2N6R Prog 2

python™

# Format JSON

# Dictionnaires

retour

```
auto = {"marque" : "Ford","modele" : "Mustang",  "annee":1964 }

annee_fab = auto["annee"]
print (annee_fab)
# 1964

auto["annee"] = 2002
print (annee_fab)
# 1964

marque = auto.pop("marque")
print(marque)
# "Ford"

print(auto.get(marque))
# None
print (auto["marque"])
# == > ERREUR
```

```
infos_sup = {"couleur" : "rouge",
             "kilometrage": 35674 }

auto.update(infos_sup)

print(auto)
# {'marque': 'Ford', 'modele': 'Mustang',
'annee': 1964, 'couleur': 'rouge',
'kilometrage': 35674}
```

# Liste dans un dictionnaire

retour

- Les dictionnaires et les listes sont souvent utilisés ensemble pour permettre de stocker de nombreuses données de façon flexible.
- Ici, un dictionnaire représentant une auto et contenant une liste d'accessoires.

```
auto = {
    "marque": "Reliant",
    "modele": "Robin",
    "annee": 1988,
    "accessoires": [
        "Marchepied chromé",
        "Moteur V8",
        "Dés en minou sur le rétroviseur"
    ]
}

print(f"Il y a {len(auto['accessoires'])} accessoires:")

for item in auto['accessoires']:
    print("- " + item)

# Il y a 3 accessoires:
# - Marche-pied chromée
# - Moteur V8
# - Dés en minou sur le rétroviseur
```

Liste

# Dictionnaires dans une liste

retour

- Ici, une liste de voitures. Chaque voiture est représentée par un dictionnaire.
- Il n'y a pas de limites aux «  niveaux de profondeur » des dictionnaires et listes.

```
autos = [
    {"marque": "Ford", "modele": "Mustang", "annee": 1964},
    {"marque": "Reliant", "modele": "Robin", "annee": 1988},
    {"marque": "Toyota", "modele": "Tercel", "annee": 1991}
]


print(f"Il y a {len(autos)} autos:")


for auto in autos:
    print(f"- {auto['marque']} {auto['modele']} {auto['annee']}")


# Il y a 3 autos:
# - Ford Mustang 1964
# - Reliant Robin 1988
# - Toyota Tercel 1991
```

# JSON

Un standard pour le transfert de données

# Formats de sérialisation

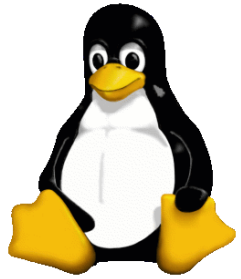
## JSON

```json
{
    "first_name": "John",
    "last_name": "Smith",
    "age": 25,
    "address": {
        "street_address": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postal_code": "10021"
    },
    "phone_numbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ],
    "sex": {
        "type": "mâle"
    }
}
```

## XML

```xml
<person>
 <firstName>John</firstName>
 <lastName>Smith</lastName>
 <age>25</age>
 <address>
   <streetAddress>21 2nd Street</streetAddress>
   <city>New York</city>
   <state>NY</state>
   <postalCode>10021</postalCode>
 </address>
 <phoneNumbers>
   <phoneNumber>
     <type>home</type>
     <number>212 555-1234</number>
   </phoneNumber>
   <phoneNumber>
     <type>fax</type>
     <number>646 555-4567</number>
   </phoneNumber>
 </phoneNumbers>
 <sex>
   <type>male</type>
 </sex>
</person>
```

## YAML



```yaml
first_name: John
last_name: Smith
age: 25
address:
  street_address: 21 2nd Street
  city: New York
  state: NY
  postal code: "10021"
phone_numbers:
  - type: home
    number: 212 555-1234
  - type: fax
    number: 646 555-4567
sex:
  type: mâle
```

Souvent utilisé dans les fichiers de configuration Linux

# JSON

- JSON (JavaScript Object Notation) est un format standard de sérialisation d'objets sous forme de données textuelles.
- Format **standard** permettant le transfert d'objets indépendant du langage.
  - Standard le plus répandu pour les échanges de données entre les applications.
  - Ensemble de règles pour format de chaines de caractères.
    - Comme les csv, un fichier JSON est **juste un fichier texte** qui respecte certaines règles.

# JSON

( Chaîne de caractères )

```
[
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "Moteur V8",
            "Dés en minou"
        ]
    },
    {
        "marque": "Toyota",
        "modele": "Tercel",
        "annee": 1991,
        "accessoires": []
    }
]
```

- Ici, on a 3 objets JSON correspondants à des voitures.
- Ils sont énumérés de façon séquentielle, séparés par des virgules (",") et contenus dans des crochets. [ ]
- Une fois convertis en objet dans Python, il s'agira d'une liste de dictionnaires ayant tous les mêmes clefs.

# JSON

```
[
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "Moteur V8",
            "Dés en minou"
        ]
    },
    {
        "marque": "Toyota",
        "modele": "Tercel",
        "annee": 1991,
        "accessoires": []
    }
]
```

Les **listes** sont entre **crochets [ ]**

- Une liste peut contenir des valeurs brutes, des dictionnaires ou d'autres tableaux.

Les **dictionnaires** sont entre **accolades { }**

- Un dictionnaire est composé d'un ou plusieurs champs composés d'une clé et d'une valeur.
- La valeur d'un champ peut être une chaîne de caractères, un nombre, un tableau ou un dictionnaire.

Chaque élément est séparé des autres par des virgules

# Convertir du contenu JSON en objet

- La méthode **loads()** du module **json** convertit du texte formaté JSON en objet natif Python
- Cet objet peut représenter une hiérarchie d'objets listes et dictionnaires

**Reponse_req.json**

```
1  [{"marque":"Ford","modele":"Must
2  ang","annee":1964,"accessoires":
3  []},{"marque":"Reliant","modele"
4  :"Robin","annee":1988,"accessoir
5  es":["MoteurV8","Désenminou"]},{
6  "marque":"Toyota","modele":"Terc
7  el","annee":1991,"accessoir
   es":[]},"marque":"Relia…
```

**json.loads("[un string]")**

```
[
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "Moteur V8",
            "Dés en minou« ]
]
```

# Convertir du contenu JSON en objet

- La méthode **loads()** du module **json** convertit du texte formaté JSON en objet natif Python
- Cet objet peut représenter une hiérarchie d'objets listes et dictionnaires

**Reponse_req.json**

```
1  [{"marque":"Ford","modele":"Must
2  ang","annee":1964,"accessoires":
3  []},{"marque":"Reliant","modele"
4  :"Robin","annee":1988,"accessoir
5  es":["MoteurV8","Désenminou"]},{
6  "marque":"Toyota","modele":"Terc
7  el","annee":1991,"accessoir
   es":[]},"marque":"Relia…
```

**json.loads("[un string]")**

```
[
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "Moteur V8",
            "Dés en minou« ]
]
```

# Convertir du contenu JSON en objet

```
import json

with open('Reponse_req.json', 'r') as file:
    text = file.read()

type(text)      # <class 'str'>
# C'est du texte brut

autos = json.loads(text)
# Convertit le contenu JSON en objet

type(autos)     # <class 'list'>
type(autos[0])  # <class 'dict'>
# C'est une liste de dictionnaires

print(autos[0]['marque'])
# Ford
```

- Dans cet exemple on lit un fichier texte appelé « Reponse_req.json » et on met le contenu (un string) dans la variable « text ».
- On utilise ensuite la fonction json.loads() pour convertir le string en un objet utilisable dans Python. Ici, une liste contenant des dictionnaires.

# Convertir du contenu JSON en objet

- La méthode **dumps()** du module **json** prend un objet dans Python et le transforme en une chaine de caractères suivant le format JSON
- On peut ensuite enregistrer cette chaine de caractères ou l'envoyer dans une requête HTTP.

```
[
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "Moteur V8",
            "Dés en minou« ]
]
```


```
[{"marque":"Ford","modele
":"Mustang","annee":1964,
"accessoires":[]},{"marqu
e":"Reliant","modele":"Ro
bin","annee":1988,"access
oires":["MoteurV8","Désen
minou"]},{"marque":"Toyot
a","modele":"Tercel","ann
ee":1991,"accessoires":[]
},"marque":"Relia…
```

# Convertir du contenu JSON en objet

## Un objet python (une liste de dictionnaires

```
autos= [
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "Moteur V8",
            "Dés en minou« ]
    ]
```


## Un string, une chaine de caractères.

```
[{"marque":"Ford","modele
":"Mustang","annee":1964,
"accessoires":[]},{"marqu
e":"Reliant","modele":"Ro
bin","annee":1988,"access
oires":["MoteurV8","Désen
minou"]},{"marque":"Toyot
a","modele":"Tercel","ann
ee":1991,"accessoires":[]
},"marque":"Relia…
```

# Convertir un objet en contenu JSON

- La méthode **dumps()** peut aussi prendre une valeur pour son paramètre « indent » afin de rendre la chaine de caractères facilement lisible par l'être humain.

```
autos= [
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "Moteur V8",
            "Dés en minou« ]
    ]
```

print(json.**dumps**(autos, indent=4)

```
>>> print(json.dumps(autos,indent=4))
[
    {
        "marque": "Ford",
        "modele": "Mustang",
        "annee": 1964,
        "accessoires": []
    },
    {
        "marque": "Reliant",
        "modele": "Robin",
        "annee": 1988,
        "accessoires": [
            "MoteurV8",
            "D\u00e9senminou"
        ]
    }
]
>>>
```