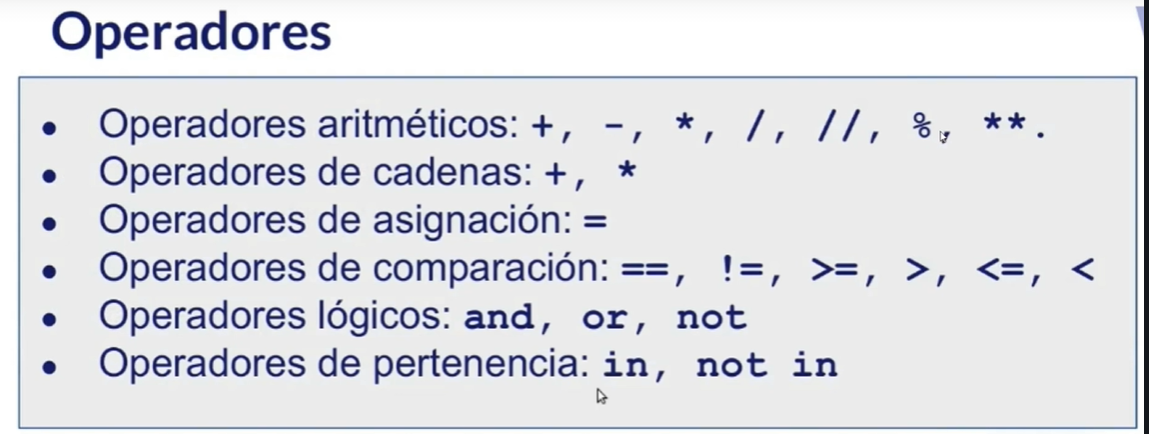
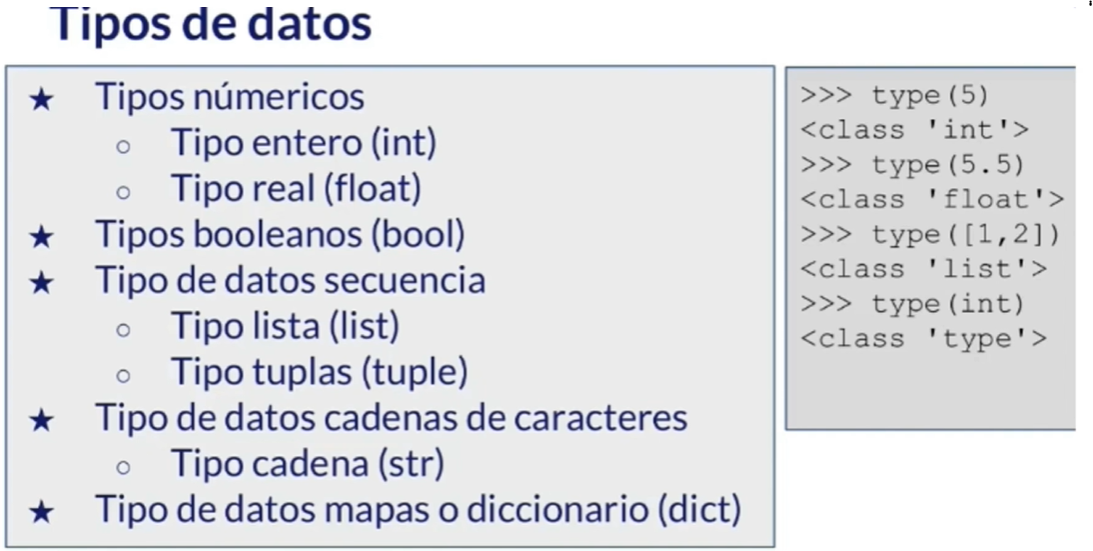
1. Hay 2 tipos de IP (pública y privada).
2. IP Privada es la que entrega el proveedor de servicio, la que uno tiene en su casa.
3. La IP Publica es la que utilizamos para salir a Internet.
4. Existen Protocolos, Puertos y Servicios.

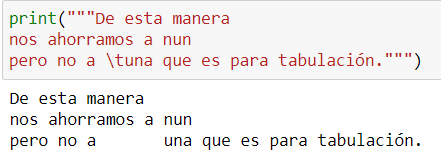
**Python**

1. Instalar **Anaconda Individual edition.**
2. #, ##, ### son comentarios, mientras más almohadillas menos importancia.
3. Las variables no son fuertemente tipado.
4. \n, salto de línea.
5. \t, tabulación.
6. Operadores:

1. Con la función **type()**, se puede saber que tipo de datos es un valor o variable.
2. Con triple comilla nos ahorramos el salto de línea, pero no la tabulación. Ejemplo:



1. Para que funcione el salto de línea o tabulación, hay que usar **print()**.
2. Se concatena con el signo +.
3. Índice de variable permite posicionarse en un string.
4. Índice se representa entre corchetes con su valor, ejemplo, profesor [2].
5. También se puede acceder de forma inversa comenzando en -1.
6. Se puede usar el índice en grupo de caracteres, de la forma:

Variable[inicio:final-1]

Ejemplo:

Profesor = “Álvaro Chirou]

Profesor[2:5] = var

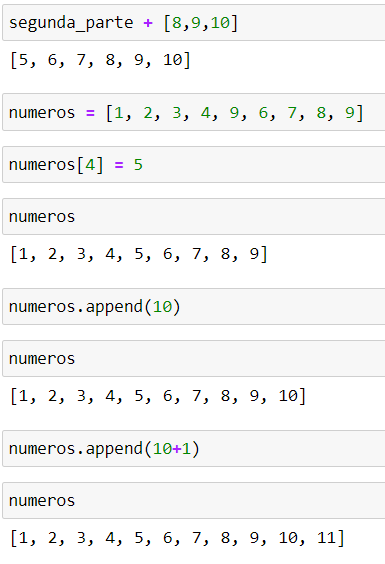
1. También puede ser:

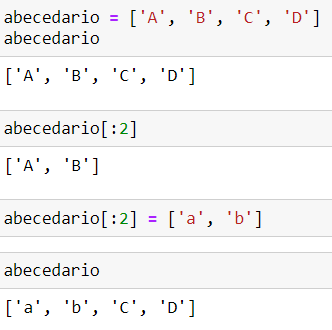
Profesor[:5]

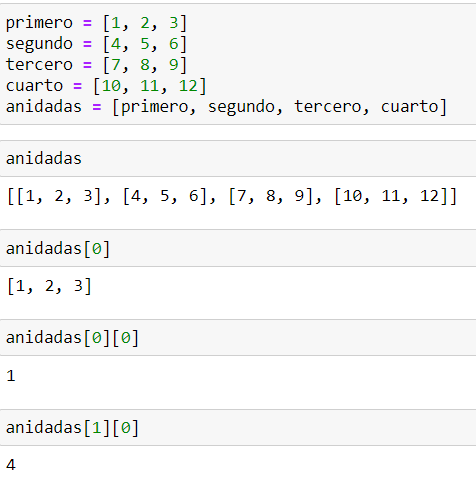
Profesor[2:]

1. Para contar la cantidad de caracteres se usa la función **len(variable)**.
2. Las **listas** van entre corchete.



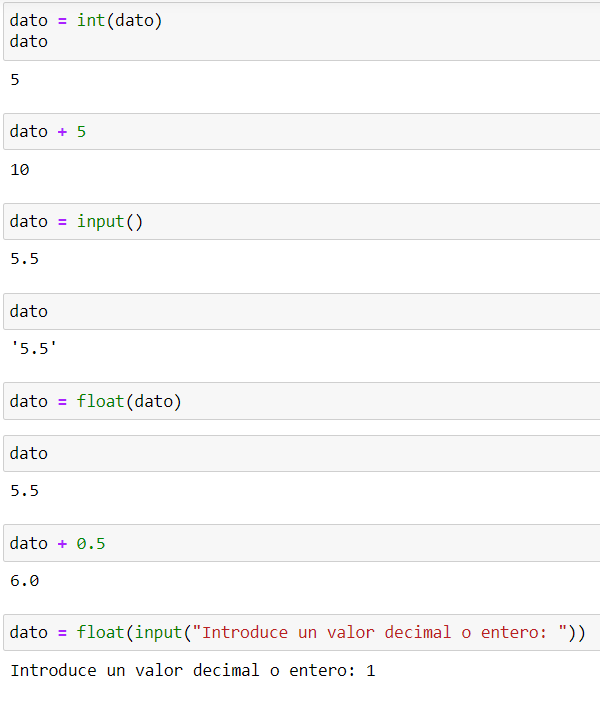


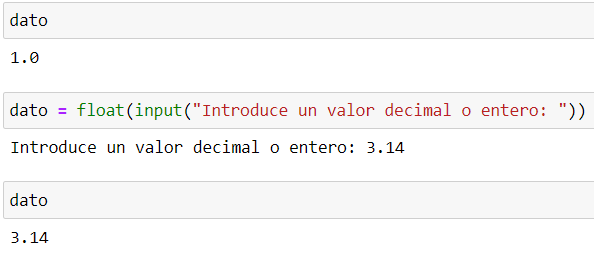




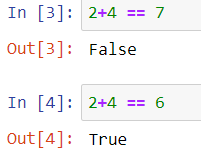
1. Datos por teclado:
   1. Se usa la función **input()**.
   2. Lo captura queda como cadena.
   3. Si es un número hay que convertirlo con la función **int()** o **float()**.







1. Operadores relacionales, lógicos y asignación. Expresiones Anidadas:
   1. Operadores lógicos:
      1. Es cuando se tiene un valor verdadero o falso.

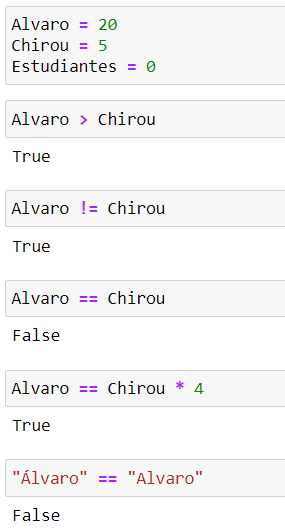


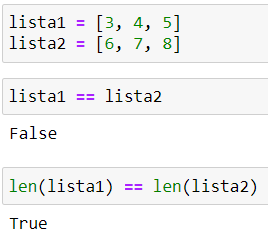
* + 1. Los operadores lógicos son NOT, AND Y OR.
    2. Python es Case Sensitive.



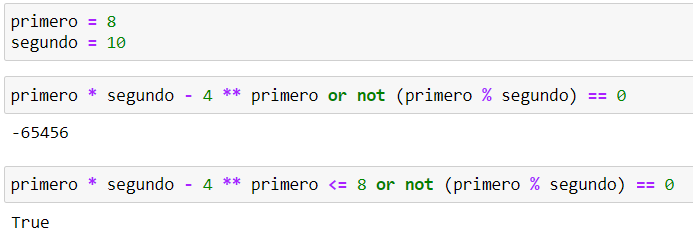


* 1. Operadores racionales:
     1. Compara 2 valores y obtiene un resultado.
     2. Son: !=, ==, <, > <=, =>.



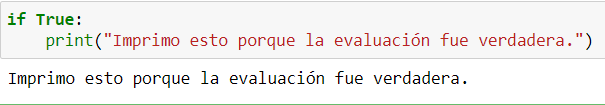


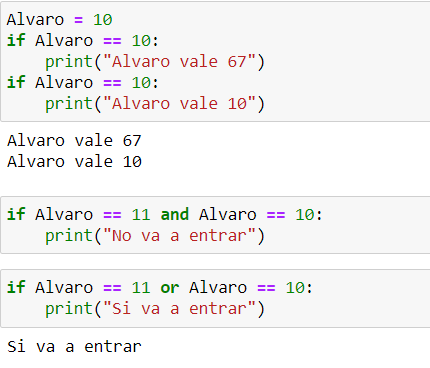
* 1. Expresiones anidadas y operadores de asignación:
     1. Primero resolver lo que esta entre paréntesis,
     2. Luego las expresiones aritméticas,
     3. Luego las expresiones relacionales,
     4. Finalmente, las expresiones lógicas.



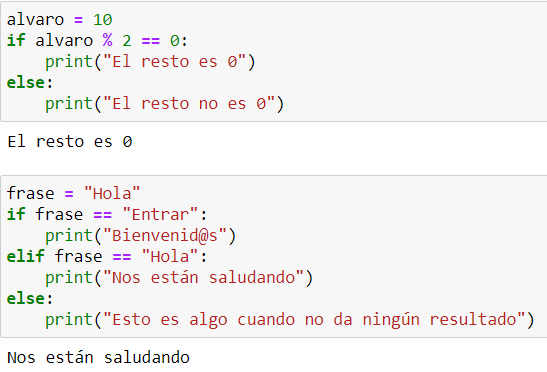
* + 1. Operadores de asignación son: =, +=, -=, \*=, /=, %=, \*\*=.

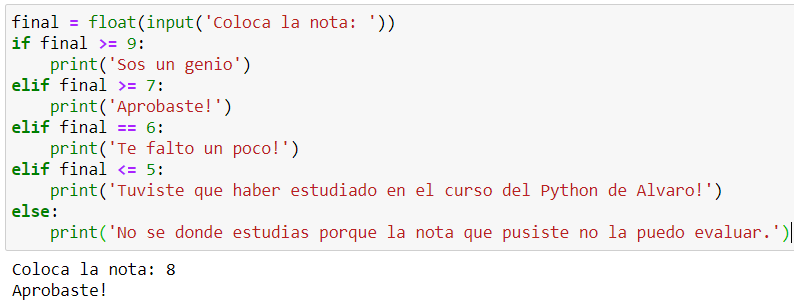
1. Estructuras de control:
   1. La tabulación siempre se respeta.
   2. IF:



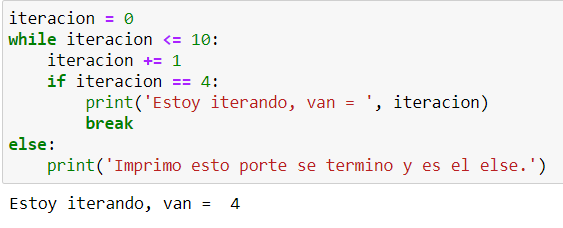


* 1. IF ELSE y ELIF:

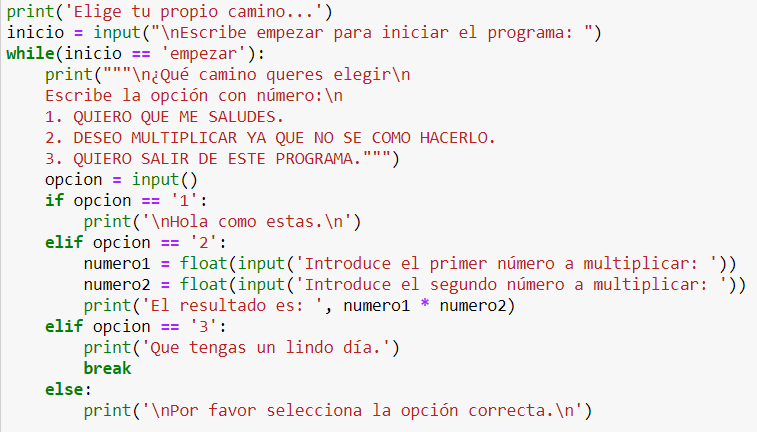




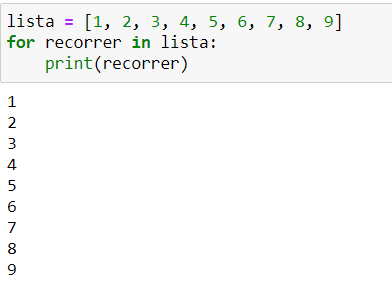
* 1. WHILE:

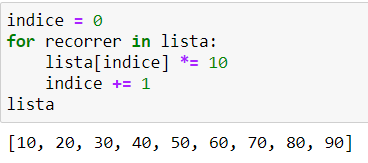


* 1. EJEMPLO:

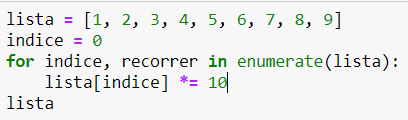


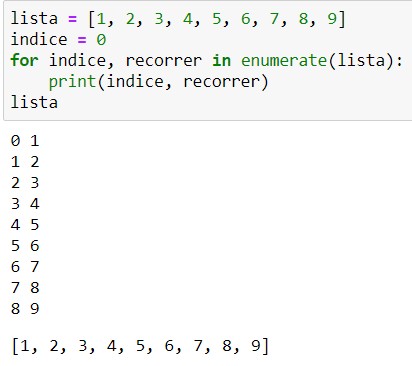
* 1. FOR:

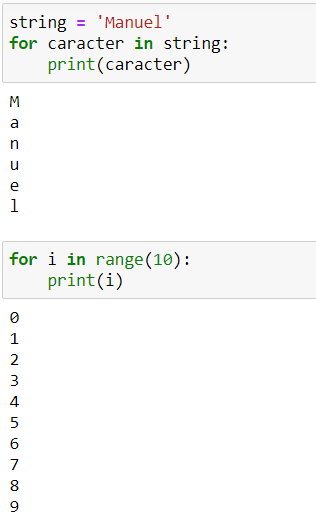


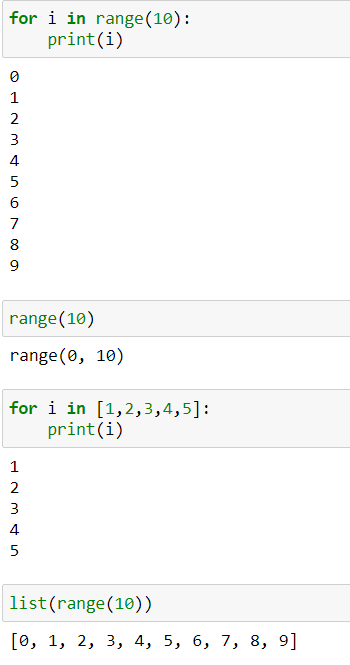


La función enumerate(), permite obtener el contador y el valor.

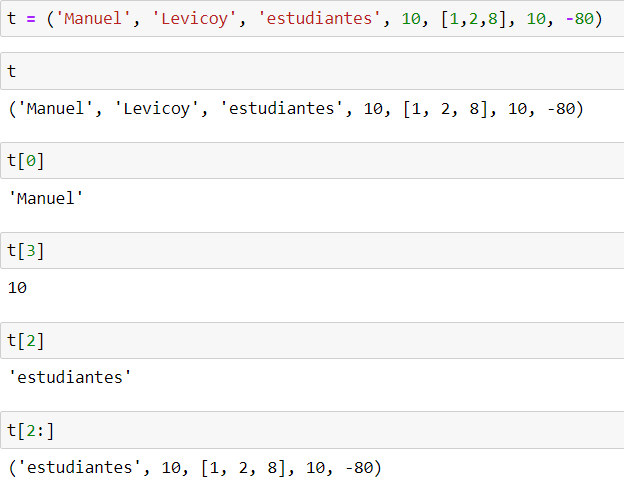




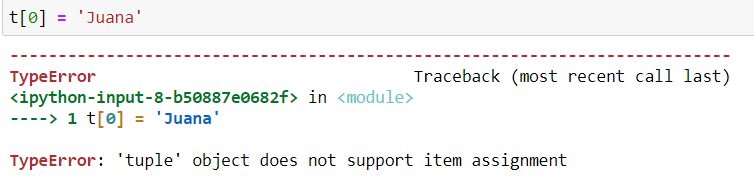




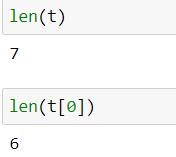
1. **Tuplas, Diccionarios, Conjuntos, Pilas y Colas**:
   1. Tuplas: son como listas, pero inmutables, es decir, no se puede modificar el contenido. Se escribe entre paréntesis.



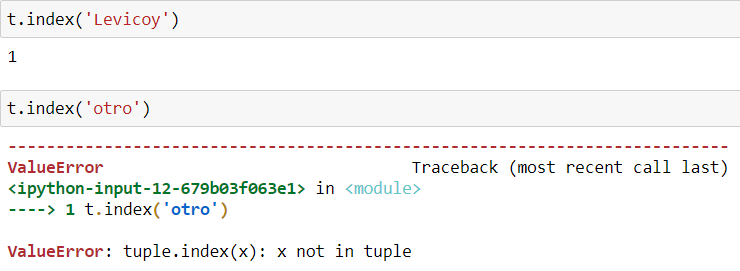
La tupla no se puede modificar, ejemplo:



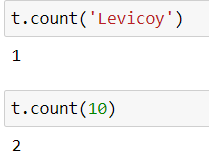
Para ver la cantidad de elementos se usa la función **len()**:



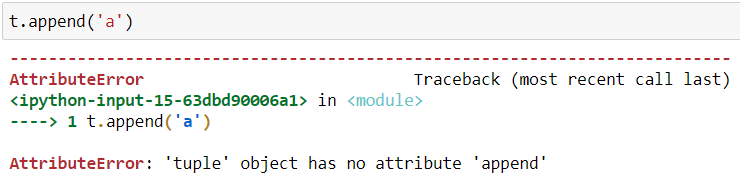
Para ver la posición de un elemento se usa la función **index()**:



Para ver cuantas veces aparece un elemento en la Tupla se usa la función **count()**:

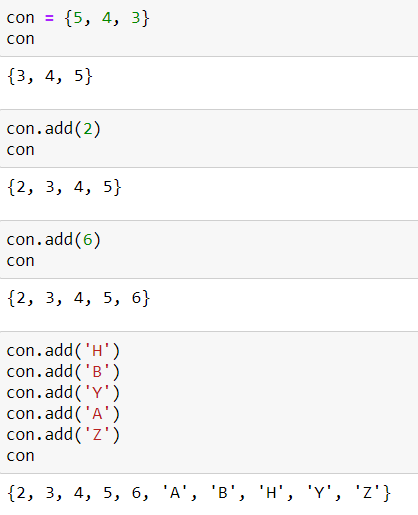


La función **append()** que permite agregar un nuevo elemento no se puede usar ya que las tuplas no se pueden modificar:

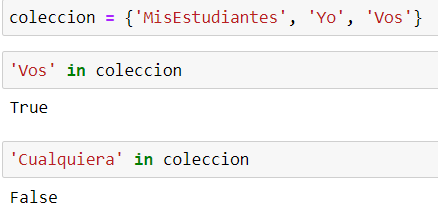


* 1. Conjuntos: Son colecciones desordenadas, permite comprobar la pertenencia de los grupos y la eliminación de los elementos duplicados. Se colocan entre llaves {}.

Para agregar elementos al conjunto:



Para ver si un elemento se encuentra en el conjunto:



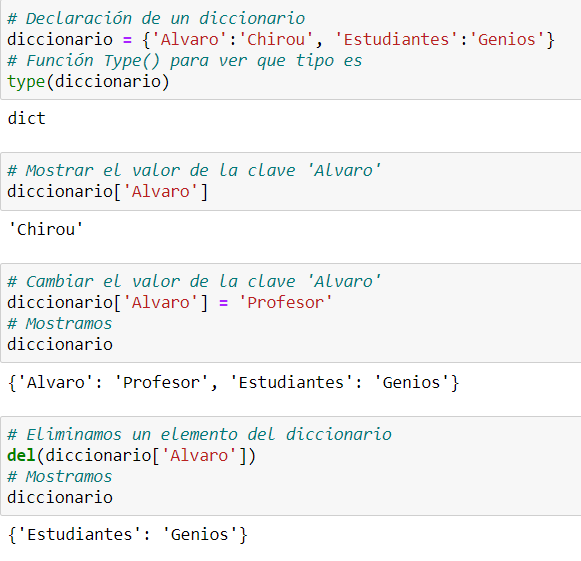
Los conjuntos no permiten elementos repetidos, por ejemplo:



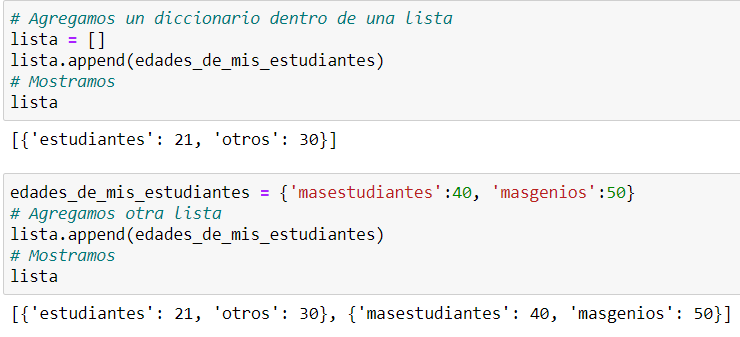
Para crear un conjunto se usa la función **set()**:



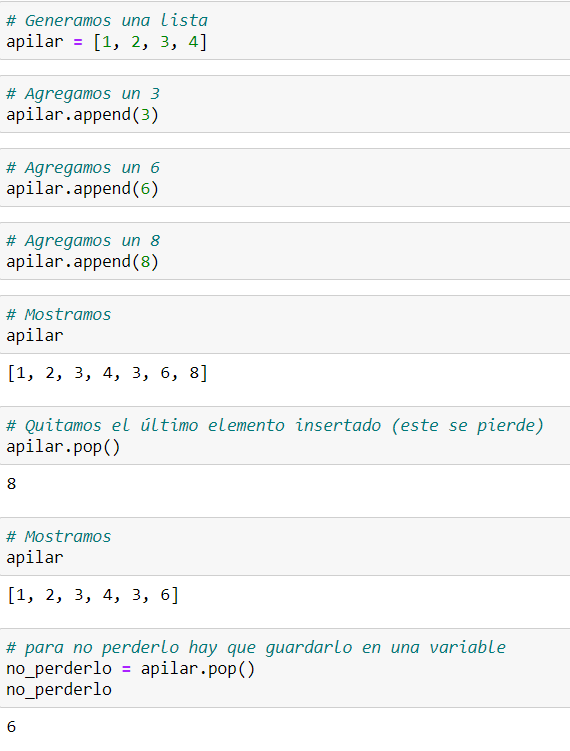
* 1. Diccionario: Son similares a las listas, ocupan una clave única, en otros leguajes se conocen como arreglos asociativos. Su forma es {clave:valor}





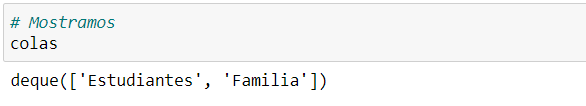


* 1. Pilas: Sale el último que ingresa.

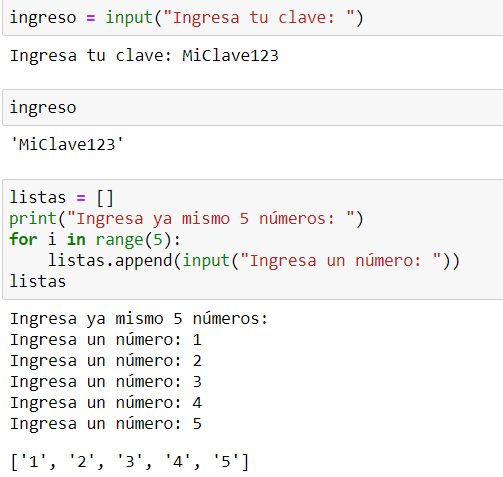


* 1. Colas: Sale el primero que ingresa.

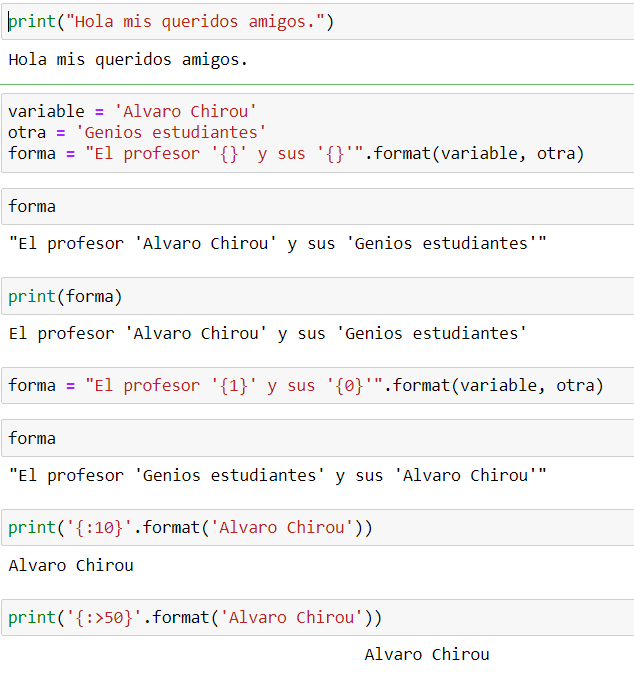




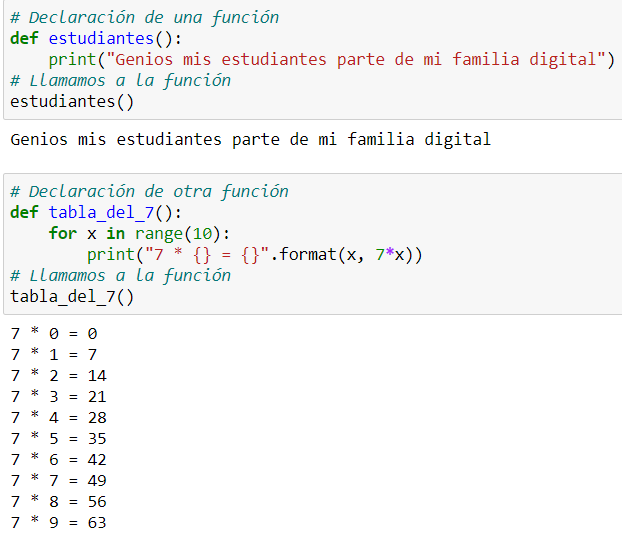
1. **Entrada por Teclado y Salida por Pantalla**
   1. Entrada por teclado:

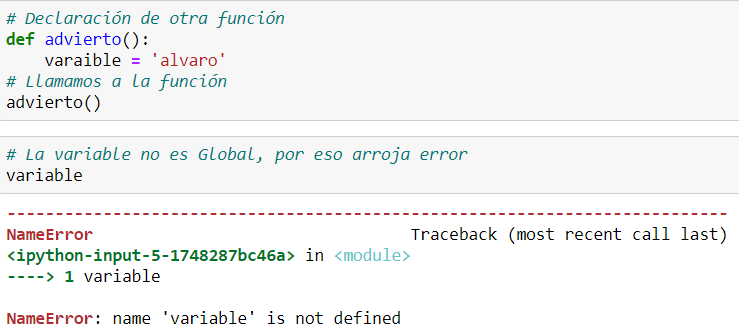


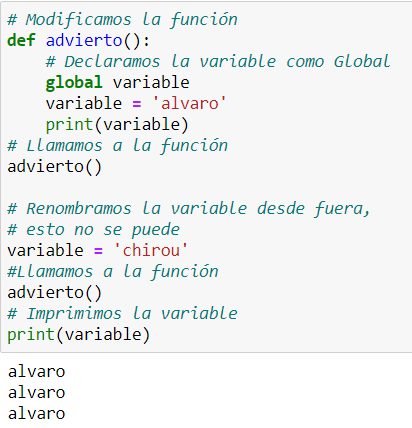
* 1. Salida por pantalla:



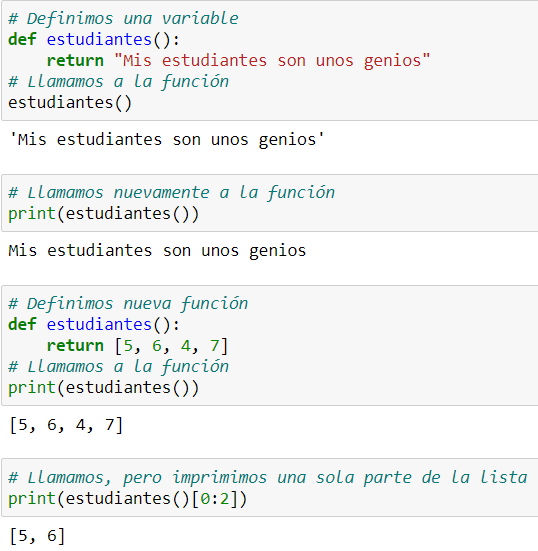
1. **Funciones**:
   1. Funciones:

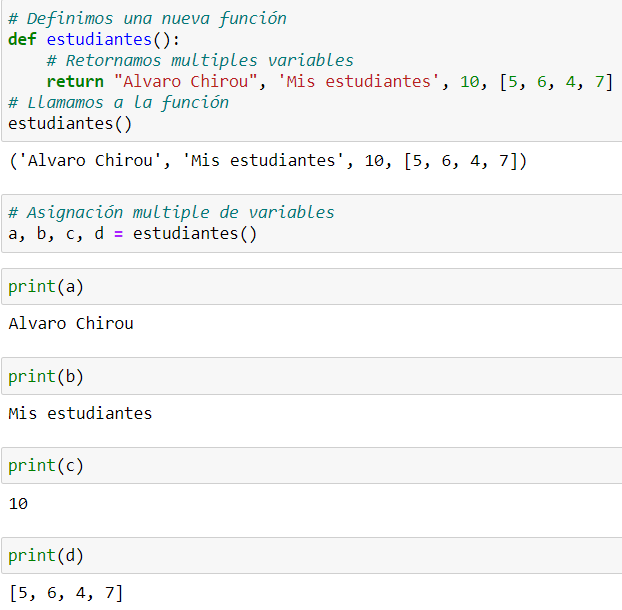


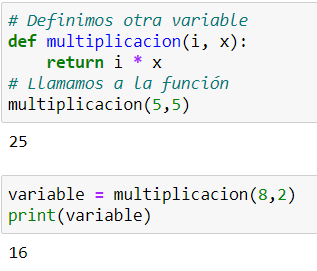




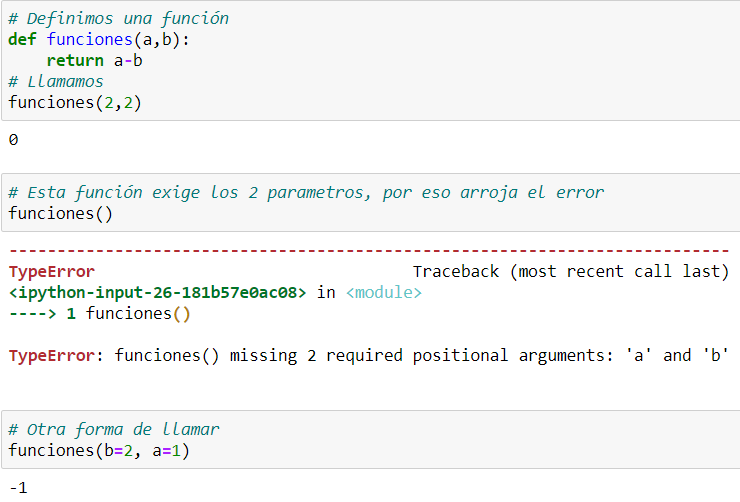
* 1. Retorno y envío de valores:

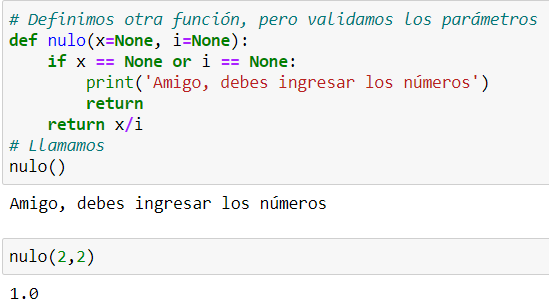




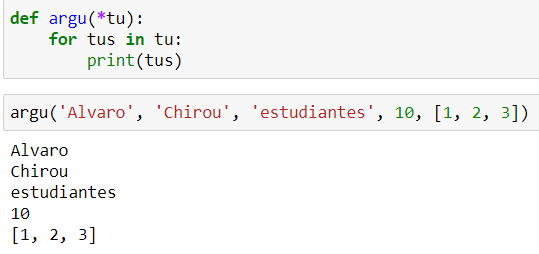


* 1. Funciones argumentos:

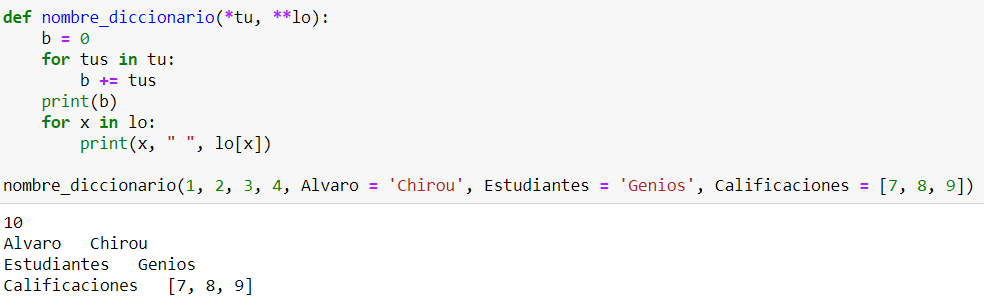




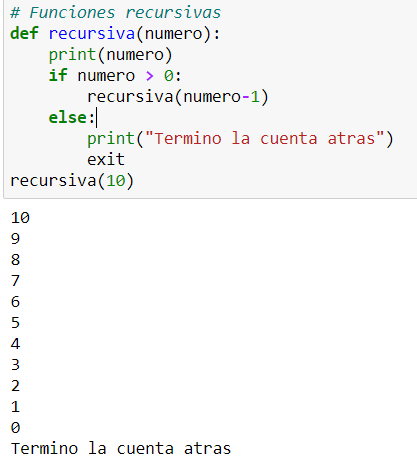
Argumentos indeterminados por posición (tuplas):

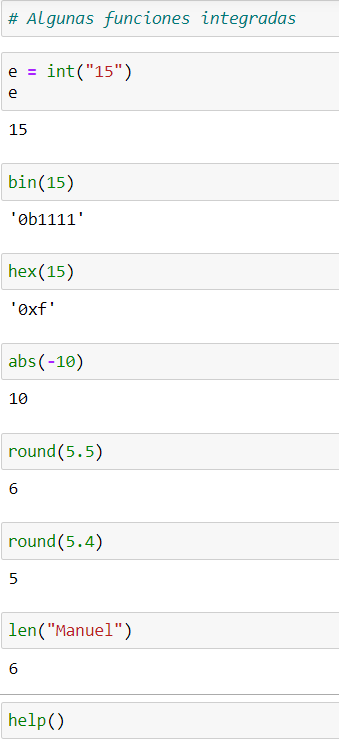


Argumentos indeterminados por nombre (diccionarios):

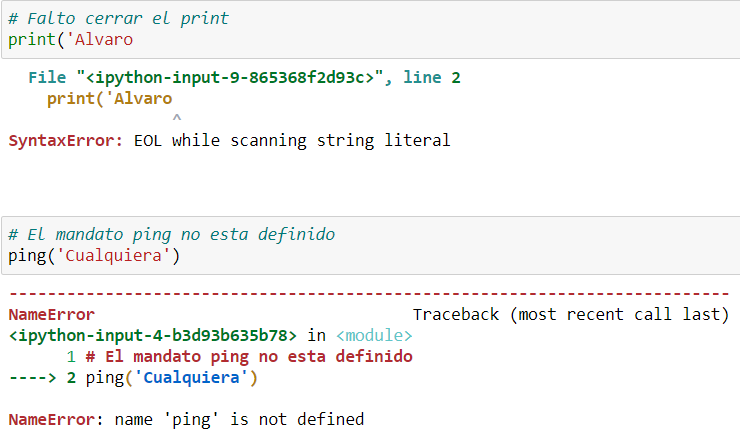


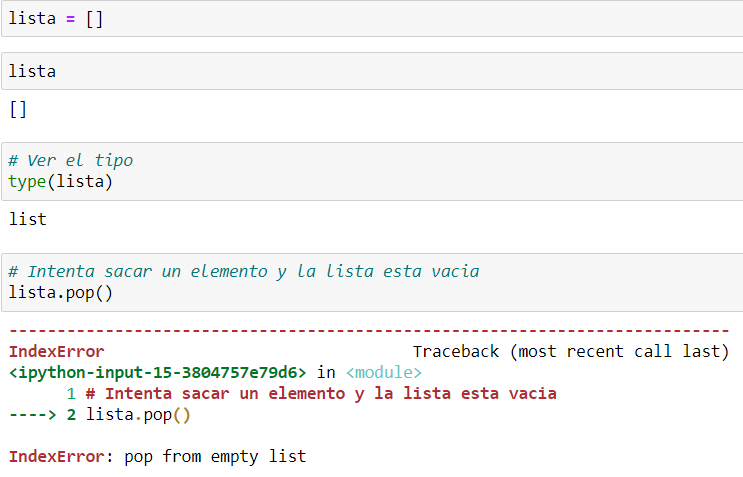
* 1. Funciones recursivas e integrada:



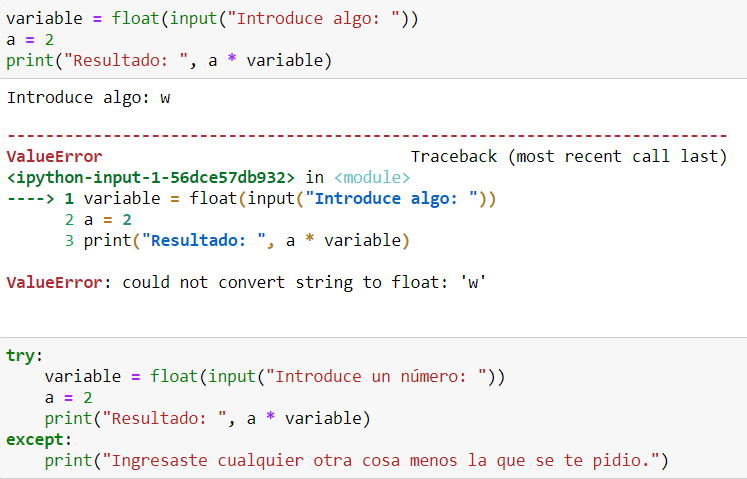


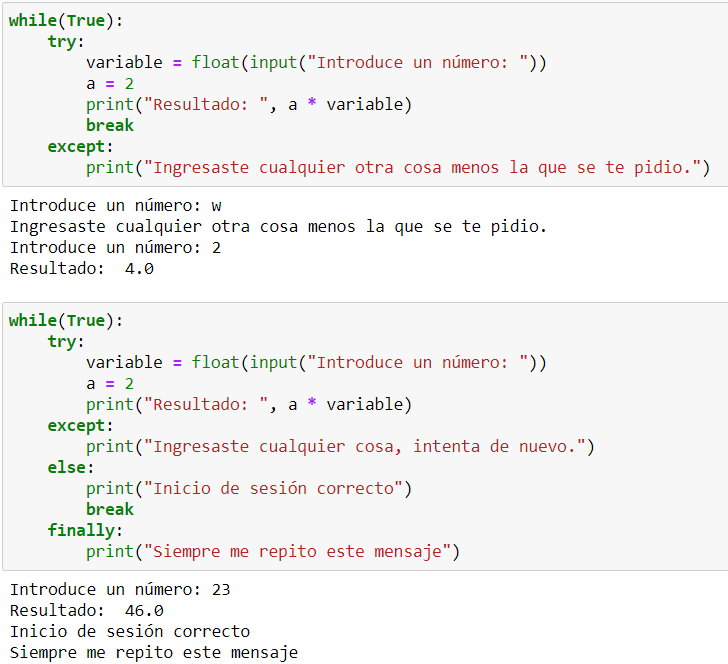
1. **Errores y Excepciones**
   1. Errores:



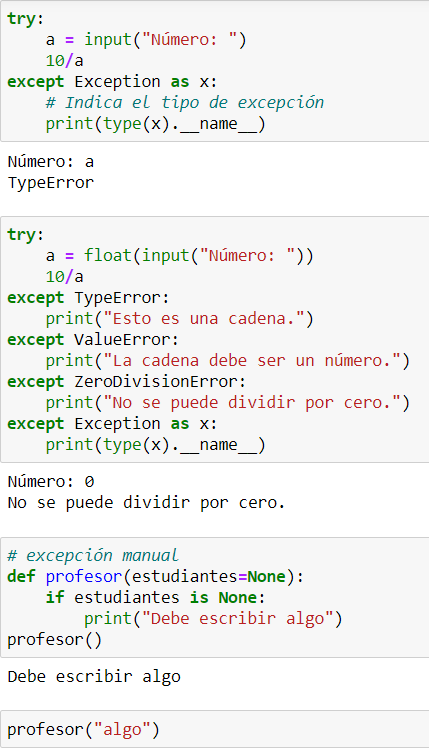


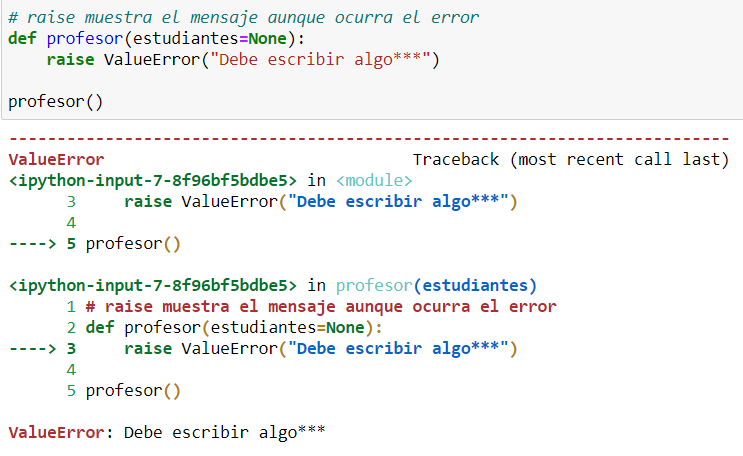
* 1. Excepciones múltiples:



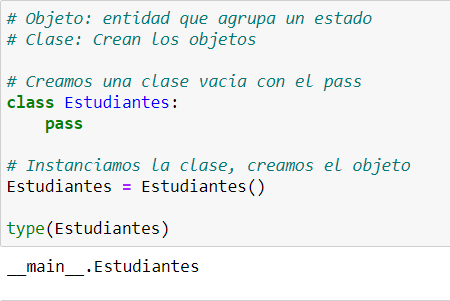


* 1. Excepciones múltiples e invocación de excepciones:

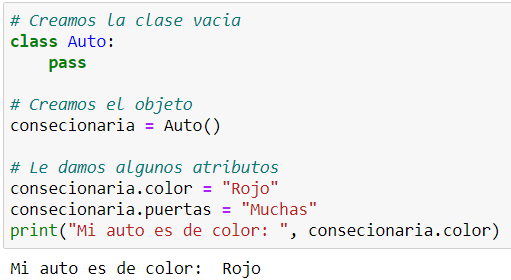


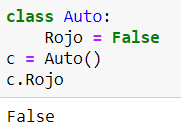


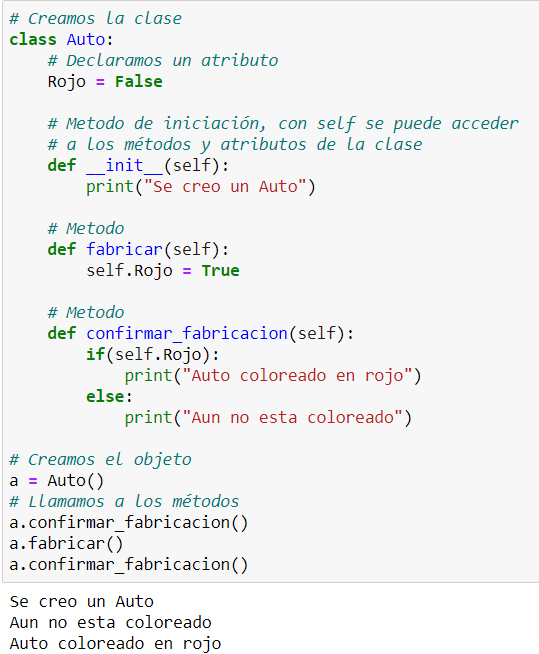
1. **POO (Programación Orientada a Objetos)**
   1. POO objetos y clases:

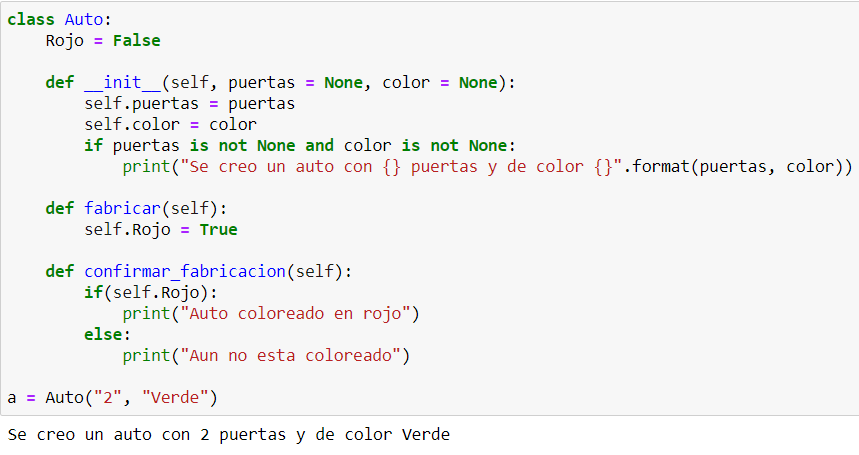


* 1. Atributos y Métodos de una clase:

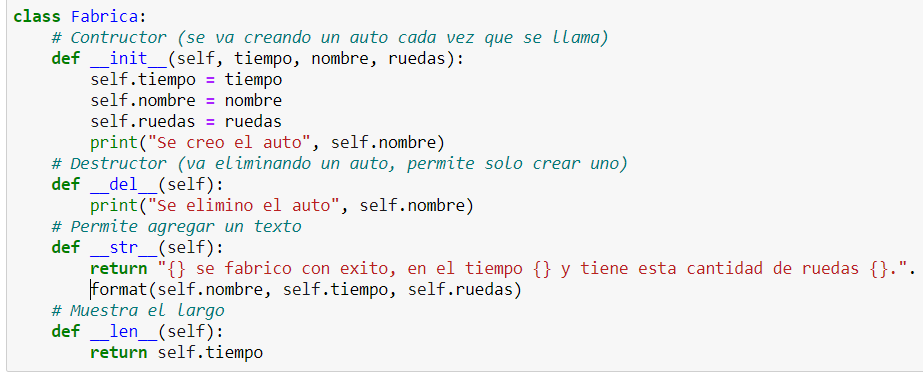






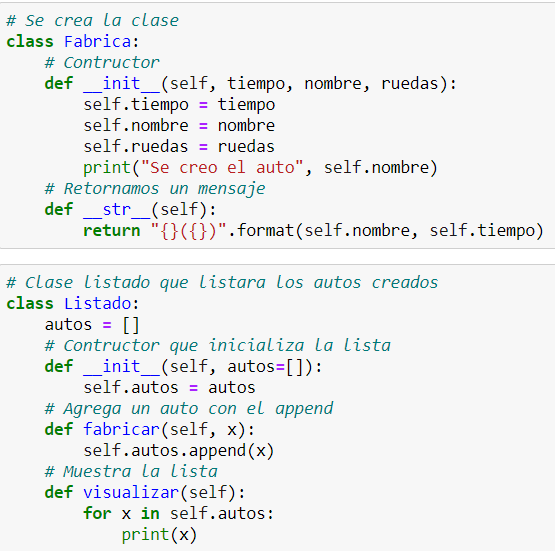


* 1. Clases – Métodos especiales:



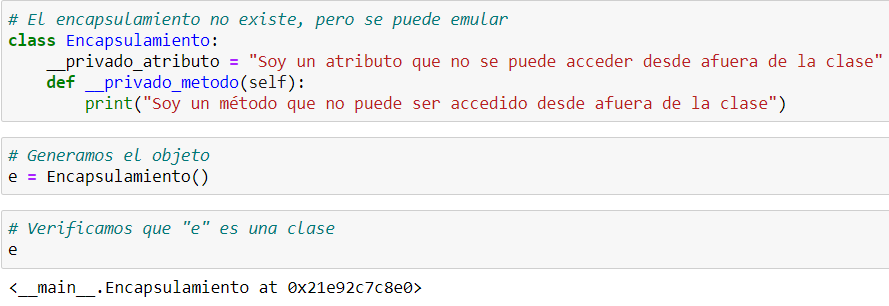


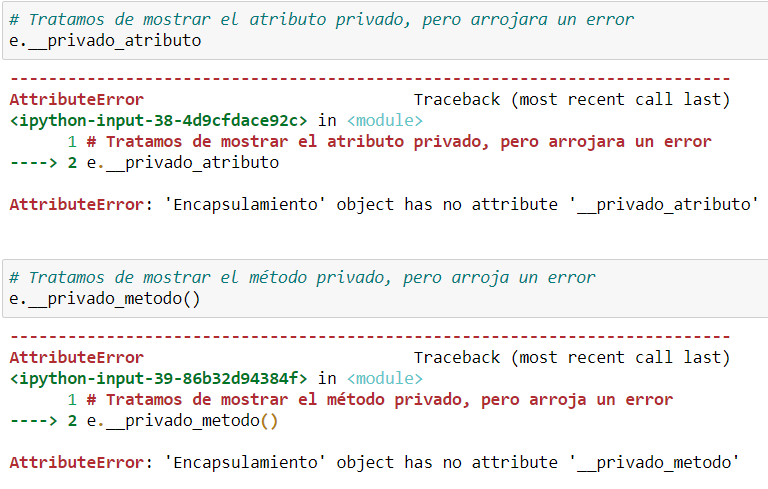
* 1. La utilidad de usar objetos embebidos:

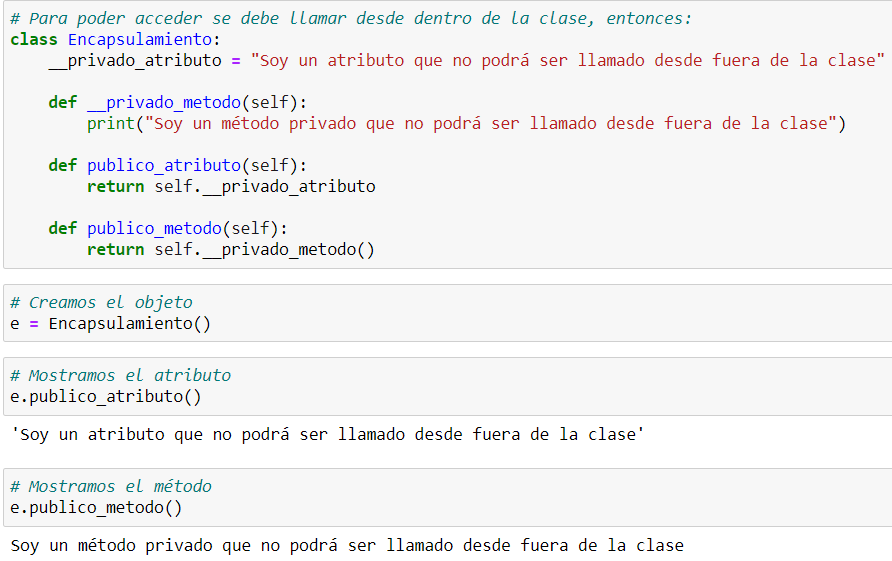




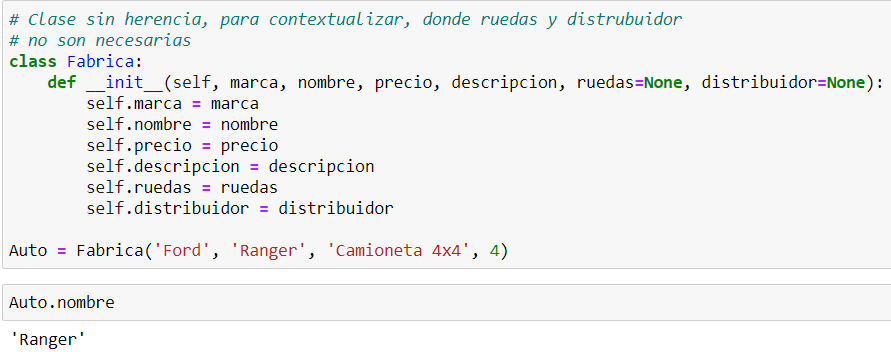
* 1. Encapsulamiento:



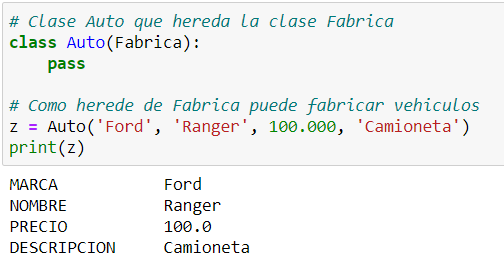




* 1. Herencia:

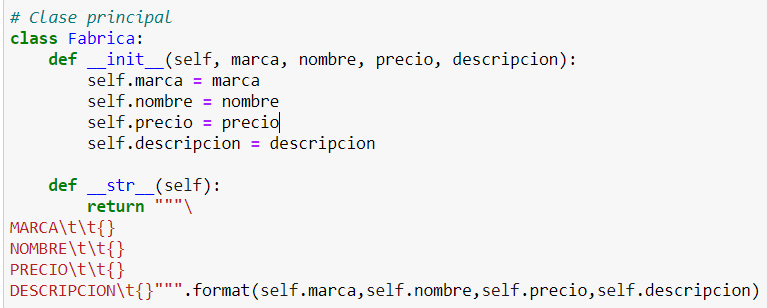




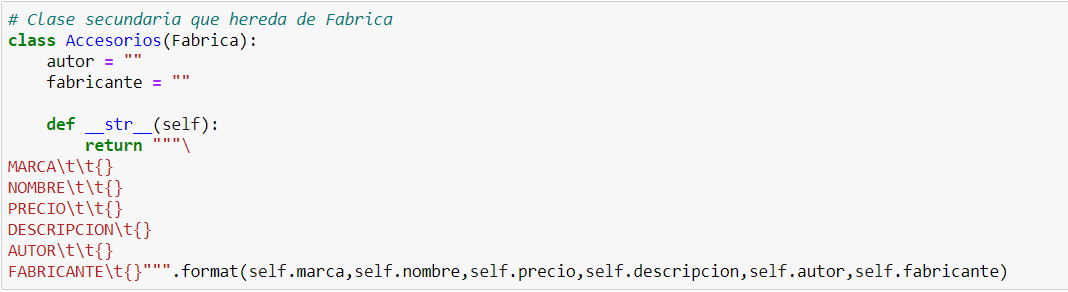




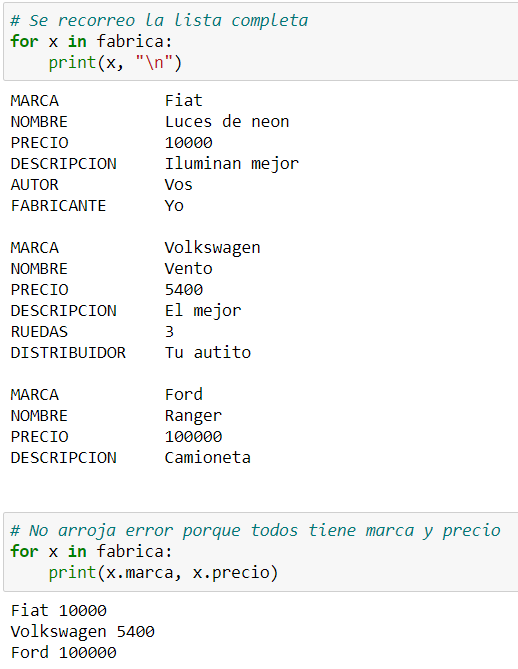
* 1. Clases heredadas y polimorfismo:

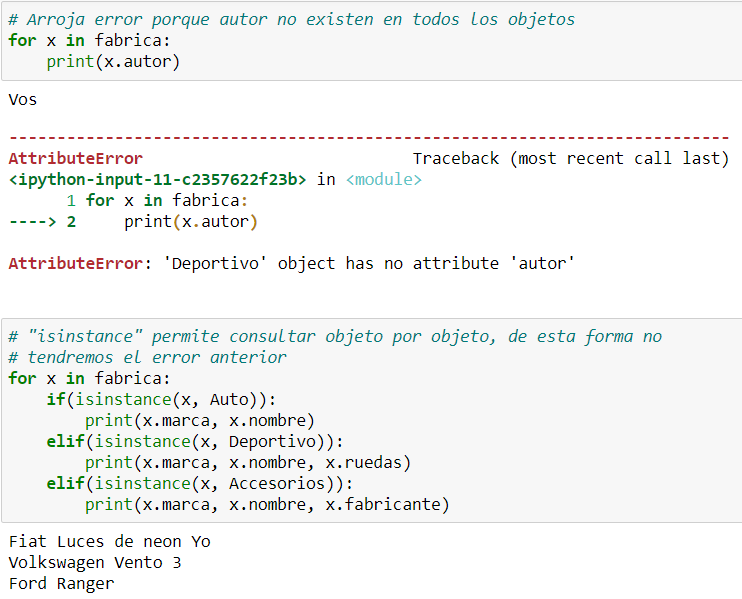


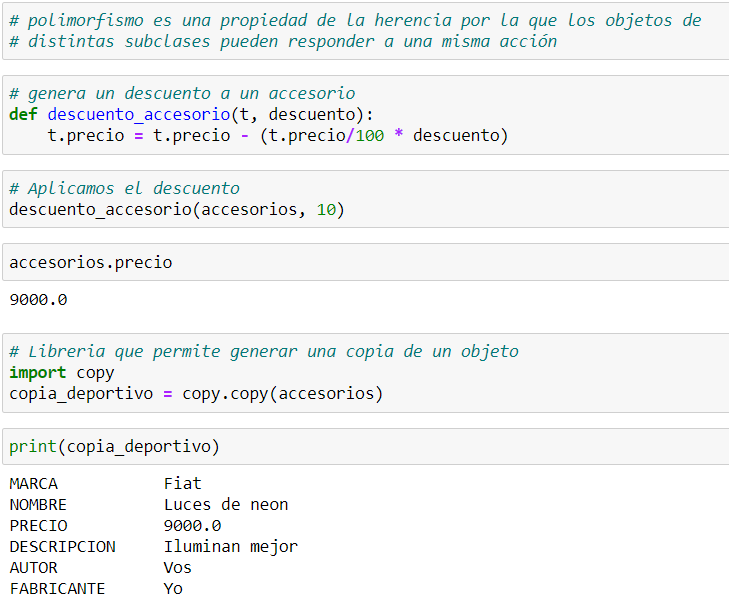




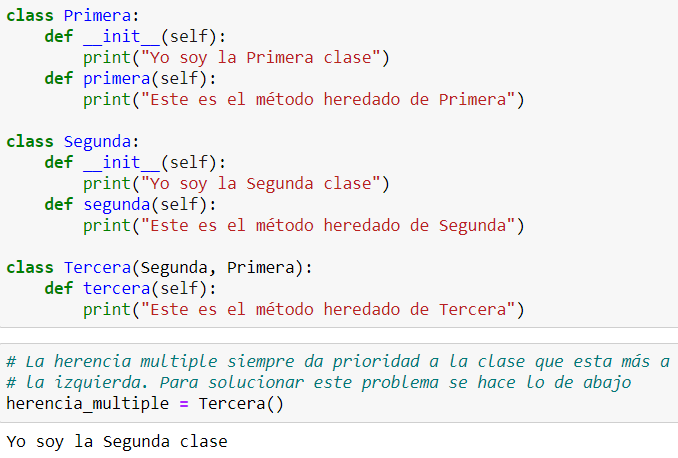


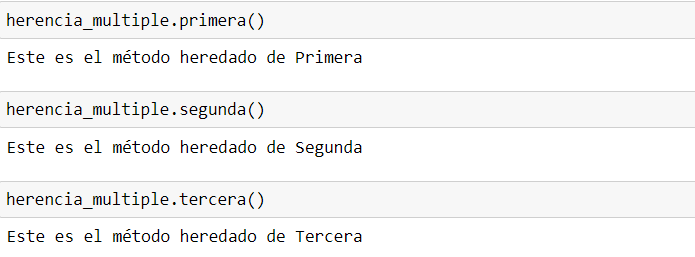






* 1. Herencias múltiples:

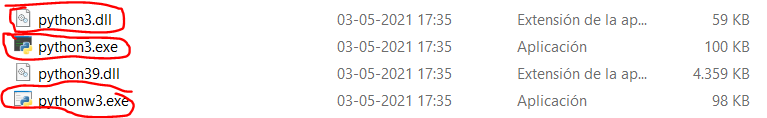




* 1. Como detectar dominios alojados en un Servidor:

**Seguridad**

1. **Preparar el entorno**
   1. Instalar Python 2 y 3 desde <https://www.python.org/downloads/>
   2. Guardar la ruta de instalación.
   3. Verificar al momento de la instalación que la ruta de Python 2 y 3 se agreguen a las variables de entorno.
   4. Renombrar los ficheros de Python 3,



* 1. Verificar las variables de entorno, si no existe crearlas,





* 1. Instalar Sublime Text.

1. **¿Cómo ejecutar archivos Python?**
   1. Ir a la carpeta donde se encuentra el archivo.
   2. Ejecutar con Python o Python3.
2. **Instalación de librerías principales a utilizar**
   1. Se utiliza **pip**:

**Pip install <libreria>**

**python -m pip install <librería>**

* 1. Instalar **requests**, **bs4** y **html5lib**:







1. **Vulnerabilidades en aplicaciones Web hechas en Wordpress**
   1. **¿Qué es WordPress y cuáles son sus principales vulnerabilidades?**

WordPress**:**

* + 1. Es un CMS o administrador de contenido.
    2. No se necesita saber programar.
    3. Es autoadministrable.
    4. Usa temas y plugin.

Vulnerabilidades

1. Su uso a nivel mundial expone a tu sitio a muchos ataques por bots o Netbots.
2. Los Plugin de terceros pueden contener errores y vulnerabilidades.
3. La ruta de Login (<https://www.pagina.cl/wp-login.php>) es conocida por todos los atacantes.
4. Al no necesitar grandes conocimientos para el uso puede ocasionar errores humanos.Al confiar en las actualizaciones automática de WordPress no se realizan los monitores constates de actualizaciones.
   1. **Escaneamos los temas de nuestros objetos**
      1. Conocer el tema de WordPress es una brecha de seguridad, ya que se puede buscar por Internet las vulnerabilidades de ese tema.
      2. La librería **requests** se dedica a las peticiones http con las páginas web. Se pueden usar métodos como GET, POST, PUT entre otros.
      3. **Bs4 import BeautifulSoup** permite analizar páginas html.



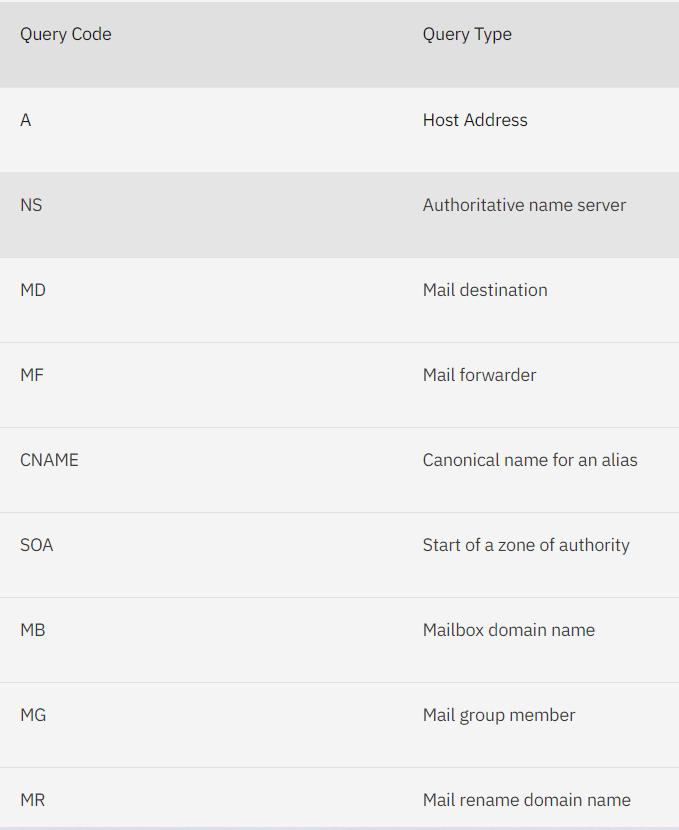
* 1. **Escanear los Plugins de nuestro objeto**

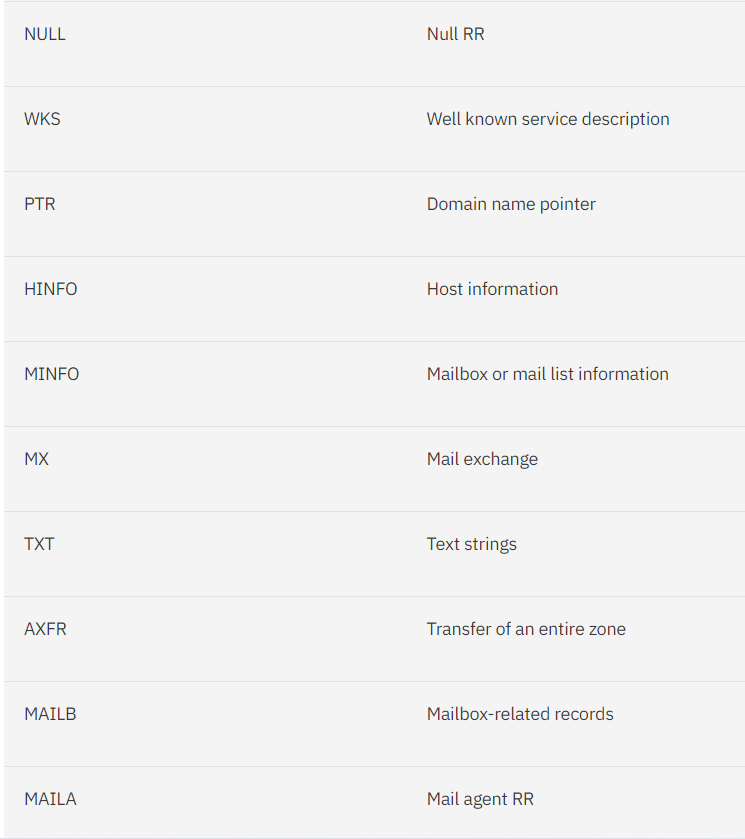


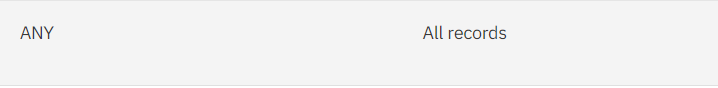
* 1. **Escanear los usuarios de nuestro objetivo**



1. **Recolectar información de servidores Web**
   1. Obtener información de nuestro objetivo a través de su DNS:











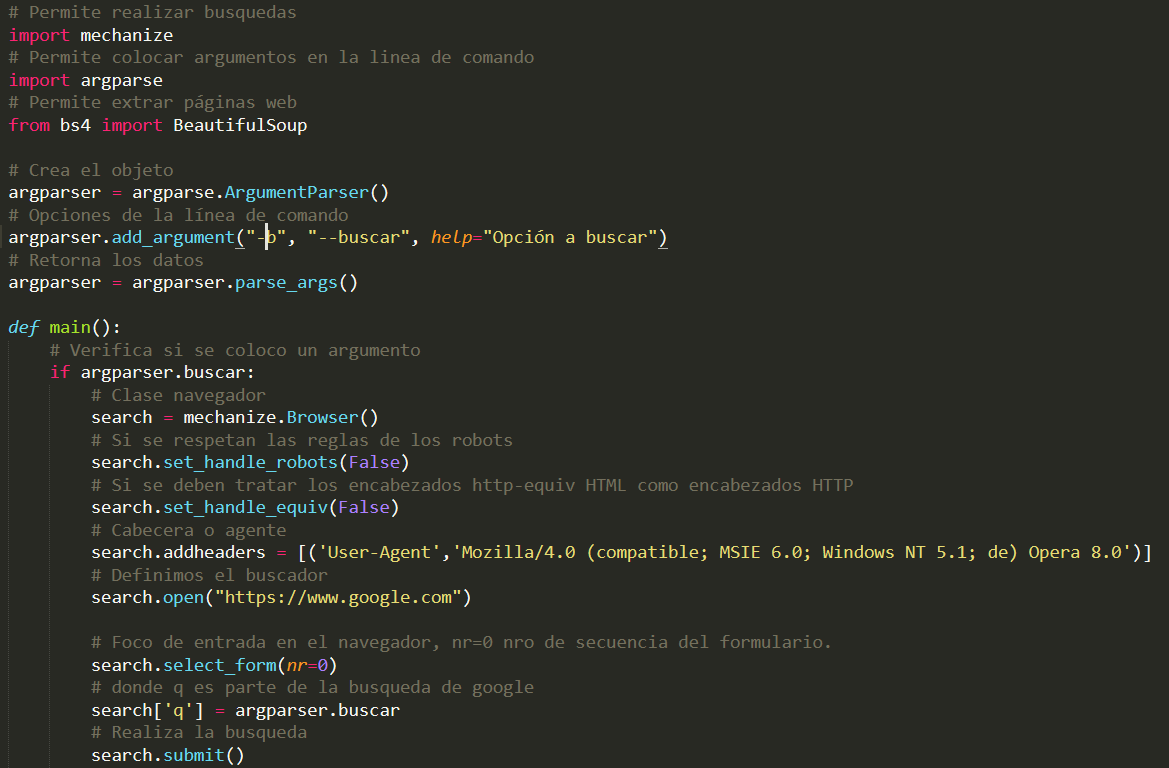
* 1. Obtenemos información del DNS con inversa lookup:
     1. Un servidor puede contener muchas páginas web.
     2. Se va a utilizar <https://viewdns.info/reverseip/>



* 1. Obtener información de la cabecera de la página WEB:



* 1. Búsqueda por Google con mechanize y web scraping:
     1. Mechanize permite interactuar con sitios web.
     2. Instalar con **pip install mechanize**

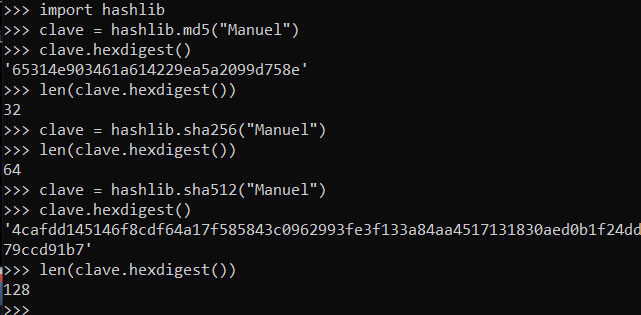




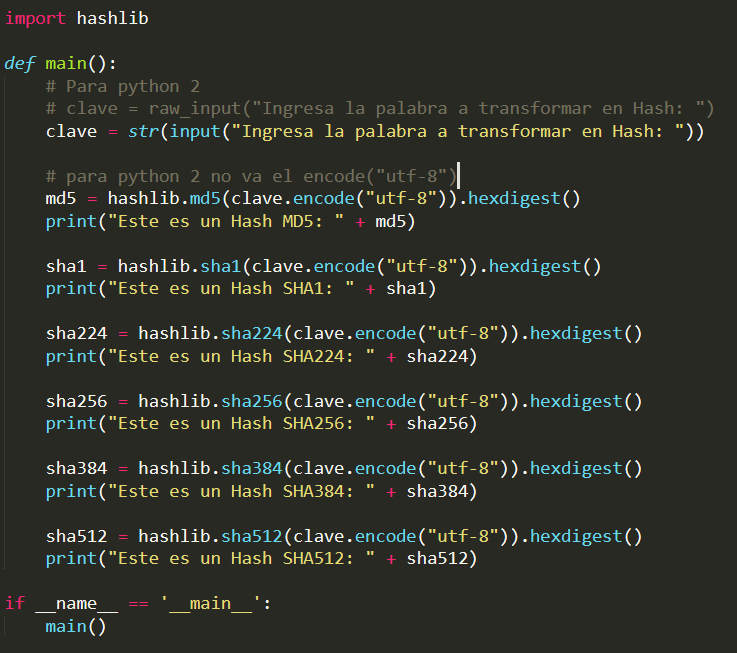
* 1. Geolocalizar servidor objetivo:



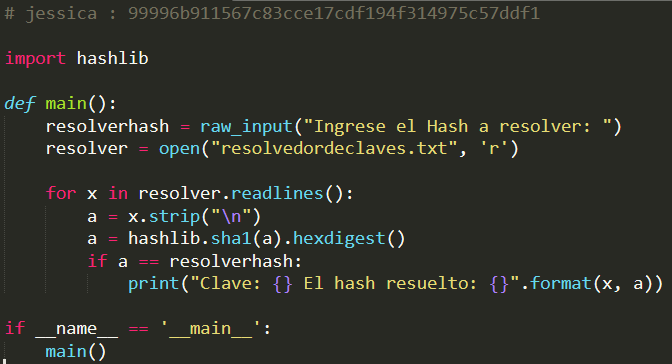
1. **Descifrando contraseñas – Rompiendo Hash**:
   1. Aprendemos sobre seguridad, que es un hash y como lo creamos con Python:
      1. Un Hash es una función matemática de resumen.
      2. Ejemplo de cómo crear alguno hash:



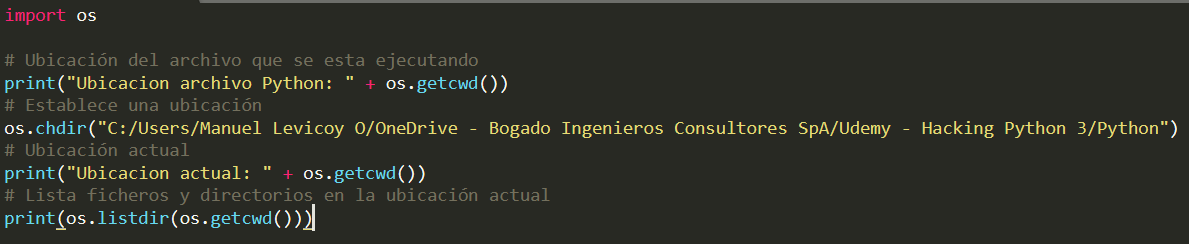
* 1. Crear diferente hash con Python:



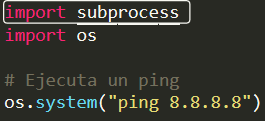
* 1. Romper Hash con fuerza bruta:

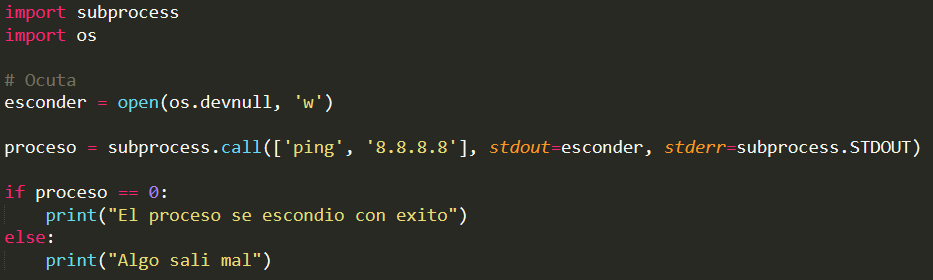


1. **Como usar Python en Sistemas Operativos**
   1. Herramientas para analizar SO:
      1. Se va a usar la librería OS desde <https://docs.python.org/es/3/library/os.html>

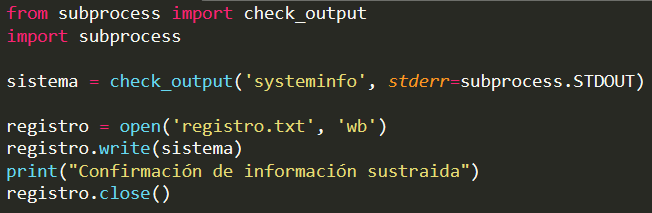


* 1. Crear procesos sin que sepa el usuario:

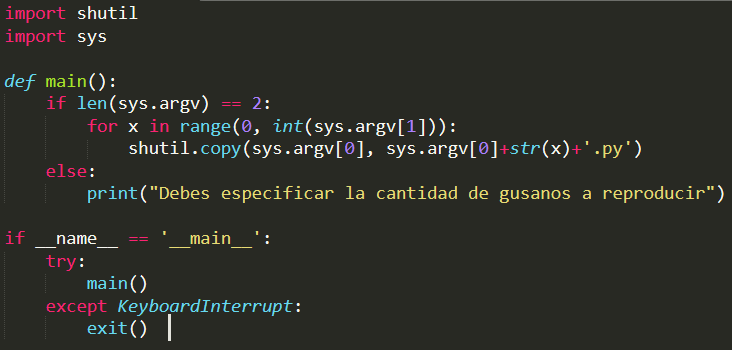




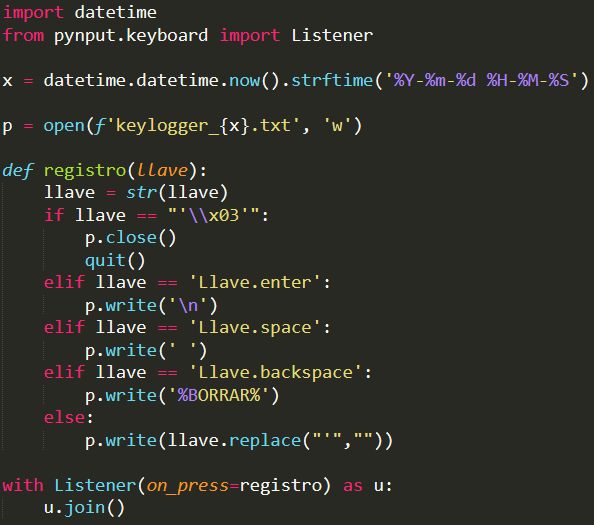
* 1. Extraer información del Sistema Operativo:



* 1. Realizar un gusano:

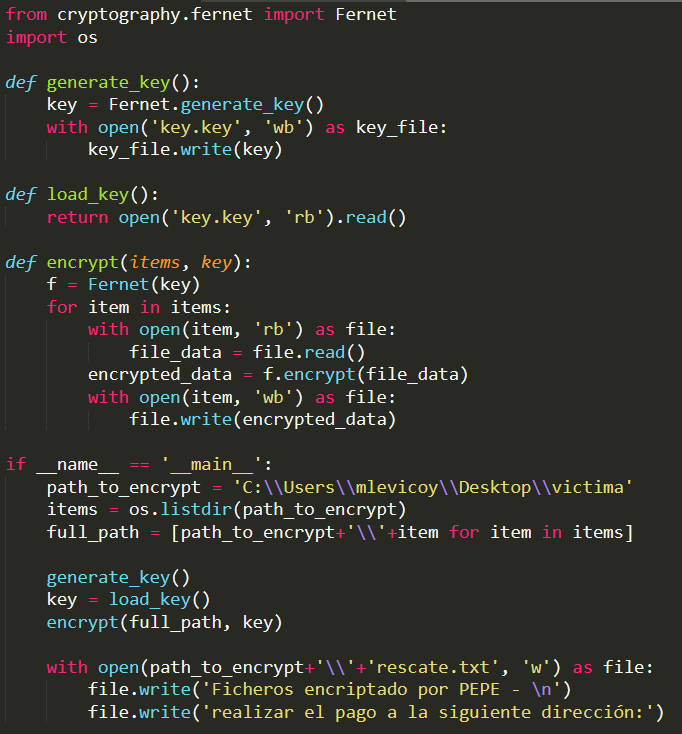


* 1. Keylogger simple:



* 1. Crear un ransomware:

Fichero “**encrypt.py**”:



Fichero “**decrypt.py**”:

