string re.match() busca en la primera línea del string y devuelve un objeto con la primera coincidencia en nuestro string resultado_match.span() devuelve la referencia de las posiciones donde hizo el “match” resultado_match.group() devuelve el element resultando de la coincidencia del “match” re.split() busca en todo el string y devuelve una lista con los elementos separados por el patron re.sub() busca en todo el string y devuelve un string con el element que coincide

Modulos/Librerias (paquetes de funciones)
Importar y usar modulos y sus funciones import modulo para importar un modulo from modulo import funcion importar solo una funcion modulo.funcion() usar una funcion de un modulo modulo.clase.funcion() para usar una funcion de una clase import modulo as md asignar un alias a un modulo
Libreria os os.getcwd() devuelve la ruta de donde estamos trabajando; se puede guardar en un variable e.g. ruta = os.getcwd() os.listdir() devuelve una lista de los archivos y carpetas donde estamos trabajando os.listdir(‘carpeta’) devuelve los contenidos de otra carpeta os.chdir(‘ruta’) cambia la carpeta en la que estes os.mkdir(‘nueva_carpeta’) crear una nueva carpeta os.rename(‘nombre_carpeta’, ‘nueva_nombre’) cambia el nombre de una carpeta os.rmdir(‘carpeta’) borra la carpeta
Libreria shutil from shutil import rmtree rmtree(‘carpeta’) borra la carpeta y subcarpetas
Abrir y cerrar ficheros Primero hay que guardar la ruta del archivo: ubicacion_carpeta = os.getcwd() nombre_archivo = “text.txt” ubicacion_archivo = ubicacion_carpeta + “/” + nombre_archivo f = open(ubicacion_archivo) abrir un archivo en variable f f.close() cerrar un archivo * IMPORTANTE * with open(ubicacion_archivo) as f: codigo e.g. variable = f.read() abre el archivo solo para ejecutar el codigo indicado (y despues lo deja)
Encoding from locale import getpreferredencoding getpreferredencoding() para saber que sistema de encoding estamos usando f = open(ubicacion_archivo, encoding="utf-8") abrir un archivo y leerlo con el encoding usado; guardar con .read()
mode: argumento opcional al abrir un archivo r – read w – write - sobreescribe x – exclusive creation, sólo crearlo si no existe todavía a – appending, añadir texto al archivo sin manipular el texto que ya habia hay que anadir otra letra: t – texto – leer en texto b – bytes – leer en bytes (no se puede usar con encoding)
f = open(ubicacion_archivo, mode = “rt”)
Leer ficheros f.read() leer el contenido de un archivo f.read(n) leer los primeros n caracteres de un archivo variable = f.read() guardar el contenido del archivo (o n caracteres de un archivo) en un variable f.readline(n) por defecto devuelve la primera linea o n lineas f.readlines() devuelve una lista de todas las lineas del archivo (cada linea es un elemento); se usa vacio sin n y list_name[x:] para seleccionar lineas especificas
Escribir en ficheros with open(ubicacion_archivo, “w”) as f: f.write(“Texto que va en el fichero.”) para escribir with open(ubicacion_archivo, “a”) as f: f.write(“Texto que va en el fichero.”) para anadir texto f.writelines(‘lista’) para anadir lineas de texto de una lista

Ficheros xml
import xml.etree.ElementTree as ET importa la librería xml variable_tree = ET.parse(‘ruta/archivo.xml’) abre el archivo variable_root = variable_tree.getroot() saca el elemento que envuelve todo (el elemento raíz) en una lista <root> <child_tag atributo1=“valor” atributo2=valor> <subchild_tag> elemento </subchild_tag> </child_tag> </root> variable_root.tag devuelve el nombre del tag del raiz variable_root.attrib devuelve los atributos del fichero variable_root.find(“tag”).find(“childtag”).text devuelve la primera ocasión en que el tag de un elemento coincida con el string variable_root.findall(“tag”).findall(“childtag”).text devuelve todos los elementos cuyos tag coincide
MySQL Connector/Python
Conectar a una base de datos import mysql.connector para importar MySQL Connector pip install mysql-connector pip install mysql-connector-Python connect() para conectar a una base de datos: variable_cnx = mysql.connector.connect (user='root', password='AlumnaAdalab', host='127.0.0.1', database='nombre_BBDD') from mysql.connector import errorcode importar errores mysql.connector.Error se puede usar en un try/except cnx.close() desconectar de la base de datos
Realizar queries variable_cursor = cnx.cursor() crear el objeto cursor que nos permite comunicar con la base de datos variable_cursor.close() desconectar el cursor variable_query = (“SQL Query”) guardar un query en un variable variable_cursor.execute(variable_query) ejecutar el query; devuelve una lista de tuplas import datetime sacar fechas en el formato AAAA-MM-DD datetime.date (AAAA, M, D) devuelve el formato de fecha variable_query = “SQL Query... %s AND %s”) query dinamica variable_cursor.execute(query, (variable1, variable2)) valores que van en lugar de los %s variable_cursor.execute(“SHOW DATABASES”) mostrar las BBDD variable_cursor.execute(“SHOW TABLES”) mostrar las tablas de la BBDD indicado en la conexión variable_cursor.execute(“SHOW TABLES”) variable_cursor.execute(“SHOW COLUMNS FROM bbdd.table”) mostrar las columnas de la tabla especificada; hay que conectarse a la bbdd information_schema
Argumentos cursor: variable_cursor = cnx.cursor ([arg=value[, arg=value]...]) buffered=True devuelve todas las filas de la bbdd raw=True el cursor no realizará las conversiones automáticas entre tipos de datos dictionary=True devuelve las filas como diccionarios named_tuple=True devuelve las filas como named tuples cursor_class un argumento que se puede usar para indicar que subclase queremos usar para instanciar el nuevo cursor

Obtener resultados de una query variable_cursor.fetchone() devuelve el primer resultado variable_cursor.fetchall() devuelve todos los resultados como iterable – cada fila es una tupla
Pandas dataframe with SQL import pandas as pd variable_df = pd.DataFrame (variable_resultado_fetchall, columns = [‘columna1’, ‘columna2’, ...]) crear un dataframe con los resultados de una query en una variable variable_df.head(n) devuelve las n primeras filas del df, o 5 por defecto variable_df = pd.read_sql_query (variable_query, variable_cnx) convertir los resultados de la query en df pd.read_sql (variable_query, variable_cnx) variable_df.to_csv (“nombre_archivo.csv”) guardar en csv variable_df.to_string() formatear el dato en string variable_df.to_latex() formatear el dato en un string que facilite la inserción en un documento latex
Crear y alterar una base de datos variable_cursor.execute(“CREATE DATABASE nombre_BBDD”) variable_cursor.execute(“CREATE TABLE nombre_tabla (nombre_columna TIPO, nombre_columna2 TIPO2)”) variable_cursor.execute(“ALTER TABLE nombre_tabla ALTERACIONES”)
Insertar datos variable_query = “INSERT INTO nombre_tabla (columna1, columna2) VALUES (%s, %s)” variable_valores = (valor1, valor2) variable_cursor.execute (variable_query, variable_valores) otro método: variable_query = “UPDATE nombre_tabla SET nombre_columna = “nuevo_valor” WHERE nombre_columna = “valor”
Insertar múltiples filas a una tabla variable_valores_en_tuplas = ((valor1columna1, valor1columna2), (valor2columna1, valor2columna2)) variable_cursor.executemany (variable_query, variable_valores_en_tuplas)
variable_conexion.commit() después de ejecutar la inserción, para que los cambios efectúen en la BBDD variable_conexion.rollback() se puede usar después de execute y antes de commit para deshacer los cambios print (variable_cursor.rowcount, “mensaje”) imprimir el número de filas en las cuales se han tomado la accion
Eliminar registros variable_query = “DROP TABLE nombre_tabla”
Añadir errores importar errorcode y usar try/except: try: accion except mysql.connector.Error as err: print(err) print("Error Code:", err.errno) print("SQLSTATE", err.sqlstate) print("Message", err.msg)

Python Cheat Sheet 3	DataFrames: estructuras en dos dimensiones	DataFrames: carga de datos	Metodos de DataFrames
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	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 estadísticos, incluyendo 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 índice utilizando uno o mas columnas; puede sustituir o ampliar un índice existente <code>inplace = True</code> los cambios sobrescriben sobre el df * cuando una columna se cambia a índice 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 <code>axis = 1</code> columnas <code>axis = 0</code> 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)
Series: estructuras en una dimension			
Crear series <code>serie = pd.Series()</code> crear serie vacía <code>serie = pd.Series(array)</code> crear serie a partir de un array con el indice por defecto <code>serie = pd.Series(array, index = [‘a’, ‘b’, ‘c’...])</code> crear una serie con indice definida; debe ser lista de la misma longitud del array <code>serie = pd.Series(lista)</code> crear una seria a partir de una lista <code>serie = pd.Series(número, indice)</code> crear una serie a partir de un escalor con la longitud igual al número de indices <code>serie = pd.Series(diccionario)</code> crear una serie a partir de un diccionario	Acceder a informacion de un DataFrame <code>df.loc[“etiqueta_fila”, “etiqueta_columna”]</code> devuelve el contenido de un campo en una columna de una fila <code>df.loc[“etiqueta_fila”,:]</code> devuelve los valores de todas las columnas de una fila <code>df.loc[:,“etiqueta_columna”]</code> devuelve los valores de todas las filas de una columna <code>df.iloc[indice_fila, indice_columna]</code> devuelve el contenido de un campo en una columna de una fila <code>df.iloc[indice_fila, :]</code> devuelve los valores de todas las columnas de una fila <code>df.iloc[:,indice_columna]</code> devuelve el contenido de un campo en una columna de una fila <code>df.loc[[lista_etiquetas_filas], [lista_etiquetas_columnas]]</code> devuelve el contenido de varias filas / varias columnas <code>df.loc[[lista_indices_filas], [lista_indices_columnas]]</code> devuelve el contenido de varias filas / varias columnas - se puede usar los indices/rangos de las listas [start:stop:step] dentro de los loc/iloc <code>df.loc[df.etiqueta > x]</code> seleccionar datos basado en una condición usando operadores comparativos <code>df.loc[(df.etiqueta > x) & (df.etiqueta == y)]</code> seleccionar datos que tienen que cumplir las dos condiciones (and) <code>df.loc[(df.etiqueta > x) (df.etiqueta == y)]</code> seleccionar datos que tienen que deben cumplir una de las dos condiciones (or) <code>df.iloc[list(df.etiqueta > x), :]</code> iloc no acepta una Serie booleana; hay que convertirla en lista <code>variable_df.head(n)</code> devuelve las n primeras filas del df, o 5 por defecto		
	Crear columnas <code>df[“nueva_columna”] = (df[“etiqueta_columna”] + x)</code> crea una nueva columna basada en otra <code>df = df.assign(nueva_columna= df[“etiqueta_columna”] + x)</code> crea una nueva basada en otra <code>df = df.assign(nueva_columna= [lista_valores])</code> crea una nueva columna de una lista de valores *tiene que ser de la misma longitud como el número de filas del dataframe* <code>df.insert(indice_nueva_columna, “nombre_columna”, valores)</code> crea una nueva columna en la indice indicada <code>allow_duplicates = True</code> parametro cuando queremos permitir columnas duplicadas (por defecto es False)		
	Eliminar columnas <code>df = df.drop(columns = [“column1”, “column2”])</code> eliminar columnas		

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

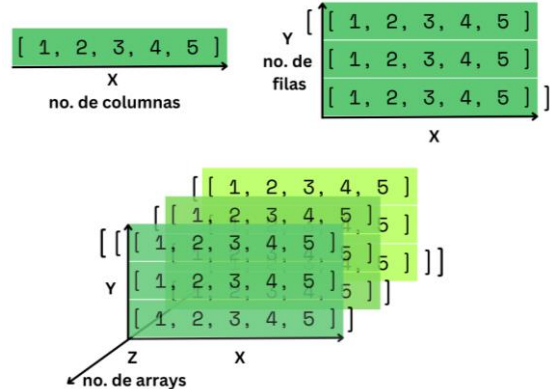
Crear otros tipos de arrays
`array = np.arange(valor_inicio, valor_final, saltos)` crea un array usando el formato [start:stop:step]
`array = np.ones(z,y,x)` crea un array de todo unos de la forma especificada
`array2 = np.ones_like(array1)` crea un array de todo unos de la forma basada en otra array
`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