# Taller Integrador - Python.

### Clase 9

### Sábado 22/05/2020 (Chivilcoy)

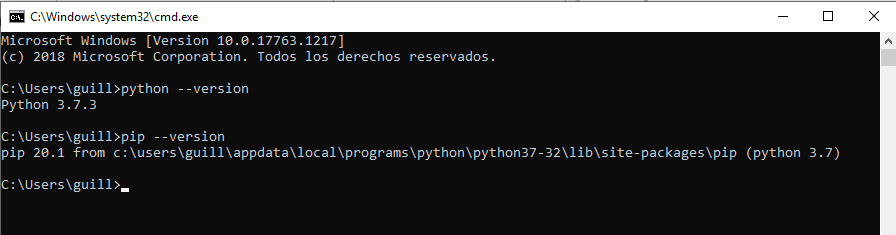
### Martes 26/05/2020 CABA.

### PIP.

PIP es un acrónimo que significa "Programa de instalación preferida". Es una utilidad de línea de comandos que le permite instalar, reinstalar o desinstalar paquetes PyPI con un comando simple y directo: "pip".

Esta utilidad viene instalada por defecto a partir de la versión 2.7 de Python.

Una manera sencilla de ver que versión tenemos instalada, tanto de Python como de PIP, es consultarlo directamente desde la línea de comandos, el viejo y querido CMD:



### Instalando un módulo.

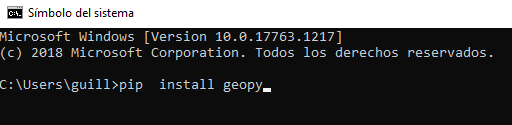
Para instalar un módulo con la utilidad PIP, lo primero es buscar que módulo queremos instalar, todos desde su página, nos dan el formato de la instrucción PIP a utilizar, por ejemplo, el módulo GEOPY que proporciona una librería de utilidades para geolocalización.

<https://pypi.org/project/geopy/>

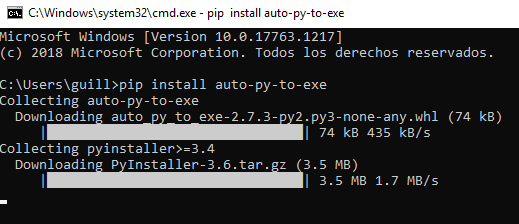
Cuando ingresamos a la web, lo primero que vemos es el formato de PIP:



Esa instrucción la ejecutamos desde el CMD:



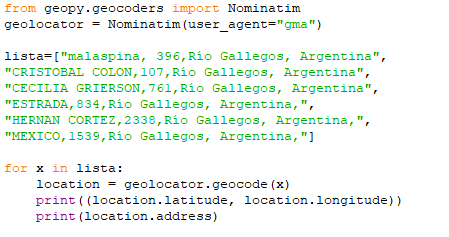
A continuación, veremos el avance de la instalación del módulo:

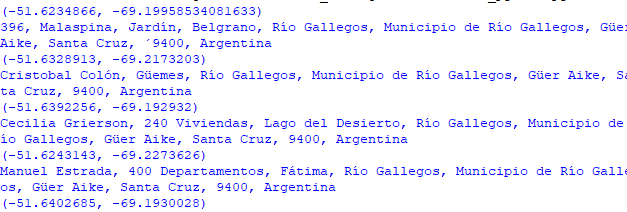


Y un mensaje de finalización, en mi caso una advertencia de que ya hay disponible una versión de PIP mas actual que la que tengo instalada, pero la instalación finalizó con éxito.

En la misma página están los ejemplos de uso y la documentación, a partir de acá es solo importar el módulo y empezar a usarlo.

Por ejemplo, creamos una lista de domicilios, y nos traemos de cada uno sus atributos de latitud y longitud.



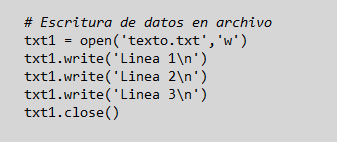


La misma web tiene un buscador para revisar los módulos disponibles e instalarlos.

### Archivos de texto (escritura).

Python nos provee de herramientas muy simples para generar un respaldo de los datos que estamos trabajando en archivos de texto.

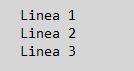
Veamos un ejemplo sencillo:



Son tres simples pasos, abrir la conexión a un archivo (exista o no), escribir línea por línea los datos que necesitamos guardar y por último, cerrar la conexión al archivo.

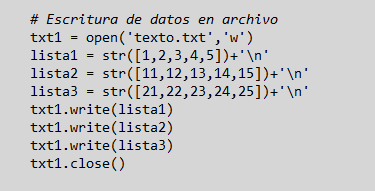
El archivo se genera en el mismo directorio que nuestra aplicación.

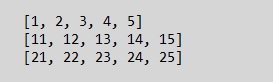
Al ejecutar el código anterior, se nos va a generar un archivo TXT llamado texto conteniendo las líneas que le pasamos:



Si volvemos a grabar datos en el mismo archivo, se pisarán los que habíamos grabado previamente.

La información a grabar en el archivo deberá ser una cadena de caracteres, si por ejemplo tenemos listas, deberemos convertirlas antes de guardarlas:





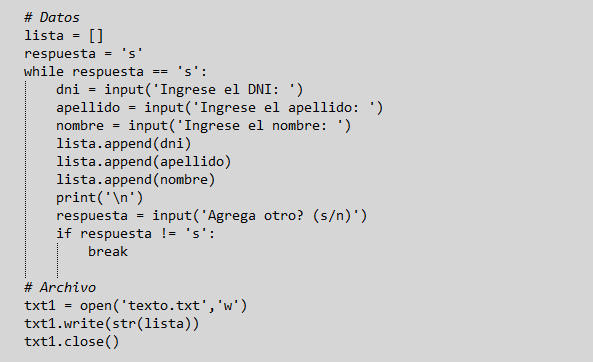
Podemos especificar la ruta donde queremos que el archivo se genere junto con el nombre:



***\*\*\* Ejercicio propuesto: Escribir un programa que solicite el ingreso por teclado de los datos de un cliente (DNI, apellido y nombre) y le pregunte al usuario si desea seguir cargando datos (las opciones son s/n).***

***Una vez terminada la carga de datos, guardarlos en un archivo.***

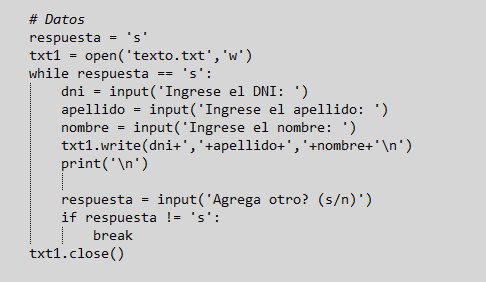
SOLUCIÓN PROPUESTA:



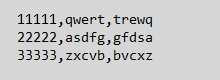
La solución propuesta de arriba nos permite grabar cada dato en una línea, lo que puede no ser la mejor solución si vamos a guardar muchos datos.

Podríamos darle algún formato de los que acostumbramos a ver en archivos de texto, como, por ejemplo, todos los campos de una fila separados por algún carácter especial y cada fila separada por un salto de línea.

Modificamos el código anterior:



Así se ven ahora los datos que ingresemos en el archivo de texto:

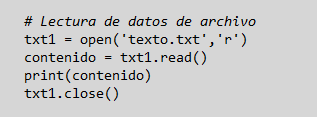


Ahora nuestros datos guardados tienen un formato más conocido, fácil de organizar.

* Podemos guardar nuestro archivo con otro formato, teniendo en cuenta que, si vamos a usar un formato de alguna aplicación, debemos conocer exactamente como lo guarda, como organiza los datos dentro del archivo para que dicha aplicación pueda leer nuestro archivo y comprenderlo.

### Archivos de texto (lectura).

De manera similar, podemos leer los archivos, por ejemplo, el que acabamos de generar.



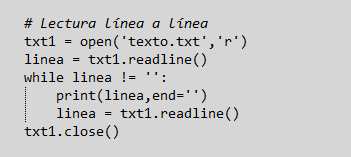
En este ejemplo simple, abrimos una conexión a nuestro archivo, mediante el método READ cargamos los datos, en nuestro caso lo cargamos en una variable, lo mostramos por pantalla y por último cerramos la conexión.

* Si al método READ no le pasamos ningún argumento, lo que estamos haciendo es leer todos los datos del archivo. Si por ejemplo le pasamos el valor 2, traerá los primeros 2 caracteres.

### Lectura línea a línea.

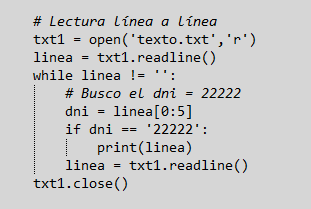
Leer nuestro archivo de datos, línea a línea, es muy sencillo mediante el método READLINE.

Vamos a ver un ejemplo con nuestros datos cargados en el ejercicio anterior



El resultado final será el mismo que al leer todo el archivo junto, pero con la posibilidad de operar con la línea leída.

Para nuestro ejemplo, tuvimos la precaución de tomar como primer dato, el número de dni que es único y sabemos que longitud tiene, por lo que nos sirve para utilizar como clave para identificar todo un registro de datos:

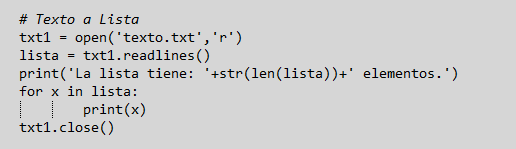


Modificamos el código que leía línea a línea para que busque un determinado número de DNI y en caso de encontrarlo, mostrar por pantalla todo el registro.

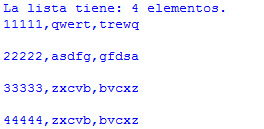
***\*\*\* Modificar el código del ejercicio para que pida un DNI por teclado y muestre un mensaje en caso de no encontrarse el DNI buscado.***

### Almacenar texto en listas.

Mediante el uso del método READLINES podemos almacenar en una lista, las líneas de texto recuperadas de nuestro archivo.



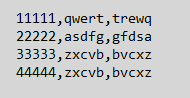
Este es el resultado que veremos por pantalla:



Mediante el uso de este método, lo que hacemos es cargar cada línea de nuestro archivo como un elemento de la lista.

### Identificar un único registro.

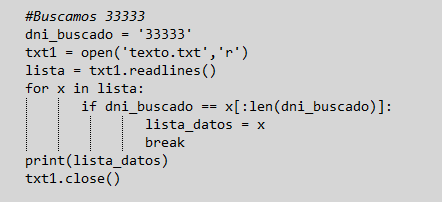
Supongamos que tenemos los siguientes datos en nuestro archivo:



Donde cada línea representa el DNI, apellido y nombre de una persona.

Supongamos que necesitamos identificar un registro en particular, necesitamos los datos de la persona cuyo DNI es igual a 33333.

Con el método READLINES vamos a poder tomar a cada línea como un elemento de la lista, así que podemos identificar aquel elemento que comience con los caracteres buscados (33333), y si además tuvimos la precaución de que este dato sea siempre de la misma longitud, podría llegar a ser más sencillo, pero supongamos que no fue así.



Explicación del código:

Cargamos el DNI buscado en la variable dni\_buscado (este dato podría ingresarlo el usuario por teclado).

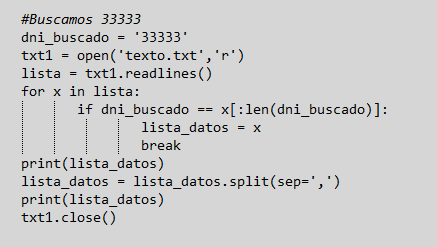
Luego recorremos todas las líneas del archivo y comparamos nuestra variable buscada con los primeros n caracteres de cada línea. El valor de n lo da el largo de nuestra variable dni\_buscado, es decir, básicamente lo que hacemos es fijarnos que línea de nuestro archivo comienza con el valor cargado en dni\_buscado.

Una vez identificada la línea que nos interesa, la guardamos en la variable lista\_datos.

Al ejecutarlo, esto es lo que veremos por pantalla:



Una vez identificada nuestra línea, podemos, mediante el método SPLIT, convertirla en una lista para poder identificar cada dato guardado en la misma:

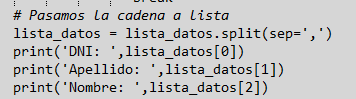


Como vemos en el código anterior, estamos mostrando por pantalla la línea tal cual la tomamos del archivo y luego, mediante el método SPLIT la convertimos en una lista, separando todo lo que este separado entre comas (el separador se pasa como argumento del método) y tomándolo como un elemento de la lista nueva, que en nuestro caso utilizamos el mismo nombre.

Así se ven ambas por pantalla:



Vemos que la primera impresión es una cadena mientras que la segunda ya está convertida en lista, de hecho, verificamos que es indexable:



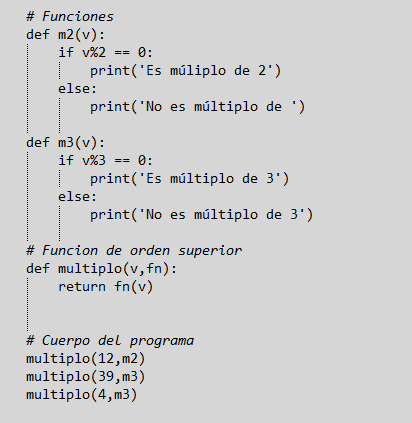
Y así lo vemos por pantalla:



# Funciones de orden superior.

Las funciones de orden superior son funciones que pueden recibir como parámetro a otras funciones.

Veamos un ejemplo simple de como poder utilizar como argumento de una función, otras funciones:



Lo que hicimos fue definir dos funciones (m2 y m3) que se encargan de recibir un número y mostrar por pantalla si el número recibido es múltiplo o no de dos (m2) y lo mismo para verificar si es múltiplo o no de 3 (m3).

A continuación, definimos nuestra función de orden superior (múltiplo) que además de recibir un valor, también recibe por parámetro otra función.

Para invocarla lo hacemos como a cualquier otra función a la que le pasamos parámetros.

Este es el resultado que deberíamos ver por pantalla:



***\*\*\* Ejercicio propuesto: Hacer una función de orden superior llamada Operador que reciba dos valores y otra función como parámetros.***

***Las funciones que pasemos por parámetro deberán ser Sum, Resta, Multiplicación y División.***

***Probar con al menos un valor cada una de ellas.***