TD : récupérer les identifiants de films sur *Allociné*

Alexandre Roulois (Université Paris Cité, LLF, CNRS)

Table of contents

Lire une page Web

Page Web = document structuré

- marqueurs autour de segments
- hiérarchie des informations
- représention arborescente

Films de SF 2010-2019 les mieux notés sur Allociné :
http://www.allocine.fr/films/notes/genre-13021/decennie-2010/

Importer le module `urllib` :

```
import urllib.request
```

Configurer la requête HTTP :

```
url = 'http://www.allocine.fr/films/notes/genre-13021/decennie-2010/'
headers = { 'User-agent' : 'HTML extractor (your name here)' }
request = urllib.request.Request(url, headers=headers)
```

Extraire le contenu HTML :

```
with urllib.request.urlopen(request) as fichier:
  html = fichier.read().decode('utf-8')
```

Analyse syntaxique

Quel est le type, au sens de type de données, de la variable `html` ?

```
# <class 'str'>
type(html)
```

Question : comment interpréter du texte brut comme texte structuré ?

Importer module *BeautifulSoup* :

- analyses syntaxiques HTML et XML
- modélisation arborescente du fichier
- méthodes pour parcourir, rechercher et modifier un arbre

```
from bs4 import BeautifulSoup
```

Lancer l'analyse syntaxique :

```
soup = BeautifulSoup(html, 'html.parser')
```

Rechercher tous les films dans le document :

- méthode `.select()` avec comme entrée un sélecteur CSS
- sélecteur `.meta-title-link` trop gourmand
- contrainte supplémentaire comme enfant de `.mdl`

```
movies = soup.select('.mdl .meta-title-link')
```

Pour chaque film, récupérer le lien hypertexte :

```
# for each movie…
for movie in movies:
  # … get the content in 'href' attribute
  href = movie.get('href')
```

Et constituer plutôt une liste propre :

```
# each link is added into a list of 'href'
hrefs = [ movie.get('href') for movie in movies ]
```

Écrire dans un fichier

Objectif : enregistrer les liens dans un fichier plat

Besoins :

- créer nouveau fichier avec droits en écriture
- insérer chaque lien dans le fichier
- un lien par ligne

```
with open('./data/allocine/links.txt', 'w') as file:
  # instructions
  pass
```

2e étape : pour chaque lien, lancer une instruction d'écriture dans le fichier

```
with open('./data/allocine/links.txt', 'w') as file:
  for href in hrefs:
    file.write(href)
```

3e étape : ajouter un retour à la ligne à chaque fois

```
with open('./data/allocine/links.txt', 'w') as file:
  for href in hrefs:
    file.write(href)
    file.write('\n')
```

Optimiser

Modulariser le code grâce à une fonction utilisateur :

```
def get_html_from_url(url, charset='utf-8'):
  """Extracts the HTML code from a single URL.

  Keyword arguments:
  url -- the url to scrape
```

```
    charset -- charset used to decode caracters
    """
    headers = { 'User-agent' : 'HTML extractor (your name here)' }
    request = urllib.request.Request(url, headers=headers)

    with urllib.request.urlopen(request) as f:
      html = f.read().decode(charset)

    return html
```

Placer la fonction et ses dépendances (`urllib.request`) dans un fichier *utils.py*, à importer comme un module dans le script :

```
import utils
```

Créer un répertoire *scrape* et placer le fichier *utils.py* à l'intérieur. Modifier l'appel au module :

```
# 'utils' module is now in a 'scrape' folder
import scrape.utils
```

Appeler ensuite la fonction au lieu de tout le code explicite :

```
# get HTML
html = scrape.utils.get_html_from_url(url)
```