Les Fichiers Semi-Structurés

Sami Souihi
v1_2020

Définition

Les données semi-structurées sont des données qui ne sont pas conformes aux normes des données structurées classiques, mais qui contiennent des balises ou d’autres types de balisage qui identifient des entités individuelles et distinctes dans les données.

Objectifs

DÉCOUVERTE DE JSON

MANIPULER DU JSON EN PYTHON

INTÉGRATION JSON & FLASK

Exemple de JSON

- "JSON" signifie "JavaScript Object Notation"
- Malgré son nom, JSON est indépendante du langage JS.
- Il permet de décrire des objets en tant que paires clé-valeur

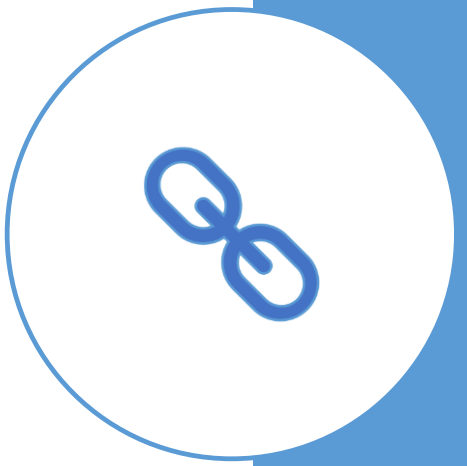

```
{
  "type":"Feature",
  "id":"63d55408-39a1-4284-b413-9afb1aa86b52",
  "geometry":{
    "type":"Point",
    "coordinates":[
      24.93218,
      60.19897
    ]
  }
}
```

Syntaxe JSON

- Un objet est un ensemble non ordonné de paires nom/valeur :
  - Les paires sont encadrées par des accolades { }
  - Le nom et la valeur sont séparés par deux points
  - Les paires sont séparées par des virgules
  - Exemple : { "nom" : "John Doe", " age" : 5 }
- Un tableau est une collection ordonnée de valeurs
  - Les valeurs sont placées entre parenthèses, [ ]
  - Les valeurs sont séparées par des virgules
  - Exemple : ["html", "xml", "css"]

Syntaxe JSON

- Une valeur peut être : Une chaîne de caractères, un nombre, un booléen, nul, un objet ou un tableau
  - Les valeurs peuvent être imbriquées
- Les chaînes de caractères sont entre guillemets et peuvent contenir l'assortiment habituel de caractères échappés
- Les nombres ont la syntaxe habituelle C/C++/Java, y compris la notation exponentielle (E)
  - Tous les nombres sont décimaux



EVAL

La méthode eval(string) compile et exécute la chaîne donnée

- La chaîne peut être une expression, une déclaration ou une séquence de déclarations
- Les expressions peuvent inclure des variables et des propriétés d'objet
- eval renvoie la valeur de la dernière expression évaluée - Lorsqu'il est appliqué à JSON, eval renvoie l'objet décrit

Comparaison entre JSON et XML

Similarités :

- Les deux sont lisibles par l'homme
- Les deux ont une syntaxe très simple
- Les deux sont hiérarchiques
- Les deux sont indépendants du langage

Différences :

- La syntaxe est différente
- JSON est moins verbeux
- JSON inclut des tableaux
- Les noms dans JSON ne doivent pas être des mots réservés par JavaScript

YAML

- YAML peut être un acronyme pour :
  - Yet Another Markup Language
  - YAML Ain't Markup Language
- Comme JSON, le but de YAML est de représenter des types de données typiques en notation lisible par l'homme
- YAML est techniquement un sur-ensemble de JSON, avec beaucoup plus de capacités (listes, casting, etc.)
- **YAML - Lorsque JSON ne suffit pas, considérez YAML**

Encodage et décodage JSON

- Le module « json » fournit une API familière aux utilisateurs des modules marshal et pickle de la bibliothèque standard.
- Les encodeurs et décodeurs de ce module conservent l'ordre d'entrée et de sortie par défaut. L'ordre n'est perdu que si les conteneurs sous-jacents ne sont pas ordonnés.
  - Avant Python 3.7, dict n'était pas garanti d'être ordonné, donc les entrées et sorties étaient généralement mélangées à moins d'utiliser explicitement un collections.OrderedDict.
  - À partir de Python 3.7, un dict conserve son ordre, il n'est donc plus nécessaire d'utiliser un collections.OrderedDict pour générer et analyser du JSON.

Sérialisation de JSON

- Que se passe-t-il lorsqu'un ordinateur traite un grand nombre d'informations ?

- La bibliothèque json expose la méthode dump() pour écrire des données dans des fichiers. Il existe également une méthode dumps() (prononcée "dump-s") pour écrire dans une chaîne de caractères Python.

- Les objets Python simples sont traduits en JSON selon une conversion assez intuitive.

Conversions

- source:

https://docs.python.org/3/library/json.html#encoders-and-decoders

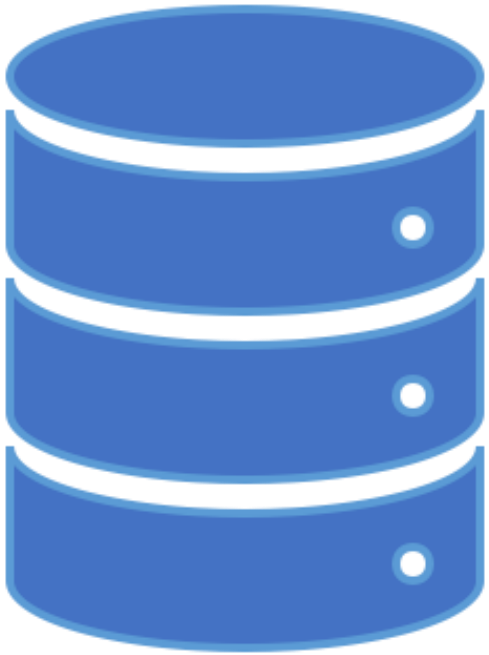
JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

JSON Read

```
import json


f =
open('data.json')

data = json.load(f)

f.close()

print(data)

print(data["feature
s"])

print(data["feature
s"][0]["geometry"])


for i in
data["features"]:


print(i["geometry"]
["coordinates"][0])
```

```
{
 "type": "FeatureCollection",
 "features": [ {
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [42.0, 21.0]
   },
  "properties": {
    "prop0": "value0"
  }
 }]
}
```

{'features':
[
{'type': 'Feature', 'geometry':
{'coordinates': [42.0, 21.0], 'type': 'Point'},
'properties': {'prop0': 'value0'}
}
],
'type': 'FeatureCollection'}



JSON write

```
import json

f = open('data.json')
data = json.load(f)
f.close()

f = open('out.json', 'w')
json.dump(data, f)
f.close()
```

Formatted printing

```
print(json.dumps(data["features"], sort_keys=True, indent=4))
[
    {
        "geometry": {
            "coordinates": [
                42.0,
                21.0
            ],
            "type": "Point"
        },
        "properties": {
            "prop0": "value0"
        },
        "type": "Feature"
    }
]
```

Dumps offre une option de formatage d'impression agréable : https://docs.python.org/3/library/json.html

Outil de controle de JSON

```
python -m json.tool < data.json
```

JSON et FLASK

- Flask possède un utilitaire appelé jsonify() qui facilite le retour des réponses JSON

- from flask import Flask
- from flask import jsonify
- app = Flask(__name__)
- @app.route('/api/get-json’)
- def hello():

  return jsonify(hello='world')

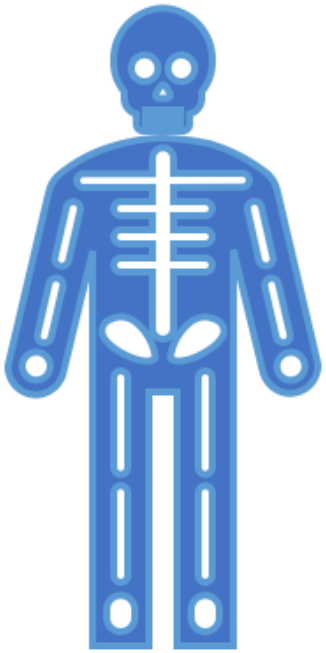
Exercice

Proposer une fonction permettant de chercher dans un dictionnaire les paramètres de santé d’une personne et de les renvoyer sous forme d’objet JSON

Exercice

Développer une app permettant de

1) Afficher les paramètres de santé d'une personne
   - Input de l'API : id personne
2) Modifier un paramètre de santé d'une personne
   - Input de l'API : id personne + champ à modifier + nouvelle valeur
3) Ajouter une nouvelle personne
   - Input de l'API : id personne + champs et valeurs à enregistrer
4) Supprimer une personne
   - Input de l'API : id personne



Les requêtes se font sous forme d'API avec flask et Python.

Les données santé sont gérées par un fichier json.

L'interaction avec l'app se fait via des pages html