S2-MAJEUR - Manipulation des Données avec Python

Chapitre 4 : Web crawling et Web scrapping - partie 1

Les langages web de balisages : HTML et XML

Langages de Balisage et le Web

Les langages de balisage structurent le contenu sur le web, facilitant la présentation et l'échange de données. Parmi les plus connus :

- HTML (HyperText Markup Language) : crée la structure et le contenu visuel des pages web.
- XML (eXtensible Markup Language) : conçu pour stocker et transporter des données de manière simple et lisible.
- XHTML (eXtensible HyperText Markup Language) : combine les strictes conformités de XML avec HTML.
- D'autres technologies basées sur XML, comme SVG pour les graphiques vectoriels ou XSLT pour transformer les documents XML, enrichissent l'écosystème web.

Ces langages soutiennent la diversité et la richesse des interactions sur le web, rendant l'information accessible et manipulable à travers différentes plateformes.

Autres techniques et langages du web des données

1. XML : Langage de balisage pour structurer les données.
2. DTD : Type de document pour définir la structure d'un document XML.
3. XSLT : Langage de transformation pour convertir des documents XML en d'autres formats.
4. XSD : Schéma XML pour définir la structure et la validation des documents XML.
5. RDF : Framework pour représenter des métadonnées et des informations liées sur le web.
6. OWL : Langage pour définir des ontologies utilisées dans le web sémantique.
7. XPath : Langage de navigation pour extraire des données à partir de documents XML.
8. XQuery : Langage de requête pour interroger des documents XML et des bases de données XML.
9. SparQL : est un langage de requête conçu pour RDF.

Structure d’arbre

On a une structure arborescente à respecter :

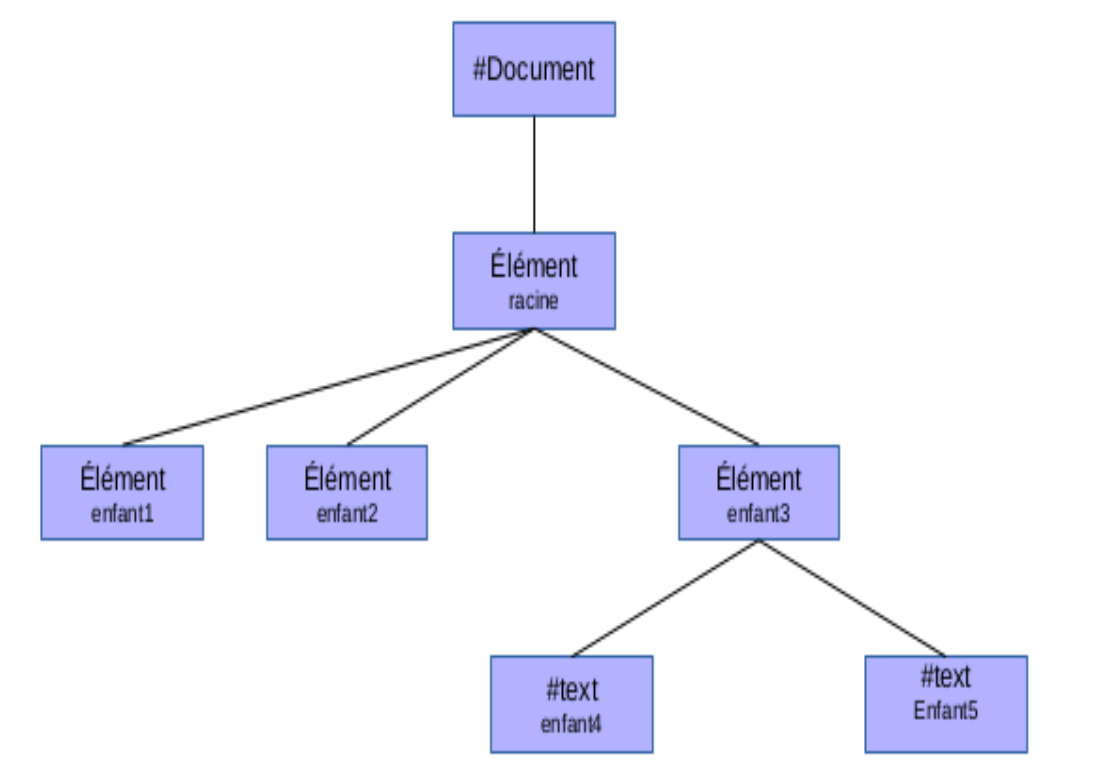

Fig.7 Une histoire de famille.

un fichier XML

On a aussi :

1. une seule racine par document
2. des éléments ouvert et fermés
3. des élements qui ne se chevauche pas

```
<parent>
    texte1
    <enfant>texte2</enfant>
    texte3
</parent>
```

Figure 1: Enter Caption

Un fichier HTML

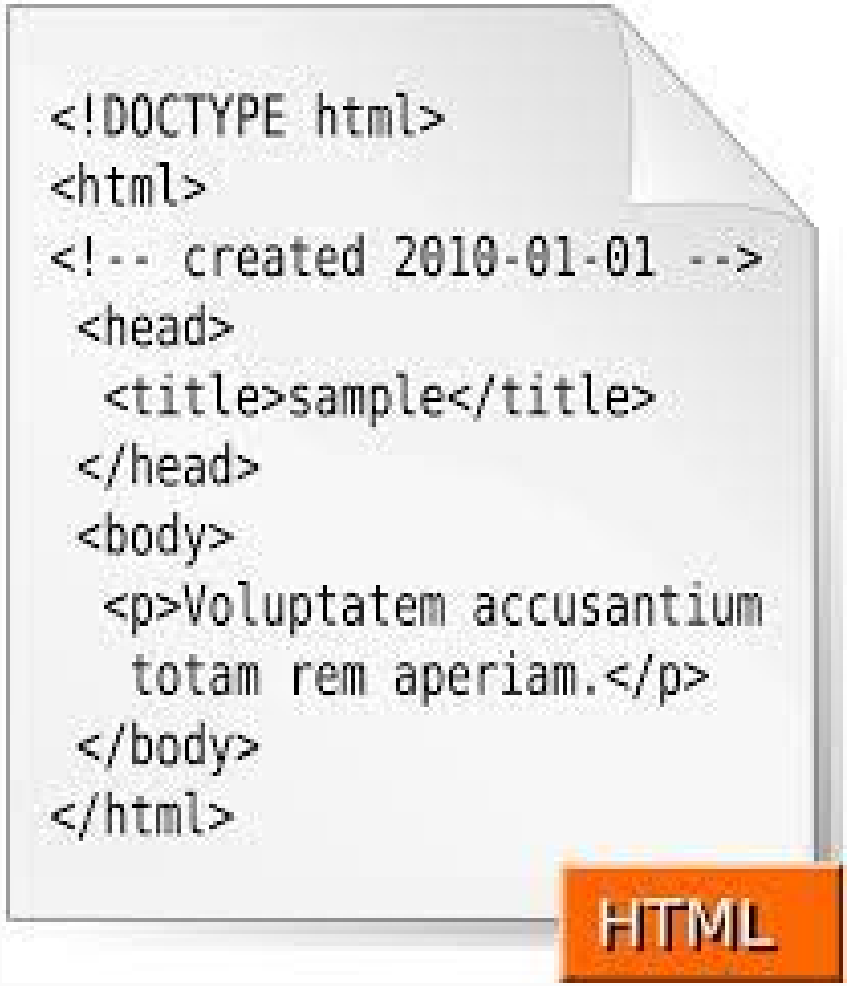

Figure 2: avec la racine html

Un fichier XML

Le format XML est au centre de nombreux processus : Formats d'enregistrement, échange de données entre applications, outils et langages de programmation, Bases de données.

```
<!-- Racine -->
▼<animaux xml:type="Une_Alimentation">
  <!-- rajouter un attribu type et la valeur entre guillement -->
 ▼<alimentation xml:type="carnivore">
    <!-- Fermer la balise et mettre le texte entre guillement -->
    <animal nom="renard"/>
    <!-- idem -->
    <animal nom="loup"/>
  </alimentation>
  <!-- rajouter un attribu type et  la valeur entre guillement -->
 ▼<alimentation xml:type="herbivore">
    <!-- rajouter la valeur entre guillement -->
    <animal nom="vache"/>
    <!-- idem -->
    <animal nom="cheval"/>
    <!-- Fermer la balise alimentation -->
  </alimentation>
</animaux>
```

Figure 3: Enter Caption

En route vers Xpath

Xpath et python

Le but ici n'est pas de faire un cours sur le web des données.

Dans cette partie on apprend à utiliser Xpath, pour pouvoir l'utiliser avec python lorsqu'on fait du scrapping.

Nous pouvons donc faire du Xpath avec les librairies pythons suivantes : lxml, xml, selenium .

Enfin, retenons que Xpath est une alternative aux Regex et autres technique de récupération des données que nous verrons plus loin.
Voir notebook (tutoriel de la séance).

XPath : Extraction d'information XML

XPath est un langage permettant d'interroger des documents XML, offrant une syntaxe et plus de 200 fonctions pour extraire des éléments, attributs et manipuler différents types de données. Il utilise des chemins d'accès pour sélectionner des nœuds, permettant des requêtes flexibles et puissantes dans divers langages de programmation.

Exemple d'utilisation XPath

Pour récupérer le texte contenu dans un élément spécifique d'un document XML, on utilise la syntaxe :
/messages/message/contenu/text()

```
<?xml version="1.0" encoding
    ="utf-8" ?>
<!DOCTYPE messages SYSTEM "
    messages.dtd" >
<messages>
    <message numero="4">
        <dest>Alice</dest>
        <contenu>Bonjour !</
            contenu>
    </message>
</messages>
```

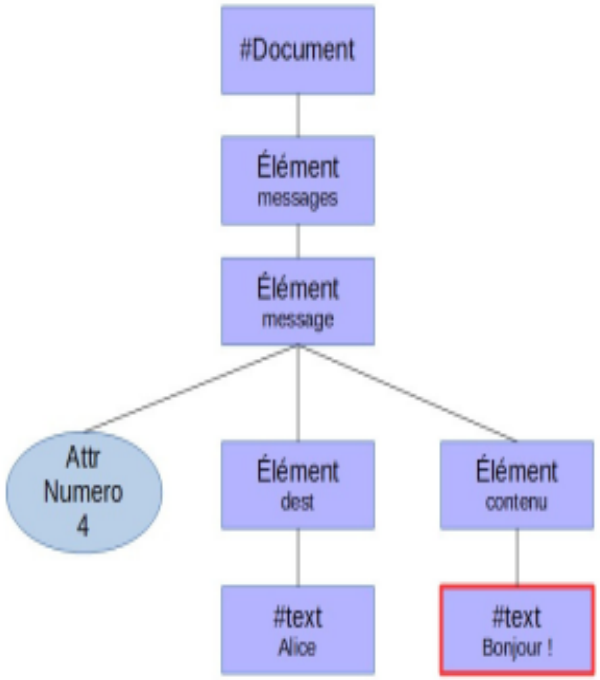

Figure 4: Un exemple

Séparateurs dans XPath

XPath définit deux principaux séparateurs pour naviguer dans les documents XML :

- ”/” indique un chemin direct depuis l’élément courant, tandis que

- ”//” permet de sauter un nombre indéfini d’éléments pour atteindre la cible spécifiée.

Exemples simples avec XPath

- `/messages/message/dest` sélectionne tous les éléments `<dest>` sous `<message>` depuis la racine `<messages>`.
- `//dest` trouve tous les éléments `<dest>` dans le document, quel que soit leur emplacement.
- `/messages//contenu` sélectionne tous les éléments **<contenu>** situés sous la racine **<messages>**

Recherche d'attributs avec XPath

Utilisez le symbole "@" pour cibler un attribut. Exemples :

- `/messages/message/@numero` pour sélectionner les attributs `numero` des éléments `message`.
- `//@numero` pour trouver tous les attributs `numero` dans le document.

Étapes supplémentaires

- . : désigne le nœud courant
- .. : désigne le nœud parent
- * : désigne tous les nœuds de ce niveau
- | : regroupe les résultats de deux expressions XPath

Exemples :
- /messages/*/@numero sélectionne tous les nœuds attributs nommé numero des éléments situés sous la racine <messages>
- //dest/@* sélectionne tous les nœuds attributs des éléments <dest> du document
- //message/dest|//message/contenu sélectionne les éléments <dest> et <contenu> sous <message> en combinant deux chemins

Conditions sur les chemins avec XPath

XPath permet de filtrer les nœuds en fonction de conditions spécifiques, utilisant des prédicats (entre crochets) pour affiner les sélections.

/messages/message[@numero=5]/contenu : sélectionne les <contenu> des messages dont l'attribut numero est 5.

Opérateurs logiques dans XPath

Les conditions peuvent intégrer des opérateurs logiques pour des comparaisons complexes, permettant des sélections précises basées sur le contenu ou les attributs des nœuds.

Opérateurs de comparaison en XPath :
- Arithmétique : + - * div mod
- Comparaison : < <= = != > =>
- logique : and or not

Exemples :
- //message[not(@numero < 5 and @numero >= 9)] : sélectionne les <message> dont l'attribut est entre 5 et 8.
- //message[@numero mod 5 = 0] : sélectionne les messages dont l'attribut numero est un multiple de 5.

Fonctions utiles en XPath

XPath inclut des fonctions pour manipuler les chaînes, calculer des valeurs ou travailler avec des dates et heures, enrichissant la capacité de requête sur les documents XML. - contains(s1,s2) : vrai si s1 contient s2
- starts-with(s1,s2) et ends-with(s1,s2) : vrai si s1 commence/finit par s2
- matches(s,motif) : vrai si s correspond à motif
- abs(nb), flootr(nb) et round(nb)
- text() : sélectionne tous les nœuds textes sous l'élément courant, y compris dans les descendants ;
- /messages/message/contenu/text()
- node() : sélectionne tous les nœuds enfants de l'élément

Fonctions utiles en XPath

- string(s) : retourne le texte de s ;
string(/messages/message[2]/contenu)
- position() : retourne l'index de l'élément dans son parent ;
/messages/message[position()<=3]
- last() : retourne le numero du dernier élément de son parent ;
//dest[position()>last()-3]
- count(expression) : compte le nombre de nœuds XML sélectionnés par l'expression ; //message[count(dest)>2]
- string-length(s) : retourne la longueur de la chaîne s
- concat(s1,s2,...) : concatène les chaînes s1, s2, etc. -
substrings(s,deb,long) : retourne les long caractères de s à partir de deb (commence à 1).

Les axes en XPath

Les axes XPath spécifient la direction à suivre pour naviguer dans le document XML, offrant une grande flexibilité pour accéder à différents nœuds relatifs au contexte courant.
- child : : pour parcourir les enfants du contexte
- descendant : : pour parcourir les enfants et petits-enfants, équivaut à //
- parent : : pour parcourir le parent du contexte, équivaut à .. avec un test sur le parent voulu
- ancestor : : pour parcourir les parents et grands-parents
- preceding-sibling : : pour parcourir les frères précédents
- following-sibling : : pour parcourir les frères suivants
- attribute : : pour parcourir les attributs du contexte, équivaut à@

Autres techniques courantes, complémentaires et / ou facultatives

Feuille de route

Avant d'aller plus loin, retenons que nous allons toucher aux points suivants :

1. **Robots.txt**
2. **Builtwith**, Wappalyzer
3. **Whois**
4. **Sitemaps**
5. **Google search optimization**
6. **RegEx**
7. **web crawling**
8. **web scrapping**

Dans le notebook tutoriel ils ne sont pas abordés dans cet ordre pour la simple raison qu'il est possible d'aller à l'essentiel, mais, dans ce cas, c'est uniquement à des fins pédagogique, car faire ne suffit pas, il faut bien faire...et pour y parvenir il faut respecter les étapes.

Robots.txt

La plupart des sites web définissent un fichier robots.txt pour informer les robots d'éventuelles restrictions concernant le crawl de leur site web.

Ces restrictions sont simplement des suggestions, que les bons citoyens du web les respecteront.

Le fichier robots.txt est une ressource précieuse à vérifier avant de crawler pour minimiser le risque d'être bloqué, et aussi pour découvrir des indices sur la structure d'un site web. Plus d'informations sur le protocole robots.txt sont disponibles sur http://www.robotstxt.org.

Builtwith, Wappalyzer

Pour identifier la technologie utilisée par un site web, nous pouvons utiliser le module **builtwith**, qui analyse un site web et renvoie les technologies utilisées.

Par exemple, pour le site example.webscraping.com, nous obtenons des informations sur les frameworks JavaScript, les langages de programmation, les frameworks web et les serveurs web utilisés.

Whois

Pour trouver le propriétaire d'un site web, nous pouvons utiliser le protocole **WHOIS** pour voir qui est le propriétaire enregistré du nom de domaine.

Un module Python appelé python-whois facilite cette tâche.

En interrogeant le domaine appspot.com avec ce module, nous obtenons des informations sur les serveurs de noms, l'organisation propriétaire et les adresses e-mail associées.

Sitemaps

Les fichiers de plan de site sont fournis par les sites web pour aider les robots à localiser leur contenu mis à jour sans avoir besoin de crawler chaque page web.

Pour plus de détails, la norme de plan de site est définie à l'adresse **http://www.sitemaps.org/protocol.html**.

Google search optimization

La taille du site web cible affectera la manière dont nous le crawle. Si le site web ne comporte que quelques centaines d'URL, comme notre exemple de site web, l'efficacité n'est pas importante.

Cependant, si le site web compte plus d'un million de pages web, télécharger chacune séquentiellement prendrait des mois. Il existe une façon de faire qui permet de résoudre ce problème.

C'est l'utilisation des raccourcis de recherche de Google , apr exemple :

1. site: -> **site:wikipedia.org Python** vous renverra vers la page wikipeida de python
2. intitle: **intitle:Python tutorial** renverra des résultats contenant "Python tutorial" dans leur titre
3. etc.

Et donc quadn vous faites du web crawling cela peut vous être utile .

RegEx : Utilisation des expressions régulières

Pour récupérer des liens qui mènent vers d'autres pages web, nous devons être un peu plus précis sur les motifs qui nous intéressent, d'où l'intérêt d'apprendre ce que l'on appelle les expressions régulières.

cette section est très bien détaillé dans le tutoriel.

Concepts fondamentaux

- **Les caractères littéraux** : Les expressions régulières peuvent contenir des caractères littéraux qui correspondent exactement à eux-mêmes. Par exemple, le motif ”abc” correspondrait littéralement à la séquence de caractères ”abc” dans une chaîne.
- **Les métacaractères** : Ce sont des caractères spéciaux qui ont une signification particulière dans les expressions régulières. Par exemple, le métacaractère ”.” correspond à n’importe quel caractère unique, le métacaractère ”^” correspond au début d’une ligne, et le métacaractère $̈correspond à la fin d’une ligne.

Concepts fondamentaux

- **Les classes de caractères** : Les classes de caractères permettent de spécifier un ensemble de caractères parmi lesquels le moteur d'expression régulière peut faire correspondre un seul caractère. Par exemple, "[abc]" correspondrait à n'importe lequel des caractères "a", "b" ou "c".
- **Les quantificateurs** : Les quantificateurs sont utilisés pour spécifier le nombre d'occurrences d'un motif à rechercher. Par exemple, **"*" signifie zéro ou plusieurs** occurrences, **"+" signifie une ou plusieurs** occurrences, et **"?" signifie zéro ou une** occurrence.

Concepts fondamentaux

- **Les groupes de capture** : Les parenthèses sont utilisées pour créer des groupes de capture, ce qui permet d'extraire des sous-parties d'une correspondance. Par exemple, dans le motif "(ab)+", le groupe "(ab)" correspondrait à la séquence "ab", et le quantificateur "+" indiquerait une ou plusieurs occurrences de ce groupe.

Exemples d'expressions régulières

- Vérifier un numéro de téléphone : `^(\d{2}) \d{9}$`
- Vérifier une adresse email :
  `[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`

web crawling

Le web crawling est un processus automatisé où des robots d'indexation parcourent le web en suivant les liens entre les pages.

Google utilise le web crawling pour découvrir de nouvelles pages web et mettre à jour son index de recherche.

Exemple : dans une page, on récupère tous les liens (hyperliens) .

Web Crawling, Web Scrapping, API

Méthodes de Récupération de Données sur le Web

1. **Web Crawling** : Robots parcourant le web pour indexer le contenu des pages.
2. **Web Scraping** : Extraction de données spécifiques de pages web via des outils comme Beautiful Soup.
3. **API** : Accès à des données structurées via des interfaces de programmation.
4. **RSS** : Abonnement à des flux de contenu pour des mises à jour régulières.
5. **Crowdsourcing** : Collecte de données via la contribution humaine.
6. **Scraping Visuel** : Simulation du comportement de navigation pour extraire des données visuelles.
7. **Capture d'Écran** : Collecte de données visuelles via des captures d'écran automatisées.

En conclusion

Le but de cette première partie consiste à vous introduire aux pratiques du web scrawling et du web scrapping.

Dans la deuxième partie, vous trouverez des techniques poussées de résupérations de données pour les captcha, téléchargement des sites web ,etc.

Ensuite dans le dernier chapitre du cours, nous verrons ce qu'est une api, comment faire le choix d'une bonne api, comment comprendre une api et comment l'utiliser.