# Script Automatización Redes

Tabla de contenido

[Script Automatización Redes 1](#_Toc109342322)

[Elementos y ficheros para su funcionamiento 1](#_Toc109342323)

[Auto\_redes\_func5.py 2](#_Toc109342324)

[Librerías: 2](#_Toc109342325)

[Estructuras esenciales: 3](#_Toc109342326)

[Func\_lectura5.py: 4](#_Toc109342327)

[Glosario de funciones y variables 7](#_Toc109342328)

[Ejecución 8](#_Toc109342329)

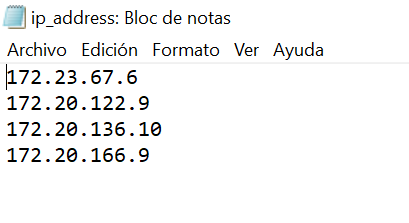
## Elementos y ficheros para su funcionamiento

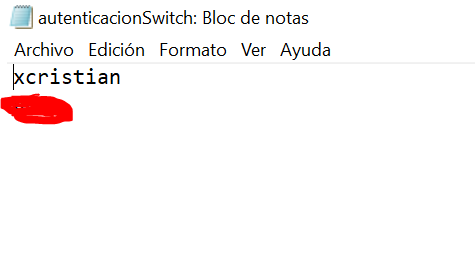
Este programa sirve para la automatización masiva de redes, para poder ejecutar de forma masiva comandos en elementos de red de forma rápida y eficiente de una sola ejecución. Este programa cuenta con dos archivos .py llamados:

*auto\_redes\_func5.py:* encargado de establecer la conexión ssh con los switches y contiene el bucle para la ejecución de comandos en cada switch.

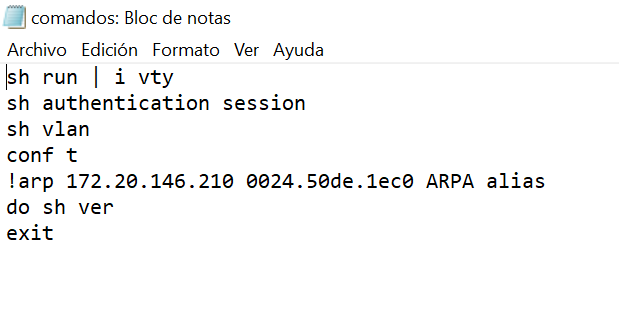
*func\_lectura5.py:* contiene las funciones de lectura de ficheros, lectura de logs, detección de pilas, puertos del switch...etc

Para el correcto funcionamiento, este script tiene 4 ficheros de texto:

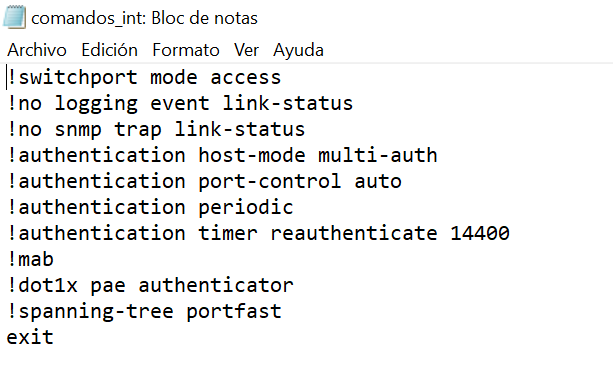
* ip\_address.txt: este fichero deberá contener las IPs de los dispositivos a los que se quiere acceder, una IP por línea.
* autenticacionSwitch.txt: aquí se deberá escribir el usuario (primera línea del fichero) y contraseña (segunda línea) para poder acceder a los dispositivos



* comandos.txt: aquí se deberán incluir los comandos globales a ejecutar en el switch, ya sea en modo ejecución o en modo configuración.



* comandos\_int.txt: este fichero tendrá los comandos a ejecutar en modo configuración, a nivel de interfaz (previamente el script habrá detectado cuantos puertos tiene el dispositivo y si existe una pila)

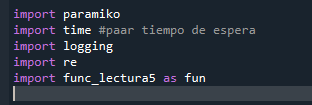


\*en este fichero hay que escribir exit como último comando

Estos ficheros de textos deben estar ubicados en la misma carpeta que los archivos .py

## Auto\_redes\_func5.py

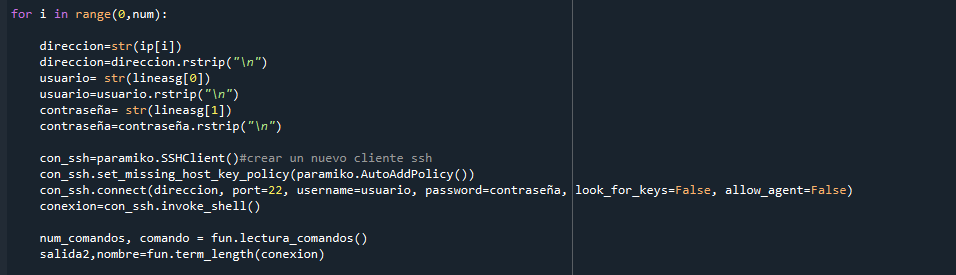
Librerías: la librería esencial para el correcto funcionamiento y poder interartuar con el swicth a través de ssh es paramiko. Otra librería esencial es la de logging, que permitirá la creación de logs de las salidas del CLI del switch. El resto de las librerías se necesitan para implementar alguna línea de código sin mayor importancia.



Destacar que la última línea es para importar las funciones del otro archivo .py, las llamadas a las funciones en este archivo deberán hacerse insertando fun. Al inicio de la línea.

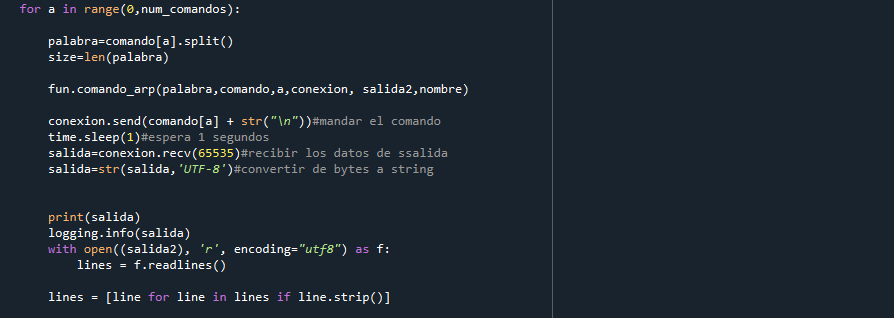
### Estructuras esenciales:

Bucle for de conexiones ssh



Este bucle permite la conexión ssh con los dispositivos del fichero que contiene las IPs, habiendo leído previamente el usuario y contraseña

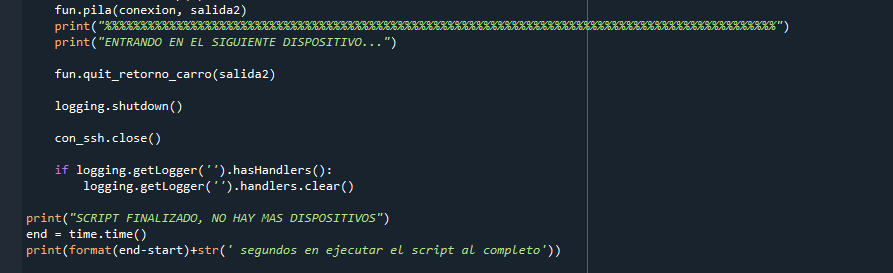
Bucle for de comandos



Este bucle ejecuta los comandos del fichero de comandos.txt uno por uno, y de forma secuencial. A continuación de este fragmento de código hay unos sentencias y condiciones “if” que dotan al script con la capacidad de entender si la ejecución de un comando ha finalizado. Por lo que, si la instrucción no ha terminado, no se ejecutará el siguiente comando.

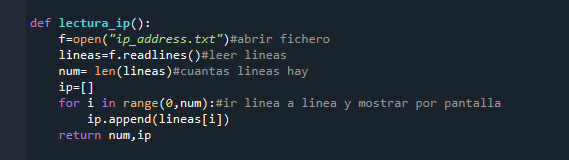
\*Cuando aparece la función logging.info() significa que lo que vaya entre paréntesis se guardará en el log, mientras que si pone print, solo se imprimirá por consola

Después se encuentra el siguiente fragmento de código, donde se llama a la función que se encarga de la detección de pilas, se eliminan los retornos de carro del log, se cierra la conexión ssh con el dispositivo y se eliminan las variables internas creadas de la sesión de logging.

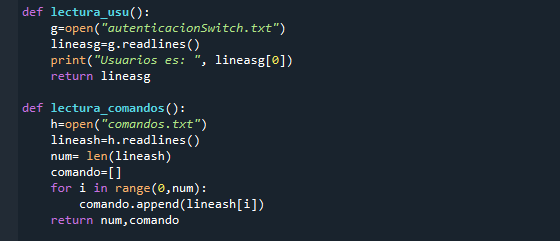


Func\_lectura5.py:

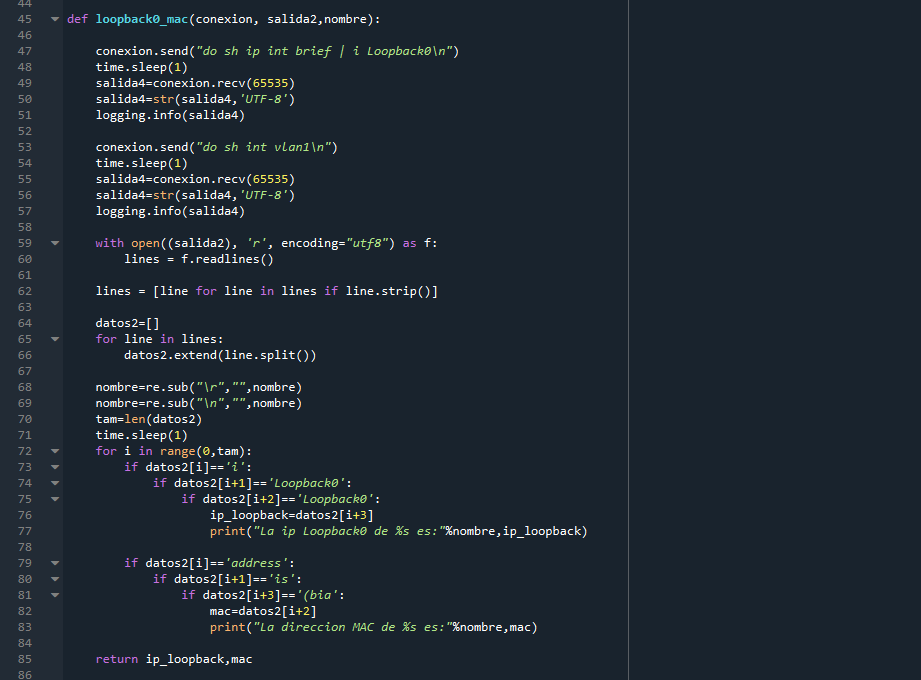
Este archivo de Python contiene varias funciones:



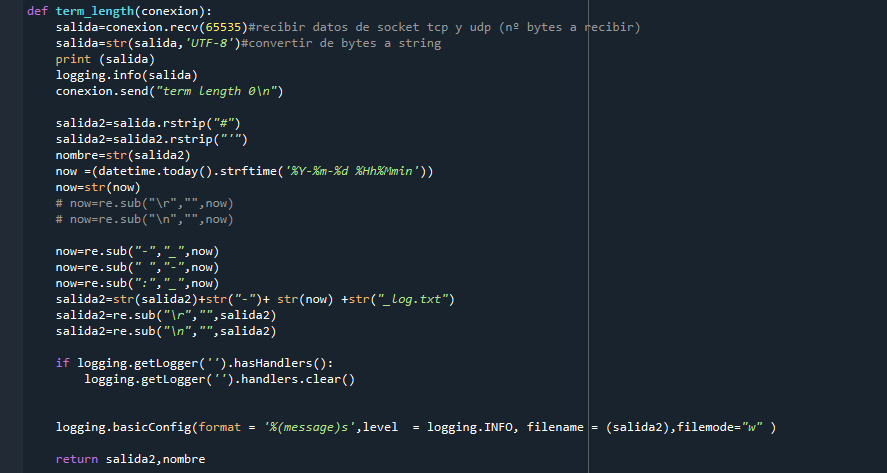
Esta función no tiene mucho misterio, lee cada línea del fichero de las direcciones IP y devuelve el número de direcciones que hay, y una lista con las propias IPs de cada switch



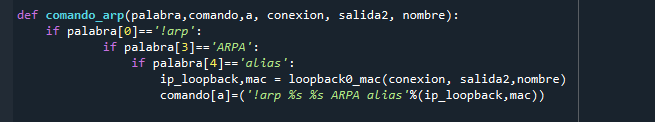
Las siguientes funciones son prácticamente idénticas, la diferencia es que leen las credenciales (usuario y contraseña) y los comandos globales. Cada línea se guardará en una lista.



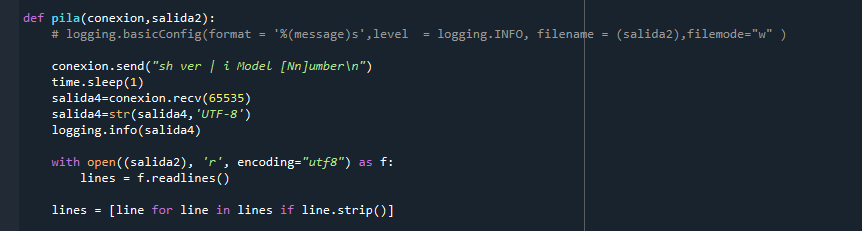
La siguiente función se utiliza para encontrar la IP de vlan1 y la dirección del Loopback0 para utilizar el comando arp…alias ARPA. Se ejecutan los comandos “do sh int vlan1” y “do sh ip int brief | i Loopback0” para obtener las respectivas direcciones. En las líneas 58 a 83, se separa las salidas en palabras y se hace un filtrado para encontrar las direcciones.



Esta función se encarga de ejecutar el comando term length para que las salidas no se queden a medias, obtiene el nombre del switch y crea la sesión logging con la respectiva fecha y hora.



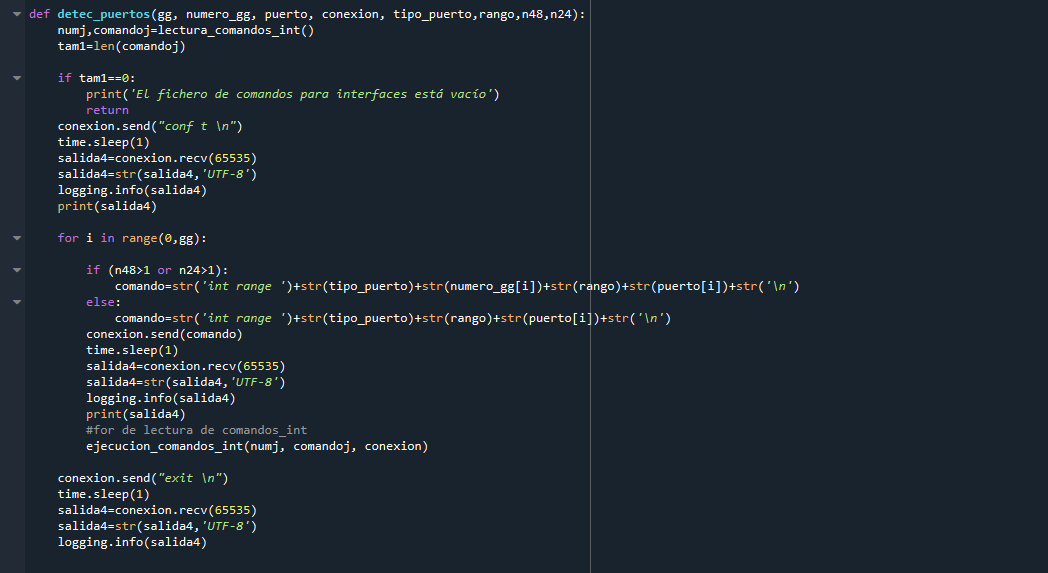
Aquí se detecta si existe el comando arp…ARPA alias en el fichero de comandos. Cabe destacar que la ip de vlan1 y loopback0 que se escriban en el fichero de comandos es indiferente, el script lo sustituirá por las direcciones correctas.



Esta función de detección de pila es la más extensa en cuanto a lineas de codigo, es la encargada de detectar cunatos switches existen, el numero de puertos y el tipo de estos (fast, giga..etc). El número de puertos y la cantidad de swicthes se detecta con el comando “sh ver | i Model [Nn]umber”. El resto de la función es simplemente tratamiento de cadenas y listas.

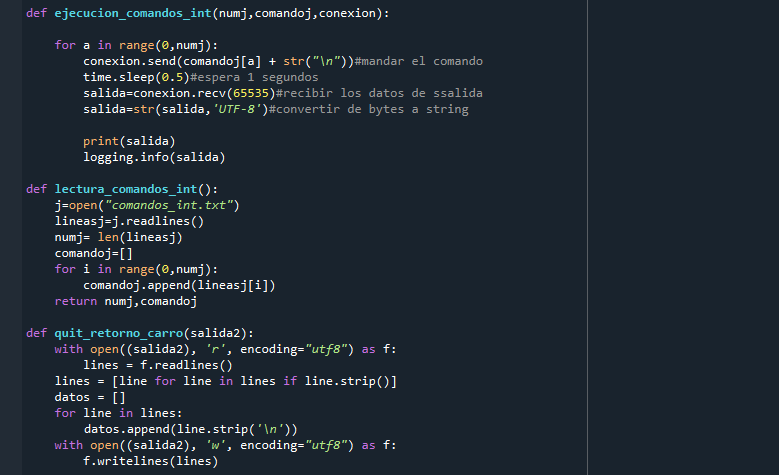
Destacar las siguientes variables:

* + puerto: es una lista en la que cada elemento indica el nuemro de puertos de cada switch, es decir, el primer elemento es eñ nº de puertos del primer switch y así sucesivamente. Ejemplo: [48,48,24,24]
  + gg: es el número total de switches, independientemente de los puertos que tengan
  + numero\_gg: es una lista con los numeros de cada switch. Ejemplo: [1,2,3…]



En esta función de la imagen anterior, es la encargada de ejecutar el comando para la selección del rango de interfaces de los switches de cada pila. Las variables n48 y n24 indican la cantidad de switches que hay con puertos 48 y 24 respectivamente que hay en pila.

En el caso de que haya pila, el comando que se ejecutará será del estilo int range g1/0/1-24. En caso contrario, una instrucción de este estilo, int range g0/1-24. Esta diferenciación se encuentra en el if y else que definen la variable comando.



En la imagen anterior tenemos la función de ejecución\_comandos\_int, que básicamente es un bucle para ejecutar los comandos del fichero comandos\_int.txt.

Lectura\_comados\_int, bucle encargado de guardar en una lista los comandos dedicados a las interfaces. La función quit\_retorno\_carro de primeras es una función bastante “fea”, abre el log y guarda cada línea de este en una lista, para cada elemento de la lista elimina el retorno “\n”, para después abrir el fichero en modo escritura “w” y sobrescribe el txt con las líneas sin el retorno de carro.

## Glosario de funciones y variables

A continuación se exponen variables y funciones que pueden dar lugar a confusión

* logging.info(x): función para guardar x en el log creado
* .readlines(): guardar en cada elemento de una lista las líneas de un objeto
* time.sleep(x): esperar x segundos en la ejecución del programa
* conexión.send: enviar un comando al CLI del switch (la variable conexión se ha referenciado como conexión ssh)
* conexión.recv: recibir la salida del CLI
* puerto: es una lista en la que cada elemento indica el nuemro de puertos de cada switch, es decir, el primer elemento es eñ nº de puertos del primer switch y así sucesivamente. Ejemplo: [48,48,24,24]
* gg: es el número total de switches, independientemente de los puertos que tengan
* numero\_gg: es una lista con los numeros de cada switch. Ejemplo: [1,2,3…]
* salida2: variable de tipo cadena que guarda el nombre de switch+fecha+hora de cuando se estableció la conexión ssh
* n48: cantidad de switches que hay en pila con 48 puertos
* n24: cantidda de switches que hay en pila con 24 puertos

## Ejecución

Para ejecutar el script es bastante sencillo, bastará con abrir en el editor Spyder los archivos .py, y ejecutar el archivo auto\_redes\_func5.py

También se puede ejecutar desde cmd ejecutando el comandos “python auto\_redes\_func5.py” en la carpeta donde esté ubicado el archivo