



avec pythonTM

Les fichiers JSON et les requêtes http



JSON

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": []
  }
]
```

- > Ici, on a 3 objets JSON correspondant 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.



```
[
  {
    "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 listes.

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

- > Un dictionnaire est composé d'un ou plusieurs champs composés d'une clé et une valeur.
- > La valeur d'un champ peut être une chaîne de caractères, un nombre, une liste 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": "Mustang", "annee": 1964, "accessoires": []}, {"marque": "Reliant", "modele": "Robin", "annee": 1988, "accessoires": ["Moteur V8", "Désenminou"]}, {"marque": "Toyota", "modele": "Tercel", "annee": 1991, "accessoires": []}, {"marque": "Relia...
```



json.loads()

```
[
  {
    "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 chaîne de caractères suivant le format JSON
- > On peut ensuite enregistrer cette chaîne 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"  
    ]  
  }  
]
```



json.dumps()

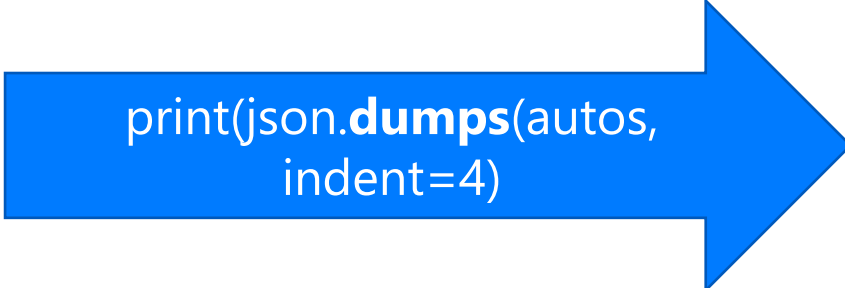
```
[{"marque": "Ford", "modele": "Mustang", "annee": 1964,  
  "accessoires": []}, {"marque": "Reliant", "modele": "Robin", "annee": 1988,  
  "accessoires": ["MoteurV8", "Dés en minou"]}, {"marque": "Toyota", "modele": "Tercel", "annee": 1991,  
  "accessoires": []}, {"marque": "Relia...
```



Convertir un objet en contenu JSON

- > La méthode **dumps()** peut aussi prendre une valeur pour son paramètre « indent » afin de rendre la chaîne 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"  
        ]  
    }  
]
```



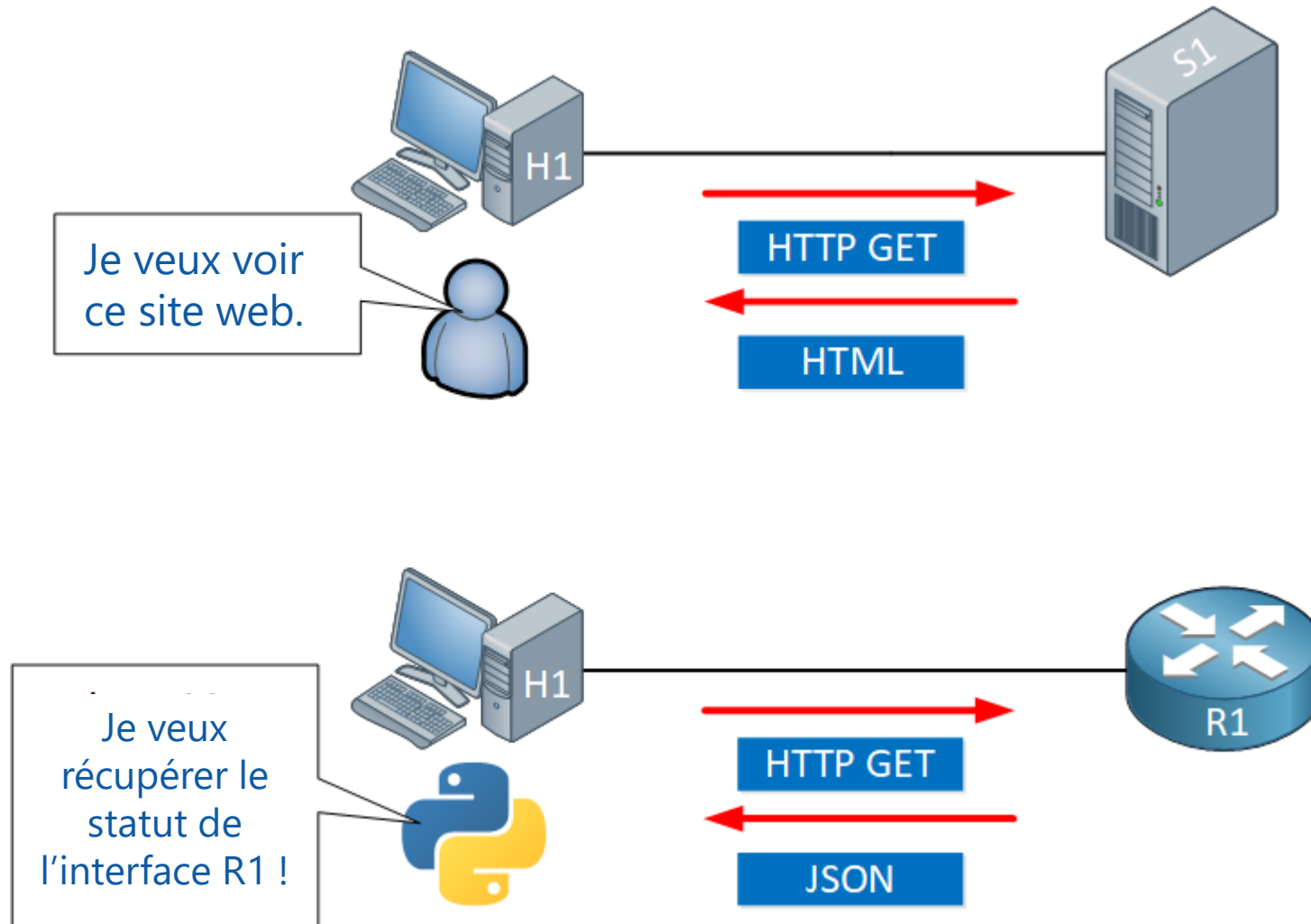
```
print(json.dumps(autos,  
    indent=4))
```

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



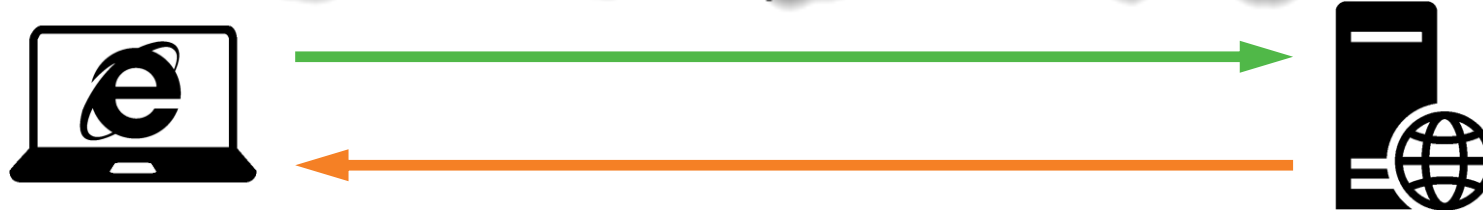

API Web

Transmission de données



Source: NetworkLessons.com

```
GET https://www.cegepmontpetit.ca/ HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Accept-Language: en-US,en;q=0.5
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: www.cegepmontpetit.ca
Connection: Keep-Alive
```



```

HTTP/1.1 200 OK
Date: Thu, 10 Oct 2019 04:25:29 GMT
Server: Apache-Coyote/1.1
Content-Type: text/html; charset=UTF-8
Set-Cookie: JSESSIONID=66BB8CA8BDAF7102EB1CF89B569BDF57; Path=/; HttpOnly
Vary: Accept-Encoding
Connection: close
Content-Length: 59803

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html
xmlns="http://www.w3.org/1999/xhtml" xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"
lang="fr"><head><title>Bienvenue ! | Cégep Édouard-Montpetit</title><meta name="description"
content="Cégep Édouard Montpetit"></meta><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"></head><body><div id="Aut"><div id="Call"><div id="Navigation">

```



Toutes les Méthodes HTTP

| Méthode | Description |
|---------|---|
| GET | Demande une ressource (un fichier, un objet, etc.) |
| POST | Envoie des données à une ressource (la requête a un <i>payload</i>) |
| PUT | Crée ou remplace une ressource sur le serveur |
| HEAD | Ne demande que les informations sur la ressource, pas son <i>payload</i> |
| DELETE | Supprime une ressource du serveur |
| OPTIONS | Obtient les options d'une ressource ou du serveur |
| TRACE | Demande au serveur de retourner ce qu'il a reçu, à des fins de diagnostic |
| CONNECT | Permet d'ouvrir un tunnel de communication (par exemple, SSL) |
| PATCH | Modifie une ressource, comme PUT, mais partiellement |

Plus de détails: <https://www.iana.org/assignments/http-methods/http-methods.xhtml>

La méthode GET



| Méthode | Description |
|---------|--|
| GET | Demande une ressource (un fichier, un objet, etc.) |

- > Une requête http est un message qu'on envoie à une adresse url avec un format standard indiquant les données avec lesquelles on veut interagir et la façon dont on veut traiter ces données.

```
response = requests.get('https://fakestoreapi.com/products?limit=5&sort=desc')
```

Adresse url où on
envoie notre requête

Paramètres de notre
requête



Utilisation de la méthode GET

> Exemples de requêtes effectuées à fakestoreapi.com

```
_req.py > ...  
import requests as rq  
import json  
  
res = rq.get('https://fakestoreapi.com/products')  
  
res = rq.get('https://fakestoreapi.com/products/9')  
  
res = rq.get('https://fakestoreapi.com/products?limit=5')  
  
res = rq.get('https://fakestoreapi.com/products?limit=5&sort=desc')
```

Tous les produits

Le produit avec l'ID 9

Les 5 premier produits

Les 5 premier produits en ordre décroissant d'ID

> Toujours consulter la documentation pour formuler une requête

↳ <https://fakestoreapi.com/docs>

Exemple détaillé en Python



Python

```
import json, requests

url = "http://date.jsontest.com/"
response = requests.get(url)
data = json.loads(response.text)

print(f"Il est { data['time'] } sur le serveur !")
```

Il faut installer le module **requests** avec:

```
pip install requests
```

Il faut s'assurer d'avoir pip déjà installé :

```
python get-pip.py
```

Il est 02-20-2023 sur le serveur

La réponse...



- La commande `requests.get()` nous retourne un objet de la classe `<Response>`.
- Cet objet possède ses propres attributs et méthodes.
- On s'intéresse particulièrement aux attributs « `status_code` », « `ok` », « `text` », « `headers` » et la méthode « `json()` ».

Réponse



```
>>> response = requests.get("http://date.jsontest.com/")

>>> print(response.status_code)
200

>>> print(response.ok)
True

>>> print(response.text)
{
  "date": "12-07-2020",
  "milliseconds_since_epoch": 1607327636741,
  "time": "07:53:56 AM"
}

>>> print(response.json())
{"date": "12-07-2020", "milliseconds_since_epoch": 1607327636741, "time": "07:53:56 AM"}

>>> print(response.headers)
{'Access-Control-Allow-Origin': '*', 'Content-Type': 'application/json', 'X-Cloud-Trace-Context':
'd9a3a2f216dc37228d6ae1f376eece11', 'Date': 'Mon, 07 Dec 2020 07:53:56 GMT', 'Server': 'Google Frontend',
'Content-Length': '100'}
```

Exemple: données météo



```
import datetime, json, requests

uri = "https://www.metaweather.com/api/location"

woeid = json.loads(
    requests.get(f"{uri}/search/?query=montr%C3%A9al").text)[0]['woeid']
today = datetime.datetime.now().strftime('%Y/%m/%d')
data = json.loads(
    requests.get(f"{uri}/{woeid}/{today}").text
)

for item in data:
    print(f"{item['created']} : {item['the_temp']}°C")
```

Fakestore → Une ressource pour exemples et exercices



Fake Store API

Fake store rest API for your e-commerce or shopping website prototype

View on GitHub



Read Docs



Fakestore

- > <https://fakestoreapi.com/>
- > Fake store api est un site web contenant un api avec lequel on peut interagir comme si nous avions toutes les autorisations.
 - > Pas limité au GET seulement
 - > On peut faire des POST, PUT, PATCH et DELETE

Routes

All HTTP methods are supported

| | |
|--------|-----------------------------|
| GET | /products |
| GET | /products/1 |
| GET | /products/categories |
| GET | /products/category/jewelery |
| GET | /cart?userId=1 |
| GET | /products?limit=5 |
| POST | /products |
| PUT | /products/1 |
| PATCH | /products/1 |
| DELETE | /products/1 |

[View Details on Docs](#)



Pour en savoir plus

- > [API Integration in Python – Part 1 – Real Python](#)
- > <https://apislist.com/>
- > <http://www.jsontest.com/>
- > [Outil: Fiddler](#)
- > [Liste d'API publics pour s'amuser](#)
- > [Authentication using Python requests - GeeksforGeeks](#)