Métodos de Strings

Éstos son los métodos de cadena que soportan tanto las cadenas de caracteres de 8 bit como los objetos Unicode:

`capitalize` ()
Devuelve una copia de la cadena con el primer carácter en mayúscula.

`count` (*sub*[, *start*[, *end*]])
Devuelve cuántas veces aparece *sub* en la cadena S`[`*start*`:`*end*`]`. Los argumentos opcionales *start* y *end* se interpretan según la notación de corte.

`encode` ([*encoding*[,*errors*]])
Devuelve una versión codoficada de la cadena. La codificación predeterminada es la codificación predeterminada de cadenas. El parámetro opcional *errors* fija el esquema de gestión de errores. Su valor predeterminado es `'strict'`, que indica que los errores de codificación harán saltar `ValueError`. Otros valores posibles son `'ignore'` (ignorar) y `'replace'` (reemplazar).

`endswith` (*suffix*[, *start*[, *end*]])
Devuelve verdadero si la cadena finaliza con el sufijo *suffix* especificado, en caso contrario falso. Si se da valor al parámetro opcional *start*, la comprobación empieza en esa posición. Si se da valor al parámetro opcional *end*, la comprobación finaliza en esa posición.

`expandtabs` ([*tabsize*])
Devuelve una copia de la cadena con todos los tabuladores expandidos a espacios. Si no se indica el paso de tabulación *tabsize* se asume `8`.

`find` (*sub*[, *start*[, *end*]])
Devuelve el menor índice de la cadena para el que *sub* se encuentre, de tal modo que *sub* quede contenido en el rango

[*start*, *end*). Los argumentos opcionales *start* y *end* se interpretan según la notación de corte. Devuelve `-1` si no se halla *sub*.

`index` (*sub*[, *start*[, *end*]])
Como `find()`, pero lanza `ValueError` si no se encuentra la subcadena.

`isalnum` ()
Devuelve verdadero si todos los caracteres de la cadena son alfanuméricos y hay al menos un carácter. En caso contrario, devuelve falso.

`isalpha` ()
Devuelve verdadero si todos los caracteres de la cadena son alfabéticos y hay al menos un carácter. En caso contrario, devuelve falso.

`isdigit` ()
Devuelve verdadero si todos los caracteres de la cadena son dígitos y hay al menos un carácter. En caso contrario, devuelve falso.

`islower` ()
Devuelve verdadero si todos los caracteres alfabéticos de la cadena están en minúscula y hay al menos un carácter susceptible de estar en minúsculas. En caso contrario, devuelve falso.

`isspace` ()
Devuelve verdadero si todos los caracteres de la cadena son espacio en blanco (lo que incluye tabuladores, espacios y retornos de carro) y hay al menos un carácter. En caso contrario, devuelve falso.

`istitle` ()
Devuelve verdadero la cadena tiene forma de título (anglosajón) y hay al menos un carácter. En caso contrario, devuelve falso. Se considera que una cadena tiene formato de título si todas sus

palabras están en minúsculas a excepción de la primera letra de cada una, que debe ser mayúscula.

`isupper` ()
Devuelve verdadero si todos los caracteres alfabéticos de la cadena están en mayúscula y hay al menos un carácter susceptible de estar en mayúsculas. En caso contrario, devuelve falso.

`join` (*seq*)
Devuelve una cadena formada ppor la concatenación de todos los elementos de la secuencia *seq*. Los elementos se separan por la cadena que proporciona el método. Se lanza `TypeError` si alguno de los elementos no es una cadena.

`ljust` (*width*)
Devuelve la cadena justificada a la izquierda en una cadena de longitud *width*. Se rellena la cadena con espacios. Se devuelve la cadena original si *width* es menor que `len(`*s*`)`.

`lower` ()
Devuelve una copia de la cadena convertida en minúsculas.

`lstrip` ()
Devuelve una copia de la cadena con el espacio inicial eliminado.

`replace` (*old, new*[*, maxsplit*])
Devuelve una copia de la cadena en la que se han sustituido todas las apariciones de *old* por *new*. Si se proporciona el argumento opcional *maxsplit*, sólo se sustituyen las primeras *maxsplit* apariciones.

`rfind` (*sub* [*,start* [*,end*]])
Devuelve el índice máximo de la cadena para el que se encuentra la subcadena *sub*, tal que *sub* está contenido en *cadena*`[`*start*`,`*end*`]`. Los argumentos opcionales *start* y *end* se interpretan según la notación de corte. Devuelve `-1` si no se encuentra *sub*.

`rindex` (*sub*[, *start*[, *end*]])
Como `rfind()` pero lanza `ValueError` si no se encuentra *sub*.

`rjust` (*width*)
Devuelve la cadena justificada a la derecha en una cadena de longitud *width*. Se rellena la cadena con espacios. Se devuelve la cadena original si *width* es menor que `len(`*s*`)`.

`rstrip` ()
Devuelve una copia de la cadena con el espacio al final suprimido.

`split` ([*sep* [,*maxsplit*]])
Devuelve una lista de las palabras de la cadena, usando *sep* como delimitador de palabras. Si se indica *maxsplit*, se devolverán como mucho *maxsplit* valores (el último elemento contendrá el resto de la cadena). Si no se especifica *sep* o es `None`, cualquier espacio en blanco sirve de separador.

`splitlines` ([*keepends*])
Devuelve una lista de las líneas de la cadena, dividiendo por límites de línea. No se incluyen los caracteres limitadores en la lista resultante salvo que se proporcione un valor verdadero en *keepends*.

`startswith` (*prefix*[, *start*[, *end*]])
Devuelve verdadero si la cadena comienza por *prefix*, en caso contrario, devuelve falso. Si se proporciona el parámetro opcional *start*, se comprueba la cadena que empieza en esa posición. Si se proporciona el parámetro opcional *end*, se comprueba la cadena hasta esa posición.

`strip` ()
Devuelve una copia de la cadena con el espacio inicial y final suprimido.

`swapcase` ()
Devuelve una copia de la cadena con las mayúsculas pasadas a minúsculas y viceversa.

`title` ()
Devuelve una versión con formato título, es decir, con todas las palabras en minúsculas excepto la primera letra, que va en mayúsculas.

`translate` (*table*[, *deletechars*])
Devuelve una copia de la cadena donde se han eliminado todos los caracteres de *deletechars* y se han traducido los caracteres restantes según la tabla de correspondencia especificada por la cadena *table*, que debe ser una cadena de longitud 256.

`upper` ()
Devuelve una copia de la cadena en mayúsculas.

Manejo de archivos en Python.

Para abrir un archivo se debe relacionar un archivo físico (el que está en el disco duro) con un dispositivo lógico (como se identificará el archivo en el programa).

Archivo_logico = open(nombre_archivo_fisico[, modo_acceso][, buffering])

Los argumentos de la función open son:

Nombre_archivo_fisico: Nombre del archivo que se desea acceder.

Mode_acceso: Determina el modo en el que el archivo tiene que ser abierto, es decir: leer, escribir, etc.

Buffering: Si el valor de buffer se establece en 0, ningún almacenamiento temporal se llevará a cabo. Si el valor es 1, el búfer se realizará por línea.

Modos de accesos:

MODO	Descripción
r	Abre un archivo de sólo lectura. El puntero del archivo se coloca en el principio del archivo.
rb	Abre un archivo de sólo lectura en formato binario.
r+	Abre un archivo para lectura y escritura. El puntero del archivo estará en el principio del archivo.
rb+	Abre un archivo para la lectura y la escritura en formato binario.
w	Abre un archivo para escribir solamente. Sobrescribe el archivo si el archivo existe.
wb	Abre un archivo para escribir sólo en formato binario. Sobrescribe el archivo si el archivo existe.
w+	Abre un archivo para escritura y lectura. Sobrescribe el archivo si este existe.
wb+	Abre un archivo, tanto para la escritura y la lectura en formato binario.

Atributos de los archivos

objeto_archivo.closed: Devuelve true si el archivo está cerrado, false en caso contrario.

objeto_archivo.mode: Devuelve el modo de acceso con el que se abrió el archivo.

objeto_archivo.name: Devuelve el nombre del archivo.

Cerrar un archivo

Para cerrar un archivo se debe utilizar la función:

```
objecto_archivo.close()
```

Para escribir en un archivo

Es importante tener en cuenta que las cadenas de caracteres(strings) de Python pueden tener datos binarios y no sólo de texto.

El método write() no añade un carácter de nueva línea ('\ n') al final de la cadena.

La sintaxis para escribir en un archivo es:

```
objecto_archivo.write(string)
```

Para leer desde un archivo

Es importante tener en cuenta que las cadenas de caracteres(strings) de Python pueden tener datos binarios y no sólo de texto.

La sintaxis para leer desde un archivo es:

```
Variable = objecto_archivo.read([count])
```

Donde [count] es la cantidad de caracteres que se desea leer, si se omite se leerá la totalidad del archivo.

A continuación hay un ejemplo de lectura de un archivo, llamado ejemplo.txt, se lee completamente el archivo y se escribe caracter a caracter.

```
ar = open("ejemplo.txt"),"r")
leido = ar.read()
a = len(leido)
for i in range (a):
print leido[i]
ar.close()
```

Un ejemplo de escritura de un archivo, llamado ejemplo1.txt.

```
ar = open("ejemplo.txt"),"w")
ar.write("lo que se desea escribir")
ar.close()
```