2N6 Programmation 2

avec python™

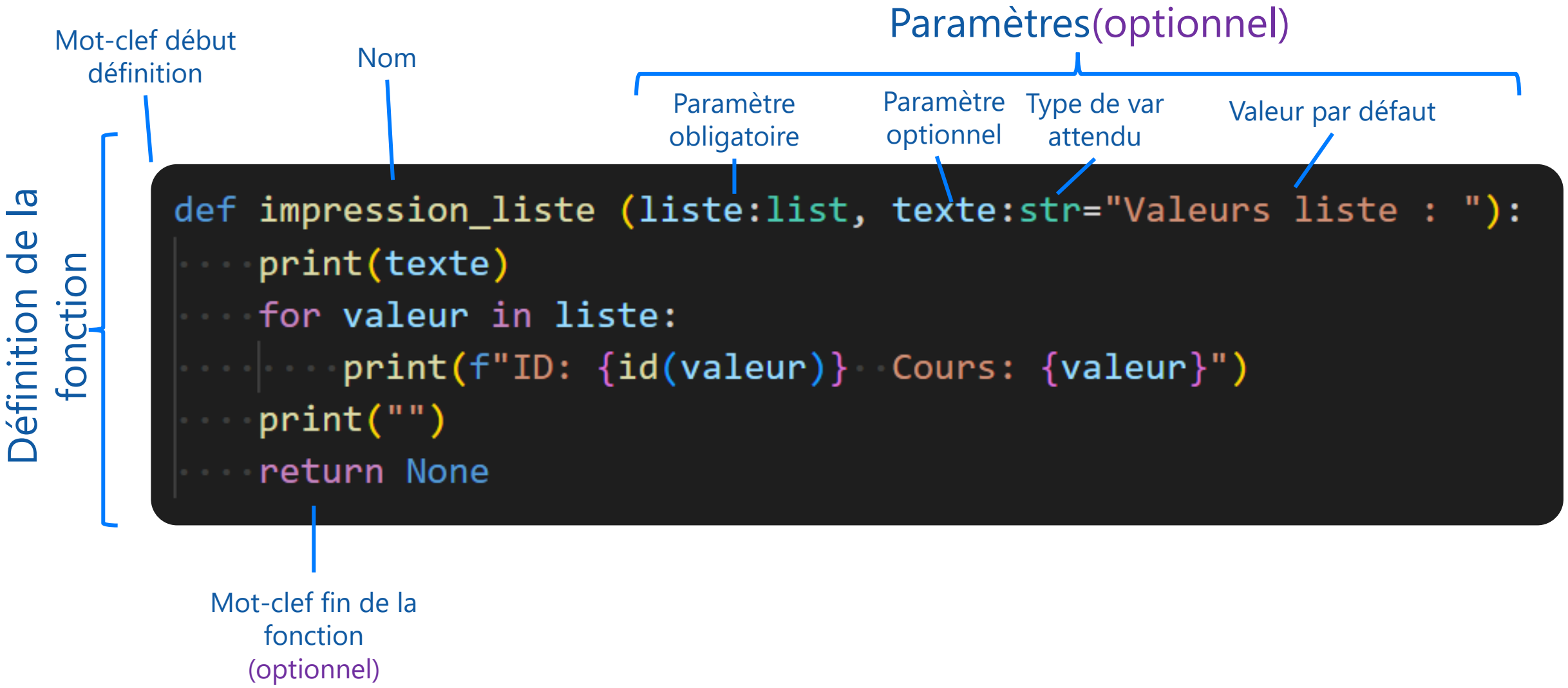
Module requests

Requêtes http

API RESTfull (communication avec sites web)

Les fonctions

retour



Résumé module JSON

retour

```
[
    {
        "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 [ ]**

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

Nous utilisons 2 fonctions :

- nom_variable = json.loads("string")
  - qui prend un str en paramètre et qui retourne un objet python (List ou Dict)
- variable_string = json.dumps(objet_python)
  - Qui prend un objet python (List ou Dict) et qui retourne un str.

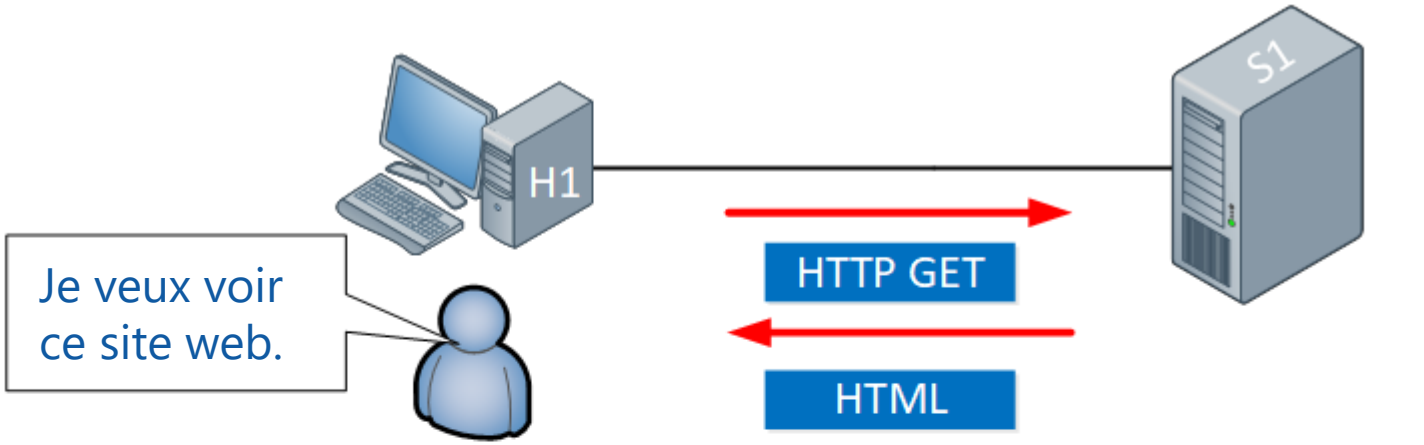
API Web

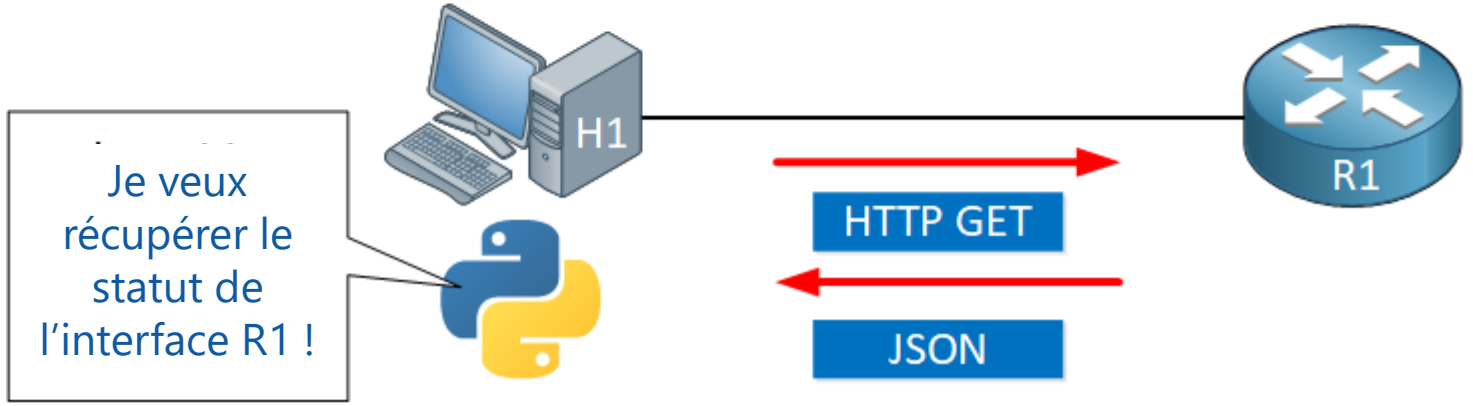
Interopérabilité

*« **L'interopérabilité** est la capacité que possède un produit ou un système, dont les interfaces sont intégralement connues, à fonctionner avec d'autres produits ou systèmes existants ou futurs et ce sans restriction d'accès ou de mise en œuvre. »*

Source: Wikipédia

Transmission de données

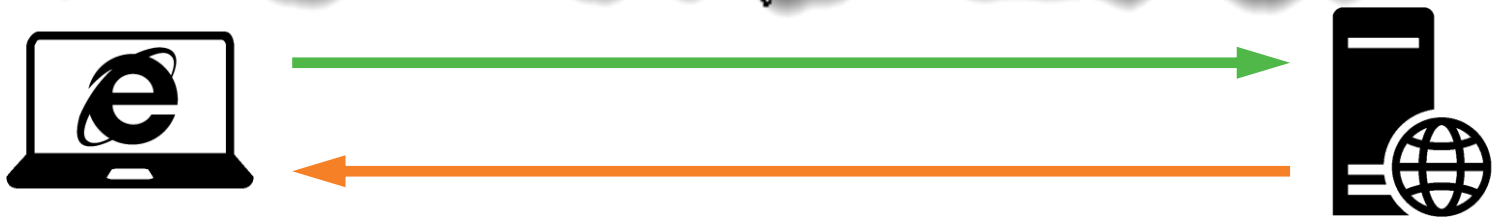




Source: NetworkLessons.com

Échange HTTP

```
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 Edouard Montpetit"></meta><meta http-equiv="Content-Type" content="text/html;
```

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 *payload*)
PUT	Crée ou remplace une ressource sur le serveur
HEAD	Ne demande que les informations sur la ressource, pas son *payload*
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

Hors du cadre du cours

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 fakestorapi.com

```python
_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

```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 est 02-20-2023 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
```

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

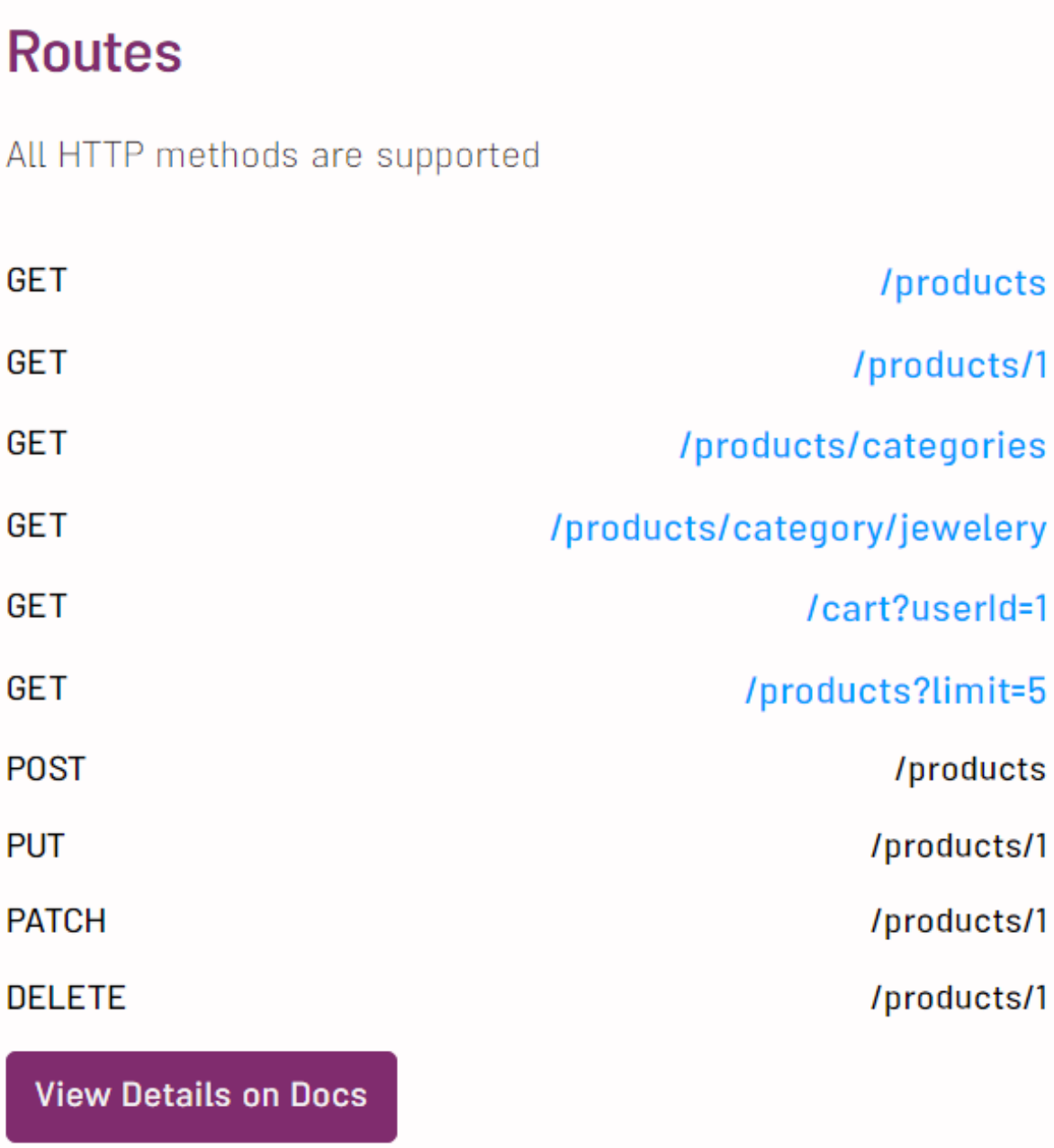
Fake store rest API for your e-commerce or shopping website prototype

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



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