HINTS

MANEJANDO ARCHIVOS CON PYTHON

Para escribir o leer cadenas de caracteres para/desde archivos (otros tipos deben ser convertidas a cadenas de caracteres), Python incorpora un tipo integrado llamado file, el cual es manipulado mediante un objeto archivo el cual fue generado a través de una función integrada en Python. A continuación, se describen los procesos típicos, y sus referencias a funciones propias del lenguaje:

- **Abrir archivo:** la forma preferida para abrir un archivo es usando la función integrada `open()`.
- **Leer archivo:** la forma preferida para leer un archivo es usando algunos de los métodos del tipo objeto file, como: `read()`, `readline()` y `readlines()`.
- **Escribir archivo:** la forma preferida para escribir un archivo es usando el método del tipo objeto file, llamado: `write()`.
- **Cerrar archivo:** la forma preferida para cerrar un archivo es usando el método del tipo objeto file, llamado: `close()`.

ARCHIVOS CON MODULO OS

El módulo **OS** de Python permite realizar operaciones dependientes del Sistema Operativo como: crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, entre otros. Este módulo tiene métodos para ver variables de entornos del Sistema Operativo con las cuales Python está trabajando.

A continuación, algunos útiles métodos del módulo **OS** que pueden ayudar a manipular archivos y carpeta en su programa Python:

Crear una nueva carpeta

```
>>> import os
>>> os.makedirs("Ana_Poleo")
```

Listar el contenido de una carpeta

```
>>> import os
>>> os.listdir("./")
['Ana_Poleo']
```

Mostrar el actual directorio de trabajo

```
>>> import os
>>> os.getcwd()
'/home/usuario/python/'
```

Mostrar el tamaño del archivo en bytes del archivo pasado en parámetro

```
>>> import os
>>> os.path.getsize("Ana_Poleo")
4096
```

¿Es un archivo el parámetro pasado?

```
>>> import os
>>> os.path.isfile("Ana_Poleo")
False
```

¿Es una carpeta el parámetro pasado?

```
>>> import os
>>> os.path.isdir("Ana_Poleo")
True
```

Cambiar directorio/carpeta

```
>>> import os
>>> os.chdir("Ana_Poleo")
>>> os.getcwd()
'/home/usuario/python/Ana_Poleo'
>>> os.listdir("./")
[]
>>> os.chdir("../")
>>> os.getcwd()
'/home/usuario/python'
```

Renombrar un archivo

```
>>> import os
>>> os.rename("Ana_Poleo","Ana_Carolina")
>>> os.listdir("./")
['Ana_Carolina']
```

Eliminar un archivo

```
>>> import os
>>> os.chdir("Ana_Carolina")
>>> archivo = open(os.getcwd()+'/datos.txt', 'w')
>>> archivo.write("Se Feliz!")
>>> archivo.close()
>>> os.getcwd()
'/home/usuario/python/Ana_Carolina'
>>> os.listdir("./")
['datos.txt']
>>> os.remove(os.getcwd()+"/datos.txt")
>>> os.listdir("./")
[]
```

Eliminar una carpeta

```
>>> os.rmdir("Ana_Carolina")
>>> os.chdir("Ana_Carolina")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
OSError: [Errno 2] No such file or directory: 'Ana_Carolina'
```

Lanza una excepción ***OSError*** cuando intenta acceder al directorio que previamente eliminó y este no encuentra.

EJEMPLOS DE ARCHIVOS

A continuación, se presentan algunos ejemplos del uso del tipo objeto file:

Ejemplo de iterar un archivo para leerlo

Se puede iterar sobre un archivo, tal como se muestra a continuación:

```
>>> archivo = open('datos.txt', 'r')
>>> for linea in archivo:
...     print linea
...
Este es una prueba

y otra prueba
>>> archivo.close()
```

Ejemplo de iterar un archivo con escritura y lectura

Se puede manipular un archivo con permisos de escritura y lectura, además de interactuar del mismo, tal como se muestra a continuación:

```
import os
print "\nCrear un archivo"
print "================"
NOMBRE_ARCHIVO = 'datos.txt'
archivo = open(NOMBRE_ARCHIVO, 'w') # abre el archivo datos.txt
archivo.write('Este es una prueba \ny otra prueba.')
archivo.close()
if NOMBRE_ARCHIVO in os.listdir("."):
    print "\nArchivo creado en la ruta: \n\n\t{0}/{1}".format(
        os.getcwd(), NOMBRE_ARCHIVO)
else:
    print "El archivo no fue creado!!!\n"
print "\n\nLeer un archivo"
print "===============\n"
archivo = open(NOMBRE_ARCHIVO, 'r')
contenido = archivo.read()
print contenido
archivo.close()
print "\n\nIterar sobre un archivo"
print "=======================\n"
archivo = open(NOMBRE_ARCHIVO, 'r')
for linea in archivo:
    print linea
print "\n"
archivo.close()
print "\nEliminar un archivo"
print "==================="
os.remove(os.getcwd()+"/"+NOMBRE_ARCHIVO)
print "\nEliminado archivo desde la ruta: \n\n\t{0}/{1}".format(
    os.getcwd(), NOMBRE_ARCHIVO)
```

Si un archivo no existe y se intenta abrir en modo lectura, se generará un error; en cambio, si se lo abre para escritura, Python se encargará de crear el archivo al momento de abrirlo, ya sea con w, a, w+ o con a+).

En caso de que no se especifique el modo, los archivos serán abiertos en modo sólo lectura (r).

Si un archivo existente se abre en modo escritura (w o w+), todos los datos anteriores son borrados y reemplazados por lo que se escriba en él.

MÉTODO APPEND

El método **append()** permite a los programas abrir archivos de datos en directorios especificados como si estuvieran en el directorio actual. Si se usa sin parámetros, **append()** muestra la lista de directorios anexados.

```
append [[<drive>:]<path>[;…]] [/x[:on|:off]] [/path:[:on|:off] [/e]
```

Lista de parámetros que acepta el método **append()**

Parámetro	Descripción
[\<drive>:]<path>	Especifica una unidad y un directorio que se anexarán.
/x:on	Aplica directorios anexados a búsquedas de archivos e inicio de aplicaciones.
/x:off	Aplica directorios anexados solo a las solicitudes para abrir archivos. La **opción /x:off** es la configuración predeterminada.
/path:on	Aplica directorios anexados a solicitudes de archivo que ya especifican una ruta de acceso. **/path:on es** la configuración predeterminada.
/path:off	Desactiva el efecto de **/path:on**.
/e	Almacena una copia de la lista de directorios anexados en una variable de entorno denominada APPEND. **/e** solo se puede usar la primera vez que use **append** después de iniciar el sistema.
;	Borra la lista de directorios anexados.
/?	Muestra la ayuda en el símbolo del sistema.

Este comando no se admite en Windows 10