El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE | MisiónTIC 2022

Ciclo 1

Semana 7

Archivos de Texto

Lectura 4 – *Implementación en Python*

Convenio de asociación no. 777 de 2021 suscrito entre el Fondo Único de Tecnologías de la Información y las Comunicaciones y la Universidad el Bosque.

Implementación en Python

Para trabajar un archivo en Python como en otros lenguajes de programación, involucra seguir siempre un esquema de tres pasos:

Abrir el archivo: se indica la ruta y el modo de trabajo, que puede tener tres variantes:

- Lectura: se lee la información del archivo, pero no se modifica ni se añade nueva.
- Escritura: se escribe la información en el archivo, por regla general, abrir un archivo para escritura borra el contenido previo del mismo.
- Lectura/Escritura: se puede leer y escribir información en el archivo.
- Adición: se añade nueva información, pero no se modifica la existente.

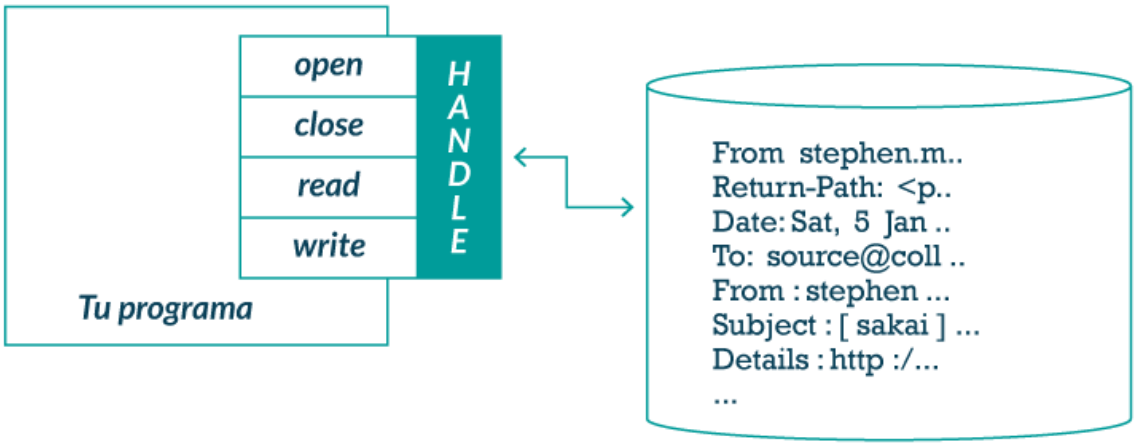


Figura 1: tomado de Severance

Al abrir un archivo se le pide al sistema operativo que lo encuentre por su nombre y se asegure que existe. Por ejemplo, el siguiente fragmento de código de un programa, abre el archivo entrada.txt, que debe estar en el directorio de trabajo donde se ejecuta el programa:

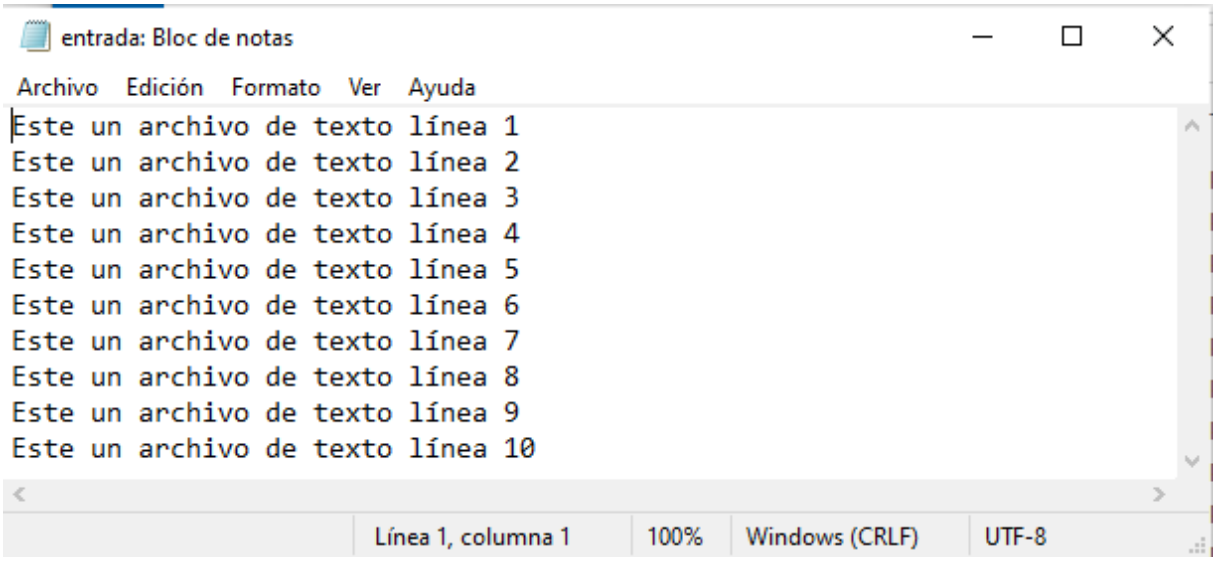
```
archivoLogico = open('entrada.txt')
print(archivoLogico)
```

```
<_io.TextIOWrapper name='entrada.txt' mode='r' encoding='cp1252'>

Process finished with exit code 0
```

En este caso se le dio la orden open y no se le estipuló el modo de trabajo, lo que muestra que por defecto él lo abrirá para lectura. Si la orden open tiene éxito, es decir, el archivo existe en la ruta esperada y se tienen los permisos apropiados, el sistema operativo nos retorna un manejador de archivo 'handle', con el cual se pueden leer los datos.

Leer o escribir en el archivo: Un archivo de texto se puede considerar como una secuencia de líneas, cada una de las cuales es también una secuencia de caracteres. Por ejemplo, el archivo entrada.txt, contiene 10 líneas así:



Cada línea al final tiene un salto de línea '\n', que realmente es un solo carácter, es el que permite ver el archivo línea por línea y no como una sola línea. En el siguiente ejemplo se puede ver como leer un archivo y contar sus líneas:

```
archivoLogico = open('entrada.txt')
contador = 0
for linea in archivoLogico:
    contador = contador + 1
print('El archivo tiene',contador,' líneas.')
```

```
El archivo tiene 10  líneas.


Process finished with exit code 0
```

En esta forma se puede leer un archivo, no importa el tamaño físico, puesto que se está leyendo por línea. Si se quiere leer un archivo completo en una sola cadena, para archivos relativamente pequeños, se debe utilizar el método read de la siguiente forma:

```
archivoLogico = open('entrada.txt')
cadena = archivoLogico.read()
print(len(cadena))
```

```
340

Process finished with exit code 0
```

Como se puede observar generó una cadena de 340 caracteres, 34 caracteres por línea, 10 líneas.

Una tarea muy común cuando trabajamos con archivos es buscar un determinado elemento, por ejemplo, en el siguiente programa se busca el carácter '7' en el archivo entrada.txt, para este caso se le puede solicitar al usuario el nombre del archivo, para que no sea necesario cambiar el programa cada vez que se requiera hacer el mismo proceso en otro archivo, así se le pide al usuario que digite el nombre y ese nombre se le pasa al open, para que se realice la operación de apertura como se ha hecho antes, la función rstrip() limpia los espacios al final de la línea:

```
archivo = input('Por favor de el nombre del archivo: ')
archivoLogico = open(archivo)
for linea in archivoLogico:
    linea = linea.rstrip()
    if linea.find('7') != - 1:
        print(linea,'SI tiene 7')
    else:
        print(linea,'NO tiene 7')
```

```
Por favor de el nombre del archivo: entrada.txt
Este un archivo de texto lÃ-nea 1 NO tiene 7
Este un archivo de texto lÃ-nea 2 NO tiene 7
Este un archivo de texto lÃ-nea 3 NO tiene 7
Este un archivo de texto lÃ-nea 4 NO tiene 7
Este un archivo de texto lÃ-nea 5 NO tiene 7
Este un archivo de texto lÃ-nea 6 NO tiene 7
Este un archivo de texto lÃ-nea 7 SI tiene 7
Este un archivo de texto lÃ-nea 8 NO tiene 7
Este un archivo de texto lÃ-nea 9 NO tiene 7
Este un archivo de texto lÃ-nea 10 NO tiene 7
```

Para la operación de escritura, el archivo debe abrirse en modo ‘w’ (write, escritura) como segundo parámetro. Si el archivo ya existía previamente, abrirlo en este modo causará que se borre todo su contenido, así que se debe tener mucho cuidado con la apertura en este modo. Si el archivo no existe, un nuevo archivo será creado. Además, el método write del manejador de archivos escribe los datos y retorna el número de caracteres escritos. El modo de escritura por defecto es texto para leer y escribir cadenas.

El manejador de archivos mantiene un seguimiento sobre la posición del archivo en que se encuentra, así que cada vez que se llame a write se escribirán los datos al final. Otro punto importante al escribir registros en el archivo es insertar el carácter de salto de línea al final de cada una de ellas. La sentencia print agrega un salto de línea al final de forma automática, pero

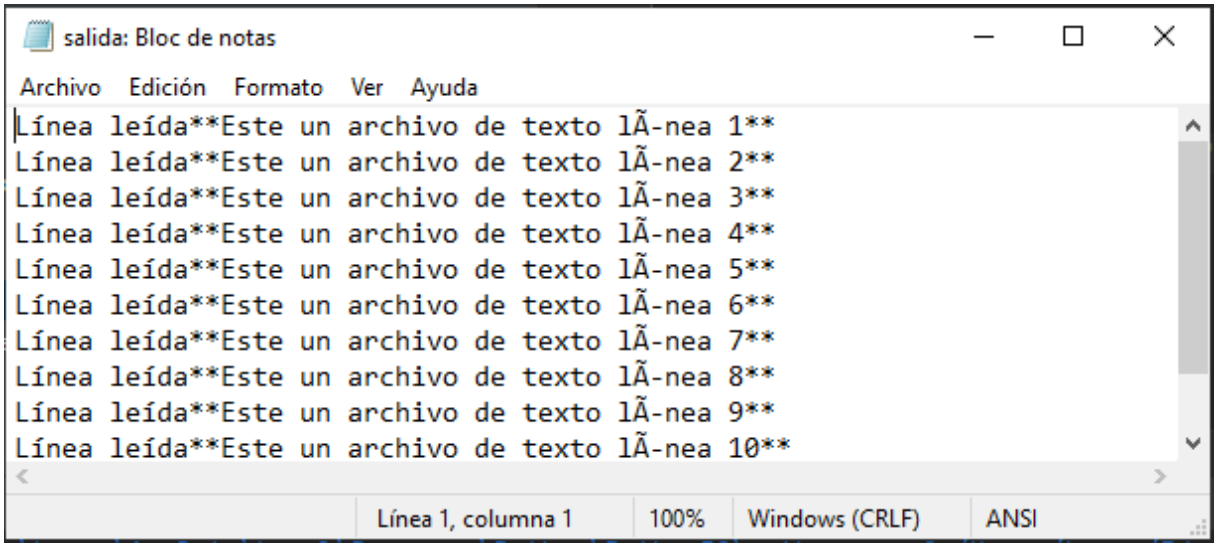
el método write no lo hace. Cuando se termine de escribir, se debe cerrar el archivo para tener la seguridad que los datos se escriban físicamente en el disco duro y no se pierdan ante un fallo de energía.

En el siguiente programa se leen los datos de un archivo y se graban en otro haciendo un cambio en cada línea. Se le solicita al usuario el nombre del archivo de entrada y el nombre del archivo de salida.

```
entrada = input('Por favor de el nombre del archivo de entrada: ')
salida = input('Por favor de el nombre del archivo de salida: ')
entradaLogico = open(entrada,'r')
salidaLogico = open(salida,'w')
for linea in entradaLogico:
    linea = linea.rstrip()
    linea = 'Línea leída**'+linea+'**\n'
    salidaLogico.write(linea)
entradaLogico.close()
salidaLogico.close()
```

```
Por favor de el nombre del archivo de entrada: entrada.txt
Por favor de el nombre del archivo de salida: salida.txt

Process finished with exit code 0
```

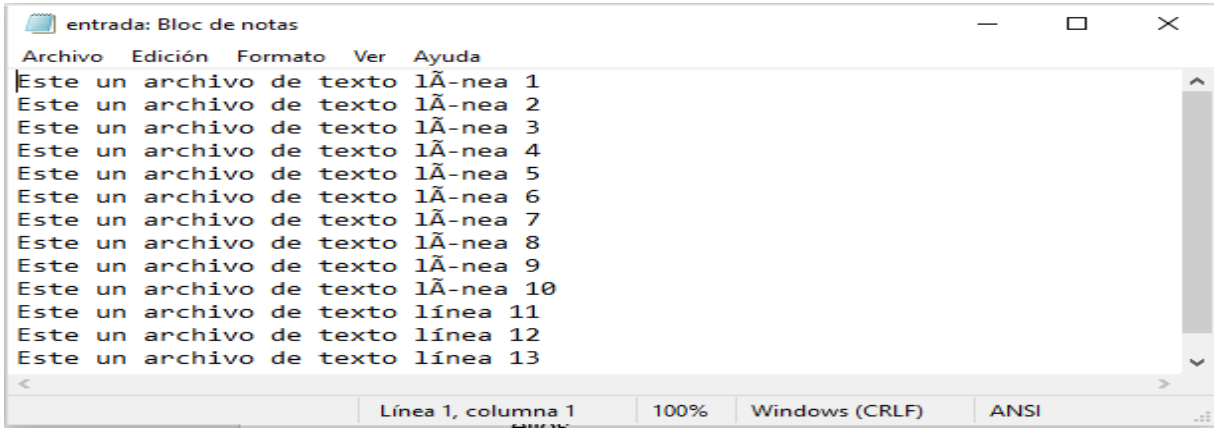


También se pueden adicionar registros al final del archivo, para eso es necesario hacer el open en modo 'a' (append, añadir al final). Ver el siguiente ejemplo que adiciona tres líneas al archivo entrada.txt.

```
entrada = input('Por favor de el nombre del archivo de entrada: ')
entradaLogico = open(entrada,'a')
for i in range(11,14):
    linea = 'Este un archivo de texto linea '+str(i)+'\n'
    entradaLogico.write(linea)
entradaLogico.close()
```

```
Por favor de el nombre del archivo de entrada: entrada.txt

Process finished with exit code 0
```



Cerrar el archivo: como se puede ver en el ejemplo anterior, el archivo de entrada y el archivo de salida fueron cerrados con el método close(), una vez no se requiere ninguna acción sobre ellos.