Nombre: Daniela Martínez Madrid

RESUMEN SOBRE FILES METHODS

close(): Es un método que cierra el archivo abierto, en un archivo cerrado no se puede leer ni escribir.

Sintaxis: `name.closed()`

closed(): Retorna una variable Booleana, indicando si el archivo está cerrado o no.

Sintaxis: `name.closed`

encoding: Especifica el Unicode en el cual está o estará escrito tu archivo.

Sintaxis: `open('sample-file.txt', encoding='utf-8')`

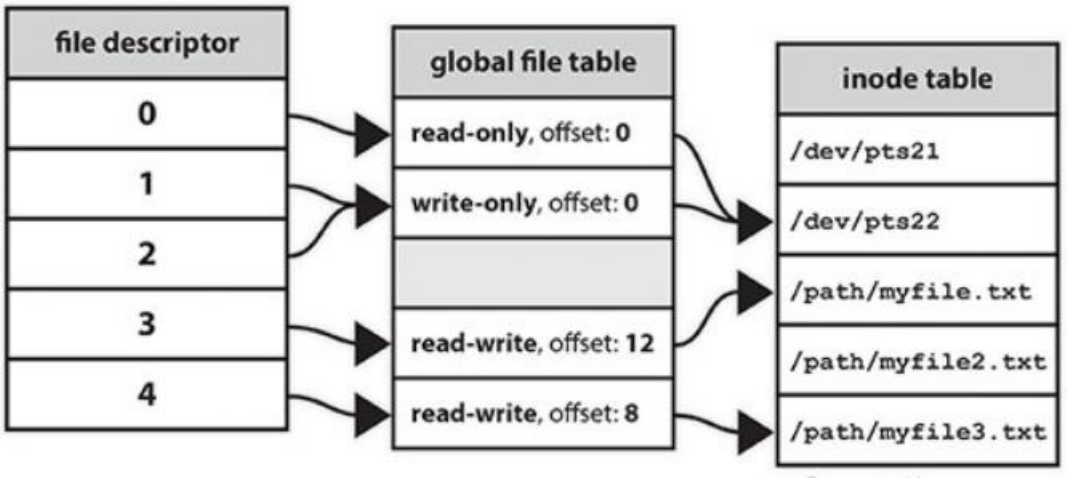
errors: Los errores asociados a la lectura y escritura de código, intentar leer un archivo que no existe producirá el error `FileNotFoundError` y si intentas escribir en un archivo abierto únicamente para lectura, el error será: `UnsupportedOperationError.` Por ejemplo:

```
Ejemplo 1.
file_handle = open('myfile.txt', 'r')

---------------------------------------------------------------------
---------
FileNotFoundError                         Traceback (most recent c
all last)
<ipython-input-14-f6e1ac4aee96> in <module>()
----> 1 file_handle = open('myfile.txt', 'r')

FileNotFoundError: [Errno 2] No such file or directory: 'myfile.tx
t'


Ejemplo 2.
file_handle = open('myfile.txt', 'w')
file_handle.read()

UnsupportedOperation                      Traceback (most recent c
all last)
<ipython-input-15-b846479bc61f> in <module>()
      1 file_handle = open('myfile.txt', 'w')
```

```
----> 2 file_handle.read()

UnsupportedOperation: not readable
```

Fileno(): Es un método que retorna el entero descriptor de archivo. Este número id entifica de forma única un archivo abierto en un sistema operativo, describiendo un recurso de datos y el como se puede acceder a este.

Los descriptores de archivos dependen del sistema operativo, y el caso de Windo ws son denominados como file handles**.**



Sintaxis: `file_object.fileno()`

Flush(): Los archivos automáticamente realizan este método en el momento de ce rrarlos , sin embargo el método flush permite borrar el búfer interno del archivo, sin necesidad de cerrarlo.
Este método no requiere ningún parámetro y no retorna nada.

Sintaxis: `fileObject.flush()`

Cuando se le aplica este método a un archivo, su contenido solo podrá ser leído y presentado.

Ejemplo 1.

```
# opening the file in read mode
fileObject = open("gfg.txt", "r")

# clearing the input buffer
fileObject.flush()

# reading the content of the file
```

```
fileContent = fileObject.read()

# displaying the content of the file
print(fileContent)

# closing the file
fileObject.close()
```

Isatty(): Este método retorna True si el archivo está conectado a un dispositivo tty (teletypewriter), como por ejemplo terminal.

Sintaxis: `fileObject.isatty()`

Modes: Para trabajar con un archivo se debe especificar el modo con el que se desea trabajar, que son los siguientes:

r : Para lectura, es el modo por defecto.
r +: Abre el archivo para lectura y escritura.
w : Abre el archivo para escritura solamente, si el archivo existe sobrescribe sobre este y en caso contrario crea un nuevo archivo.
w +: Abre el archivo para escritura y lectura.
rb : Abre el archivo solamente para lectura en formato binario.
rb +: Abre el archivo en modo lectura y escritura en formato binario.
wb +: Abre el archivo para lectura y escritura en formato binario y además sobre escribe si el archivo existe, de no ser así, crea uno nuevo.
a : Abre el archivo para anexar. En un archivo existente el anexo irá en el puntero final, y de no existir, creará un archivo nuevo.
ab : es igual que el modo a, pero en este caso se trabaja con formato binario.
a+ : Abre el archivo para anexar y leer.
ab +: Anexar y leer en formato binario.
x : Abrir para una creación exclusiva, en caso de existir el archivo el método fallará.

Sintaxis: `file_object  = open("filename", "mode")`

Name: Retorna el nombre completo del archivo.

Sintaxis: `fileObject.name`

Newlines: Retorna tipo de nuevas líneas encontradas mientras se lee el archivo, estos pueden ser: *'r'*, *'n'*, *'rn'*, **None o tuple.**

Sintaxis: `fileObject.newlines`

Next: Es usado cuando el archivo es usado como un iterador. El método retorna el siguiente input line

Sintaxis: `fileObject.next()`

Ejemplo:

```
fo = open("foo.txt", "rw+")
print "Name of the file: ", fo.name

# Assuming file has following 5 lines
# This is 1st line
# This is 2nd line
# This is 3rd line
# This is 4th line
# This is 5th line

for index in range(5):
   line = fo.next()
   print "Line No %d - %s" % (index, line)

# Close opend file
fo.close()
```

Return:

```
Name of the file: foo.txt
Line No 0 - This is 1st line

Line No 1 - This is 2nd line

Line No 2 - This is 3rd line

Line No 3 - This is 4th line

Line No 4 - This is 5th line
```

Read(): Retorna la lectura de todo el texto, sin embargo, también se puede especificar el número de caracteres que deben ser leídos.

Sintaxis: `fileObject.read(), lectura de todo el archivo.`

`fileObject.read(número de caracteres), número especifico de caracteres.`

Readline(): Lee una línea entera de un archivo, también se puede especificar el número de bytes que se leen en la línea.

Sintaxis: `fileObject.readline( size )`

Ejemplo:

```
# Open a file
fo = open("foo.txt", "rw+")
print "Name of the file: ", fo.name

# Assuming file has following 5 lines
# This is 1st line
# This is 2nd line
# This is 3rd line
# This is 4th line
# This is 5th line

line = fo.readline()
print "Read Line: %s" % (line)

line = fo.readline(5)
print "Read Line: %s" % (line)

# Close opend file
Fo.close()
```

Readlines(): Retorna todas las filas en una línea en formato de lista, donde cada elemento de la lista es una línea del archivo.

Sintaxis: `fileObject.readlines()`

Ejemplos:

```
Assuming file as file = example.txt:
Python is the best programming language in the world in 2020
Edureka is the biggest Ed-tech platform to learn python
Python programming is as easy as writing a program in simple
English language

Print(file.readlines())

Return

['Python is the best programming language in the world in 2020',
'Edureka is the biggest Ed-tech platform to learn python',
'Python programming is as easy as writing a program in simple
English language']
```

Además, podemos especificar el número de bytes que queremos retomar del archivo, bajo el parámetro `sizehint`.

Sintaxis: `fileObject.readlines( sizehint )`

Seek(): Es usado para cambiar la posición el puntero (file handle) a una posición especifica, el file handle actúa como el cursor, que indica el inicio y fin donde ha sido leído o escrito el archivo.

Sintaxis: `fileObject.seek(offset, from_what)`
Donde:
offset: Número de bytes que serán movidas hacia delante del puntero.
from_what: define un punto de referencia

El argumento from_what acepta tres valores:
0: Fija el punto de referencia en el inicio del archivo.
1: Fija el punto de referencia en la posición actual del archivo.
2: Fija el punto de referencia en el final del archivo.
Por defecto, se usa el argumento 0.

Tell(): Retorna el byte donde se encuentra el puntero.

Sintaxis: `fileObject.tell()`

Truncate(): Cambia el tamaño del archivo a un número dado de bytes. Si el tamaño no se especifica, la posición actual será usada.

Sintaxis: `fileObject.truncate(size)`

Ejemplo:
Assuming file as f: “Hello! Welcome to my world.”

```
f = open("file.txt", "a")
f.truncate(20)
f.close()

#open and read the file after the truncate:
f = open("file.txt", "r")
print(f.read())
Hello! Welcome to my
```

Existen dos maneras de escribir en un archivo: write() y writelines():

Write(): Inserta la cadena en una única línea en el texto del archivo.

Sintaxis: `fileObject.write(string)`

Writelines(): Es usado es para insertar varias cadenas en un solo comando. Dado una lista de elementos donde estos son cadenas, cada elemento de esta lista se inserta al archivo.
Sintaxis: `fileObject.write(string)`

Ejemplo:

```
# Python program to demonstrate
# writing to file

# Opening a file
file1 = open('myfile.txt', 'w')
L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]
s = "Hello\n"

# Writing a string to file
file1.write(s)

# Writing multiple strings
# at a time
file1.writelines(L)

# Closing file
file1.close()

# Checking if the data is
# written to file or not
file1 = open('myfile.txt', 'r')
print(file1.read())
file1.close()
```

Retorn

```
Hello
This is Delhi
This is Paris
This is London
```

Para Python 2, existe el método xreadlines(), el cual dejo de existir para las versiones superiores de Python 3.

Xreadlines(): retorna un iterador, el cual consiste de leer cada las líneas de un archivo , una a la vez.

Sintaxis: `fileObject. xreadlines()`
Además, podemos indicar el máximo número n de líneas que deseamos en el iterador, así:

```
fileObject. xreadlines()[n]
```