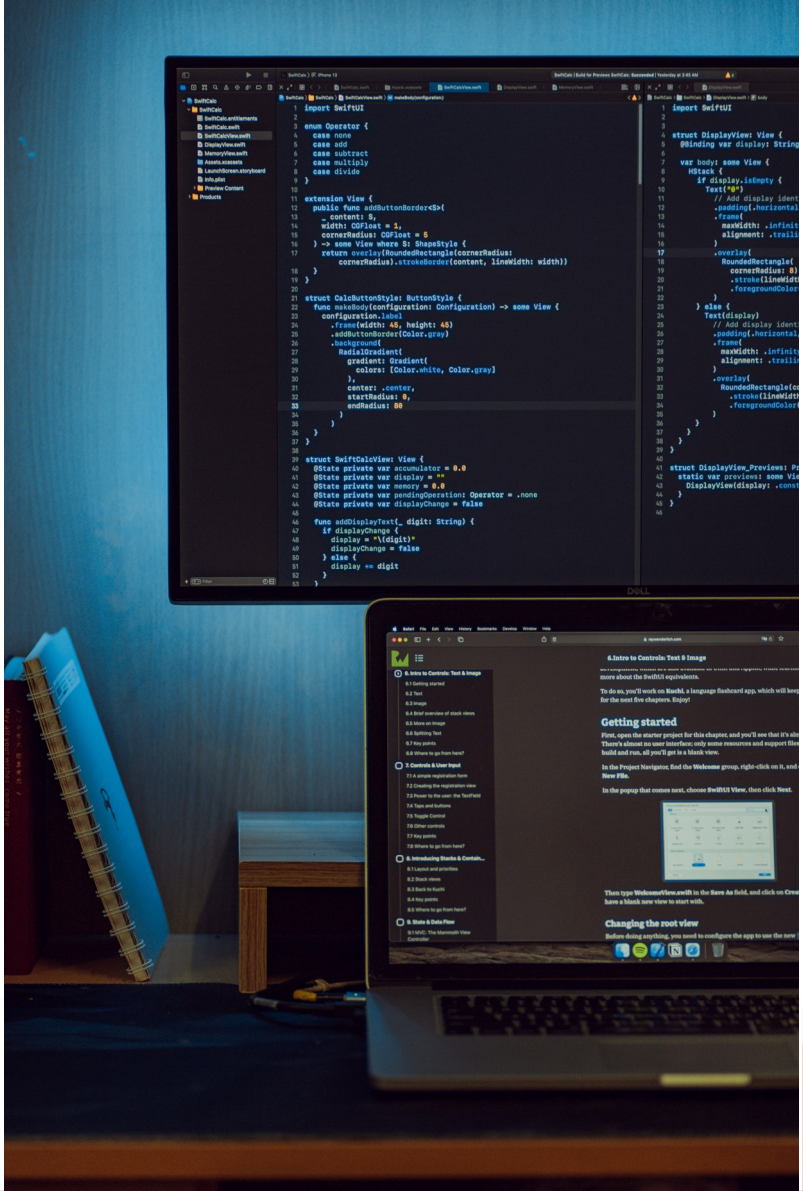
Curso Python

Escuela Politécnica Superior

JSON



UNIVERSIDAD NEBRIJA

Profesor:
José Ignacio Naranjo
jnaranjo@nebrija.es

¿Qué es JSON?

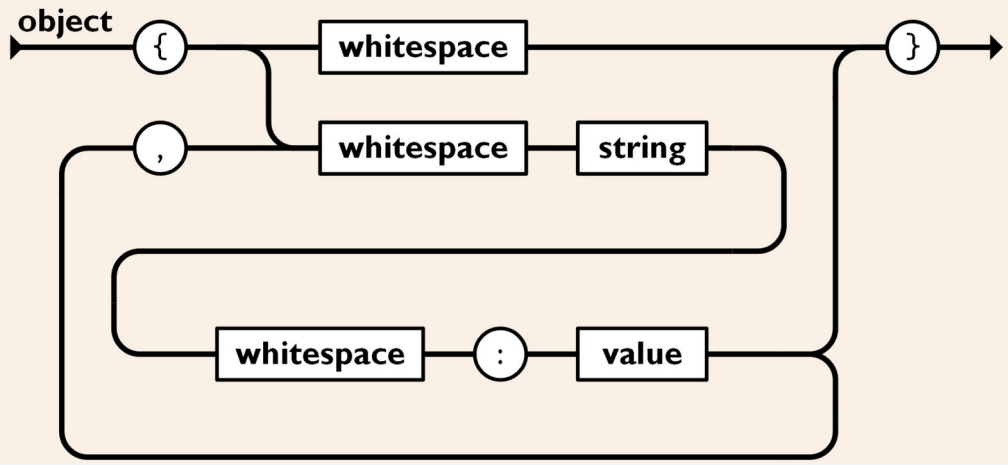
JSON → JavaScript Object Notation

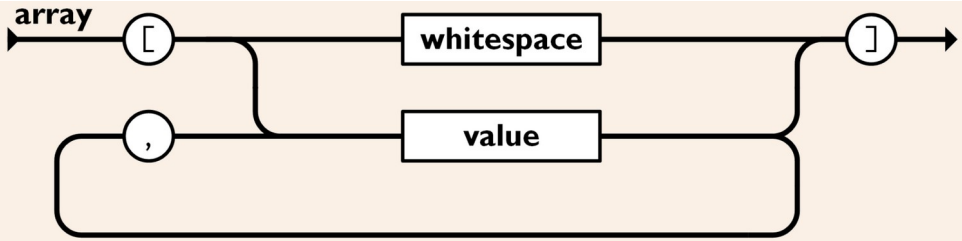
- Formato de intercambio de datos
- Fácil de entender para las personas
- Fácil de parsear y generar para las máquinas
- Usado en APIs de Internet para devolver datos

¿Cómo es un JSON?

Su sintaxis se basa en dos estructuras conocidas por los programadores:

1) Colecciones de clave/valor → Dict en Python
2) Listas de valores → List de Python





Ejemplos de servicios con JSON

Ergast → https://ergast.com/mrd/

Nasa → https://api.nasa.gov/

Ayuntamiento de Madrid → https://datos.madrid.es

ISBN → https://openlibrary.org/books/OL7650903M.json

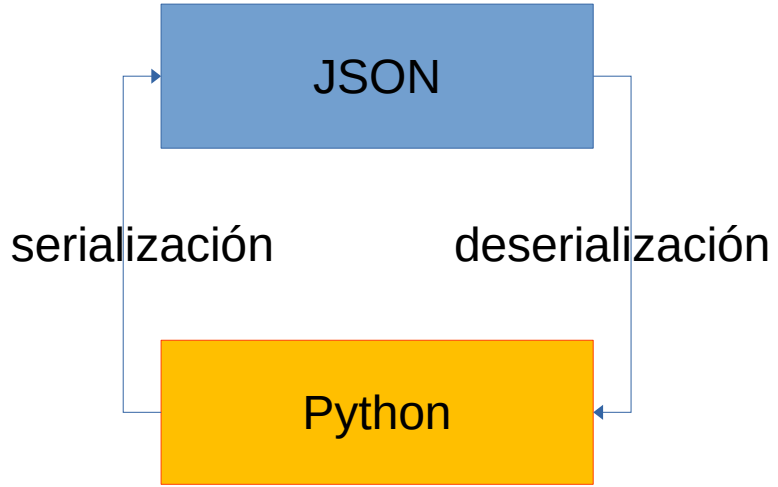
Ejemplo de JSON


```
{
    "season": "2023",
    "round": 1,
    "url": "https://en.wikipedia.org/wiki/2023_Bahrain_Grand_Prix",
    "raceName": "Bahrain Grand Prix",
    "date": "2023-03-05",
    "time": "15:00:00Z",
    "Qualifying": {
        "date": "2023-03-04",
        "time": "15:00:00Z"
    }
}
```


Más conceptos

- Serialización → Proceso en el cual se transforma de un objeto en memoria a un formato transmitible.

- Deserialización → Proceso en el cual se transforma de información transmitible a un objeto en memoria.



¿Cómo leer un JSON?

En el módulo json de Python disponemos de los métodos:

- dump() → Escribe la información en un objeto de tipo fichero y con formato JSON
- dumps() → Escribe la información en cadenas de texto con formato JSON

¿Cómo escribir un JSON?

En el módulo json de Python disponemos de los métodos:

- dump() → Escribe la información en un objeto de tipo fichero y con formato JSON
- dumps() → Escribe la información en cadenas de texto con formato JSON

Python object →

Python	JSON
dict	object
list, tuple	array
str	string
int, long, float	number
True	true
False	false
None	null

→ JSON

¿Cómo leer un JSON?

En el módulo json de Python disponemos de los métodos:

- load() → Lee la información de un objeto de tipo fichero y devuelve un objeto
- loads() → Lee la información desde una cadena de texto y devuelve un objeto

¿Y qué pasa con nuestras clases?

¿Si tenemos una lista de la clase Piloto?

Hasta ahora hemos visto como serializa y deserializa tipos básicos del lenguaje.

```python
class Piloto:
    def __init__(self, nombre, numero, edad):
        self.__nombre = nombre
        self.__numero = numero
        self.__edad = edad


if __name__ == "__main__":
    data = [
        Piloto("Fernando Alonso", "14", 42),
        Piloto("Carlos Sáinz", "55", 27)
    ]

    print(data)
    print(type(data))
    json_str = json.dumps(data)
    print(json_str)
    print(type(json_str))
```

```
Traceback (most recent call last):
  File "/home/josnar/PycharmProjects/Nebrija/curso_python_basico_json/clases_propias/serializacion_error.py", line 19, in
    json_str = json.dumps(data)
  File "/usr/lib/python3.10/json/__init__.py", line 231, in dumps
    return _default_encoder.encode(obj)
  File "/usr/lib/python3.10/json/encoder.py", line 199, in encode
    chunks = self.iterencode(o, _one_shot=True)
  File "/usr/lib/python3.10/json/encoder.py", line 257, in iterencode
    return _iterencode(o, 0)
  File "/usr/lib/python3.10/json/encoder.py", line 179, in default
    raise TypeError(f'Object of type {o.__class__.__name__} '
TypeError: Object of type Piloto is not JSON serializable
```

Podemos construir un dict con nuestro objeto

La serialización/deserialización de objetos en Python es un poco compleja.

Vamos a usar una aproximación muy sencilla: Vamos a hacer que nuestro objeto sepa transformarse en un diccionario. Y también haremos que sea capaz de construirse desde un objeto.

```python
class Piloto:
    def __init__(self, nombre, numero, edad):
        self.__nombre = nombre
        self.__numero = numero
        self.__edad = edad

    def to_serializable(self):
        return {
            "nombre": self.__nombre,
            "numero": self.__numero,
            "edad": self.__edad
        }

    @classmethod
    def from_json(cls, data):
        return Piloto(data["nombre"], data["numero"], data["edad"])
```