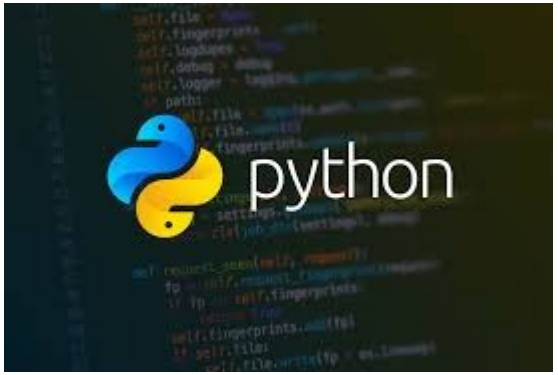
CODERHOUSE

Comisión: 50190

Clase 8

- Operaciones sobre archivos



CODERHOUSE

Operaciones sobre Archivos

- Creación && Apertura
  - open(file, mode)
  - file: path y nombre del archivo
  - mode: modo del archivo
- Escritura
  - write(“Texto a escribir”)
- Eliminar
  - os.remove(filename)
- Lectura
  - read()
  - readlines()
  - readline()

Modos de Apertura

"r" - Lectura (Read) - Valor por default. Abre un archivo para lectura. Da error si el archivo no existe

"a" - Agregar (Append) - Abre el archivo para agregar informacion. Si el archivo no existe lo crea

"w" - Escritura (Write) - Abre el archivo para escritura. Si el archivo no existe lo crea.

"x" - Crea un archivo (Create) - Crea un archivo. Retorna un error si el archivo existe.

Con el modo, tambien se puede especificar si el archivo es binario o de texto.

"t" - Texto - Default value.

"b" - Binario - Binary mode,

Archivos JSON

JSON es un formato para el intercambio de datos basado en texto. Por lo que es fácil de leer tanto para una persona como para una maquina. El nombre es un acrónimo de las siglas en inglés de JavaScript Object Notation. Lo que indica que su origen se encuentra vinculado al lenguaje JavaScript. Aunque hoy en día puede ser utilizado desde casi todos los lenguajes de programación. JSON se ha hecho fuerte como alternativa a XML, otro formato de intercambio de datos que requiere más metainformación y, por lo tanto, consume más ancho de banda y recursos.

Los datos en los archivos JSON son pares de propiedad valor separados por dos puntos. Estos pares se separan mediante comas y se encierran entre llaves. El valor de una propiedad puede ser otro objeto JSON, lo que ofrece una gran flexibilidad a la hora de estructurar información. Esta estructura de datos recuerda mucho a los diccionarios de Python.

JSON – Tipos de Datos I

String:

En JSON los strings siempre deben ir entre comillas dobles y la key debe ser una secuencia de caracteres:

{“Nombre”:“Juan”}

Números

En el caso de los números, estos siempre deberán ser enteros o decimales:

{“Edad”:“30”}

Null

Identifica una key a la que aún no se le asigna un valor

{“Apellido”:null}

JSON – Tipos de Datos II

Objetos

Por su parte, cuando asignamos objetos a un valor necesitamos emplear signos de llave para contener la información del conjunto. La información contenida debe mantener el mismo formato, lo que da una apariencia mucho más simple y limpia:

{“Empleado”:{“Nombre”:“Juan”, “Edad”:“30”, “Ciudad”:“Madrid”}}

Arreglos

Los arreglos deben ingresarse de una forma bastante similar al formato y sintaxis de los strings y números. La única diferencia radica en que no utilizaremos signos de llaves, sino corchetes, para contener los datos:

{“Empleados”:[“Juan”, 30, “Pedro”]}

Booleanos

Los valores booleanos se utilizan cuando la información solo puede ser verdadera (true) o falsa (false). Por ejemplo, cuando una venta ha sido concretada o se requiere autenticación:

{“Venta”:true}

JSON – Python

Python	JSON Equivalent
`dict`	object
`list`, `tuple`	array
`str`	string
`int`, `float`, `int`	number
`True`	true
`False`	false
`None`	null

JSON – Libreria json

Es parte de la libreria standard

Dumps: Diccionario → json -- Convierte un diccionario en un string json

```
>>> import json
>>> print(json.dumps({'4': 5, '6': 7}, sort_keys=True, indent=4))
{
    "4": 5,
    "6": 7
}
```

loads: json → diccionario -- Convierte un string json en un diccionario