08. Manipulacion Archivos

August 26, 2024

1 Manejo de archivos

El manejo de archivos es una parte importante de cualquier aplicación web.

Python tiene varias funciones para crear, leer, actualizar y eliminar archivos.

1.1 Función `open`

La función clave para trabajar con archivos en Python es `open()`. La función `open()` toma dos parámetros: *nombre_archivo* y *modo* .

Hay cuatro métodos (modos) diferentes para abrir un archivo:

- "r"- Leer - Valor predeterminado. Abre un archivo para leerlo. Se produce un error si el archivo no existe.
- "a"- Anexar - Abre un archivo para anexarlo, crea el archivo si no existe.
- "w"- Escribir - Abre un archivo para escribir, crea el archivo si no existe.
- "x"- Crear - Crea el archivo especificado, devuelve un error si el archivo existe.

Además, puede especificar si el archivo debe manejarse en modo binario o de texto.

- "t"- Texto - Valor predeterminado. Modo texto.
- "b"- Binario - Modo binario (por ejemplo, imágenes).

1.2 Sintaxis

La función `open()` abre un archivo y lo devuelve como un objeto de archivo.

```
open(file, mode)
```

1.2.1 Valores predeterminados

Parámetro	Descripción
file	La ubicación y nombre del archivo
mode	Una cadena que define el modo como se abre el archivo `"r"` - **Lectura** - Valor por defecto. Abre un archivo para lectura, envía error si el archivo no existe. `"a"` - **Añadir** - Abre un archivo para añadir, crea el archivo si no existe.

Parámetro	Descripción
	`"w"` - **Escritura** - Abre un archivo para escritura, crea el archivo si no existe. `"x"` - **Creación** - Crea un archivo, devuelve un error si el archivo ya existe. Adicionalmente se puede especificar si el archivo será manejado como texto o binario `"t"` - **Texto** - Modo texto, valor por defecto. `"b"` - **Binario** - Modo binario (i.e. imágenes)

Para abrir un archivo para su lectura es suficiente especificar el nombre del archivo:

```
f = open("demofile.txt")
```

Este código es equivalente al siguiente:

```
f = open("demofile.txt", "rt")
```

Dado que `'r'` para lectura y `'t'` para texto son los valores por defecto, no es necesario incluirlos.

1.3 Abrir un archivo localmente

Supongamos que tenemos el archivo `demofile.txt`, ubicado en la misma carpeta que el script de Python:

Archivo demofile.txt

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Para abrir el archivo, se utiliza la función incorporada `open()`. Esta función devuelve un objeto tipo archivo, que tiene un método `read()` para leer el contenido del archivo:

[ ]:
```
f = open("demofile.txt", 'r')
print(f.read())
```

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!Now the file has more content!
```

1.4 Abrir un archivo en Google Drive

Supongamos que tenemos el archivo `demofile.txt`, ubicado en Google Drive:

Archivo demofile.txt

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Además un notebook de Python se encuentra en la misma carpeta. Para abrir un archivo es necesario utilizar la función `google.colab` dentro del paquete `drive`. Una vez que se haya incluído el paquete, ahora es necesario *montar* la unidad de almacenamiento de Google Drive en una ubicación dentro del sistema de archivos virtual: `/drive`. Cabe mencionar que esta dirección apuntará a la raíz de la unidad de almacenamiento en Google Drive.

```
from google.colab import drive
drive.mount("/drive")
```

Al ejecutar la función `drive.mount`, Google Drive preguntará por autorización para tener acceso a su unidad.

Posterior a esto, sólo es necesario utilizar la función open tal y como lo hicimos anteriormente.

```
f = open("/drive/My Drive/demofile.txt", 'r')
print(f.read())
```

Observe que se precede al nombre del archivo la ubicación `/drive/My Drive/`, que es donde se montó virtualmente la unidad de almacenamiento de Google Drive.

1.5 Leer partes de un archivo

Por defecto el método `read()` devuelve todo el texto, pero se puede especificar cuantos caracteres se desea leer.

[ ]:
```
f = open("demofile.txt", "r")
print(f.read(5))
```

```
Hello
```

1.6 Leer líneas del archivo

Es posible leer una línea de un archivo mediante el método `readline()`.

[ ]:
```
f = open("demofile.txt", "r")
print(f.readline())
```

```
Hello! Welcome to demofile.txt
```

[ ]:
```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

```
Hello! Welcome to demofile.txt

This file is for testing purposes.
```

Si se utiliza un ciclo `for`, se puede leer todo el archivo línea por línea.

[ ]:
```
f = open("demofile.txt", "r")
for x in f:
```

```
    print(x)
```

```
Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!Now the file has more content!
```

1.7 Cerrar un archivo

Es una biena práctica cerrar un archivo después de utilizarse. Para ello se debe utilizar el método `close()`.

[ ]:
```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

```
Hello! Welcome to demofile.txt
```

Se puede verificar si un archivo está cerrado mediante la propiedad booleana `closed`.

[ ]:
```
print(f'El archivo está cerrado?: {f.closed}')
```

```
El archivo está cerrado?: True
```

1.8 Escribir en un archivo existente

Para escribir en un archivo existente se debe agregar un parámetro al método `open()`.

- `'a'`. Añadir contenido al final del archivo
- `'w'`. Escribir en el archivo. Sobreescribe cualquier contenido existente.

Ejemplo. Abrir el archivo `demofile2.txt` y añadir contenido.

[ ]:
```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!Now the file has more content!Now the file has more content!
```

Ejemplo. Abrir el archivo `demofile2.txt` y sobrescribir contenido.

[ ]:
```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```

```
#open and read the file after the overwriting:
f = open("demofile3.txt", "r")
print(f.read())
```

```
Woops! I have deleted the content!
```

1.9 Crear un nuevo archivo

Para crear un nuevo archivo en Python se debe utilizar el método `open()` con uno de los siguientes parámetros.

- 'x'. Crear - crea el archivo y devuelve un error si el archivo existe.
- 'a'. Añadir - crea el archivo si es que no existe previamente.
- 'w'. Escribir - crea el archivo si es que no existe previamente.

Ejemplo. Crea un archivo llamado `myfile.txt`.

[ ]:
```
f = open("myfile.txt", "x")
```

Ejemplo. Crea un nuevo archivo `myfile.txt` si es que no existe previamente.

[ ]:
```
f = open("myfile.txt", "w")
```

1.10 Borrar un archivo

Para borrar archivos en Python se debe importar el módulo `os` y utlizar el módulo `os.remove()`.

[ ]:
```
import os
os.remove("myfile.txt")
```

[ ]:
```
import os
if os.path.exists("myfile.txt"):
  os.remove("myfile.txt")
else:
  print("The file does not exist")
```

```
The file does not exist
```

2 Referencias

- Manejo de archivos en Python.
- Lutz M., Learning Python, O'Reilly. 2009