PRO-UT3-A7. Uso de JSON

Introducción

El formato JSON

JavaScript Object Notation (JSON) es un formato basado en **texto plano** para representar datos estructurados en la sintaxis de objetos de JavaScript.

Es comúnmente utilizado para transmitir datos en aplicaciones web y móviles(por ejemplo: enviar datos desde el servidor al cliente, o viceversa).

La mayoría de lenguajes de programación disponen de librerías que permiten trabajar con dicho formato. Esto, unido a la facilidad de su interpretación por parte de los programadores lo ha convertido en el formato más extendido a la hora de intercambiar información entre equipos que ejecutan distintos lenguajes de programación.

Otra de las ventajas de JSON es que al permitirnos convertir estructuras de datos complejas a texto podemos almacenarlas usando este formato directamente en un archivo o en una base de datos.

Convertir diccionario a JSON

Para convertir un objeto de tipo diccionario de Python al formato de texto **JSON** lo podemos hacer de la siguiente manera:

```
# import json module
import json

# list of dictionaries of employee data
dict = {
 'Mathematics' : 145,
 'Done': True,
 'Lecturer': 'Allan Smith'
}


# convert into json
dict_json = json.dumps(dict, indent = 2)

# display
print(dict_json)
```

El resultado sería:

```
{
  "Mathematics": 145,
  "Done": true,
  "Lecturer": "Allan Smith"
}
```

El parámetro `indent = 2` del método `dump()` le indica que el texto generado debe estar indentado usando 2 espacios para la indentación.

Si no le pasaramos dicho parámetro a `dump()` el resultado sería:

```
{"Mathematics": 145, "Done": true, "Lecturer": "Allan Smith"}
```

Fíjate que en una sola línea de texto tenemos toda la información del diccionario.

Convertir listas a JSON

También podemos convertir listas a JSON:

```
# import json module
import json

names = ['Yae', 'Samuel', 'Liz', 'Ammon']

# convert into json
names_json = json.dumps(names)

# display
print(names_json)
```

El resultado sería:

```
["Yae", "Samuel", "Liz", "Ammon"]
```

Mezclando estructuras de datos complejas

Tambíen podemos convertir a JSON estructuras de datos más complejas, como por ejemplo un diccionario que contenga a su vez una lista:

```
import json

person = {
 'Name' : "Allan Smith",
 'Man': True,
 "age" : 45,
 'siblings': ["Yae","Samuel","Liz","Ammon"],
 "Pet" : None
}

person_json = json.dumps(person, indent = 2)

print(person_json)
print(type(person_json))
```

El resultado sería

```
{
  "Name": "Allan Smith",
  "Man": true,
  "age": 45,
  "siblings": [
    "Yae",
    "Samuel",
    "Liz",
    "Ammon"
  ],
  "Pet": null
}
<class 'str'>
```

El diccionario se ha convertido en una cadena de texto.

Decodificando JSON

La utilidad de **JSON** no está en poder convertir objetos que almacenan estructuras de datos complejas a dicho formato, sino en que también podemos hacer la operación inversa; convertir texto en formato JSON a las estructuras de datos originales.

Para ello podemos usar el método `loads()`

```
import json

person_json = '''
{
  "Name": "Allan Smith",
  "Man": true,
  "age": 45,
  "siblings": [
    "Yae",
    "Samuel",
    "Liz",
    "Ammon"
  ],
  "Pet": null
}
'''

# convert into json


person = json.loads(person_json)
# display
print(person)
print(type(person))
print(person_json['siblings'][1])
```

El resultado sería:

```
{'Name': 'Allan Smith', 'Man': True, 'age': 45, 'siblings': ['Yae', 'Samuel',
'Liz', 'Ammon'], 'Pet': None}
<class 'dict'>
Samuel
```

Recursos

- Kiprono Elijah Koech: guía de uso de JSON y diccionarios en Python