

# Curso Python en 8 clases

## Clase 3: Archivos

**Autor:** Sebastián Bassi [sbassi@gmail.com](mailto:sbassi@gmail.com)

**Versión:** 1.0

**Licencia:** Creative Commons BY-NC-SA 2.5. ([ver texto completo](#))



## Archivos

En líneas generales los archivos se leen y se escriben con 3 pasos:

Lectura:

1. Abrir (open)
2. Leer (read, readlines, readline)
3. Cerrar (close)

Escritura:

1. Abrir (open)
2. Guardar (write)
3. Cerrar (close)

## Leyendo un archivo

La función open crea un *file handle* (una referencia a un archivo) sobre el que se puede usar para leer los datos:

```
open(filename[, mode[, bufsize]])
```

Ejemplo de generación de filehandler asociador al archivo *mi\_archivo.txt*.

```
fh = open('mi_archivo.txt', 'r')
```

La 'r' indica "modod de lectura" y es el modo por defecto.

Sobre el file handle se puede:

- read(n): Lee n bytes, por defecto lee el archivo entero.
- readline(): Devuelve str con una sola línea
- readlines(): Devuelve una lista con str por cada línea

Distintas maneras de leer un archivo:

```
fh = open('archivo.txt')
contenido = fh.read()
print contenido

fh = open('archivo.txt')
contenido = fh.readlines()
print contenido
```

```

contenido = ''
fh = open('archivo.txt')
while True:
    line = fh.readline()
    contenido += line
    if line=='':
        break
print contenido

# Para todos:
fh.close()

```

Apertura secuencial de un archivo:

```

fh = open('archivo.txt')
contenido = ''
for line in fh:
    contenido += line
fh.close()

```

Leyendo con 'with' (Python 2.6 en adelante)

Manera genérica:

```

with EXPRESION as VARIABLE:
    BLOQUE DE CODIGO

```

Ejemplo:

```

with open('archivo.txt') as fh:
    for line in fh:
        print line

```

## Escribiendo archivos

Modos de escritura:

- w: Write, graba un archivo nuevo, si existe, borrarlo.
- a: Append (agregar), agrega información al final de un archivo pre-existente. Si no existe, crea uno nuevo (uso típico: logs).

Ejemplo:

```

fh = open('/home/yo/archivo.txt', 'w')
fh.write('1\n2\n3\n4\n5\n')
fh.close()

```

## Archivos CSV (Comma separated file)

Este tipo de archivos es un "estandar de facto" para muchas aplicaciones, incluyendo planillas de cálculo.

Archivo CSV:

```
Sebastián,Perez,33,23566777
Jose,Martinez,23,42121329
Karina,Gonzalez,32,24159857
Maria Laura,Yaños,19,43852144
```

Archivo CSV con otro separador (;):

```
Sebastián;Perez;33;23566777
Jose;Martinez;23;42121329
Karina;Gonzalez;32;24159857
Maria Laura;Yaños;19;43852144
```

## Leyendo CSV sin usar módulo CSV

Con las herramientas vistas hasta este momento podemos procesar dichos archivos:

```
fh = open('archivo.txt')
for line in fh:
    linea = line.split(',')
    # procesar elementos:
    nombre = linea[0]
    apellido = linea[1]
    # etc, etc
fh.close()
```

Aunque Python tiene un módulo especialmente preparado para esto:

## CSV con módulo CSV

Uso del módulo CSV:

```
import csv
lineas = csv.reader(open('fn.csv'))
for line in lineas:
    nombre = line[0]
    apellido = line[1]
Cambiando separador (delimitador):
lineas = csv.reader(open('fn.csv'), delimiter=';')
```

Dialectos:

No todos los CSV son hechos iguales, hay diferencias sutiles como las comillas en los campos de texto y otras diferencias menores que agregan algunos programas.

```
>>> csv.list_dialects()
['excel-tab', 'excel']
```

## Escribiendo archivos CSV

```
>>> import csv
>>> spamWriter = csv.writer(open('eggs.csv', 'w'), delimiter=' ',
...                          quotechar='|', quoting=csv.QUOTE_MINIMAL)
>>> spamWriter.writerow(['Spam'] * 5 + ['Baked Beans'])
>>> spamWriter.writerow(['Spam', 'Lovely Spam', 'Wonderful Spam'])
```

## Lectura y escritura de Excel: xlrd y xlwt

Estos módulos no vienen con Python, pero pueden ser instalados fácilmente con *Easy\_install* (<http://pypi.python.org/pypi/setuptools>).

## Leyendo Excel

Programa que usa **xlrd** para leer la primera hoja de una planilla llamada `sampladata.xls` que contiene 2 columnas:

```
import xlrd
iedb = {} # Empty dictionary
book = xlrd.open_workbook('sampladata.xls')
sh = book.sheet_by_index(0)
for i in range(1,sh.nrows): #skips fist line.
    iedb[sh.cell_value(rowx=i, colx=1)] = \
        sh.cell_value(rowx=i, colx=2)
```

## Creando planillas Excel

Este programa crea una planilla usando **xlwt**:

```
import xlwt
list1 = [1,2,3,4,5]
list2 = [234,267,281,301,331]
wb = xlwt.Workbook()
ws = wb.add_sheet('Primera hoja')
ws.write(0,0,'Column A')
ws.write(0,1,'Column B')
i = 1
for x,y in zip(list1,list2): #Walk 2 list @ once.
    ws.write(i,0,x) # Fila, Columna, Datos.
    ws.write(i,1,y)
    i += 1
wb.save('mynewfile.xls')
```

## Pickle: Persistencia simple de datos

Cualquier dato (variable) en Python puede grabarse a disco usando Pickle o cPickle. La única diferencia entre Pickle y cPickle es que este último está escrito en C y por lo tanto es mas rápido.

## Grabar

```
>>> spdict = {'blue':'azul','red':'rojo'}
>>> spdict
{'blue': 'azul', 'red': 'rojo'}
>>> import cPickle
>>> fh = open('spict.data','w')
>>> cPickle.dump(spdict, fh)
>>> fh.close()
```

## Leer

Con *load* "levantamos" el objeto que habia sido almacenado con *dump*:

```
>>> spdict = cPickle.load(open('spict.data'))
>>> spdict
{'blue': 'azul', 'red': 'rojo'}
```

## Mas información

- Archivos: <http://www.devshed.com/c/a/Python/File-Management-in-Python>
- Pickle: <http://docs.python.org/library/pickle.html>