😎 Archivos

Abrir archivo

Para abrir un archivo es usando la función integrada **open()**

Leer archivo

Para leer un archivo es puede sere usado algunos de los métodos del tipo objeto file como **read()**, **readline()** y **readlines()**

Escribir archivo

Para escribir un archivo es usando el método del tipo objeto file llamado **write()**

Cerrar archivo

Para cerrar un archivo es usando el método del tipo objeto file llamado **close()**

Tiene la siguiente sintaxis:

archivo = open(nombre_archivo, modo, buff)

Parámetros:

> **nombre_archivo**: este argumento es un objeto string que contiene el nombre del archivo al que queremos acceder.

> **modo**: el modo de acceso determina el modo en el que el archivo ha sido abierto, por ejemplo, leer, escribir, anexar etc. Una lista completa de todos los valores posibles se especifica más adelante.

> **buff**: si el valor de buff(buffering) es 0 no se realiza ningún buffering. Si el valor es 1, se ejecuta un buffer de línea.

Modificadores:

> **r** -> abre un archivo solo para solo lectura: El puntero al archivo está localizado al comienzo del archivo. Este es el modo predeterminado de la función.

> **rb** -> abre un archivo solo para lectura en formato binario: El puntero al archivo está localizado al comienzo del archivo. Este es el modo predeterminado de la función.

> **r+** -> abre un archivo para escritura y lectura: El puntero del archivo está localizado al comienzo del archivo.

> **w** -> abre un archivo solo para escritura: Sobreescribe el archivo si este ya existe. Si el archivo no existe, crea un nuevo archivo para escritura.

> **wb** -> abre un archivo solo para escritura en formato binario: Sobreescribe el archivo si este ya existe. Si el archivo no existe, crea un nuevo archivo para escritura.

> **w+** -> abre un archivo para escritura y lectura: Sobreescribe el archivo si este ya existe. Si el archivo no existe, crea un nuevo archivo para escritura.

> **a** -> abre un archivo para su anexo: El puntero del archivo esta al final del archivo si este existe. Es decir, el archivo está en modo anexo. Si el archivo no existe, crea un nuevo archivo para escritura.

> **Nota: los nombres [foo, bar, foobar] son genéricos que se usan usualmente para mencionar un nombre que se ignora.**

Abrir para lectura ('r' de read, 't' de text)

arch = open('foo.txt', 'rt')

Abrir para escritura ('w' de write, 't' de text)

Abrir para escritura ('w' de write, 't' de text)

Para leer el archivo:

arch = f.read(nombre_archivo) #una parte arch = f.read([parte])

Para escribir en el archivo:

arch.write(' texto')

Para cerrar los archivos:

arch.close()

with ... as ...

Su sintaxis general es la siguiente:

with EXPRESIÓN as VARIABLE:

    BLOQUE DE INSTRUCCIONES

En el caso de los archivos, la expresión es una llamada a la función open y la variable es la conexión con el archivo:

with open("ARCHIVO") as archivo:

    BLOQUE DE INSTRUCCIONES

La función open puede tener varios argumentos. Los más importantes son:

with open("ARCHIVO", mode="MODO", encoding="CODIFICACIÓN") as archivo:

    BLOQUE DE INSTRUCCIONES

In [ ]:
```
salarios={}
with open('archs/Base_salarios.csv', 'rt') as f:
    for line in f:
        print(line)
```

BLOQUE DE INSTRUCCIONES

- **"ARCHIVO" es la ruta absoluta o relativa hasta el archivo.**
- **"MODO" indica si el archivo se abre para leer, escribir o ambas cosas y si se trata de un archivo de texto o binario. El modo predeterminado es lectura en modo texto.**
- **"CODIFICACIÓN" indica el juego de caracteres del archivo: "utf-8" (UTF-8), "cp1252" (ASCII para Europa occidental), etc.**

Nota: En algunos sistemas operativos es probable que se deba especificar el encoding agregando `encoding='utf8'` como parámetro al comando open.

Qué codificación utiliza nuestro sistema?

In [11]:
```
import locale
print(locale.getpreferredencoding())
```

```
cp1252
```

Las operaciones que pueden realizarse sobre un archivo:

Abrir archivos en python mediante el objeto file

Sin with as

f = open("archivotext.txt", "w") #Creamos el archivo f.write("Creando archivo de texto en python") I f.close()

Con with as

with open ("archivotext.txt", "w") as f: f.write("Creando archivo de texto en python usando whit as") #<-Escribimos en el archivo f.close()

Métodos del objeto file en python

seek(byte)

Este método se corresponde al puntero que te mencionaba y junto al método tell() nos permitirá ubicarnos al final o al principio de los archivos. Pero el método seek() fue diseñado para archivos binarios, no para archivos de texto. Este método moverá el puntero hacia el byte indicado como argumento. seek (desplazamiento, desde_donde) Si colocamos un cero '0' como argumento a seek nos posiciona al inicio del archivo.

tell

Devuelve el puntero a una posición en un momento dado. Junto con seek() nos permitirán posicionarnos al principio o al final. tell() nos devuelve un entero que indica la posición del puntero.

read([bytes])

Nos permite leer un archivo completo, salvo que indiquemos los bytes entonces solo leerá hasta los bytes determinados.

Ejemplo:

with open ("un_archivo.txt", "r", encoding='utf8') as f: # Abrimos de modo Read (r)

```
    contenido = f.read() #<-Lo abrimos utilizando el metodo read y lo almacenamos en la variable
```

print (contenido) #Imprimimos la variable que tiene el contenido

readline()

El método readline lee una línea por vez. Si colocamos un solo readline nos leerá una sola línea y si no especificamos a partir de donde será la primera por defecto. Y cada vez que hagamos un readline() tomará la siguiente línea.

with open ("letra_pueblo_blanco.txt", "r") as f: #Abrimos el archivo de modo read (r)

```
    linea1 = f.readline() #<-Almacenamos en variable linea1 el método readline()
    linea2 = f.readline() #<-Volvemos a almacenar en variable linea2 el método readline()
    print (linea1)
    print (linea2)
```

Suponiendo que quisiéramos imprimir a partir de la línea 19 a la 31 inclusive, deberíamos hacerlo iterando el archivo de texto con la función predefinida: enumerate e imprimir solo las líneas que nos interesan.

with open ("letra_pueblo_blanco.txt", "r", encoding='utf-8') as f: #Abrimos como modo read (r)

```
    una_parte=(linea for i,linea in enumerate(f) if i>=19 and i<=31)
    for linea in una_parte:
    print (linea)
```

readlines()

Lee todas las líneas en forma de lista separando las líneas con el carácter de escape para los saltos de línea "/n".

with open ("letra_pueblo_blanco.txt", "r", encoding='utf-8') as f:

```
    texto = f.readlines()
    print (texto)
```

Se puede escribir en el archivo

- **Con la función print() añadiendo el argumento file= archivo, donde archivo es la variable utilizada en la expresión with ... as ...**
- **Con el método write sobre el objeto archivo, donde archivo es la variable utilizada en la expresión with ... as ... La función print() añade un salto de línea al final de la cadena agregada al archivo, pero el método write no lo hace, por lo que habrá que agregarlo explícitamente.**

[w] Cada vez que se ejecuta este programa, se crea nuevamente el archivo, por lo que este sólo contiene la palabra Hola.

ruta = "prueba.txt" with open(ruta, mode="w", encoding="utf-8") as archivo:

```
    print("Hola", file= archivo)
```

ruta = "prueba.txt" with open(ruta, mode="w", encoding="utf-8") as archivo:

```
    archivo.write("Hola\n")
```

[x] Al ejecutar por segunda vez este programa (o a la primera si el archivo ya existe), se genera un error.

ruta = "prueba.txt" with open(ruta, mode="x", encoding="utf-8") as archivo:

```
    print("Hola", file= archivo)
```

ruta = "prueba.txt" with open(ruta, mode="x", encoding="utf-8") as archivo:

```
    archivo.write("Hola\n")
```

[a] Cada vez que se ejecuta este programa, se añade al archivo una línea con la palabra Hola (si el archivo no existe, se crea)

ruta = "prueba.txt" with open(ruta, mode="a", encoding="utf-8") as archivo:

```
print("Hola", file= archivo)
```

ruta = "prueba.txt" with open(ruta, mode="a", encoding="utf-8") as archivo:

```
archivo.write("Hola\n")
```

r+ De esta manera nuestra cadena se agregará al final

with open ("letra_pueblo_blanco.txt", "r+") as f: f.write("\n\nPueblo Blanco - Joan Manuel Serrat") contenido= f.read() print (contenido)

writelines Nos permite escribir una secuencia (elemento iterable) dentro de él línea a línea por ejemplo: Una lista

Lista = ["Banana\n", "Sandia\n", "Mandarina\n", "Naranjas\n", "Papaya\n", "Manzana\n"]

with open ("frutas.txt", "w+") as f:

```
f.writelines(Lista)
contenido = f.read()
print (contenido)
```

Abrir automáticamente en el navegador

Si el archivo que se ha creado con Python es un página web, puede ser útil que se abra en el navegador para ver el resultado inmediatamente. Para ello se puede utilizar la función open() del módulo webbrowser de la biblioteca estándar, que abre el archivo indicado con la aplicación asociada en el sistema operativo.

In [12]:

```
import webbrowser

ruta = "prueba.html"
with open(ruta, mode="w", encoding="utf-8") as archivo:
    print("<!DOCTYPE html>", file=archivo)
    print('<html lang="es">', file=archivo)
    print("<head>", file=archivo)
    print(' <meta charset="utf-8">', file=archivo)
    print(" <title>HTML 5</title>", file=archivo)
    print( '<meta name="viewport" content="width=device-width, initial-scale=1.0">', file=archivo,)
    print("</head>", file=archivo)
    print("", file=archivo)
    print("<body>", file=archivo)
    print(" <p>Esta página es una página HTML válida.</p>",file=archivo)
    print("</body>", file=archivo)
    print("</html>", file=archivo)
    webbrowser.open(ruta)
```

Archivos con modulo os

El módulo os de Python le permite a usted realizar operaciones dependiente del Sistema Operativo como crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, etc. Este módulo tiene métodos para ver variables de entornos del Sistema Operativo con las cuales Python está trabajando en mucho más.

Crear una nueva carpeta

os.makedirs("Mi_carpeta")

Listar el contenidos de una carpeta

os.listdir("./")

Mostrar el actual directorio de trabajo

os.getcwd()

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

os.path.getsize("Mi_carpeta")

¿Es un archivo el parámetro pasado?

os.path.isfile("Mi_carpeta")

¿Es una carpeta el parámetro pasado?

os.path.isdir("Mi_carpeta")

Cambiar directorio/carpeta

os.chdir("Mi_carpeta")

os.getcwd()

os.listdir("./")

os.chdir("../")

os.getcwd()

Renombrar un archivo

os.rename("Mi_Carpeta","Tu_carpeta")

os.listdir("./")

Eliminar un archivo

os.chdir("Tu_carpeta")

archivo = open(os.getcwd() + '/archivo_texto.txt', 'w')

archivo.write("Hoy es un gran día!")

archivo.close()

os.getcwd()

os.listdir("./")

os.remove(os.getcwd() + "/archivo_texto.txt")

os.listdir("./")

Eliminar una carpeta

os.rmdir("Tu_carpeta")

os.chdir("Tu_carpeta")

Traceback (most recent call last): File "", line 1, in OSError: [Errno 2] No such file or directory: 'Tu_carpeta'

```
# Lanza una excepción OSError cuando intenta acceder al directorio que previamente elimino y este no encuentr
a
```

In [ ]: