Introducción a la Programación con PYTHON

José Rodriguez

Unidad 8.- Gestión de Ficheros

UNIDAD 3: GESTIÓN DE FICHEROS

ÍNDICE

1.- **Lectura/escritura de ficheros**

La forma más habitual de abrir un fichero para escritura es la función open()

```
ruta = "c:\\prueba.txt"
with open(ruta, mode="w", encoding="utf-8") as fichero:
    print("Hola y Adiós", file=fichero)
```

Si analizamos el script vemos que, en primer lugar, definimos la ruta del fichero. Después, invocamos al método open con los tres parámetros necesarios: ruta, modo y tipo de codificación. Evidentemente, si el fichero no existe lo crea y si existe machaca su contenido. Por último, indicamos el tipo de codificación que, generalmente, suele ser utf-8 para incluir las tildes y algunos caracteres especiales como la ´ñ.

Los modos de escritura son:

- "x": únicamente crear el fichero (da error si ya existe el fichero)
- "w": escribir (crea el fichero si no existe y borra el contenido anterior del fichero)
- "a": añadir (crea el fichero si no existe, no borra el contenido existente y escribe al final del fichero)

Una vez abierto el fichero, la siguiente operación debe ser llenarlo de contenido. Para ello disponemos de varios métodos.

Se puede escribir en el fichero

- con la función print() añadiendo el argumento file=fichero, donde fichero es la variable utilizada en la expresión with ... as ...
- con el método write sobre el objeto fichero, donde fichero es la variable utilizada en la expresión with ... as ...

La función print() añade un salto de línea al final de la cadena añadida al fichero, pero el método write no lo hace, por lo que habrá que añadirlo explícitamente. Por tanto, el código anterior escrito con el método write sería

```
ruta = "c:\\ejemplo.txt"
with open(ruta, mode="a", encoding="utf-8") as fichero:
    fichero.write("Hola y Adiós" + "\n")
```

Donde se ha insertado en el código el salto de línea “\n. También hemos escogido el módo “a” y se añade al contenido previo del fichero.

Abrir varios ficheros a la vez

Se pueden abrir varios ficheros a la vez en una única construcción with ... mediante varias funciones open(...) as .... separadas con comas. Cada fichero se puede abrir en un modo distinto.

```
ruta_1 = "c:\\prueba_1.txt"
ruta_2 = "c:\\prueba_2.txt"

with open(ruta_1, mode="w", encoding="utf-8") as fichero_1, open(ruta_2, mode="w", encoding="utf-8") as fichero_2:
    print("Hola", file=fichero_1)
    print("Adios", file=fichero_2)
```

Lectura de ficheros

El procedimiento para leer los archivos de texto más sencillo es hacerlo de una vez con el método readlines. Una vez abierto el archivo solamente se ha de llamar a este método para obtener el contenido. Por ejemplo, se puede usar el siguiente código.

En este ejemplo vamos a leer el fichero escrito en los ejemplos anteriores

```
ruta = "c:\\ejemplo.txt"
with open(ruta, mode="r", encoding="utf-8") as fichero:
    lines = fichero.readlines()
    print(lines)
fichero.close()
```

lines es un array formado por todas las líneas. Podemos leer una sola línea

```
print(lines[0])
```

Otra posibilidad es obtener el número de líneas del fichero

```
ruta = "c:\\ejemplo.txt"
with open(ruta, mode="r", encoding="utf-8") as fichero:
    lines = fichero.readlines()
l= len(lines)
print(l)
fichero.close()
```

Y después recorrer con un bucle todas las líneas y mostrarlas por pantalla

```
l= len(lines)
i=0
while i<l:
    print(lines[i])
    i=i+1
fichero.close()
```

también podemos leer línea a línea y guardarlas en una lista

```
ruta = "c:\\ejemplo.txt"
lineas = []
with open(ruta, mode="r", encoding="utf-8") as fichero:
 for line in fichero:
    lineas.append(line)
fichero.close()
l=len(lineas)
i=0
while i<l:
    print(lineas[i])
    i=i+1
```

Modos de lectura

Los modos de lectura son:

- "r": únicamente leer el fichero (da error si no existe el fichero, lee desde el principio y es posible desplazarse)
- "r+": leer y escribir (da error si no existe el fichero, lee desde el principio y es posible desplazarse, pero sólo escribe al final del fichero

Para leer un fichero completo usamos el método read()

```
with open('c:/prueba.txt', 'r') as f:
    contenido = f.read()
    print(contenido)
```

2.- Gestión de Directorios

Para listar o recorrer un directorio en Python basta con usar la función listdir() del módulo os. Esta función recibe como argumento una ruta del sistema de ficheros y devuelve una lista con los nombres de los archivos y carpetas que contiene. Si no se pasa ningún argumento, el directorio de referencia es la carpeta actual.

Listado de todos los ficheros del directorio raíz.

```
import os
ruta = 'c:/'
contenido = os.listdir(ruta)
for fichero in contenido:
    print (fichero)
```

Listar ficheros jpg del directorio raíz

```
import os
ruta = 'c:/'
contenido = os.listdir(ruta)
imagenes = []
for fichero in contenido:
    if os.path.isfile(os.path.join(ruta, fichero)) and fichero.endswith('.jpg'):
        imagenes.append(fichero)
print(imagenes)
```

donde hemos utilizado el método endswith() para obtener la terminación del fichero y la hemos comparado con .jpg. también hemos utilizado el método join para obtener el path absoluto del fichero uniendo la carpeta y el nombre del fichero

Si debes filtrar de algún modo los elementos devueltos por listdir(), la mejor forma de recorrer un directorio en Python es usar la función scandir(), contenida también en el módulo os.

```
import os
ruta = 'c:/'
with os.scandir(ruta) as ficheros:
    for fichero in ficheros:
        print(fichero.name)
```

Otra librería que permite listar directorios es pathlib

```
import pathlib
ruta = 'c:/'
directorio = pathlib.Path(ruta)
for fichero in directorio.iterdir():
    print(fichero.name)
```

Podemos utilizar el método is_file() que devuelve true o false para detectar si un elemento de la búsqueda es fichero.

```
import os
ruta = 'c:/'
with os.scandir(ruta) as ficheros:
    for fichero in ficheros:
        if fichero.is_file():
            print(fichero.name)
```

De la misma forma, podemos filtrar sólo los subdirectorios

```
import os
ruta = 'c:/'
with os.scandir(ruta) as ficheros:
    for fichero in ficheros:
        if fichero.is_dir():
            print(fichero.name)
```

Otro método bastante útil es getcwd() que devuelve el directorio de trabajo

```
import os
a=os.getcwd()
print(a)
```

Para crear un directorio usamos el método mkdir(path)

```
import os
os.mkdir("c:/prueba4")
```

3.- Abrir un fichero en el navegador

El módulo webbrowser incluye funciones para abrir URLs en navegadores en forma interactiva. El módulo incluye un registro de los navegadores disponibles en caso de que haya múltiples opciones disponibles en el sistema. Puede también controlarse con la variable de entorno BROWSER.

```
import webbrowser
webbrowser.open('http//localhost')
```

El resultado sería que el navegador por defecto mostraría la URL indicada como parámetro

Si queremos abrir la página en una pestaña nueva usamos el método new

```
import webbrowser
webbrowser.open_new('https://localhost')
```

Si el fichero que se ha creado con Python es un página web, puede ser de utilidad que el fichero se abra automáticamente en el navegador para ver el resultado inmediatamente.

Para ello se puede utilizar la función open() del módulo webbrowser de la biblioteca estándar, que abre el fichero indicado con la aplicación asociada en el sistema operativo.

En este ejemplo creamos un fichero con formato HTML y lo abrimos en el navegador.

```
import webbrowser
ruta = "prueba.html"
with open(ruta, mode="w", encoding="utf-8") as fichero:
    print("<!DOCTYPE html>", file=fichero)
    print('<html lang="es">', file=fichero)
    print("<head>", file=fichero)
    print('  <meta charset="utf-8">', file=fichero)
    print("  <title>HTML 5</title>", file=fichero)
```

```
    print(
        '  <meta name="viewport" content="width=device-width, initial-scale=1.0">',
        file=fichero,
    )
    print("</head>", file=fichero)
    print("", file=fichero)
    print("<body>", file=fichero)
    print("  <p>Hola Mundo.</p>", file=fichero)
    print("</body>", file=fichero)
    print("</html>", file=fichero)

webbrowser.open(ruta)
```