APIs REQUESTS

¿Qué aprenderás en esta práctica?

- Que es un JSON y como recorrerlo.
- Como consultar APIs públicas y obtener datos específicos.
- Además, también buscarás nuevas Apis públicas, aprenderás a consultarlas y a publicar datos en tu propia API.

1. JSON

JSON (JavaScript Object Notation) es un formato ligero y legible para representar datos estructurados. Se usa mucho en APIs, almacenamiento y configuración. Su objetivo: ser fácil de leer/escribir por humanos y fácil de generar/parsear por máquinas. Podemos decir también que históricamente ha sido el sucesor de XML para muchas tareas.

Conceptos clave

- **Objeto**: colección de pares `clave: valor` encerrada entre llaves `{ ... }`.
  - Se representa con `{}`.
  - Las claves **siempre** son cadenas entre comillas dobles (`"clave"`).
  - Ejemplo mínimo: `{"nombre": "Ana", "edad": 30}` → **objeto**.
- **Array**: lista ordenada de valores encerrada entre corchetes `[ ... ]`.
  - Se representa con `[]`.
  - Ejemplo: `["manzana", "pera", "naranja"]` → **array**.

- **Formato**: sensible a comillas y comas:

  - Las claves y cadenas usan comillas dobles `"`.
  - No se permiten comentarios.

- **Codificación**: normalmente UTF-8. Contenido HTTP: `Content-Type: application/json; charset=utf-8`.

Ejemplo básico:

```json
{
  "alumno": "Luis",
  "curso": "DAW",
  "activo": true,
  "nota": 8.5
}
```

Ejemplo array:

```json
[
  { "id": 1, "nombre": "HTML" },
  { "id": 2, "nombre": "CSS"  },
  { "id": 3, "nombre": "JS"   }
]
```

Ejemplo completo:

```json
{
  "proyecto": "WebApp",
  "equipo": [
    { "nombre": "Ana", "rol": "Frontend" },
    { "nombre": "Luis", "rol": "Backend"  }
  ],
  "repos": {
    "frontend": "https://git.example/f",
    "backend": "https://git.example/b"
  }
}
```

Código entero:

```
URL = "https://rickandmortyapi.com/api/character"


    respuesta = requests.get(URL)
    datos = respuesta.json()
    for p in datos["results"]:
            print(p['name'])
```

Línea a línea:

```
respuesta = requests.get(URL)
```

Envía una petición HTTP GET a la URL usando la librería requests y guarda la respuesta en la variable "**respuesta**".

respuesta contiene: un objeto requests.Response. Dentro de ese objeto hay varias cosas útiles:

- respuesta.status_code (entero): código HTTP (ej. 200).
- respuesta.headers: las cabeceras HTTP.
- respuesta.text (cadena): el cuerpo de la respuesta en texto (normalmente JSON en formato texto).

Todavía no hemos convertido el JSON en estructuras Python; `respuesta` es el contenedor de la respuesta HTTP.

```
datos = respuesta.json()
```

Llamamos al método .json() del Response para parsear el texto JSON recibido y convertirlo en una estructura de Python.

Ahora datos contiene: un diccionario de Python (dict) — resultado del parseo del JSON.

```
{
  "info": { "count": 826, "pages": 42, ... },
  "results": [
    {"id": 1, "name": "Rick Sanchez", "status": "Alive", ...},
    {"id": 2, "name": "Morty Smith", "status": "Alive", ...},
    ...
  ]
}
```

Si os fijáis, “results” contiene una lista o una array de elementos. Con la siguiente línea recorreríamos uno a uno. Cada elemento de esa lista es otro diccionario ☺

```
for p in datos["results"]:
        print(p['name'])
```

Así, p contiene cada personaje de “results” que es un diccionario y buscaría si existe una clave llamada ‘name’ y sacaríamos por pantalla su valor.

2. Práctica.

Tenéis en el aula virtual una main.py donde solo está operativo la función de devolver todos los personajes de la primera página de la API de RickYMorty (Qué es justamente la que hemos comentado). Descargaos el main.py y deberéis de implementar el resto de las funcionalidades.

Tareas

1. Entra en la página de la API de RickYMorty y échale un vistazo al apartado de Docs en la zona de REST:
    a. https://rickandmortyapi.com/documentation

2. Llame en un navegador a los personajes de la serie: https://rickandmortyapi.com/api/character

3. Para que sea más legible y para que entiendas mejor de que está compuesto este JSON vete a la siguiente página web: https://jsonformatter.curiousconcept.com/

4. Copia y pega el JSON y dale a: Process.

5. Ahora podrás ver en mucho más visual como está compuesto este JSON. Deberías de ver algo así:

   Puedes abrir o cerrar los diccionarios y las listas. De esta manera te darás cuenta rápidamente de que results es una lista y cada elemento puedes ocultarlo o no.

6. Ahora que ya sabes con qué tipo de JSON vas a trabajar toca ponerse manos a la obra e implementar las funciones que faltan.

Tareas 2

Cuando acabes, busca otra API Rest pública que puedas consultar. Mira un poco su documentación y entiende cómo funciona.

Tareas

1. Levanta una FastAPI como hemos hecho en anteriores proyectos.
2. En la raíz de tu API “/” debería de poner la URL de tu API Rest pública elegida y como va a funcionar el siguiente punto.
3. Crea un endPoint “/petición” para que otras personas puedan interactuar con esta API. Te pongo el ejemplo con RickYMorty.

- Sabemos que la API de RickYMorty en esta URL: https://rickandmortyapi.com/api/character
- Si en cambio, lees algo más la DOC sabrás que https://rickandmortyapi.com/api/character/1 devolverá solo los datos del personaje con ID número 1, en este caso Rick.
- Quiero que en /petición por método POST se te envíe un dato clave para poder filtrar algún personaje, ubicación o algo de tu API rest. Especifica en "/" cómo funciona "/petición" y que opciones tiene el usuario.
- Con el ejemplo de RickYMorty el usuario podría enviarte un "1" y tu tendrías que devolverle el contenido de Rick llamando a: https://rickandmortyapi.com/api/character/1