copyNB

October 28, 2019

1 1. Manejo de archivos

Tanto en python como en muchos otros lenguajes de programación es de suma utilidad el manejo de archivos. Ya sea para leer información relevante que no se encuentra en la memoria volátil o para escribir los resultados de una serie de cálculos.

1. Section **??**
2. Section **??**
3. Section **??**
4. Section **??**
5. Section **??**

1.1 Abrir y cerrar archivos

Antes de poder manipular cualquier archivo es necesario 'abrirlo' para poder manipularlo. Esto lo haremos asignando a una variable el archivo deseado a través de la función **open()**, la cuál recibe el nombre del archivo por abrir y el *modo* en el que será abierto. Más adelante hablaremos más sobre cuáles son estas modalidades. Para cerrar un archivo se utiliza la función **close()**. Es recomendable cerrar un archivo una vez que lo hayas terminado de usar.

En el siguiente código veremos la sintaxis utilizada para abrir y cerrar un archivo titulado *'archivo1.txt'*.

```
archivo = open('archivo1.txt', 'rt') # Aquí podemos ver que el modo de apertura utilizado es '

# Aquí irá el código para manipular el archivo

archivo.close() # Aquí se cierra el archivo
```

Como pudimos ver, este proceso es bastante sencillo pero debemos poner especial atención en cuál será el propósito del archivo a abrir ya que podemos obtener errores si no manejamos el archivo correctamente.

> **Ejercicio** - £De qué tipo es la variable que 'aloja' al archivo? - £Qué crees que signifique el modo 'rt'?

Regresar al Inicio

1.2 Lectura de archivos

El primer uso que se le puede dar a un archivo es para realizar su lectura. Esto se logra usando el modo **r** (por la palabra *read* en inglés) seguido de la letra **t** o **b** dependiendo si el archivo se debe manejar como un archivo de **t**exto o como un **b**inario. Cabe destacar que el modo *'r'* **solamente permite lectura** y que regresa un error si no encuentra el archivo especificado.

Para leer todo el contenido de un archivo utilizamos la función **read()** veamos el siguiente ejemplo.

```
In [2]: archivo = open('../sources/archivo1.txt', 'rt') # Usamos la 't' ya que usaremos un arc

        contenido = archivo.read()

        print(contenido)

        archivo.close()
```

```
Has abierto un archivo!
Descrube todas las formas de manipular archivos en Python.
```

También podemos leer un número específico de caracteres con la función **read()**. Por ejemplo, para leer los primeros 10 caracteres debemos escribir `archivo.read(10)`.

```
In [4]: archivo = open('../sources/archivo1.txt', 'rt')

        print(archivo.read(10))

        archivo.close()
```

```
Has abiert
```

Otra función útil es **readline()** con la cuál podemos leer una linea completa del archivo. Podemos leer varias lineas sucesivamente de la siguiente forma:

```
In [11]: archivo = open('../sources/archivo1.txt', 'rt')

         print(archivo.readline()) # Imprime la primer linea de texto
         print(archivo.readline()) # Imprime la segunda linea de texto

         archivo.close()
```

```
Has abierto un archivo!

Descrube todas las formas de manipular archivos en Python.
```

También podemos **iterar** sobre el archivo para imprimir todo su contenido de la forma que ya hemos visto en capítulos anteriores con la sentencia **for**.

```
In [12]: archivo = open('../sources/archivo1.txt', 'rt')

         for u in archivo: # Así recorremos objetos iterables
             print(u)

         archivo.close()
```

```
Has abierto un archivo!

Descrube todas las formas de manipular archivos en Python.
```

Por último veremos la manera de imprimir la ruta y nombre del archivo que estamos manejando. Esto es con el *atributo* **name**. Observa cómo se usa.

```
In [13]: archivo = open('../sources/archivo1.txt', 'rt')

         print(archivo.name) # Se imprime la ruta y el nombre del archivo

         archivo.close()
```

```
../sources/archivo1.txt
```

Regresar al Inicio

1.3 Escritura de archivos

La siguiente utilidad de los archivos es cuando los usamos para llevar un 'registro' que se almacenará en la computadora y que contendrá la información necesaria para algún proceso. Para abrir un archivo y escribir en él se usa el modo **w** (que viene de *write* en inglés) seguido de la **t** o **b** dependiendo del tipo de archivo. La particularidad de este modo de escritura es que sobreescribe la información que pueda contener el archivo por abrir. Es decir, borra todo y escribe nueva información. Con este modo se crea un archivo nuevo si no encuentra el archivo especificado.

La función necesaria para esto es **write()** en la que debemos específicar qué vamos a escribir.

Veamos un ejemplo en el que leemos la información de un archivo para después escribir sobre él.

```
In [19]: archivo = open('../sources/archivo2.txt', 'rt') # Primero leemos el archivo
         print(archivo.read())
         archivo.close()

         archivo = open('../sources/archivo2.txt', 'wt') # Volvemos a abrir el archivo pero es
         archivo.write("Información Actualizada") # Usamos la función write() para escribir en
         archivo.close()

         archivo = open('../sources/archivo2.txt', 'rt') # Volvemos a leer el archivo
         print(archivo.read())
         archivo.close()
```

```
Información Actualizada
Información Actualizada
```

Cuando no deseamos sobreescribir nuestra información sino escribir más en el archivo usamos la modalidad **a** con la que se añade información al final del archivo. Usemos este modo en un ejemplo.

```
In [25]: archivo = open('../sources/archivo2.txt', 'at')
         archivo.write("\nInformación añadida al final") # Observa la secuencia de escape \n p
         archivo.close()

         archivo = open('../sources/archivo2.txt', 'rt') # Leemos el archivo
         print(archivo.read())
         archivo.close()
```

```
Información Actualizada
Información añadida al final
Información añadida al final
Información añadida al final
Información añadida al final
```

> **Ejercicio** - £Se puede utilizar la función **read()** cuando abrimos el archivo en la modalidad **wt** ? - Crea un archivo de texto llamado 'info.txt' y almacena en él tu nombre y edad.

Regresar al Inicio

1.4 Funciones extra

Ahora hablemos sobre otras modalidades de apertura de archivos, la primera es la modalidad **x** la cuál sirve para crear un nuevo archivo y regresará un error si el archivo ya existe. Nótese que no se puede leer ni escribir en el archivo con esta modalidad pero puede ser útil para verificar la existencia de un archivo y tomar decisiones sobre cómo manipularlo.

La otra modalidad es aquella que nos permite leer y escribir en un archivo tal como ya lo podemos hacer. Simplemente se usa el modo **r+** para tener control total sobre un archivo.

Finalmente vamos a conocer la función **seek()** la cuál sirve para posicionarnos en una parte específica del archivo. Habrás notado que tanto la lectura como la escritura de archivos se realiza en el inicio o en el final de un archivo pero no podíamos posicionarnos en algún punto específico para realizar una acción. Ahora con la función **seek()** podemos indicar en qué linea del archivo queremos posicionarnos para leer o escribir.

Aquí hay un ejemplo:

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

print(archivo.read()) # Al realizar esta acción la posición para realizar acciones en el archi
archivo.seek(0) # Regresamos al inicio del archivo
archivo.write("Cambio ")
```

```
print(archivo.read()) #Imprime la información del archivo escrita después de lo que acabamos d
archivo.seek(0) # Regresamos al inicio del archivo
print(archivo.read())

archivo.close()
```

> **Ejercicio** - Prueba el modo **r+** para leer y escribir sobre un archivo. - ***Experimenta*** con la función **seek()** para modificar un archivo de texto nuevo con múltiples lineas de texto

Regresar al Inicio

2 Sección Yeyecoa

---

1. Escribe un programa que le pida al usuario las coordenadas de dos puntos en el plano y calcule la distancia que existe entre ellos. Después de hacer el cálculo debe crear un archivo de texto en el que escriba en la primer línea las coordenadas del primer punto, en la segunda línea las coordenadas del segundo punto y en la tercera debe escribir el mensaje 'Distancia entre los puntos:' seguido del resultado de tu cálculo. Finalmente debe imprimir un mensaje anunciando en dónde(ruta y nombre del archivo) se encuentran los resultados.

Regresar al Inicio