Python: ficheros

SISTEMAS DE GESTIÓN EMPRESARIAL – 148FA (DAM)

Trabajando con archivos

Python nos permite trabajar en dos niveles diferentes con respecto al sistema de archivos y directorios.

1. **A través del módulo os:** facilita el trabajo con todo el sistema de archivos y directorios, a nivel del propios Sistema Operativo.
2. **El nivel que nos permite trabajar con archivos**, manipulando su lectura y escritura a nivel de la aplicación y tratando a cada archivo como un objeto.

Objeto ***file***

Objeto file

Manipular una de ellas como un objeto File, es posible, cuando a ésta, se le asigna como valor un archivo.

Para asignar a una variable un valor de tipo file, se utiliza la función **open()**

La función integrada open(), recibe dos **parámetros**:

- El primero de ellos, es **la ruta hacia el archivo** que se desea abrir
- Y el segundo, **el modo** en el cual abrirlo

Objeto file

Modos de apertura: para qué queremos abrir el archivo (lectura, escritura, creación…)

- Cuando abrimos un archivo, estamos creando un **puntero**. Este puntero se posicionará en un determinado lugar del archivo (generalmente al principio o al final).
- Podremos mover el puntero a una nueva posición mediante el número de *byte* correspondiente
- Este puntero, se creará -en inicio- dependiendo del modo de apertura indicado, el cuál será indicado a la función ***open()*** como una *string* en su segundo parámetro.

Modos de apertura de los archivos

La opción open() puede recibir un parámetro para indicar el modo en el que se abrirá el archivo. Los tres modos principales de apertura son:

1. **Modo de sólo lectura** (r): no es posible realizar modificaciones al archivo, sólo leer su contenido.
2. **Modo de sólo escritura** (w): el archivo es vaciado si existe y si no, lo crea.
3. **Modo de escritura posicionándose al final del archivo** (a): se crea el archivo, si no existe. En caso de que exista se posiciona al final, manteniendo el contenido original.

En caso de que no se especifique, se abrirá en modo de sólo lectura (r)

Indicador	Modo de apertura	Ubicación del puntero
r	Solo lectura	Al inicio del archivo
rb	Solo lectura en modo binario	Al inicio del archivo
r+	Lectura y escritura	Al inicio del archivo
rb+	Lectura y escritura en modo binario	Al inicio del archivo
w	Solo escritura. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
wb	Solo escritura en modo binario. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
w+	Escritura y lectura. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
wb+	Escritura y lectura en modo binario. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
a	Añadido (agregar contenido). Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
ab	Añadido en modo binario (agregar contenido). Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
a+	Añadido (agregar contenido) y lectura. Crea el archivo si éste no existe.	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
ab+	Añadido (agregar contenido) y lectura en modo binario. Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo

Objeto file: métodos

Método: read([bytes]) Lee todo el contenido de un archivo. Si se le pasa la longitud de bytes, leerá solo el contenido hasta la longitud indicada.

```
archivo = open(“fichero.txt", "r")
contenido = archivo.read()
print (contenido)
```

Objeto file: métodos

Método: readline([bytes]) Lee una línea del archivo.

```
archivo = open(“fichero.txt", "r")
linea1 = archivo.readline()
print (linea1)
```

```
linea = archivo.readline()
while linea !='':
    # procesar línea
    linea=archivo.readline()

# Otro modo
for linea in archivo:
    # procesar línea
```

Objeto file: métodos

Método: readlines() Lee todas las líneas de un archivo.

```
archivo = open(“fichero.txt", "r")
for linea in archivo.readlines():
     print (linea)

# Guardo en una variable todas las líneas
lineas = archivo.readlines()
```

Objeto file: métodos

Método: seek(byte) Mueve el puntero hacia el byte indicado

```
archivo = open(“fichero.txt", "r")
contenido = archivo.read()
# el puntero queda
# al final del documento
archivo.seek(0)
```

Objeto file: métodos

Método: close() Cierra un archivo.

```
archivo = open(“fichero.txt", "r")
contenido = archivo.read()
archivo.close()
print (contenido)
```

Ejemplo open y close

Ejemplo: apertura de un archivo en el que imprime todas las líneas con su número

```
archivo = open(“archivo.txt”)
i = 1
for linea in archivo:
    linea = linea.rstrip(‘\n’)
    print (‘{0}: {1}’.format(i, linea))
    i+=1
archivo.close()
```

Objeto file: métodos

Método: tell() Retorna la posición actual del puntero.

```
archivo = open(“fichero.txt", "r")
linea1 = archivo.readline()
mas = archivo.read(archivo.tell() * 2)
if archivo.tell() > 50:
    archivo.seek(50)
```

Objeto file: métodos

Método: write(cadena) Escribe cadena dentro del archivo.

```
archivo = open(“fichero.txt", "r+")
contenido = archivo.read()
final_de_archivo = archivo.tell()

archivo.write('Nueva linea')
archivo.seek(final_de_archivo)
nuevo_contenido = archivo.read()

print (nuevo_contenido)
# Nueva linea
```

Objeto file: métodos

Método: writelines(secuencia). Escribe una lista de cadenas

```
archivo = open(“fichero.txt", "r+")
contenido = archivo.read()
final_de_archivo = archivo.tell()
lista = ['Línea 1\n', 'Línea 2']
archivo.writelines(lista)
archivo.seek(final_de_archivo)
print (archivo.readline())
# Línea 1
print (archivo.readline())
# Línea 2
```

Objeto file: propiedades

Las propiedades principales del objeto file son las siguientes:

- **closed**: devuelve True si el archivo se ha cerrado. De lo contrario, False.
- **mode**: devuelve el modo de apertura.
- **name**: devuelve el nombre del archivo
- **encoding**: devuelve la codificación de caracteres de un archivo de texto

Agregar información a un archivo

- Si abrimos el archivo en **Modo de escritura posicionándose al final del archivo** (a), podemos escribir al final de él.
- De este modo se puede escribir un archivo de **log**, en el que se pueda ver los distintos eventos que han ido sucediendo tras la ejecución de nuestro programa.
- En los archivos de log suele utilizarse la fecha y hora actual en la que ha ocurrido el evento. En Python se utiliza **datetime** para obtener la fecha y hora actual.

Ejemplo de programa para generar archivo de log

Persistencia de datos

- **Persistencia de datos**: capacidad de guardar la información de un programa para poder utilizarla en otro momento.
- **Guardar archivo + Cerrar archivo + Abrir archivo**

Persistencia de datos

Ejemplo:

- Queremos guardar la información de los participantes de un torneo junto a su puntuación y tiempo
- Dicha información se puede guardar en tuplas:
- [(nombre1, puntos1), (nombre2, puntos2), (nombre3, puntos3), …]
- Toda la información estará guardada en un archivo de texto: cada tupla ocupará una línea y cada valor de cada tupla estará separado por una coma.
- **Nota**: será necesario convertir la puntuación de cada jugador en una cadena, a pesar de ser numérico. Y al abrir el archivos será necesario convertirlo de nuevo en un valor numérico.

Archivos CSV

- Los archivos CSV (Comma Separated Values) suelen usarse para transferir datos entre programas.
- Es un formato sencillo en el que los valores generalmente están separados por comas, tabuladores u otros elementos (siempre siendo el mismo el que los separa).

  “Nombre”,”Apellido”,”DNI”
  “Jon”,”Aguirre”,”11223344K”
  “Iratxe”,”Gil”,”12345678B”

- Los archivos csv pueden ser considerados como una pequeña base de datos, en los que la primera línea será la cabecera y las siguientes los datos separados por comas.

Archivos CSV

Método: **csv.reader(nombre_archivo, otros parámetros)**: sirve para leer datos de un archivo csv
Método: **csv.writer(nombre_archivo, otros parámetros)**: sirve para escribir datos a un archivo csv

Como ejemplo, se pueden añadir estos parámetros:

- delimiter: cadena usada para separar campos. Por defecto es ‘ , ’
- strict: si se establece como True, lanza una excepción Error cuando halle una mala entrada csv
- …

Archivos CSV

Método: **csv.DictReader**: crea un objeto que mapea la información leída a un diccionario, cuyas claves están dadas por el parámetro "*fieldnames*". Es un parámetro opcional, pero si no se especifica, la primera fila de datos se considera como las claves del diccionario.

Método: **csv.DictWriter**: Escribe los datos a un archivo csv. El parámetro "*fieldnames*" define la secuencia de claves del diccionario, para indicar el orden en el que los datos serán escritos en el archivo csv.

```
import csv
with open('nombre_archivo.csv') as csvfichero:
reader = csv.DictReader(csvfichero)
for fila in reader:
        print(fila['nombre'], fila['apellido'])
```

Archivos CSV

```
import csv

resultados = []
with open(‘nombre_archivo.csv') as csvfichero:
    reader = csv.DictReader(csvfichero)
    for fila in reader:
        resultados.append(fila)
    print (resultados)
```

csv.Dictreader

```
import csv

with open('nombre_archivo.csv') as csvfichero:
    reader = csv.reader(csvfichero, delimiter=',', quotechar=',',
                        quoting=csv.QUOTE_MINIMAL)
    for fila in reader:
        print(fila)
```

csv. reader

Archivos CSV

csv.writer

```
import csv

datos = [[“Nombre", “Apellido", “Nota"],
          [‘Aitor', ‘Alonso', 'A'],
          [‘Leire', ‘Aguirre', 'B']]

mi_fichero = open(‘archivo_ejemplo.csv', 'w')
with mi_fichero :
    writer = csv.writer(mi_fichero)
    writer.writerows(datos)

print(“Proceso de escritura completado")
```

Archivos CSV

```
datos = [{'Nota': 'B', 'Nombre': 'Aitor', 'Apellido': ‘Urrutia'},
{'Nota': 'A', 'Nombre': 'Miren', 'Apellido': 'Rodriguez'},
{'Nota': 'C', 'Nombre': 'Mikel', 'Apellido': 'Loidi'},
{'Nota': 'B', 'Nombre': 'Juan', 'Apellido': 'Lopez'},
{'Nota': 'A', 'Nombre': 'Olaia', 'Apellido': 'Vela'}]
```


Queremos escribir estos datos en un archivo csv

Archivos CSV

csv.Dictwriter

```
import csv

with open('datos_ejemplo.csv', 'w') as csvfichero:
    campos = ['Nombre', 'Apellido', 'Nota']
    writer = csv.DictWriter(csvfichero, fieldnames= campos)

    writer.writeheader()
    writer.writerow({'Nota': 'B', 'Nombre': 'Aitor', 'Apellido': 'Urrutia'})
    writer.writerow({'Nota': 'A', 'Nombre': 'Miren’, 'Apellido': 'Rodriguez'})
    writer.writerow({'Nota': 'C', 'Nombre': 'Mikel', 'Apellido': 'Loidi'})
    writer.writerow({'Nota': 'B', 'Nombre': 'Jose', 'Apellido': 'Lopez'})
    writer.writerow({'Nota': 'A', 'Nombre': 'Olaia', 'Apellido': 'Vela'})


print(“Proceso de escritura completado")
```

Escribe una línea

Archivos CSV

```
import csv

with open('archivo_csv_alumnos.csv', 'w') as csvfichero:
    campos = ['Nombre', 'Apellido', 'Nota']
    writer = csv.DictWriter(csvfichero, fieldnames= campos)

    writer.writeheader()
    writer.writerows([{'Nota': 'B', 'Nombre': 'Aitor', 'Apellido': 'Urrutia'},
                      {'Nota': 'A', 'Nombre': 'Miren', 'Apellido': 'Rodriguez'},
                      {'Nota': 'C', 'Nombre': 'Mikel', 'Apellido': 'Loidi'},
                      {'Nota': 'B', 'Nombre': 'Juan', 'Apellido': 'Lopez'},
                      {'Nota': 'A', 'Nombre': 'Olaia', 'Apellido': 'Vela'}])

print("Proceso de escritura completado")
```

csv.Dictwriter

→ Escribe todas las líneas

Archivos CSV

Ejercicios:

1. Crea un programa en el que lea un archivo csv y que imprima los resultados por pantalla. Deberás crear un archivo csv o por código o a través Excel, Calc o un Bloc de notas

2. Crea una lista de datos para después guardarlos en un archivo csv nuevo.