

BeautifulSoup

ou le doux potage de la moisson



La bibliothèque python *BeautifulSoup* permet d'extraire des informations d'un site web, ou encore d'un document XML, avec quelques lignes de code.

Son nom est tiré d'une chanson de la fausse tortue, dans *Alice au pays des merveilles* :

« Ô doux potage,
Ô mets délicieux !
Ah ! pour partage,
Quoi de plus précieux ?
Plonger dans ma soupière
Cette vaste cuillère
Est un bonheur
Qui me réjouit le cœur. »

(source: [Wikisource](#))

BeautifulSoup - Guide simplifié

Recette de base (1)

Les premières lignes d'un script moissonnant des données à l'aide de *BeautifulSoup* sont souvent les mêmes.

1. On importe d'abord les modules dont on a besoin, notamment BeautifulSoup.

```
1  # coding: utf-8
2
3  import csv
4  import requests
5  from bs4 import BeautifulSoup
6
```

2. Si on veut moissonner un site web, il est utile de mettre l'URL dans une variable qu'on peut appeler *url*.

```
10
11  url = "http://www.admfincs.forces.gc.ca/apps
12
```

3. On personnalise aussi les entêtes qui seront transmises au site en question, histoire de faire du journalisme à visière levée. Dans certaines circonstances, il peut être nécessaire d'ajouter des éléments à ces entêtes, voire de carrément s'en passer.

```
13  entetes = {
14      "User-Agent": "Jean-Hugues Roy - Reque
15      "From": "tarantino.quentin@uqam.ca"
16  }
```


BeautifulSoup - Guide simplifié

Recette de base (2)

Les premières lignes d'un script moissonnant des données à l'aide de *BeautifulSoup* sont souvent les mêmes.

4. Ensuite, on utilise *requests* pour placer le contenu de la page web dans la variable... *contenu*.

```
17
18 contenu = requests.get(url, headers=entetes)
19
```

5. Et enfin, on analyse («*parse*») le contenu afin de le rendre facilement interrogeable par les différentes fonctions offertes par BeautifulSoup. Ces fonctions vont pouvoir être appliquées à la variable *page*.

```
19
20 page = BeautifulSoup(contenu.text, "html.parser")
21
```

Il peut arriver qu'on doive commencer un moissonnage dans un ou des fichiers *.html* qu'on a téléchargés localement, sur son ordinateur. Dans ces cas, l'étape 4 est inutile puisqu'on ne se connecte pas dans le web et la syntaxe de l'étape 5 change quelque peu. Si le fichier *.html* qu'on avait téléchargé s'appelait *index.html*, voici le qu'il faudrait plutôt entrer.

```
78
79 page = BeautifulSoup(open("index.html"), "html.parser")
80
```

BeautifulSoup - Guide simplifié

Recette de base (3)

Il peut arriver qu'on doive commencer un moissonnage dans un ou des fichiers .html qu'on a téléchargés localement, sur son ordinateur.

4. Ensuite, on utilise *requests* pour placer le contenu de la page web dans la variable... *contenu*.

```
17
18 contenu = requests.get(url, headers=entetes)
19
```

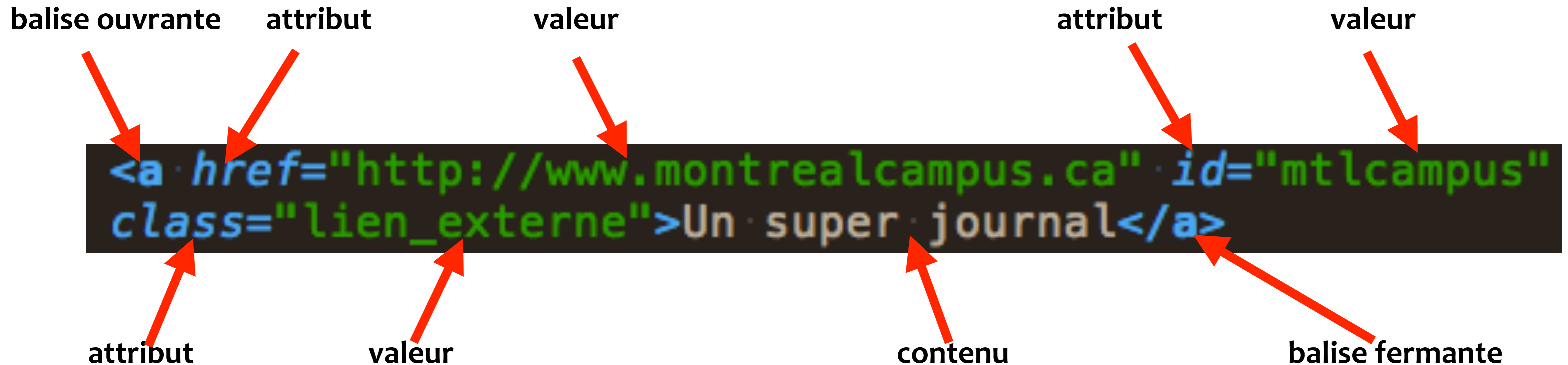
5. Et enfin, on analyse («*parse*») le contenu afin de le rendre facilement interrogeable par les différentes fonctions offertes par BeautifulSoup. Ces fonctions vont pouvoir être appliquées à la variable *page*.

```
19
20 page = BeautifulSoup(contenu.text, "html.parser")
21
```

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML

Tout bon moissonnage commence par un examen du code HTML de la page qui nous intéresse. On se souvient tous de la structure d'un élément HTML. Chaque élément est fait d'une balise, qui peut être accompagnée d'attributs, de valeurs et de contenu.



BeautifulSoup peut nous permettre d'accéder à chacun de ces éléments.

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> par balise

La façon la plus simple d'accéder à un élément HTML est de le faire par nom de balise à l'aide de la fonction `.find()`

`.find()`

```
<a href="http://www.montrealcampus.ca" id="mtlcampus" class="lien_externe">Un super journal</a>
```

Ainsi, si on écrit `.find("a")`, ici, *BeautifulSoup* nous retourne tout le HTML de l'élément `<a>`.

```
<a href="http://www.montrealcampus.ca" id="mtlcampus" class="lien_externe">Un super journal</a>
```

À noter qu'on pourrait écrire aussi tout simplement `.a`, et ce serait la même chose que d'écrire `.find("a")`

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> par attribut (par id)

Il peut arriver qu'on ne veuille sélectionner que des éléments qui ont un *id* particulier.

, id=" "

```
<a href="http://www.montrealcampus.ca" id="mtlcampus" class="lien_externe">Un super journal</a>
```

Ainsi, s'il y a plusieurs éléments `<a>` dans une page, mais que seul celui ou ceux dont le `id` est `mtlcampus` nous intéresse(nt), il suffit d'écrire `.find("a", id="mtlcampus")`. *BeautifulSoup* nous retournera alors l'ensemble du code HTML de la balise `<a>` correspondante.

```
<a href="http://www.montrealcampus.ca" id="mtlcampus" class="lien_externe">Un super journal</a>
```

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> par attribut (par *class*)

La même syntaxe s'applique à tout attribut présent dans un élément HTML, sauf deux. Il y a l'attribut *class* qui exige une syntaxe particulière.

```
, class_=" "
```

```
<a href="http://www.montrealcampus.ca" id="mtlcampus"  
class="lien_externe">Un super journal</a>
```

Ainsi, s'il y a plusieurs éléments `<a>` dans une page, mais que seul celui ou ceux dont la classe est, dans ce cas-ci, `lien_externe` nous intéresse(nt), il suffit d'écrire `.find("a", class_="lien_externe")`. *BeautifulSoup* nous retournera alors l'ensemble du code HTML de la balise `<a>` correspondante.

```
<a href="http://www.montrealcampus.ca" id="mtlcampus" class="lien_externe">Un super journal</a>
```


BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> par attribut (par *name*)

Il y a aussi l'attribut *name*, qu'on rencontre par exemple dans des éléments *meta*.

```
<meta name="language" content="fr">
```

Ici, la syntaxe à utiliser pour repérer des éléments semblables est la suivante :

```
, attrs={"name": "valeur"}
```

Ainsi, dans le cas de l'exemple ci-dessus, il faudrait écrire `.find("meta", attrs={"name": "language"})`. *BeautifulSoup* nous retournera alors l'ensemble du code HTML de la balise `<meta>` en question.

```
<meta name="language" content="fr">
```

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> le contenu

Quand on moissonne des pages web, la plupart du temps, ce qui nous intéresse, c'est le contenu. Pour le récupérer, il suffit d'utiliser la fonction `.text`

`.text`

```
<a href="http://www.montrealcampus.ca" id="mtlcampus"
class="lien_externe">Un super journal</a>
```

Ainsi, si on écrit `.find("a").text`, *BeautifulSoup* nous retourne le contenu de l'élément `<a>`.

Un super journal

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> les valeurs des attributs

Quand on moissonne des pages web, souvent, aussi, ce qui nous intéresse, ce sont les hyperliens contenus dans des balises `<a>`. Pour les récupérer, il faut savoir qu'un attribut et sa valeur sont transformés par *BeautifulSoup* en un petit dictionnaire (comme on l'a vu avec l'attribut *name*) dont la clé est l'attribut et la valeur... sa valeur.

```
["attribut"]
```

```
<a href="http://www.montrealcampus.ca" id="mtlcampus"
class="lien_externe">Un super journal</a>
```

Donc, si on écrit `.find("a")["href"]`, ou encore `.a["href"]`, *BeautifulSoup* nous retourne la valeur de l'attribut `href`, à savoir l'URL vers lequel pointe l'hyperlien formé par l'élément `<a>`.

```
http://www.montrealcampus.ca
```


BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> plusieurs éléments


Quand on moissonne des pages web, en fait, on veut souvent récupérer un ensemble d'éléments semblables. La fonction `.find_all()` est alors très utile.

`.find_all()`

Par exemple, si on cherche à récupérer toutes les lignes (balise `<tr>`) d'un tableau qui nous intéresse dans une page web donnée, il suffit d'écrire `.find_all("tr")`. *BeautifulSoup* créera alors une **liste** dans laquelle il placera chacun des éléments `<tr>`. C'est ainsi qu'on pourra créer une boucle pour travailler avec chacun de ces éléments et y appliquer d'autres fonctions de *BeautifulSoup*.

Dans un site du ministère de la Défense, par exemple, la commande ci-dessous, à gauche, produira l'affichage reproduit partiellement ci-dessous, à droite :

```
for ligne in page.find_all("tr"):
    print(ligne.a)
```



```
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191173">DADS INC</a>
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191174">ALPHA CHEMI
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191175">pro-expert
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191176">Garage Leon
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191177">BCL X-ray C
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191178">PwC</a>
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191179">UNITED STAT
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191180">Severn Tren
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191181">Frank's Ele
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191182">Golder Asso
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191183">Orion Const
<a href="contract-contrat-fra.asp?q=3&y=2016&id=id191184">Dexter Con
```

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> enchaîner les fonctions

Comme dans toute fonction en python, il est possible d'enchaîner les fonctions offertes par *BeautifulSoup*.

```
▼<div class="conteneurColonnesImbriquees">
  ▼<div class="colonneImbriquee imbGauche">
    <!-- Placeholder: permet d'afficher le titre sur 1 colonne.
    cas quel. (default: s'affiche) -->
    ▼<h1>
      ▼<span id=
        "ctl00_ColCentre_TitreSurUneColonne_SiteMapPathTitreContenu
        ">
        ▶<a href=
          "#ctl00_ColCentre_TitreSurUneColonne_SiteMapPathTitreCont
          enu_SkipLink">...</a>
          <span>Journal des débats de l'Assemblée
          nationale</span>
```

Cela peut être utile quand le contenu qui nous intéresse se trouve dans une balise très commune. Dans l'exemple ci-contre, on veut le contenu d'une balise `<h1>`. Comme il y a plusieurs autres balises `<h1>` dans cette page, la façon de distinguer celle qu'on veut est de dire à l'intérieur de quelle autre balise elle se trouve.

Ici, donc, on entrera la commande à gauche.

À droite, vous en voyez le résultat :

```
print(page.find("div", class_="imbGauche").h1.text)
```



Journal des débats de l'Assemblée nationale

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> trouver l'élément suivant

Quand les éléments qu'on cherche n'ont ni classe, ni *id* qui puissent les rendre facilement identifiables, on peut alors les retrouver grâce à du contenu se trouvant tout près.

`.find_next()`

```
▼<tr>
  <th>
    Salles de bain (nb)</th>
  <td>
    2,5 salles de bain</td>
</tr>
```

Dans l'exemple ci-contre, on cherche à extraire le texte « **2,5 salles de bain** ». Mais voilà, il se trouve dans une balise `<td>` insérée dans une balise `<tr>` qui ressemblent à des dizaines de balises semblables dans la même page.

Ici, on peut donc d'abord repérer l'élément qui précède, puis utiliser la fonction `.find_next()`

Ici, donc, on entrera la commande à gauche.

```
for i in page.find_all("th"):
    if i.text.strip() == "Salles de bain (nb)":
        print(i.find_next("td").text.strip())
```

À droite, vous en voyez le résultat :

2,5 salles de bain

Accéder à des éléments HTML -> erreurs les plus courantes (1)

Dans mon expérience, voici les erreurs les plus couramment rencontrées en faisant du moissonnage.

Cas 1 - Le serveur me refuse l'accès après un certain temps.

Quand on fait plusieurs requêtes à un serveur, il arrive qu'il refuse qu'on s'y connecte. D'autres sont programmés pour détecter les requêtes qui s'enchaînent à quelques millisecondes d'intervalle.

Une parade, ici, est de permettre au serveur de respirer un peu entre chacune de vos requêtes.

Insérez un `sleep` d'une seconde ou moins (un tiers de seconde, dans l'exemple ci-dessous).

```
time.sleep(0.33)
```

Il faut cependant ajouter le module `time` au début de votre script.

```
import time
```

Accéder à des éléments HTML -> erreurs les plus courantes (2)

Dans mon expérience, voici les erreurs les plus couramment rencontrées en faisant du moissonnage.

Cas 2 - Erreur de connexion

Commencez par **vérifier si l'URL que vous demandez est valide!** C'est la source de nombreuses erreurs.

Parfois, aussi, changer le *http* par un *https* dans l'URL (ou vice-versa) a pu régler des problèmes.

```
url="http://www.assnat.qc.ca/
```



```
url="https://www.assnat.qc.ca/
```

C'est dommage pour la transparence, mais il arrive enfin que les entêtes personnalisées qu'on envoie ne soient pas reconnues par le serveur. Il suffit alors de les enlever en transformant cette ligne :

```
contenu = requests.get(url, headers=entetes)
```

Par ceci :

```
contenu = requests.get(url)
```

Accéder à des éléments HTML -> erreurs les plus courantes (3)

Dans mon expérience, voici les erreurs les plus couramment rencontrées en faisant du moissonnage.

Cas 3 - Autres types d'erreurs

Selon les erreurs que je recevais, j'ai aussi, dans le passé, modifié les entêtes que j'envoyais au serveur pour régler différents problèmes en ajoutant les items suivants dans ma variable *entetes* :

Item ajouté

```
entetes = {  
    "User-Agent": "Jean-Hugues Roy - Requête envoyée",  
    "From": "tarantino.quentin@uqam.ca",  
    "Connection": "Keep-Alive"  
}
```

Item ajouté

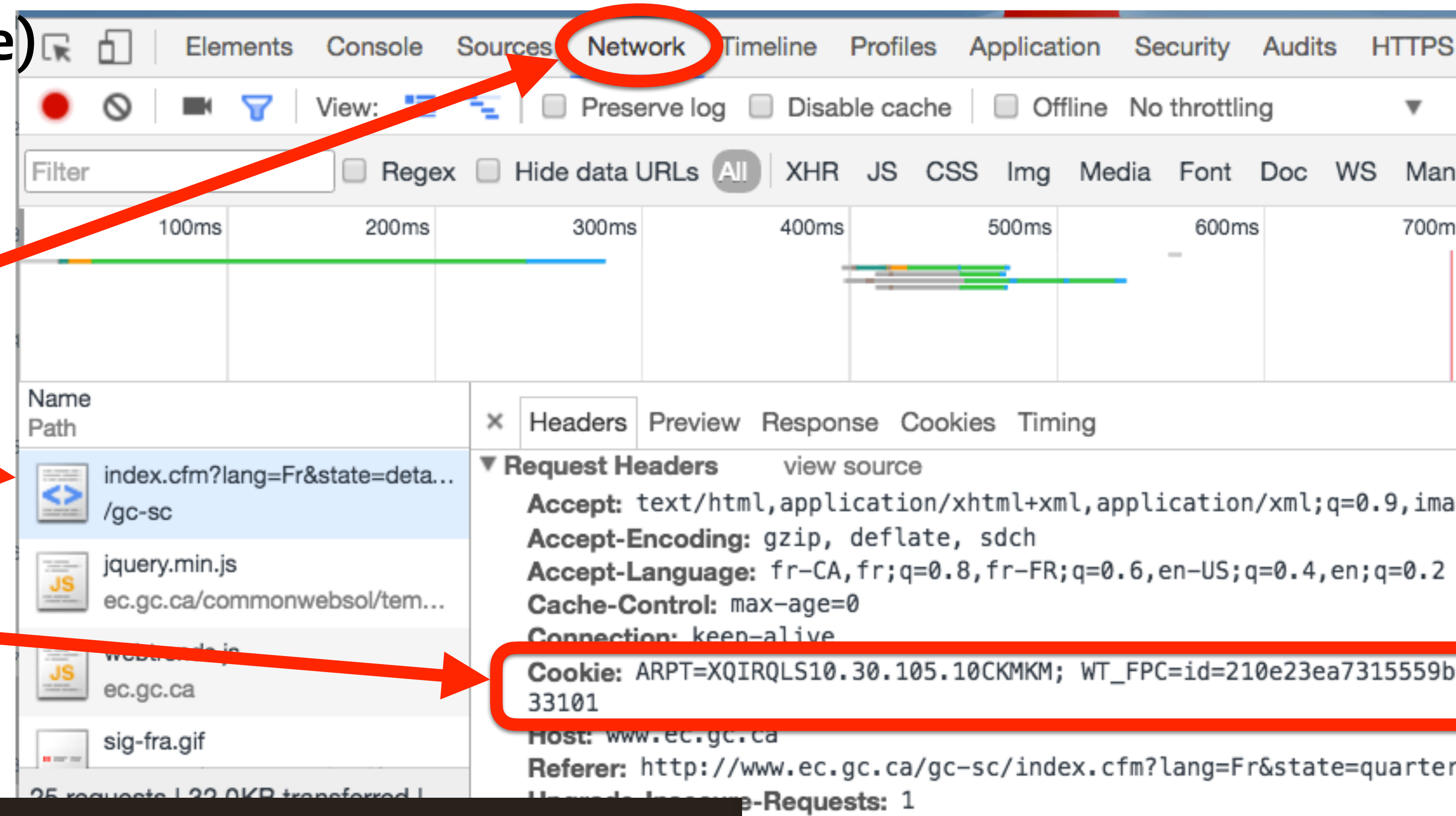
```
entetes = {  
    "User-Agent": "Jean-Hugues Roy - Requête envoyée",  
    "From": "tarantino.quentin@uqam.ca",  
    "Accept-Encoding": "identity"  
}
```


BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> erreurs les plus courantes (4)

Cas 3 - Autres types d'erreurs (suite)

1. Pour envoyer d'autres types d'entêtes, allez dans les options de développement de votre navigateur et sélectionnez l'onglet « Network ».
2. Choisissez le premier fichier qui apparaît dans la liste ci-dessous.
3. Rechargez la page (cmd-R ou ctrl-R).
4. Copiez le cookie qui est envoyé au site et collez-le dans votre variable entetes. Cela pourrait permettre de débloquer certains sites capricieux.



```
entetes = {  
    "Cookie": "ARPT=XQIRQLS10.30.105.10CKMKM; WT_FPC=id=210e23ea7315559b1cc1489884433101:"  
}
```

Si toutes ces solutions ne fonctionnent pas, votre ami est toujours Google!

BeautifulSoup - Guide simplifié

Accéder à des éléments HTML -> dernier truc -> les balises *meta*

Je vous ai montré, tout à l'heure, comment accéder au contenu des balises `<meta>`. Allez voir ce qu'elles contiennent. Vous y trouverez souvent des informations intéressantes. On peut, par exemple, y trouver des coordonnées géographiques.

```
<meta property="og:latitude" content="45.5164257">  
<meta property="og:longitude" content="-73.5579767">
```

Pour accéder aux coordonnées cachées dans l'attribut `content`, il suffirait d'écrire des commandes ressemblant à ceci :

```
print(page.find("meta", property="og:latitude")["content"])  
print(page.find("meta", property="og:longitude")["content"])
```

BeautifulSoup retournerait alors cela :

```
45.5164257  
-73.5579767
```


BeautifulSoup - Guide simplifié

Documentation complète

Ce guide simplifié pourrait ne pas contenir l'info que vous cherchez. Je vous conseille alors de consulter les documentations complètes de **BeautifulSoup** et de **requests** qui toutes deux peuvent s'avérer utiles.



Bons moissonnages !