

- **GESTIÓ DE FITXERS I FORMATS D'EMMAGATZEMAMENT D'INFORMACIÓ**
  - 1. Fitxers
  - 2. Tipus de fitxers
  - 3. Treball amb Python
  - 4. Format CSV
  - 5. FORMAT JSON
  - 6. JOC amb JSON

# GESTIÓ DE FITXERS I FORMATS D'EMMAGATZEMAMENT D'INFORMACIÓ

---

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.

## 1. 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.

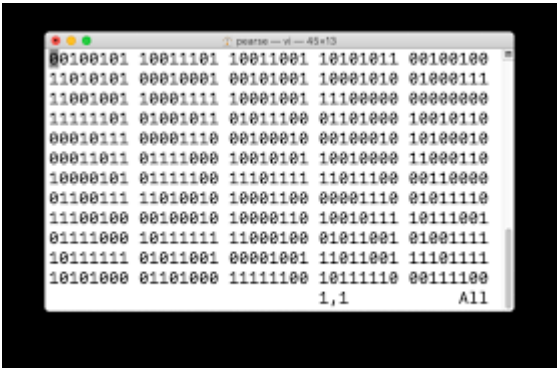
## 2. Tipus de fitxers

Els fitxers poden ser:

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

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

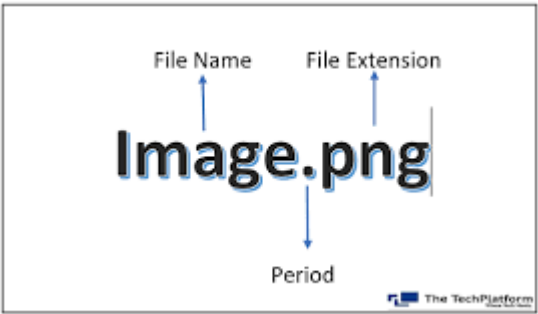
**Binariis**, quan emmagatzemen informació en format no llegible directament (com **.jpg**, **.exe**).



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



3. Treball amb Python

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.

Mode	Tipus	Descripció
r	Lectura	Obre el fitxer per llegir-lo. L'error es produeix si el fitxer no existeix.
w	Escriptura	Crea un fitxer nou (si no existeix) o sobreescriu-lo si ja existeix.
a	Afegir	Obre el fitxer per afegir contingut al final sense esborrar el que ja hi ha.
x	Creació	Crea un fitxer nou, però genera un error si el fitxer ja existeix.
b	Binari	Per obrir fitxers en mode binari (per exemple, per llegir o escriure imatges).
t	Text	Per treballar amb fitxers de text. No cal especificar-lo, ja que és el mode per defecte.
+	Lectura/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.

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 línia 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 línia 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 línia, 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'accedeixen amb un índex, i que el primer és el 0: `linia[0]`, `linia[1]`, etc. Si tot va bé, ha d'haver desaparegut el segon 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)
```

### Inserir 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)
```

## 4. 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)
```

## 5. 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

#### Parelles clau-valor

JSON es basa en dos tipus d'estructures fonamentals:

- 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"
```

#### Objectes

Els **objectes** JSON són representats amb **{}** (claus). Aquest objecte té dos parells clau-valor.

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

#### Llistes

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
}
]
```

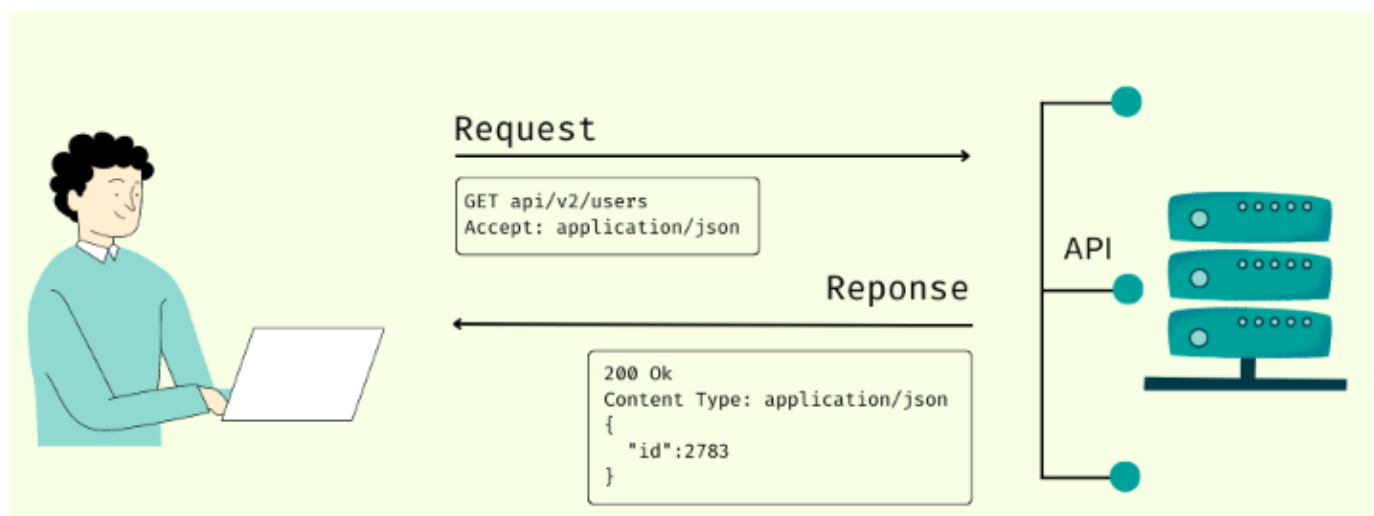
## Treball amb Python

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

## On es fa servir JSON?

Els arxius JSON tenen aplicacions molt variades, d'entre elles destaquen les següents:

- **APIs Web:** comunicació entre aplicacions i serveis.



- **Bases de dades NoSQL:** MongoDB utilitza JSON per emmagatzemar dades.



### Person

name	phone
John	3634
Sue	6343

### Orders

personName	date	product
John	2002	Gizmo
John	2004	Gadget
Sue	2002	Gadget

```
{
  "Person": [
    {
      "name": "John",
      "phone": 3646,
      "Orders": [
        {
          "date": 2002,
          "product": "Gizmo"
        },
        {
          "date": 2004,
          "product": "Gadget"
        }
      ]
    },
    {
      "name": "Sue",
      "phone": 6343,
      "Orders": [
        {
          "date": 2002,
          "product": "Gadget"
        }
      ]
    }
  ]
}
```

- **Configuració d'aplicacions:** molts programes utilitzen arxius .json per guardar configuracions.

```
{
  "SmartCursorToggle": true,
  "MapEnabled": true,
  "InvasionBarMode": 2,
  "AutoSave": true,
  "AutoPause": false,
  "Language": 1,
  "PlacementPreview": true,
  "GoreVisualsAllowed": true,
  "VolumeSound": 0.518797,
  "VolumeAmbient": 0.5714286,
  "VolumeMusic": 0.5263158,
  "UseExperimentalFeatures": true,
  "Fullscreen": false,
  "WindowMaximized": false,
}
```

Amb aquest joc aprendrem a [guardar i llegir dades amb Python i JSON](#).

### Guardar i llegir a un arxiu JSON

```
import json

arxiu = open("configuracio.json", "w")

configuracio = {}
configuracio["nom"] = "Dani"
configuracio["professio"] = "professor"

json.dump(configuracio, arxiu, indent=2)
```

Això generarà un fitxer `configuracio.json` amb el següent format:

```
{
  "nom": "Dani",
  "professio": "professor"
}
```

Ara, per fer el contrari:



```
import json
# Obrir el fitxer en mode lectura
arxiu = open("configuracio.json", "r")
configuracio = json.load(arxiu) # Carregar el JSON a un diccionari

# Extreure i mostrar els valors directament
print(f"Nom: {configuracio['nom']}")
print(f"Professió: {configuracio['professio']}")
```

## 6. JOC amb JSON

En aquest exemple veurem com utilitzar JSON per emmagatzemar la configuració i les dades de les partides, de manera que quedin guardades en un fitxer quan el programa acabi, i puguem carregar-les de nou quan el programa es torni a obrir per utilitzar aquestes dades.

A més del fitxer Python amb el joc, crearem un fitxer de text `config.json` en el qual guardarem tota la informació que necessitem.

Nombre ^	Fecha de modificación	Tipo	Tamaño
 config	04/02/2025 18:06	Archivo JSON	1 KB
 juego	04/02/2025 18:08	Archivo PY	3 KB

### Arxiu de configuració

El contenido del archivo .json será el siguiente.

```
{
  "usuario": "Dani",
  "dificultad": 10,
  "victorias": 2,
  "derrotas": 0
}
```

### Arxiu de joc

L'arxiu quedarà tal que així

```
import json # Mòdul per treballar amb fitxers i dades en format JSON
(serialització i deserialització).
import random # Mòdul per generar valors aleatoris, com números, eleccions
aleatòries en llistes, etc.
import os # Mòdul per interactuar amb el sistema operatiu, com gestionar fitxers
i directoris.

# Nombre del archivo JSON donde guardaremos los parámetros
CONFIG_FILE = "config.json"

# Función para guardar los parámetros en un archivo JSON
def guardar_configuracion(data, filename=CONFIG_FILE):
    with open(filename, "w") as file:
        json.dump(data, file, indent=4)
    print(f"Configuración guardada en {filename}")

# Función para leer los parámetros desde un archivo JSON
def cargar_configuracion(filename=CONFIG_FILE):
    if os.path.exists(filename): # Verifica si el archivo exist
        with open(filename, "r") as file:
            return json.load(file)
    else:
        print("Veo que es tu primera vez.")
        nombre = input("¿Cómo te llamas?")
        config_data = {}
        config_data["usuario"] = nombre
        config_data["dificultad"] = 10
        config_data["victorias"] = 0
        config_data["derrotas"] = 0
        return config_data # Retorna valores por defecto si el archivo no existe

config_data = cargar_configuracion()

print("Bienvenido", config_data["usuario"], "Qué deseas hacer")
print("1.Jugar")
print("2.Cambiar dificultad")
accion = int(input())
if accion == 2:
    dificultad = config_data["dificultad"]
    print("La dificultad actual es:", dificultad)
    dificultad = int(input("Elige nueva dificultad (1-5)"))
    print("Has cambiado a dificultad:", dificultad)
    config_data["dificultad"] = dificultad
    # Guardar la configuración inicial
    guardar_configuracion(config_data)
elif accion == 1:
    numero = random.randint(1, 10)
    print("🎲 Adivina un número del 1 al 10. Tienes 3 intentos.")
    for intento in range(1, 4):
        numero_elegido = int(input(f"Intento {intento}: "))
        if numero_elegido == numero:
```

```
        print(f"🎉 ¡Felicidades! Adivinaste el número {numero} en {intento} intentos.")
        config_data["victorias"] = config_data["victorias"] + 1
        break
    elif numero_elegido < numero:
        print("📉 El número es mayor.")
    else:
        print("📈 El número es menor.")

# Si es el último intento y no acertó, muestra el número correcto
if intento == 3:
    print(f"❌ Has perdido. El número era {numero}.")
    config_data["derrotas"] = config_data["derrotas"] + 1
guardar_configuracion(config_data)
```