Archivos

Apuntes versión 1.0

Víctor Araya Sánchez

2017

Objetivos de este apunte

Al terminar de leer, hacer los ejercicios propuestos de este apunte y trabajar en clases serás capaz de:

- Leer información alojada en archivos de texto plano a través de programas desarrollados en el lenguaje de Programación Python.
- Almacenar información en archivos de texto plano, resguardado criterios de orden y buenas prácticas, a través de programas desarrollados en el lenguaje de Programación Python.

1. Introducción

El presente apunte tiene como finalidad presentar las principales formas de manipular archivos de texto planos.

Para comenzar es bueno saber qué es un archivo y conocer qué de qué tipos de archivo existen.

> **archivo**
>
> Del lat. *archīvum,* y este del gr. ἀρχεῖον *archeîon.*
>
> **1.** m. Conjunto ordenado de documentos que una persona, una sociedad, una institución, etc., producen en el ejercicio de sus funciones o actividades.
>
> **2.** m. Lugar donde se custodian uno o varios **archivos.**
>
> **3.** m. Acción y efecto de **archivar** (‖ guardar documentos en un **archivo**). *Entregó la documentación para proceder a su archivo.*
>
> **4.** m. Acción y efecto de **archivar** (‖ dar por terminado un asunto). *El juez ordenó el archivo del caso.*
>
> **5.** m. *Inform.* Conjunto de datos almacenados en la memoria de una computadora que puede manejarse con una instrucción única.
>
> **6.** m. *Col.* **oficina.**
>
> **7.** m. p. us. Persona a quien se confía un secreto o recónditas intimidades y sabe guardarlas.
>
> **8.** m. p. us. Persona que posee en grado sumo una perfección o conjunto de perfecciones. *Archivo de la cortesía, de la lealtad.*

En esta ocasión nos fijaremos en la quinta acepción, propia de la informática. Desde esta perspectiva un archivo corresponde a una agrupación de datos, de distintos tipos, organizados bajo una

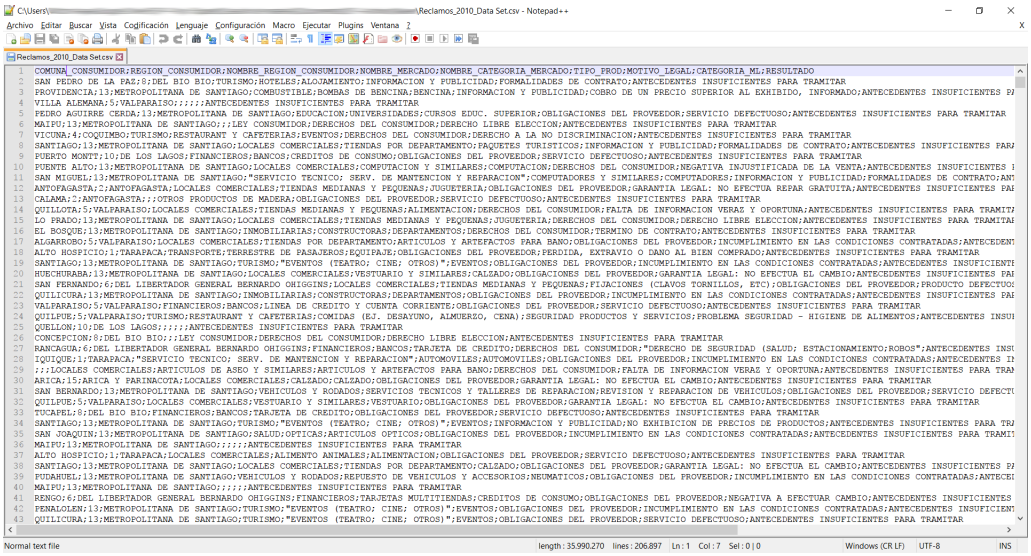


Figura 1: Esto es un ejemplo de un archivo de texto plano, abierto en Notepad++

etiqueta o nombre que lo identifica.

Hay distintas formas de clasificar archivos, por ejemplo, por el software que los utiliza (que podría ser equivalente a la clasificación por extensión del archivo), por el tiempo de información que contiene, por la finalidad que cumple, entre otras posibilidades.

En particular nos preocuparemos por abordar un tipo particular de archivo: los archivos de texto plano. Estos archivos almacenan información en forma de caracteres legibles para el ser humano, sin inclusión de elementos de formato estético o de orden adicional (como fuentes en negrita, itálica, alineación de texto, entre otros), sin elementos multimediales (como enlaces, imágenes, audios, videos) ni ninguna otra incorporación especial.

Es regular que los archivos de texto plano se almacenen con extensión .txt, pero eso no significa que sea la única extensión posible. Un ejemplo concreto de ello son los archivo .py, los cuales son scripts (o programas) escritos en Python que corresponden también a texto plano. De igual forma archivos con extensión .xml, .html y muchos otros.

En particular, estos últimos dos tipos específicos de archivos de texto plano codifican la información en modo de etiquetas (palabras o frases encerradas en <>), y se utilizan para el intercambio de información entre sistemas.

A lo largo de este apunte haremos énfasis en algunas tareas específicas que se pueden realizar con archivos, particularmente:

- Abrir y cerrar un archivo
- Leer un archivo
- Escribir en un archivo nuevo
- Añadir información a un archivo existente

2. Apertura y cerrado de archivos

Antes de poder trabajar con un archivo es necesario abrirlo. Abrir un archivo permite la manipulación de la información contenida dentro del archivo de distintas formas, como explicaremos a continuación.

Para abrir un archivo utilizaremos la función nativa `open` como sigue:

```
archivo = open("dirección/y/nombreDelArchivo.extensión", "modo")
```

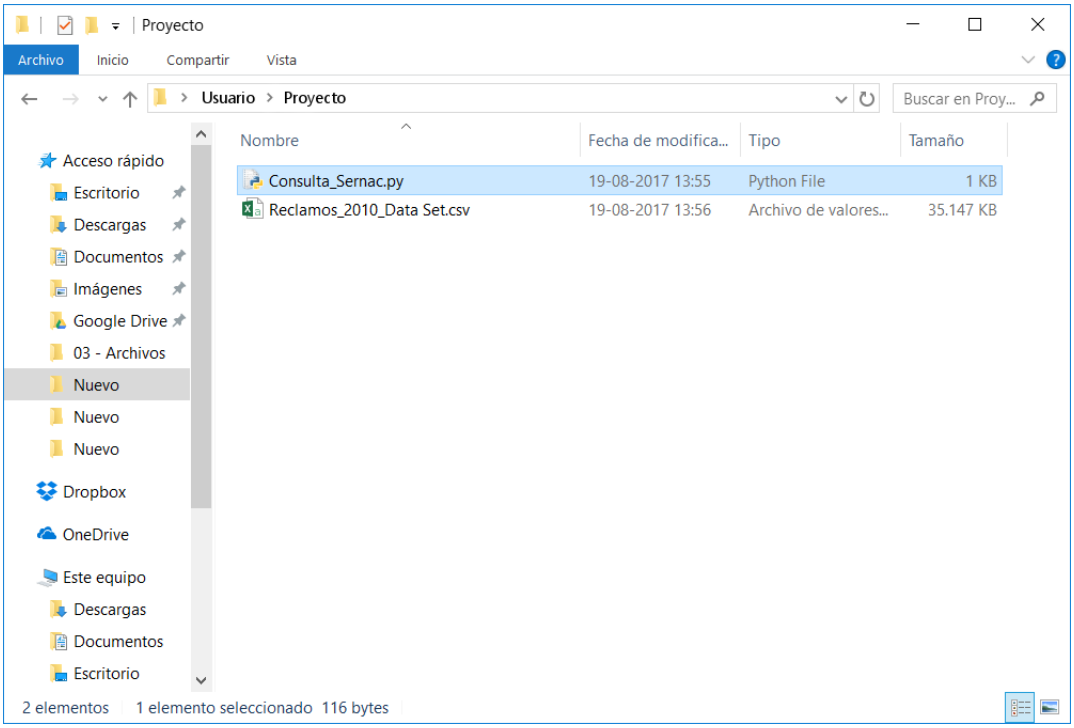
Donde:

- `archivo` es una variable que servirá como referencia para el archivo con el que trabajaremos, es decir, cada vez que queramos trabajar con el archivo deberemos trabajar con la variable asignada a este.
- `dirección/y/nombreDelArchivo.extensión` corresponde, literalmente, a la dirección donde se ubica el archivo y al nombre de este, incluyendo su extensión. Es importante notar que tanto este argumento como el que sigue son `strings`, por ende, van entre comillas.
- `modo` es lo que explicita de qué forma queremos trabajar con el archivo. A efectos del curso consideraremos tres posibles modos:
  - `r` – read, permite leer el archivo, no así modificarlo (es el "modo sólo de lectura")
  - `w` – write, permite escribir en el archivo (Si el archivo no existe crea uno nuevo, si ya existe lo sobre escribe perdiendo toda la información que tuviese antes)
  - `a` – append, permite añadir información al archivo, partiendo desde el final de este (al igual que w, con este modo si el archivo no existe se creará, pero en caso que ya existiera añadirá información a la ya existente en vez de partir desde un archivo en blanco)

3. Rutas de archivo

Haremos una pausa para precisar sobre la ubicación de un archivo y como dar su ruta o path.

Supongamos que estamos trabajando en la carpeta **Proyecto**, en dicha carpeta tenemos alojado nuestro script llamado **Consulta_Sernac.py** y un archivo de texto llamado **Reclamos_2010_Data Set.csv**, tal como se ve en la imagen. El archivo .csv puedes encontrarlo en:
http://datos.gob.cl/dataset/3162.



Para nombrar al archivo .csv[1] al querer abrirlo, desde nuestro programa, existen dos formas de hacerlo; a través del *path absoluto* y a través del *path relativo*.

El path absoluto corresponde a la ruta absoluta, textual y completamente detallada en la que se encuentra el archivo. En el ejemplo, la carpeta proyecto está en la carpeta de usuario, por ende, su ruta o path absoluto es **C:/Users/Usuario/Proyecto/Reclamos_2010_Data Set.csv**[2]. Como notarás, la ruta parte desde la letra de la unidad en que se ubica el documento, pasando por cada carpeta en la que se aloja, hasta llegar al nombre mismo del archivo. Esta forma de nombrar archivos o directorios (carpetas) es muy precisa, no obstante, es extensa y puede ser muy compleja de trabajar cuando se trabaja en dispositivos extraíbles como pendrives.

El path relativo, como su nombre lo indica, describe la posición de un archivo respecto a la posición en que me encuentro actualmente. En nuestro ejemplo la ruta o path relativo de nuestro archivo .csv es **./Reclamos_2010_Data Set.csv**, donde . simboliza la carpeta actual. Si dentro de la carpeta **Proyecto** hubiera una carpeta **Documentos** y en su interior estuviera el archivo csv, entonces la ruta sería **./Documentos/Reclamos_2010_Data Set.csv**. Es importante aclara que la carpeta desde donde se hace la referencia es la que aloja el script o programa, en nuestro caso la carpeta **Proyecto**.

[1] Es importante notar que, a pesar que un archivo **.csv** se visualiza como disponible para abrir con Microsoft Excel, en esencia es un archivo de texto plano. Para verificarlo puedes dar botón secundario sobre él y utilizar la opción **abrir con** del menú contextual, escogiendo abrir con **Block de notas** o **Notepad** según el idioma en que esté tu sistema operativo. ¿Qué pasa si haces el mismo con un archivo con extensión **.xls** o **.xlsx**? (**Precaución:** Desmarca la casilla que dice "Usar siempre esta aplicación para abrir los archivos…", de lo contrario tus archivos de Microsoft Excel seguirán abriéndose siempre con block de notas).

[2] Al revisar la ruta desde el explorador de Microsoft Windows notarás que se usa \ en vez de /, por lo que es necesario hacer el cambio para que Python lo interprete apropiadamente.

Ejemplo

Si quisiéramos abrir el documento **Reclamos_2010_Data Set.csv** en modo lectura escribiremos:

```
archivo = open("C:/Users/Usuario/Proyecto/Reclamos_2010_Data Set.
    csv", "r")
```

Actividad

Escribe la instrucción para abrir el mismo documento anterior, usando su path relativo en modo de "agregar".

Cada vez que uno ingresa a una sala, oficina, edificio u otra infraestructura que tenga una puerta que debe abrirse para entrar, las normas de buena educación señalan que luego de abrir la puerta y entrar al recinto, la puerta debe ser cerrada. Te sugerimos fuertemente que recuerdes esto cada vez que ingrese a un edificio, sobre todo cuando esté calefaccionado y en invierno. Este comentario no es solo para recordar una norma de buena educación, sino que hace referencia a las buenas prácticas en la programación, en este caso, cada vez que abras un archivo y lo uses debes cerrarlo.

Para cerrar un archivo que fue abierto asociándolo a la variable `archivo`, debes cerrarlo así:

```
archivo.close()
```

El cerrar un archivo luego de usarlo no solo responde a una buena práctica, sino que es crucial para el buen funcionamiento de nuestros programas y del equipo en que estés trabajando. Por ejemplo, supón que estás trabajando con un archivo alojado en tu pendrive con un programa que no cierra el archivo al termina de usarlo. A pesar que el programa termine su ejecución, para el sistema operativo el archivo seguirá en uso, por lo que no te dejará sacar el pendrive de manera segura.

Otro punto importante, antes de continuar, es que recuerdes usar el archivo antes de cerrarlo, ya sea leyéndolo o escribiendo sobre él, de lo contrario no podrás manipular la información que contiene.

4. Lectura de archivos

Mientras el documento está abierto es posible leer su contenido, siempre y cuando el modo de apertura haya sido "`r`".

Para realizar la lectura puedes utilizar alguno de los métodos que siguen, considerando que el archivo esté asociado a la variable `documento`:

- `documento.read()` – De esta manera se lee todo el documento y queda disponible como un único string que incluye todas las líneas del archivo.
- `documento.read(bytes)` – Al añadir el argumento opcional bytes la lectura se realiza hasta esa cantidad de bytes. En lo práctico un byte equivale a un carácter, incluyendo a los caracteres especiales como tabulación ("`\t`") y salto de línea ("`\n`"), entre otros.
- `documento.readline()` – Con este método se lee la primera línea no leída del archivo, es decir, la primera vez que se usa lee la primera línea, luego la segunda, la tercera y así sucesivamente.?? – incluye el salto de línea
- `documento.readlines()` – Así se lee todo el documento como una lista, en donde cada línea del archivo es un elemento de la lista, incluyendo el salto de línea.

Para acceder de manera más práctica la información debes asignar una variable a la información leída como sigue:

```
textoCompleto = documento.read()
```

Puedes cambiar el método de lectura según sea pertinente.

Actividad

Responde las siguientes preguntas:

- ¿Qué haría el siguiente ciclo?

```
for i in range(11):
    print archivo.readline()
```

- Así como read soporta un argumento opcional, ¿readline y readlines lo hacen? ¿Qué resulta si es que es así?

También es posible recorrer a través de un ciclo `for`, como se verá en el ejemplo que sigue a continuación.

Ejemplo

En el siguiente ejemplo calcularemos la cantidad de reclamos efectuados al SERNAC por región durante el año 2010. Los datos puedes obtenerlos desde: http://datos.gob.cl/dataset/3162.

```
#!/usr/bin/env python
#-*- coding: cp1252 -*-

# Verificar la cantidad de reclamos por región

def abrirArchivo(nombre):
    archivo = open(nombre, "r")
    listadoInformacion = []
    for linea in archivo:
        linea = linea.strip().split(";")
        listadoInformacion.append(linea)
    archivo.close()
    return listadoInformacion

def verificarRegiones(datos):
    regionesParticipantes = []
    for linea in datos:
        region = linea[2]
        if region not in regionesParticipantes:
            regionesParticipantes.append(region)
        else:
            pass
    regionesParticipantes = regionesParticipantes[1:len(
        regionesParticipantes)]
    return regionesParticipantes

def contarReclamos(datos, regiones):
    reclamos = []
    for region in regiones:
        conteo = 0
        for linea in datos:
            if region == linea[2]:
                conteo += 1
            else:
                pass
        reclamos.append(conteo)
    return reclamos


datos2010 = abrirArchivo("./Reclamos_2010_Data Set.csv")
regionesParticipantes = verificarRegiones(datos2010)
totalesReclamos = contarReclamos(datos2010, regionesParticipantes
    )
print regionesParticipantes
print totalesReclamos
```

Actividad

Comenta el código siguiendo las buenas prácticas que hemos descrito hasta ahora.

¿Qué hacen los métodos `.strip()`, `.slipt()` y `.join()`?

Ejercicio Propuesto

Chile es un país con alta actividad sísmica, por lo que manejar información de este tipo es de gran importancia. Se te ha solicitado estimar el promedio anual y mensual (de cada año) de magnitudes de sismos desde 2003 a 2015 a través de un programa desarrollado en Python. Para realizar dicha tarea han puesto a tu disposición el documento alojado en: http://benjad.github.io/2015/08/21/base-de-datos-sismos-chile/.

Desarrolla un script en Python que muestre por pantalla la información solicitada permitiendo que el usuario decida el año y/o mes del que quiere saber el promedio de magnitud.

5. Escritura de archivos

Supongamos ahora que nuestra misión es escribir información a un archivo. Mientras el documento está abierto es posible escribir sobre él o agregar contenido, siempre y cuando el modo de apertura haya sido `"w"` o `."a"`.

Para realizar la escritura puedes utilizar alguno de los métodos que siguen, considerando que el archivo esté asociado a la variable `documento`:

- `documento.write("texto")` – De esta manera se escribe en el archivo objetivo a partir desde el último lugar que se escribiera en él. Si se habrío en modo `"w"` la primera escritura será como si el archivo estuviera en blando, si fue abierto en modo `."a"`, la escritura comenzará a partir de la posición siguiente al último carácter existente en el archivo. Su argumento debe ser un `string`.

A diferencia de la lectura, con la escritura no se hace la asignación a una variable, sino que la instrucción se da por sí sola, dado que su fin no es acceder a la información, sino dirigir la información hacia un archivo.

Actividad

Existe otro método para escribir en archivos: `writelines()`. ¿Cómo funciona? ¿Qué tipo de argumentos recibe?

Ejemplo

Continuando con el ejemplo del SERNAC, añadiremos una funcionalidad a nuestro programa; ahora organizará la información apareando el nombre de la región con la cantidad de reclamos respectiva y lo escribirá a un archivo de texto que se almacenará en la misma carpeta que esté alojado el script.

```
#!/usr/bin/env python
#-*- coding: cp1252 -*-

# Verificar la cantidad de reclamos por región e imprime dicha
    información a un archivo de texto

def abrirArchivo(nombre):
    archivo = open(nombre, "r")
    listadoInformacion = []
    for linea in archivo:
        linea = linea.strip().split(";")
        listadoInformacion.append(linea)
    archivo.close()
    return listadoInformacion

def verificarRegiones(datos):
    regionesParticipantes = []
    for linea in datos:
        region = linea[2]
        if region not in regionesParticipantes:
            regionesParticipantes.append(region)
        else:
            pass
    regionesParticipantes = regionesParticipantes[1:len(
        regionesParticipantes)]
    return regionesParticipantes

def contarReclamos(datos, regiones):
    reclamos = []
    for region in regiones:
        conteo = 0
        for linea in datos:
            if region == linea[2]:
                conteo += 1
            else:
                pass
```

```
        reclamos.append(conteo)
    return reclamos

def guardarInformacion(archivo, regiones, cantidades):
    nuevoDocumento = open(archivo, "w")
    contador = 0
    while contador < len(regiones):
        regionFrecuencia = "La región " + regiones[contador] + "
            tuvo un total de " + str(cantidades[contador]) + " de
            reclamos durante el año 2010.\n"
        nuevoDocumento.write(regionFrecuencia)
        contador += 1
    nuevoDocumento.close()
    return True


datos2010 = abrirArchivo(".\Reclamos_2010_Data Set.csv")
regionesParticipantes = verificarRegiones(datos2010)
totalesReclamos = contarReclamos(datos2010, regionesParticipantes
    )
print regionesParticipantes
print totalesReclamos
guardarInformacion("./Información de reclamos.txt",
    regionesParticipantes, totalesReclamos)
```

Actividad

Incluye los comentarios necesarios en los nuevos tramos de código de nuestro programa.

Ejercicio Propuesto

Se te solicita que el programa que estima los promedios de sismos incluya tres nuevas funcionalidades:

- El programa debe detectar el sismo de mayor magnitud en el tramo escogido por el usuario.
- El programa debe detectar el sismo de menor magnitud en el tramo escogido por el usuario.
- El programa debe imprimir la información calculada en un archivo de texto con la siguiente estructura:

> "En el <mes> del <año> la magnitud promedio de los sismos producidos fue de <promedio de magnitudes>.
>
> El sismo de mayor intensidad fue de <max magnitud> y el de menor intensidad fue de <min magnitud>."

Modifica tu script del ejercicio propuesto anterior para cumplir con los nuevos requerimientos.