PYTHON

Módulo 7 – Manejo de archivos

MANEJO DE ARCHIVOS

ENTRADA Y SALIDA DE DATOS DE FICHERO

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Tipos de ficheros:
    - De texto
      - Páginas web (es un fichero de texto)
      - Datos (puede ser de texto)
    - Ejecutables
    - Imágenes
    - Sonidos
    - Videos
    - Datos

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Características de los ficheros:
    - Tienen un nombre
    - Tienen un tamaño
    - Las operaciones básicas son lectura y escritura
    - Antes de operar con un fichero es necesario abrirlo
    - Después de operar con un fichero es necesario cerrarlo
    - Solo un programa puede escribir en un fichero en un momento determinado

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Apertura:
    - *identificador_fichero* = open(*"nombre_fichero"*, *character*)

Character	Meaning
`'r'`	open for reading (default)
`'w'`	open for writing, truncating the file first
`'x'`	open for exclusive creation, failing if the file already exists
`'a'`	open for writing, appending to the end of the file if it exists
`'b'`	binary mode
`'t'`	text mode (default)
`'+'`	open for updating (reading and writing)

https://docs.python.org/3/library/functions.html#open

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Apertura:
    - *identificador_fichero* = open(*"nombre_fichero"*, *modo*)
    - *modo:*
      - "r"→Lectura
      - "w"→Escritura
      - "a"→ Añadir (*append*)
      - "x"→Apertura para creación. Si el fichero existe genera error.
      - "rb","wb","ab","xb" →Modo binario (bytes)
      - "rt","wt","at","xt" →Modo texto (por defecto)
      - "a+"→Añadir y escritura. Se posiciona al final del fichero.
      - "w+"→Lectura y escritura. Trunca el fichero en la apertura.
      - "r+"→Lectura y escritura. Respeta el fichero en la apertura. Se posiciona al principio del fichero.

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Apertura:
    - “r+” →Lee y escribe

```
fichero = open("datos.txt","r+")
linea = fichero.readline()
print(linea)
fichero.write("Nueva linea")
fichero.close()
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Apertura:
    - “w+” →Trunca, no lee nada y escribe.

```
fichero = open("datos.txt","w+")
linea = fichero.readline()
print(linea)
fichero.write("Nueva linea")
fichero.close()
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Apertura:
    - “w+” →Trunca, escribe, posiciona y lee.

```
fichero = open("datos.txt","w+")
fichero.write("Nueva linea")
fichero.seek(0) #Posiciona el puntero en el inicio del fichero
linea = fichero.readline()
print(linea)
fichero.close()
```

El método seek posiciona el puntero del fichero en la posición indicada

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Apertura:
    - Excepciones:
      - FileNotFoundError
      - IOError
      - OSError

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Cierre:
    - *identificador_fichero.*close()

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Apertura ¿sin cierre?:
    - Usando la sentencia with.
    - No requiere el cierre del fichero. Se realiza automáticamente.

```
with open("salida.txt","r") as fichero:
    contenido=fichero.readlines()
    print(contenido)
print("Fichero cerrado:" + str(fichero.closed))
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura:
    - *identificador_fichero.*read() → Permite todo el contenido de un fichero.

```
fichero = open("entrada.txt","r")
todo = fichero.read()
print(todo)
fichero.close()
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura:
    - *identificador_fichero.*read(*numero_caracteres*) → Permite leer un número determinado de caracteres.
    - Es más eficiente leer múltiplos de 512 caracteres (bytes)

```
fichero = open("salida.txt","r")
items = fichero.read(3)
print(items)
fichero.close()
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura:
    - Línea a línea mediante bucle for (el final de fichero se detecta automáticamente)

```
fichero = open("salida.txt","r")
for linea in fichero:
    print(linea)
fichero.close()
print("\nFin de la lectura")
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura:
    - Carácter a carácter mediante bucle for (cuando lee “” es que ha llegado al final de fichero –EOF-)

```
fichero = open("ficheros_basico.py","r")
c = fichero.read(1)
while c!="":
    print(c,end="")
    c = fichero.read(1)
fichero.close()
print("\nFin de la lectura")
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura:
    - *identificador_fichero.*readline() → Permite leer una línea completa.
    - El retorno incluye un salto de línea final.

```
fichero = open("salida.txt","r")
linea = fichero.readline()
print(linea)
fichero.close()
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura:
    - Si el fichero se ha leído por completo, la lectura devuelve una cadena vacía “”.

```
fichero = open("salida.txt","r")
while True:
    caracter = fichero.read(1)
    if(caracter)=="":
        break
    else:
        print(caracter,end="")
fichero.close()
print("\nFin de la lectura")
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura:
    - Si el fichero se ha leído por completo, la lectura devuelve una cadena vacía “”.

```
fichero = open("salida.txt","r")
while True:
    linea = fichero.readline()
    if(linea)=="":
        break
    else:
        print(linea,end="")
fichero.close()
print("\nFin de la lectura")
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura con el método readlines.
  - Devuelve una lista de string.

```
fichero = open("salida.txt","r")
contenido = fichero.readlines()
print(contenido)
fichero.close()
```

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Escritura:
    - Aperturas con “w” y “a”
    - Método write
    - Método writelines → Permite escribir múltiples líneas (además de estructuras de datos). No incluye los saltos de línea.
    - Método writable →Indica si está abierto para escritura.

MANEJO DE ARCHIVOS

- Entrada y salida de datos de fichero
  - Lectura y escritura de bytes.
  - Ejemplo: copiar el contenido de un fichero.

```
f_entrada = open("entrada.pdf","rb")
f_salida = open("salida.pdf","wb")
caracter = f_entrada.read(1)
while caracter:
    f_salida.write(caracter)
    caracter = f_entrada.read(1)
f_salida.close()
f_entrada.close()
```

MANEJO DE ARCHIVOS

MANEJO DE DOCUMENTOS JSON

MANEJO DE ARCHIVOS

- Ejemplo:


```
{

"Title":"Indiana Jones and the Last Crusade",
"Year":"1989",
"Rated":"PG-13","Released":"24 May 1989","Runtime":"127 min","Genre":"Action, Adventure","Director":"Steven
Spielberg","Writer":"Jeffrey Boam, George Lucas, Menno Meyjes",
"Actors":"Harrison Ford, Sean Connery, Alison Doody",
"Plot":"In 1938, after his father Professor Henry Jones, Sr. goes missing while pursuing the Holy Grail, Professor
Henry \"Indiana\" Jones, Jr. finds himself up against Adolf Hitler's Nazis again to stop them from obtaining its
powers.",
"Language":"English, German, Greek, Arabic","Country":"United States","Awards":"Won 1 Oscar. 8 wins & 22 nominations
total","Poster":"https://m.media-
amazon.com/images/M/MV5BMjNkMzc2N2QtNjVlNS00ZTk5LTg0MTgtODY2MDAwNTMwZjBjXkEyXkFqcGdeQXVyNDk3NzU2MTQ@._V1_SX300.jpg",
"Ratings":
            [
            {"Source":"Internet Movie Database","Value":"8.2/10"},
            {"Source":"Rotten Tomatoes","Value":"88%"},
            {"Source":"Metacritic","Value":"65/100"}
            ],
"Metascore":"65","imdbRating":"8.2",
"imdbVotes":"714,187",
"imdbID":"tt0097576",
"Type":"movie","DVD":"13 May 2008","BoxOffice":"$197,171,806","Production":"Paramount Pictures, Lucasfilm
Ltd.","Website":"N/A","Response":"True"
}
```

MANEJO DE ARCHIVOS

- JSON en Python.
  - import json
  - Objeto json.
  - Métodos:
    - loads → Convierte un texto en formato json a diccionario.
    - load → Permite leer un fichero json, obteniendo un diccionario.
    - dump →Permite escribir un diccionario en un fichero.
    - dumps →Convierte un diccionario a texto en formato json.

MANEJO DE ARCHIVOS

- Uso de load

```
import json
fichero = open("indiana-jones.json","r")
diccionario = json.load(fichero)
fichero.close()
#Recorrer todos los elementos
for k, v in diccionario.items():
  print(str(k)+":"+str(v))
#Acceder a un elemento complejo
print(diccionario["Ratings"][0]["Source"])
```

MANEJO DE ARCHIVOS

- Uso de loads

```
import json
fichero = open("indiana-jones.json","r")
contenido = fichero.read()
fichero.close()
#Recorrer todos los elementos
diccionario = json.loads(contenido)
for k, v in diccionario.items():
  print(str(k)+":"+str(v))
#Acceder a un elemento complejo
print(diccionario["Ratings"][0]["Source"])
```

MANEJO DE ARCHIVOS

- Uso de dumps

```
import json
diccionario = {
  "Saturno devorando a su hijo":["Francisco de Goya",1823],
  "La condesa de Vilches": ["Federico de Madrazo",1853],
  "La tentación de San Antonio": ["Salvador Dalí",1946],
  "Retrato de Encarna y su hija":["Antonio Ortiz Echagüe",1926]
}
with open("salida-json.json","w") as fichero:
  fichero.write(json.dumps(diccionario))
```

MANEJO DE ARCHIVOS

- Uso de dumps

```
import json
diccionario = {
  "Saturno devorando a su hijo":["Francisco de Goya",1823],
  "La condesa de Vilches": ["Federico de Madrazo",1853],
  "La tentación de San Antonio": ["Salvador Dalí",1946],
  "Retrato de Encarna y su hija":["Antonio Ortiz Echagüe",1926]
}
contendido_json = json.dumps(diccionario)
print(contendido_json)
```

MANEJO DE ARCHIVOS

MANEJO DE DOCUMENTOS XML

MANEJO DE ARCHIVOS

- Formato XML
- XML en Python:
  - https://docs.python.org/3/library/xml.html
    - Procesadores (parsers):
      - DOM
      - SAX
      - MiniDOM

MANEJO DE ARCHIVOS

- Ejemplo de fichero XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<mensaje importancia="alta">
    <de>Raquel</de>
    <a>David</a>
    <asunto>Recogida</asunto>
    <texto>Te espero a las 8 en la puerta de la oficina</texto>
</mensaje>
```

MANEJO DE ARCHIVOS

- Leer ficheros XML con minidom:

```
from xml.dom import minidom
doc = minidom.parse("mensaje.xml")

mensaje = doc.getElementsByTagName("mensaje")[0]
importancia = mensaje.getAttribute("importancia")#Acceso al atributo
print(importancia)
emisor = doc.getElementsByTagName("de")[0]
print(emisor.firstChild.data)#Acceso al texto
```

MANEJO DE ARCHIVOS

BASES DE DATOS SQLITE3

https://www.sqlite.org/index.html

MANEJO DE ARCHIVOS

- Base de datos embebida en Python3.
  - import sqlite3
- Apertura y cierre de conexión:
  - connection = sqlite3.connect(database_file_name)
  - connection.close()

MANEJO DE ARCHIVOS

- Tipos de datos:
  - https://www.sqlite.org/datatype3.html

Each value stored in an SQLite database (or manipulated by the database engine) has one of the following storage classes:

- **NULL**. The value is a NULL value.
- **INTEGER**. The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
- **REAL**. The value is a floating point value, stored as an 8-byte IEEE floating point number.
- **TEXT**. The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).
- **BLOB**. The value is a blob of data, stored exactly as it was input.

MANEJO DE ARCHIVOS

- Base de datos embebida en Python3.
  - import sqlite3
- Apertura y cierre de conexión:
  - connection = sqlite3.connect(database_file_name)
  - connection.close()

MANEJO DE ARCHIVOS

- Creación del esquema:

```
def create_squema():
    cursor = connection.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS clientes (id INTEGER PRIMARY KEY
AUTOINCREMENT, nombre TEXT NOT NULL, edad INTEGER NOT NULL)")
    connection.commit()
    cursor.close()
```

MANEJO DE ARCHIVOS

- Insertar elemento:

```
def insertar_persona(nombre, edad):
    cursor = connection.cursor()
    cursor.execute(f"INSERT INTO clientes (nombre, edad) VALUES ('{nombre}',{edad})")
    connection.commit()
    cursor.close()
```

MANEJO DE ARCHIVOS

- Insertar elementos:

```
def insertar_personas(lista_personas):
    cursor = connection.cursor()
    for personas in lista_personas:
        cursor.execute(f"INSERT INTO clientes (nombre, edad) VALUES
('{personas[0]}',{personas[1]})")
    connection.commit()
    cursor.close()
```

MANEJO DE ARCHIVOS

- Insertar lista:

```
def insertar_personas_execute_many(lista_personas):
    cursor = connection.cursor()
    cursor.executemany("INSERT INTO clientes (nombre, edad) VALUES (?,?)",lista_personas)
    connection.commit()
    cursor.close()
```

MANEJO DE ARCHIVOS

- Borrar todos los elementos de una tabla:

```
def delete_all_personas():
    cursor = connection.cursor()
    cursor.execute("DELETE FROM clientes")
    connection.commit()
    cursor.close()
```

MANEJO DE ARCHIVOS

- Borrar un elemento de una tabla:

```
def delete_persona_by_id(id):
    cursor = connection.cursor()
    cursor.execute("DELETE from clientes WHERE id=?",(id,))
    modificaciones = cursor.rowcount
    connection.commit()
    cursor.close()
    return(modificaciones)
```

MANEJO DE ARCHIVOS

- Modificar un elemento:

```
def update_persona(id, nombre):
    cursor = connection.cursor()
    cursor.execute("UPDATE clientes SET nombre=? WHERE id=?",(nombre,id))
    connection.commit()
    cursor.close()
```

MANEJO DE ARCHIVOS

- Obtener todos los elemento:
  - Fechall()→Devuelve una lista de tuplas.

```
def consultar_personas():
    cursor = connection.cursor()
    registros = cursor.execute("SELECT * from clientes").fetchall()
    cursor.close()
    return registros
```