Lectura y escritura de archivos en Python

Mario González
Facultad de Ingeniería y Ciencias Aplicadas
Universidad de las Américas, Quito, Ecuador

Agosto, 2021

Introducción y tipos de archivos

- En Python no es necesario importar ninguna librería para leer y escribir archivos.
- El primer paso es crear un objeto archivo (file).
- Esto se consigue usando la función **`open`**.

Tipos de archivos

- Los archivos se clasifican usualmente en archivos de texto y archivos binarios.
- Un archivo de texto a menudo está estructurado en una secuencia de líneas, y una línea es una secuencia de caracteres.
- Una línea está terminada por un caracter EOL (End Of Line).
- El caracter EOL más común es `\n`.
- Esto indica que el siguiente caracter será tratado como una nueva línea.
- Un archivo binario es básicamente cualquier archivo, que no es un archivo de texto.
- Los archivos binarios sólo pueden ser procesados por la aplicación que conoce la estructura del archivo.

Función `open()`

- Para abrir un archivo para lectura o escritura se utiliza la función predefinida (built-in function): **`open()`**.
- Esta función devuelve un objeto archivo (file) y es usada comúnmente con dos argumentos.
- La sintaxis es: **`file_object = open(filename, mode)`**.
- **`file_object`** es la variable que hace referencia al objeto file.
- El primer argumento (filename) es un string con el nombre del archivo.
- El segundo argumento (mode) describe la forma en que el archivo será usado (es también un string).

Argumento `mode`

Sintaxis es: **`file_object = open(filename, mode).`**

Los modos pueden ser:

- 'r': el archivo será solo de lectura.
- 'w': sólo escritura, un archivo existente con el mismo nombre será borrado.
- 'a': abre el archivo en modo agregar, cualquier dato escrito en el archivo será agregado al final.
- 'r+': abre el archivo para lectura y escritura.
- El argumento **`mode`** es opcional, si es omitido será asumido 'r'.

Como crear un archivo de texto

- El siguiente código en Python crea el archivo de texto prueba.txt

```
file = open("prueba.txt", "w")
file.write("Hola mundo en el archivo\n")
file.write("y agregamos otra linea")
file.close()
```

- El método **`write`** toma un parámetro, que es el string a ser escrito.
- Para empezar una nueva línea después de los datos, agregamos el caracter **`\n`** al final.
- Cuando hemos terminado de agregar datos al archivo, llamamos el método **`close`** para cerrar el archivo y liberar recursos del sistema. El archivo no podrá ser accedido fuera del programa hasta que no lo hayamos cerrado.
- Después de llamar **`close`**, si intentamos acceder al objeto file, tendremos un error.

Como leer un archivo de texto I

- Podemos usar diferentes métodos para leer un archivo de texto.

```
file = open("prueba.txt", "r")
print(file.read())
file.close()
```

- **`read()`** devuelve un string conteniendo todos los caracteres en el archivo.
- Podemos utilizar **`read(n)`, donde n** determina el número de caracteres devueltos.

```
file = open("prueba.txt", "r")
print(file.read(4))
file.close()
```

- Devuelve un string con los primeros 4 caracteres.

Como leer un archivo de texto II

```
file = open("prueba.txt", "r")
print(file.readline())
```

- **`readline()`** lee un archivo línea por línea en lugar de todo el archivo a la vez.
- Use **`readline()`** cuando quiera obtener la primera línea del archivo, sucesivas llamadas a **`readline()`** devolverán las líneas sucesivas.
- Lee cada línea del archivo devolviendo un string con los caracteres hasta encontrar **`\n`**.
- **`readlines()`** devuelve el archivo completo como una lista de strings separadas por **`\n`**.

```
file = open("prueba.txt", "r")
print(file.readlines())
```

Manipulando archivos de texto

- Para recorrer un archivo de texto, podemos iterar a través del objeto file. Esto es eficiente en términos de memoria, rápido y produce un código simple.

```
archivo = open("prueba.txt", "r")
for linea in archivo:
    print(linea, end=' ')
```

Sentencia `with`:

- Otra forma de manipular archivos es con la sentencia `with`.
- La sentencia `with`, nos da una mejor sintaxis y manejo de excepciones.

```
with open("prueba.txt") as f:
    for line in f:
        print(line, end=' ')
```

Dividiendo líneas

```
with open('data.txt', 'r') as f:
    data = f.readlines()

    for line in data:
        words = line.split()
        print(words)
```

- La función **`split`**, divide el string en una lista de acuerdo al separador especificado, si no se especifica, asume los espacios en blanco como separador.

Numpy `loadtxt` y `savetxt`

- Numpy es el paquete fundamental para computación científica con Python.

```
import numpy as np
x = np.loadtxt('data.txt')
array([[ 3.,  6.,  2.],
       [ 4.,  8.,  7.],
       [ 7.,  5.,  8.]])

np.savetxt('d2.txt',x*2,fmt='%2.0f')
```

Ejercicios

- Lea los siguientes datos de archivos de texto:

a)
```
 6 12  4
 8 16 14
14 10 16
```

b)
```
6 12 4 8 16 14 14 10 16
```

c)
```
6
12
4
8
16
14
14
10
16
```

d)
```
 6,12,  4
 8,16,14
14,10,16
```

e)
```
6,12,4,8,16,14,14,10,16
```

- Lea y grafique los datos en el siguiente archivo.