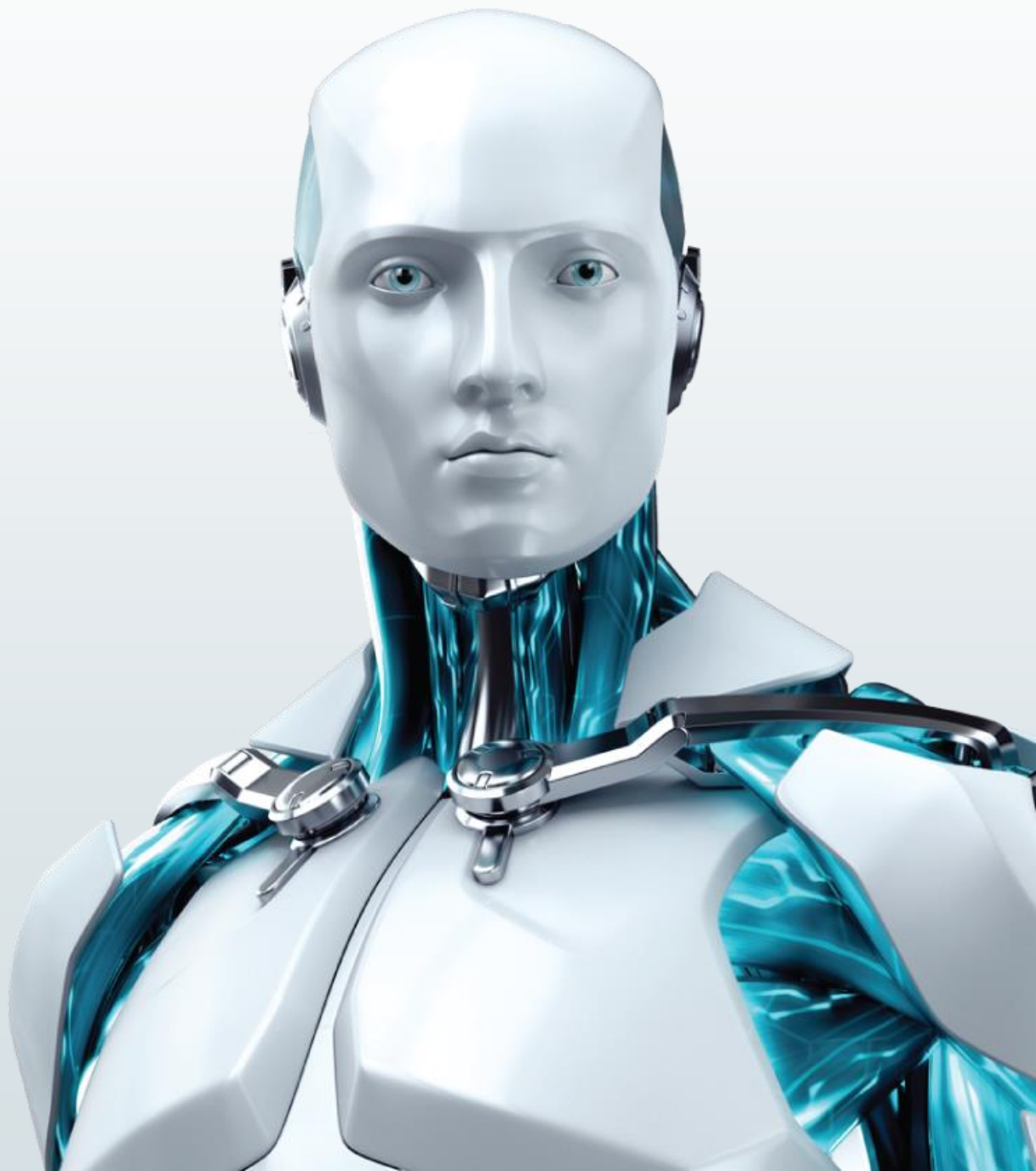


# Manejo de archivos



## Contenido

Manejo de archivos .....	3
¿Qué es un archivo? .....	3
Escribir en archivos .....	3
Leer de archivos .....	4
Leyendo la configuración desde un archivo .....	5
Conectando a la base de datos .....	6
Conexión a la base de datos .....	6
Accediendo a recursos web .....	7
Conclusión .....	7

# Introducción

Saber cómo utilizar archivos es un punto muy importante para cualquier programador. Aprender a manipular archivos de texto, conexiones a bases de datos, acceso a páginas web nos permite interactuar con otros sistemas, almacenar nuestra información para utilizarla en otro momento y utilizar datos persistentes cuando sea necesario.

Es por ello que en el presente módulo haremos una introducción a la escritura y lectura desde archivos en Python. Además veremos cómo conectarnos a una base de datos para hacer consultas y utilizar la información que en ella existe.

## Manejo de archivos

Leer y escribir Archivos, conectarnos a una base de datos para consultar o insertar información o saber de qué manera leer información ajena a nuestro código es muy importante para programar. Si en algún momento queremos guardar algún valor, dato, o información para tenerla accesible la próxima vez que ejecutemos nuestro programa, necesitamos saber cómo almacenarla de forma correcta.

Podemos utilizar los archivos de texto para cargar información en nuestro programa, para crear un registro de errores o lo que va haciendo o incluso como un archivo de configuración. Al momento de conectarnos a una base de datos, podemos agregar información, consultar por datos o actualizar.

## ¿Qué es un archivo?

Un archivo es una secuencia de bytes que se aloja en el disco rígido para almacenar información, datos, imágenes incluyendo también archivos ejecutables. Todo lo que se guarda en las computadoras son archivos, y de los formatos más diferentes que se puedan imaginar. Pero qué es lo que pasa cuando estamos escribiendo un programa y necesitamos cargar información que se encuentra en el disco, ¿cómo accedemos a esos datos?

Siempre que queramos leer o escribir en un archivo vamos a tener que decirle a nuestro programa dónde está ese recurso y de qué manera lo tenemos que abrir. Abrir un archivo en Python es mucho más sencillo de lo que ustedes se imaginan, solo debemos utilizar la función ***open()***, pasarle los parámetros correspondientes a la ruta o nombre del archivo y el modo con el que lo queremos abrir.

Para ver el detalle de cómo llamar a la función ***open()***, abran una consola de Python e ingresen ***“help(open)”***, podrán ver cómo pasar los parámetros a la función. En caso de que quieran un poco más de detalle sobre los modos de apertura de un archivo pueden ingresar en la misma consola el comando ***“file.\_\_doc\_\_”*** para que muestre por pantalla todo el detalle de los tres tipos de apertura de un archivo:

- ***r***: Abre el archivo en modo lectura (*read*) para poder leer su contenido.
- ***w***: Abre el archivo en modo escritura (*write*), para poder escribir en él.
- ***a***: Abre el archivo en modo de agregación (*append*), para continuar agregando contenido hacia el final del mismo

Además existe una opción para abrir los archivos en modo binario (***b***), y les queda a ustedes como tarea ver qué diferencia tiene esta opción con las anteriores.

## Escribir en archivos

Cuando desde nuestro programa, queramos escribir un archivo en el disco de la computadora debemos invocar a la función ***open()*** y pasarle como parámetro del modo de apertura la opción de escritura (***“w”***). Por ejemplo, podemos crear un archivo que se llame ***números.txt*** y escribir en él los números del 1 al 100:

```
escribir_archivos.py x
1 #Codigo de ejemplo para ver como escribir archivos en
  Python
2
3 #Vamos a crear un archivo numeros.txt y vamos a
  escribir en
4 #el los numeros del 1 al 100
5
6 archivo_salida = open("numeros.txt", "w")
7
8 for n in range(1,101):
9     archivo_salida.write("%d\n" % n)
10
11 archivo_salida.close()
```

Imagen 1 - Escribiendo en un archivo de texto

Primero utilizamos la función `open()`, con sus correspondientes parámetros para abrir el archivo, luego mediante un `for`, recorremos los números del 1 al 100 y escribimos el número en el archivo utilizando la función `write()`. Finalmente, una vez que salimos del bucle, cerramos el archivo llamando a la función `close()`. Siempre que se abra un archivo, debemos cerrarlo para evitar perder información.

¿Cómo creen que se podría hacer para modificar el archivo y agregarle los números del 101 al 200?

## Leer de archivos

Ahora que ya hablamos acerca de cómo escribir en archivos, tenemos que ir al otro extremo de la cuestión. Cuando guardamos información en el disco de la computadora, es porque en algún momento vamos a intentar acceder a ella. Para hacerlo, tenemos que saber cómo abrir un archivo para lectura y acceder a su contenido.

Para leer un archivo en Python, debemos abrirlo con el modo de lectura ("`r`") y luego podremos utilizar las funciones `read()`, `readline()` o `readlines()`.

Como se podrán imaginar, existen algunas diferencias entre las tres funciones que enumeramos antes para leer un archivo. La función `read()` lee, por defecto todo el archivo hasta alcanzar el EOF (End Of File), en caso de que sea necesario se le puede pasar el tamaño de bytes máximo a leer.

La segunda función que mencionamos, `readline()`, lee una línea completa del archivo. A esta función se le puede pasar como parámetro la cantidad de bytes a leer, si es menor a la longitud de la línea solo nos traerá esos bytes. Finalmente, `readlines()`, lee las líneas del archivo y nos devuelve un array con todas las líneas. Por ejemplo, el siguiente código lee un archivo de texto utilizando la función `readlines()` e imprime sus líneas por pantalla, avanzando de una a la vez y eliminando el carácter especial de nueva línea "`\n`":

```
leer_archivo.py x
1 #Codigo de ejemplo para ver como leer un archivo que es
2 # guardado en el disco
3
4 #Vamos a abrir el archivo numeros.txt y vamos a leer el
5 # imprimirlo por pantalla
6
7 archivo_entrada = open("numeros.txt", "r")
8
9 for linea in archivo_entrada.readlines():
10     print linea.strip("\n")
11
12 archivo_entrada.close()
```

Imagen 2 - Leer desde un archivo de texto

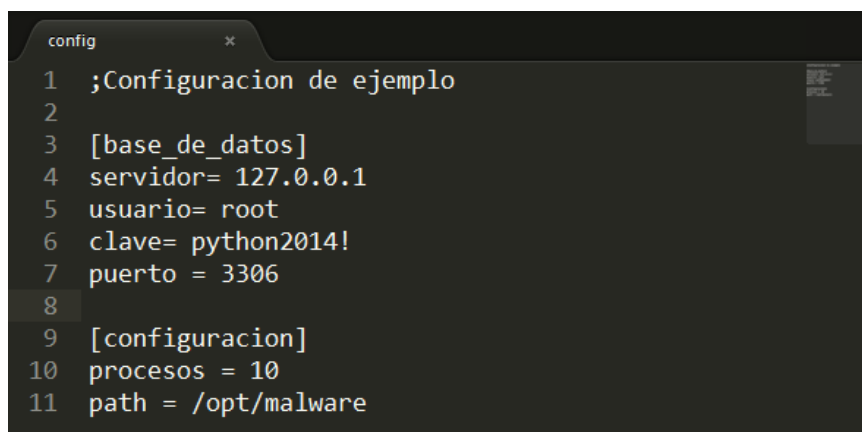
Con estas tres funciones, cualquiera que se ajuste al caso o la necesidad que tengamos podemos ir leyendo el archivo. Si en algún momento llegamos a necesitar posicionarnos en algún lugar del archivo, podemos utilizar la función **`seek()`**, averiguen cómo se utiliza.

## Leyendo la configuración desde un archivo

Además de las funciones para leer y escribir en archivos, existen algunos módulos de Python que nos ayudan para determinadas funciones. Si se llegaran a encontrar con la necesidad de leer información desde un archivo de configuración, el módulo de **`ConfigParser`** que incluye Python facilita mucho esta tarea.

Para darles un ejemplo, pueden observar el siguiente archivo de configuración que usaremos a modo de ejemplo:

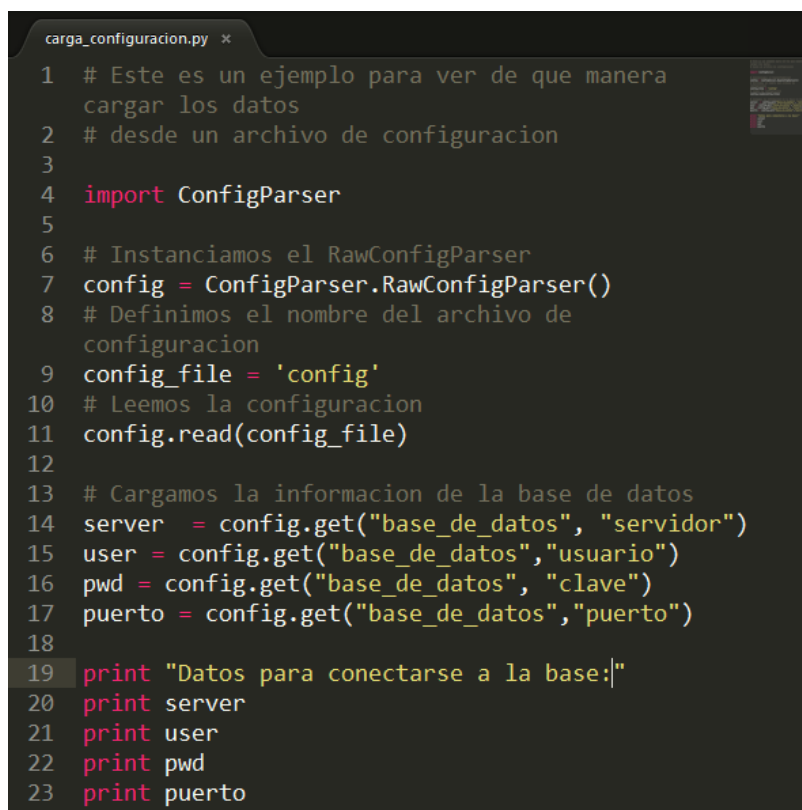
En ese archivo se especifican los parámetros de conexión a una base de datos, en donde se guardan los datos. Las secciones se separan entre “[ ]” y los parámetros y valores se separan con un “=”. Este archivo tiene dos secciones y en el siguiente ***script*** veremos cómo hacer para leer la información:



```
1 ;Configuracion de ejemplo
2
3 [base_de_datos]
4 servidor= 127.0.0.1
5 usuario= root
6 clave= python2014!
7 puerto = 3306
8
9 [configuracion]
10 procesos = 10
11 path = /opt/malware
```

*Imagen 3 - Archivo de configuración*

Para leer los datos de la configuración de ejemplo utilizaremos un pequeño programita que cargará valores específicos del archivo para luego imprimirlos por pantalla:



```
carga_configuracion.py x
1 # Este es un ejemplo para ver de que manera
  cargar los datos
2 # desde un archivo de configuracion
3
4 import ConfigParser
5
6 # Instanciamos el RawConfigParser
7 config = ConfigParser.RawConfigParser()
8 # Definimos el nombre del archivo de
  configuracion
9 config_file = 'config'
10 # Leemos la configuracion
11 config.read(config_file)
12
13 # Cargamos la informacion de la base de datos
14 server = config.get("base_de_datos", "servidor")
15 user = config.get("base_de_datos", "usuario")
16 pwd = config.get("base_de_datos", "clave")
17 puerto = config.get("base_de_datos", "puerto")
18
19 print "Datos para conectarse a la base:"
20 print server
21 print user
22 print pwd
23 print puerto
```

*Imagen 4 - Leyendo un archivo de configuración*

El **script** `carga_configuracion.py` leerá el archivo para luego imprimir por la pantalla los valores. En un sistema real, esto permitiría almacenar los datos fuera de nuestro código para que llegado el caso de tener que modificarlos, no sea necesario cambiar el código.

## Conectando a la base de datos

En la sección anterior, cuando hablamos acerca de cómo leer un archivo de configuración, presentamos un caso de ejemplo en el que se leía desde un archivo los datos para establecer una conexión con una base de datos. Ahora, nos vamos a ocupar de lograr esa conexión a una base de datos y realizar consultas.

Actualmente existen un gran número diferentes de bases de datos. ¿Qué es una base de datos? Una base de datos es una colección de datos que tienen un contexto en común y se almacenan para ser utilizados cuando sea necesario.

Para el desarrollo del curso, vamos a estar utilizando **MySQL**, para mostrar como conectarnos a una base de datos y algunas maneras sencillas de insertar nuevos valores, o realizar consultas. Para aquellos que se decidan por probar los **scripts** que utilizaremos durante estas partes del curso. Les queda como tarea del hogar el descargar e instalar el software de **MySQL** para tener acceso a él. Además, este no es el único desafío sino que también hay que instalar las librerías necesarias de Python para conectarse, en otras palabras tienen que instalar MySQLdb<sup>1</sup>.

Una vez que tengan acceso a la base de datos, y recopilen la información, podrán probar los **scripts** de esta sección.

## Conexión a la base de datos

A continuación se muestra un script que nos permite conectarnos a una base de datos de MySQL en el servidor local. El usuario se llama "python", y se autentica con el password "programando" a la base "curso\_python":

---

<sup>1</sup> Según su plataforma el software puede variar: <http://stackoverflow.com/questions/372885/how-do-i-connect-to-a-mysql-database-in-python>



```
base_datos.py x
1 import MySQLdb
2
3 bd = MySQLdb.connect(host="localhost", # El servidor
4                       user="python", # Tu usuario
5                       passwd="programando", # Tu pass word
6                       db="curso_python") # El nombre de la base
7
8 cur = bd.cursor()
9
10 # Ejecutamos una sentencia en la base
11 cur.execute("SELECT * FROM TU_TABLA")
12
13 # Imprimimos las columnas
14 for row in cur.fetchall() :
15     print row
```

Imagen 5 - Conexión a la base de datos

El *script* es bastante claro. Primero creamos la variable `bd` que representa la conexión a la base de datos y con la cual vamos a poder interactuar. Luego creamos un *cursor*, para poder ejecutar consultas a la base. Luego nos traemos todos los registros de la base “TU\_TABLA”. Finalmente, imprimimos todas las columnas por la pantalla.

Es recomendable tomarse un tiempo para armar y practicar qué otras instrucciones se pueden ejecutar, como así también entender qué hacen los métodos `execute()` y `fetchall()`.

Al tener una conexión a una base de datos podemos insertar, consultar o modificar los valores y parámetros de la base de datos, obviando que para lograr todo esto hace falta un poco de tiempo, práctica y dedicación.

## Accediendo a recursos web

Otro recurso que puede ser de interés para muchos de nosotros son las páginas web. No vamos a hablar acerca de desarrollo de sitios web y cosas por el estilo. Sin embargo, es importante que entendamos algunas maneras básicas de interactuar con sitios web, descargarlos y ver su contenido.

Para este cometido existen librerías que ya vienen incluidas en Python y otras que se pueden instalar como *BeautifulSoup*<sup>2</sup>, u otros *frameworks* de Python para interactuar con sitios web. El módulo que vamos a necesitar para interactuar con una página web es *urllib2*.

## Conclusión

A lo largo del presente módulo hemos repasado de qué manera interactuar con archivos de texto, vimos cómo utilizar funciones básicas para interactuar con archivos de configuración y bases de datos. Es una realidad que estamos hablando de la punta del iceberg y que existe mucho camino por recorrer. Un lenguaje de programación tan completo y extenso como es Python lleva tiempo de aprender y existe un número de librerías para las más remotas funcionalidades.

Python cuenta con librerías para *debuggers*, herramientas de *pentesting*, *fuzzers* y muchas cosas más. Si bien no las vamos a ver en detalle en este curso las vamos a mencionar más adelante en otros módulos. Ahora lo importante es practicar, leer de archivos e ingresarlos en una base de datos, leer de la base de datos y escribir los datos en un archivo.

---

<sup>2</sup> BeautifulSoup: <http://www.crummy.com/software/BeautifulSoup/>