GESTIÓ DE FITXERS I FORMATS D'EMMAGATZEMAMENT D'INFORMACIÓ

1. Introducció

Aquesta unitat didàctica té com a objectiu que l'alumnat adquireixi els coneixements i habilitats necessaris per gestionar fitxers de dades, comprendre els diferents formats d'emmagatzematge d'informació (CSV, JSON) i utilitzar mètodes d'accés adequats.

Fitxers

Un fitxer és un conjunt de **dades emmagatzemades** en un dispositiu de memòria (com el disc dur, SSD, o memòria flash) que es poden llegir, escriure i gestionar mitjançant un sistema operatiu.

Els fitxers permeten **organitzar** i **conservar informació** de manera persistenta perquè estigui disponible fins i tot després de tancar un programa o apagar l'ordinador.

---

2. Tipus de fitxers

Els fitxers poden ser:

De `text`, quan contenen dades llegibles per humans (com ara **.txt**, **.csv**)

```
John Doe
Jane Smith
Alice Johnson
```

`Binaris`, quan emmagatzemen informació en format no llegible directament (com **.jpg**, **.exe**).

```
01010101101010101011101010101011
```

TREBALL AMB FITXERS A PYTHON

Estructura bàsica d'un fitxer

Un fitxer és una seqüència de bytes que es guarda en un dispositiu d'emmagatzematge. Té:

- un **nom**
- una **extensió**
- **permisos** d'accés

Permisos d'accés

Els permisos d'accés a fitxers en Windows es gestionen per mitjà del sistema de seguretat de fitxers, que **permet establir qui pot llegir, escriure o executar un fitxer** o directori. Aquests permisos es poden aplicar tant a **fitxers** individuals com a **carpetes** i es poden personalitzar per a **usuaris** i **grups** específics.

Tipus de permisos d'accés

- **Lectura** (Read):
  - Permet llegir el contingut del fitxer o la llista de fitxers d'una carpeta.
  - No permet modificar ni executar el fitxer.
- **Escriptura** (Write):
  - Permet modificar el contingut d'un fitxer o afegir nous fitxers a una carpeta.
  - No permet llegir ni executar el fitxer.

Tipus de permisos d'accés:

- **Execució** (Execute):
  - Permet executar fitxers que siguin programes o scripts.
  - Aquest permís s'afegeix en fitxers executables com `.exe, .bat, .cmd`, o scripts com `.ps1`.
- **Modificació** (Modify):
  - Permet llegir, escriure i eliminar fitxers, així com canviar-ne el contingut.
  - No permet canviar els permisos d'altres usuaris.
- **Control total** (Full Control):
  - Inclou tots els permisos: llegir, escriure, executar, modificar, eliminar, i també permet modificar els permisos d'altres usuaris.
  - És el permís més alt i permet la gestió completa del fitxer o carpeta.

Python ens permet treballar amb fitxers, utilitzant diferents funcions que ja venen implementades.

Modes d'obertura d'arxius en Python

Per treballar amb un fitxer, primer cal obrir-lo en un mode específic (lectura, escriptura, afegir, etc.), i després tancar-lo per evitar pèrdues de dades o errors en el sistema.

1. `'r'`: **lectura** (default). Obre el fitxer per llegir-lo. L'error es produeix si el fitxer no existeix.
2. `'w'`: **escriptura**. Crea un fitxer nou (si no existeix) o sobreescriu-lo si ja existeix.
3. `'a'`: **afegir**. Obre el fitxer per afegir contingut al final sense esborrar el que ja hi ha.
4. `'x'`: creació **exclusiva**. Crea un fitxer nou, però genera un error si el fitxer ja existeix.

Modes d'obertura d'arxius en Python

5. `'b'`: **binari**. Específic per obrir fitxers en mode binari (per exemple, per llegir o escriure imatges).
6. `'t'`: **text** (default). Utilitzat per treballar amb fitxers de text. No és necessari especificar-lo ja que és el mode per defecte.
7. `'+'`: **lectura** i **escriptura**. Permet llegir i escriure al mateix temps (combinat amb altres modes, per exemple `'r+'` o `'w+'`).

Exemples d'obertura d'arxius:

- `open('arxiu.txt', 'r')`: Només lectura.
- `open('arxiu.txt', 'w')`: Esborrar i escriure.
- `open('arxiu.txt', 'a')`: Afegir al final.
- `open('arxiu.txt', 'rb')`: Llegir en mode binari.

Treball amb Python

Per a llegir un arxiu l'hem de obrir en **mode lectura**, utilitzant la funció `open()`:

```
f = open("arxiu.txt", "r")
```

Una vegada obert, ja podem llegir-lo amb la funció `read()`:

```
f = open("arxiu.txt", "r")
text = f.read()
```

Quan l'hem acabat de llegir, es important tancar el fitxer amb la funció `close()`:

```
f = open("arxiu.txt", "r")
text = f.read()
f.close()
```

Imprimir contingut per pantalla

Si volem imprimir per pantalla tot el contingut de l'arxiu:

```
f = open("arxiu.txt", "r")
text = f.read()
print(text)
f.close()
```

Llegir línea per línia

También podemos ir leyendo línea por línea

```
f = open("arxiu.txt", "r")
text = f.read()
print(text)
f.close()

for linea in f:
    print(linea)

archivo.close()
```

Eliminar un archivo

Para borrar un archivo podemos utilizar la librería os. Para ello, al principio de nuestro archivo debemos importarla.

```
# Eliminar un archivo llamado "test.txt"
import os
os.remove("archivo.txt")
```

Mode escriptura

Si utilizamos el argumento `w`, abriremos el archivo en modo escritura. De este modo:

- Si el archivo ya existe, se **sobrescribe** completamente, borrando el contenido previo.
- Si el archivo no existe, se crea un **nuevo archivo**.

```
f = open("alumnes.txt", "w")
f.write("Daniel")
f.close()
```

Mode annexar

Si utilizamos el argumento `a`, abriremos el archivo en modo anexar. De este modo:

- Si el archivo ya existe, se añade nuevo contenido al final del archivo sin eliminar el contenido existente.
- Si el archivo no existe, se crea un nuevo archivo.

```

```

```
# abrir el archivo
f = open('alumnes.txt', 'a')
f.write('Pepe')
```

Si volem afegir el contingut a una nova línia, haurem de fer servir el caràcter d'escapament `\n`. Aquest farà que en el lloc on estigui col·locat es faci un bot de línia.

Esborrarem el contingut previ amb `w`

```
f = open('alumnes.txt', 'w')
file.write("Dani\nAna")
file.write("\n")
file.write("Pepe")
```

Ara tindrem un arxiu amb 3 línies.

Activitat 1

Crea un programa que et demani el nom, pes, edat i altura, i calculi el teu índex de masa corporal . A continuació, hauràs de guardar a cada línea les dades d'una persona.

$$ IMC = \frac{\text{pes (kg)}}{\text{altura (m)}^2} $$

El resultat haurà de ser:

```
Dani (40) 75 kg 1.74 m IMC: 24.77
```

A continuació veurem com esborrar una línea, canviar el contingut o inserir línies al mig de l'arxiu.

Borrar una línea concreta

Para borrar una línea específica, puedes leer todas las líneas, eliminar la línea que no quieres, y luego escribir de nuevo el archivo.

```
f = open('alumnes.txt', 'r')
linies = f.readlines()
del linies[1]
f = open('alumnes.txt', 'w')
file.writelines(linies)
```

Línies és una llista, recorda, que els seus elements s'acceceixen amb un índex, i que el primer és el 0: `línia[0]`, `linia[1]`, etc. Si tot va bé, ha d'haver desaparegut el segón nom.

Modificar una línea concreta

Si deseas modificar una línea específica, puedes seguir un proceso similar, pero actualizando el contenido de la línea en memoria antes de escribirlo de nuevo.

```
f = open('alumnes.txt', 'r')
linies = f.readlines()
linies[1] = "Lola\n"
f = open('alumnes.txt', 'w')
file.writelines(linies)
```

Insertar una línea en una posición concreta

Si deseas insertar una línea en una posición específica, puedes insertar la nueva línea en la lista de líneas antes de escribir de nuevo el archivo.

```
f = open('alumnes.txt', 'r')
linies = f.readlines()
linies.insert(1, "Joan\n")
f = open('alumnes.txt', 'w')
file.writelines(linies)
```

Format CSV

Els fitxers CSV (Comma-Separated Values) són fitxers de text on les dades es separen per comes o altres delimitadors. Són útils per emmagatzemar taules de dades i intercanviar informació entre aplicacions.

```
name,age
Lionel Messi,35
Cristiano Ronaldo,38
Kylian Mbappé,25
```

Guardar dades en un fitxer CSV

Hi ha un mòdul de Python per treballar amb fitxers CSV.

```
import csv

dades = [
    ["Nom", "Edat", "Ciutat"],
    ["John", 28, "New York"],
    ["Alice", 30, "Los Angeles"],
    ["Bob", 25, "Chicago"]
]

with open('treballadors.csv', mode='w', newline='') as file:
    escriptor_csv = csv.writer(file)

    for linia in dades:
        escriptor_csv.writerow(linia)
```

Llegir des d'un fitxer

```
import csv

with open('treballadors.csv', mode='r', newline='') as arxiu:
    lector_csv = csv.DictReader(arxiu)

    # Read each row as a dictionary
    for row in lector_csv:
        print(row)
```

Format **JSON**

`JSON` (JavaScript Object Notation) és un format lleuger per a l'emmagatzematge i intercanvi de dades. És fàcilment llegible per humans i estructurat de manera que també és senzill de processar per les màquines. S'utilitza àmpliament en aplicacions web, APIs i bases de dades.

Característiques principals de JSON

- Textual i **llegible**: utilitza una sintaxi clara basada en claus i valors.
- Lleuger: ocupa poc espai i no conté caràcters innecessaris.
- Independent del llenguatge: encara que es basa en **JavaScript**, es pot utilitzar amb **Python**, **Java**, **C#**, etc.
- Basat en estructures de dades: s'organitza amb objectes i llistes.

Estructura bàsica de JSON

JSON es basa en dos tipus d'estructures fonamentals:

- Els `objectes` JSON són representats amb **{}** (claus).
- Les `claus` són sempre strings (text entre cometes dobles): **name, age**
- Els `valors` poden ser strings, nombres, booleans, arrays o altres objectes: **Lionel Messi, 35**.

```
{
  "name": "Lionel Messi",
  "age": 35
}
```

Aquest objecte té dos parells clau-valor.

Les `llistes` o arrays JSON son representades amb **[]** (claudàtors) i contenen una llista ordenada de valors. Els valors poden ser de qualsevol tipus compatible amb JSON. Aquest objecte té un parell clau-valor, i el valor és una llista amb 3 elements.

```
{
  "players": [
    {
      "name": "Lionel Messi",
      "position": "Forward",
      "team": "Paris Saint-Germain",
      "nationality": "Argentine",
      "age": 35
    },
    {
      "name": "Cristiano Ronaldo",
      "position": "Forward",
      "team": "Al Nassr",
      "nationality": "Portuguese",
      "age": 38
    },
    {
      "name": "Kylian Mbappé",
      "position": "Forward",
      "team": "Paris Saint-Germain",
      "nationality": "French",
      "age": 25
    }
  ]
}
```

Treball amb Python

Python inclou un `mòdul json` per treballar amb fitxers JSON fàcilment Guardar i carregar dades en format JSON.

On es fa servir JSON?

- APIs Web: comunicació entre aplicacions i serveis.
- Bases de dades NoSQL: MongoDB utilitza JSON per emmagatzemar dades.
- Configuració d'aplicacions: molts programes utilitzen arxius .json per guardar configuracions.