

Python Cheatsheet – JSON

Tema: Manejo de JSON en Python **Curso:** Google IT Automation with Python **Objetivo:** Leer, escribir, convertir y manipular datos JSON de forma eficiente.

Introducción

JSON (JavaScript Object Notation) es el formato estándar para intercambiar datos en APIs, archivos de configuración y automatización.

En Python se maneja con el módulo estándar:

```
import json
```

Convertir JSON a Python (Decodificación)

◆ `json.loads()` — JSON **string** → **objeto Python**

```
import json

json_str = '{"nombre": "Axel", "edad": 30}'
data = json.loads(json_str)

print(data["nombre"]) # Axel
```

Resultado de conversiones comunes

JSON	Python
object {}	dict
array []	list
string	str
number	int / float
true / false	True / False
null	None

Convertir Python a JSON (Codificación)

◆ `json.dumps()` — objeto Python → **string JSON**

```
persona = {"nombre": "Axel", "activo": True}

json_str = json.dumps(persona)
print(json_str)
# {"nombre": "Axel", "activo": true}
```

Leer y Escribir Archivos JSON

Leer archivo `.json`

```
with open("datos.json", "r") as f:
    data = json.load(f)

print(data)
```

Escribir archivo `.json`

```
with open("salida.json", "w") as f:
    json.dump(data, f, indent=4)
```

◆ `indent=4` → guarda bonito / legible.

Formato bonito (pretty print)

```
print(json.dumps(data, indent=4, sort_keys=True))
```

- `indent=4` → sangría
- `sort_keys=True` → ordena claves alfabéticamente

Acceso a datos en JSON

Dado:

```
data = {  
    "user": {  
        "name": "Axel",  
        "skills": ["Python", "SQL", "ML"]  
    }  
}
```

Acceder a valores

```
data["user"]["name"]  
data["user"]["skills"][0] # Python
```

Modificar JSON (Python dict)

```
data["user"]["name"] = "Axel L."  
data["user"]["skills"].append("Docker")
```

Validar JSON (try/except)

```
try:  
    data = json.loads(json_str)  
except json.JSONDecodeError:  
    print("❌ JSON inválido")
```

Convertir JSON desde API con requests

```
import requests  
  
r = requests.get("https://jsonplaceholder.typicode.com/posts/1")  
data = r.json()  
  
print(data["title"])
```



Manejar JSON con tipos especiales

`json` solo acepta tipos básicos. Para guardar cosas como fecha, usar conversión manual:

```
from datetime import datetime
data = {"fecha": datetime.now().isoformat()}
json.dumps(data)
```



Convertir keys numéricas a strings

En JSON, las claves SIEMPRE deben ser strings.

Python → JSON lo arregla solo:

```
data = {1: "uno", 2: "dos"}
json.dumps(data)
# {"1": "uno", "2": "dos"}
```



Errores comunes

Error	Causa	Solución
<code>JSONDecodeError</code>	JSON malformado	Revisar comillas, llaves
<code>TypeError</code>	Objeto no serializable	Convertir manualmente (fechas, sets)
Archivo vacío	<code>json.load()</code> falla	Validar tamaño antes de leer



Ejemplo completo realista (guardar y cargar configuración)

```
import json

config = {
    "usuario": "Axel",
    "preferencias": {
        "tema": "dark",
        "autosave": True
    }
}
```

```
# Guardar
with open("config.json", "w") as f:
    json.dump(config, f, indent=4)

# Leer
with open("config.json", "r") as f:
    data = json.load(f)

print("Tema:", data["preferencias"]["tema"])
```