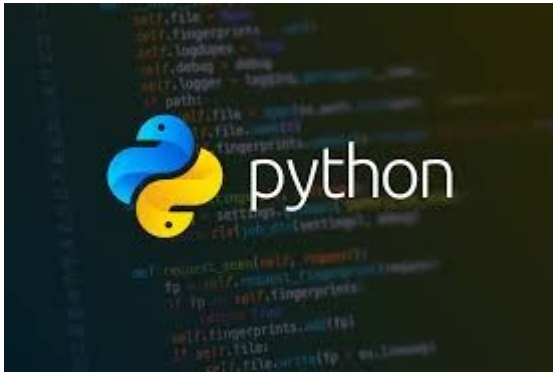
Introducción al manejo de archivos

Python Flex - 75130

Temas

- Manejo de archivos y Datos



Objetivos

- Persistencia
- Archivos - Concepto
- Archivos – Operaciones
- Archivos texto
- Archivos json
- Archivos csv

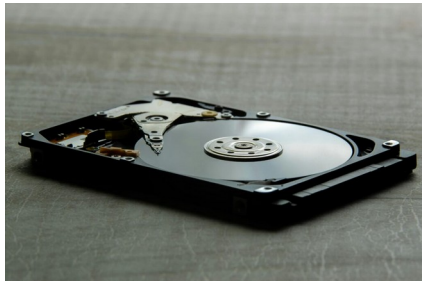
Persistencia I

Cuando uno piensa en la palabra persistencia, lo relaciona a la capacidad humana de “aguantar”, “perdurar” o “de no dejar de hacer una acción”, en este curso no hablaremos de esas cosas obviamente, pero sí de la persistencia de los datos.

Persistencia II

Hasta el momento todos los programas que realizamos ya tenían datos; por ejemplo: las edades de las personas, nombres, apellidos, lista de números, etc. Pero todos esos datos había que generarlos cada vez que usábamos nuestro programa y los datos se perdían de un ejemplo al otro o entre las clases.

La persistencia es lo que nos va a permitir guardar y recuperar los datos que se generaron en algún programa (ya sea nuestro o ajeno).



Persistencia III

Definición

La persistencia en programación se refiere a la capacidad de almacenar y recuperar datos de manera permanente, de tal forma que estos datos sobrevivan al cierre de la aplicación que los creó. Esto permite que la información sea reutilizable en futuras ejecuciones del programa.

Persistencia IV

Datos Efímeros vs. Datos Persistentes

Datos Efímeros: Son aquellos datos que solo existen mientras el programa está en ejecución. Una vez que el programa se cierra, estos datos se pierden. Un ejemplo de datos efímeros podría ser una variable en memoria que se utiliza para cálculos temporales.

Datos Persistentes: Son aquellos que se almacenan de manera permanente, de modo que pueden ser recuperados y utilizados en futuras ejecuciones del programa. Estos datos se guardan en medios de almacenamiento no volátiles, como discos duros o SSDs.

Tipos de Persistencia

El guardado de datos se puede hacer en dos grandes estructuras.

* En base de datos

* En archivos

Base de datos

Las bases de datos son sin duda la mejor alternativa para almacenar y explorar los datos. Estas son complejas y por lo general necesitan de la instalación de algún motor o programa que nos permita utilizarlas.

Trabajaremos con ellas mas adelante en este curso; así que por ahora las ignoraremos.

Archivos

Los archivos son la forma más antigua, primitiva y simple de almacenar datos.

Aún se sigue utilizando este mecanismo en más de una aplicacion; incluso en programas bastante sofisticados.

En esta clase aprenderemos a guardar datos en archivos y recuperarlos.

Tipos de Archivos I

Hay dos grandes tipos de archivos:

Los **binarios** son aquellos archivos que mejoran su eficiencia pero los datos están guardados bajo agrupaciones de bytes; lo que hace que solo la pc pueda decodificarlos.

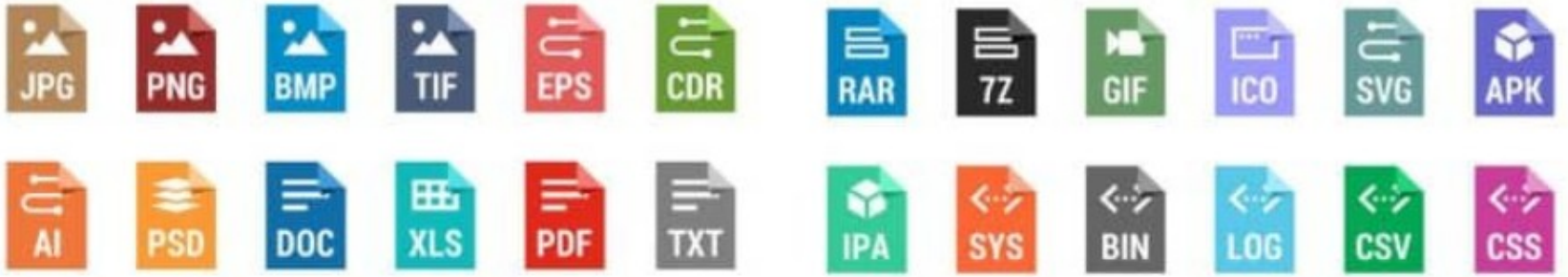
Los de **texto** son lo que uno imagina, texto que guarda el dato en particular de una forma bastante descriptiva e intuitiva.

Ver ejemplos!!!!!

Tipos de Archivos II

Por simplicidad en el curso trabajaremos solo con archivos de texto.

A su vez, los archivos de texto pueden ser de muchas extensiones, las más clásicas son: .txt, .doc, .docx, .xml, .csv, .json, etc.



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 para 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