Archivos

PROTECO

¿Qué son?

En las computadoras, los archivos son contenedores de información. Estos se pueden ver como los archivos tradicionales que hay en las oficinas. Y al igual que estos, los archivos de computadora pueden consistir de prácticamente cualquier cosa.

Ejemplos

Algunos ejemplos de archivos son:

- Una imagen jpg
- Un archivo de texto plano
- Un audio mp3
- Un PDF
- Un documento de Word

Archivos en Python

Con Python podemos manejar los archivos de una forma sencilla. En particular, en el curso vamos a trabajar con archivos de texto plano.

Archivos

Primero es necesario crear un objeto de tipo ‘archivo’. Esto se hace con la función open.

```
archivo = open('nuevo.txt','w')
```

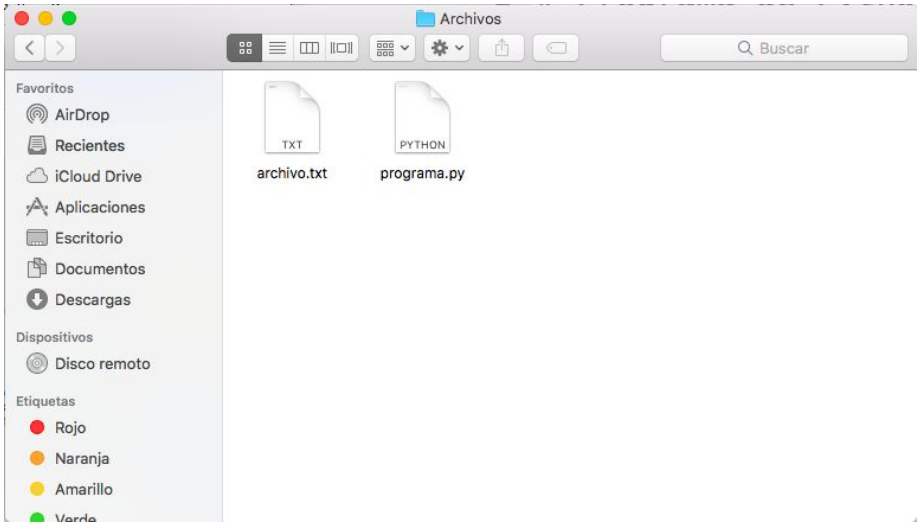
Al igual que con los diccionarios, las listas o los demás tipos de objetos que hemos visto, los objetos de tipo archivo cuentan con métodos.

Archivos

En general, los métodos de los objetos tipo archivo nos van a permitir leer y escribir el contenido de un archivo que se encuentre en la computadora.

Archivos

Para poder acceder al contenido de un archivo, se debe colocar el archivo en la misma carpeta o localización en el que encuentra el script con el programa.



Lectura

Primero necesitamos crear el objeto de tipo archivo en modo lectura. A continuación usamos su método read para guardar todo el texto del archivo en una variable.

```
archivo = open('archivo.txt','r')

texto = archivo.read()
```

Escritura

Hay dos principales formas de escribir el contenido de un archivo, en la primera creamos un archivo o sobreescribimos uno ya existente y en el segundo caso, podemos crear un archivo o escribir al final uno ya existente.

Escritura

Con el primer método, primero tenemos que crear el objeto archivo en modo escritura. Después podemos usar el método write para poder escribir contenido al archivo.

```
archivo = open('archivo.txt','w')

archivo.write("Texto para el archivo")
```

De esta forma, en caso de que el archivo no existiera previamente, se crea, en caso de que ya existiera, se sobreescribe.

Escritura

Para el segundo método, se crea el objeto de la siguiente manera.

```
archivo = open('archivo.txt','a')

archivo.write("Texto que se agrega al final")
```

De esta forma, si el archivo ya existía, agregamos el texto al final del mismo, si no existía, se crea el archivo y agrega el texto.

Archivos

Al crear el objeto archivo, se puede ver que hay dos parámetros. El nombre del archivo que se encuentra en la computadora y el modo de apertura.

```
archivo = open('nuevo.txt','w')
```

Modo de apertura

Los modos de apertura definen que vamos a poder hacer con el archivo, ya sea poder leer su contenido, poder sobreescribir su contenido o poder agregar al final.

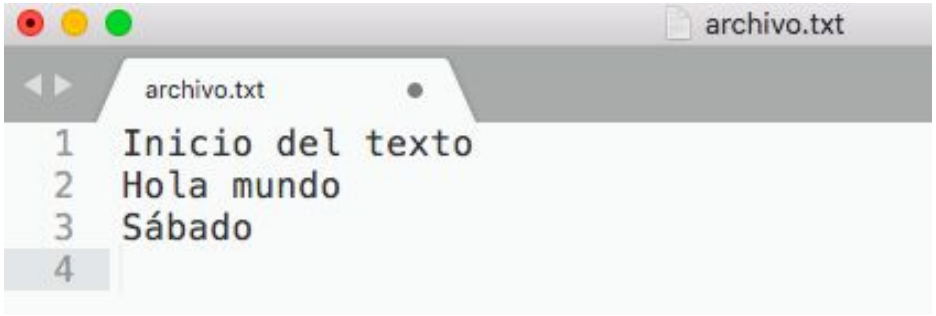
Si abrimos un archivo para lectura y tratamos de escribir en él, va a haber un error al ejecutar el código. Esto pasa de igual forma al revés.

Modo de apertura

- w -> Abre el archivo para escritura, en caso de ya existir, borra todo su contenido. (write)
- r -> Abre archivo para la lectura, en caso de que no exista, hay un error. (read)
- a -> Abre el archivo para escritura, en caso de no existir, lo crea, en caso de que ya existiera, permite la escritura pero esta es SIEMPRE AL FINAL del archivo. (append)
- + -> Permite que el archivo sea tanto de escritura como de lectura (‘w+’, ‘r+’, ‘a+’)

Puntero

Si quisiéramos escribir algo al final y después leer todo el contenido del siguiente archivo podríamos hacerlo con un programa de Python que abra el archivo en modo de ‘append’ para escritura y lectura.




```
archivo = open('archivo.txt','a+')
```

Puntero

Si ejecutamos el siguiente código, el texto se escribiría correctamente, sin embargo, la lectura estaría vacía. Esto se debe a la localización del puntero.

```
>>> archivo = open('archivo.txt','a+')
>>> archivo.write('Final del texto')
15
>>> texto = archivo.read()
>>> texto
''
```

Puntero

Cuando escribimos o leemos algo, el puntero se coloca al final de lo que leímos o escribimos. En el caso anterior al haber escrito algo al final del archivo. El puntero se coloca al final del texto. Para poder realizar una lectura de todo el texto, se necesita colocar el puntero al inicio. Esto se hace con el método seek.

```
>>> archivo.seek(0)
0
>>> texto = archivo.read()
>>> texto
'Inicio del texto\nHola mundo\nSábado\nFinal del texto'
```

Cierre

En general cuando terminamos de trabajar con un archivo, guardamos y lo cerramos, en Python es similar. Esto se hace con el método close.

```
arhivo.close()
```

En general esto se hace de forma automática, pero por buenas prácticas en recomendable hacerlo manualmente.