Introduction au Web Scraping

Définition et concepts

Le **web scraping** est une technique automatisée d'extraction de données depuis des sites web. BeautifulSoup est une bibliothèque Python qui facilite cette tâche en permettant de parser le HTML et d'en extraire les informations structurées.

Important

Éthique du scraping : Toujours respecter robots.txt, limiter la fréquence des requêtes, et vérifier les conditions d'utilisation des sites.

Installation des dépendances

```
# Installation des packages n cessaires
pip install requests beautifulsoup4 lxml pandas numpy
```

Rappel Complet de BeautifulSoup

Structure de base d'un script

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# 1. Requ te HTTP
url = "https://example.com"
response = requests.get(url)

# 2. V rification du statut
if response.status_code == 200:
    # 3. Parsing du HTML
    soup = BeautifulSoup(response.content, 'html.parser')

    # 4. Extraction des donn es
    # Code d'extraction ici...
else:
    print(f"Erreur {response.status_code}")
```

Méthodes de sélection principales

Par nom de balise


```
# Premier    lment    d'un type
soup.find('div')

# Tous les    lments    d'un type
soup.find_all('div')

# Avec limite
soup.find_all('div', limit=5)
```

Par attributs CSS

```
# Par classe
soup.find_all('div', class_='product')
soup.select('div.product')

# Par ID
soup.find('div', id='header')
soup.select('#header')

# Combinaison
soup.select('div.product.special')
```

Par attributs personnalisés

```
# Recherche par attribut data
soup.find_all('div', attrs={'data-category': 'books'})

# Recherche avec expression r guli re
import re
soup.find_all('div', id=re.compile('^product_'))
```

Navigation dans l'arbre DOM

Relations parent-enfant

```
# Parent direct
element.parent

# Parents r cursifs
for parent in element.parents:
    print(parent.name)

# Enfants directs
for child in element.children:
    print(child)

```



```
# Tous les descendants
for descendant in element.descendants:
    print(descendant)
```

Relations fraternelles

```
# Fr re suivant
next_sibling = element.next_sibling

# Fr re pr c dent
prev_sibling = element.previous_sibling

# Tous les fr res suivants
for sibling in element.next_siblings:
    print(sibling)
```

Extraction des données

Textes et contenus

```
# Texte direct
text = element.get_text()

# Texte avec s parateur
text = element.get_text(separator=' ', strip=True)

# Contenu HTML interne
html_content = str(element)

# Attributs sp cifiques
link = element['href']
src = element['src']
class_list = element.get('class', [])
```

Gestion des données manquantes

```
# M thode s curis e
def safe_extract(element, selector, default='N/A'):
    found = element.select_one(selector)
    return found.get_text(strip=True) if found else default

# Application
title = safe_extract(product, 'h3.title')
price = safe_extract(product, '.price')
```


Sélecteurs CSS avancés

```
# Combinaisons complexes
soup.select('div.products > ul > li:first-child')
soup.select('a[href^="https://"]')  # Commence par
soup.select('img[src$=".jpg"]')     # Termine par
soup.select('div:not(.excluded)')   # Exclusion
```

Gestion des erreurs et bonnes pratiques

Gestion robuste des requêtes

```
import requests
from requests.adapters import HTTPAdapter
from requests.packages.urllib3.util.retry import Retry

def create_session_with_retries():
    session = requests.Session()

    # Strat gie de retry
    retry_strategy = Retry(
        total=3,
        backoff_factor=1,
        status_forcelist=[429, 500, 502, 503, 504],
    )

    adapter = HTTPAdapter(max_retries=retry_strategy)
    session.mount("http://", adapter)
    session.mount("https://", adapter)

    return session
```

Respect des serveurs

```
import time
import random

def respectful_delay(min_delay=1, max_delay=3):
    """Pause al atoire entre les requ tes"""
    time.sleep(random.uniform(min_delay, max_delay))

def check_robots_txt(base_url):
    """V rifier robots.txt"""
    try:
        robots_url = f"{base_url}/robots.txt"
        response = requests.get(robots_url)
        print("Robots.txt:", response.text[:500])
```



```
    except:
        print("Robots.txt non accessible")
```

Exercices Challenges

Exercice 1 : Books to Scrape - Extraction complète

Site : `https://books.toscrape.com/`
Objectif : Créer un scraper qui extrait pour chaque livre :

- Titre complet et URL détaillée
- Prix (converti en float)
- Note sur 5 (convertie en numérique)
- Catégorie principale et secondaire
- Description complète (page détail)
- Stock disponible (nombre d'exemplaires)
- URL de l'image haute définition

Challenge :

- Gérer automatiquement la pagination sur toutes les pages
- Structurer les données en JSON hiérarchique
- Gérer les erreurs 404 sur les pages détail
- Sauvegarder avec timestamp dans le nom de fichier

Exercice 2 : Quotes to Scrape - Graphe de données

Site : `http://quotes.toscrape.com/`
Objectif :

- Extraire toutes les citations avec leurs métadonnées
- Pour chaque auteur, visiter sa page et extraire :
  - Biographie complète
  - Date et lieu de naissance
  - Date de décès (si applicable)
- Créer des relations : Citation $\rightarrow$ Auteur $\rightarrow$ Tags

Challenge :

- Implémenter un système de cache pour éviter les requêtes répétées
- Construire un graphe de relations entre entités
- Exporter en format GraphML ou GEXF
- Détecter les auteurs les plus cités

Exercice 3 : Fake Jobs - Filtres avancés

Site : `https://realpython.github.io/fake-jobs/`
Objectif :

— Extraire uniquement les offres contenant "Python"

— Nettoyer et standardiser les dates

— Classer par type de contrat et localisation

— Extraire et valider les URLs d'application

Challenge :

— Créer un système de filtres dynamiques en ligne de commande

— Implémenter la détection de doublons

— Générer des statistiques par ville et type de contrat

— Sauvegarder en CSV avec encodage UTF-8

Exercice 4 : Analyse de marché livresque

Site : `https://books.toscrape.com/`
Objectif :

— Calculer le prix moyen par note et par catégorie

— Identifier les tendances de prix

— Détecter les livres en rupture de stock

— Analyser la distribution des ratings

Challenge :

— Générer un rapport PDF avec matplotlib

— Créer des visualisations interactives

— Implémenter des alertes prix

— Calculer la corrélation note/prix

Exercice 5 : Navigation catégorielle avancée

Site : `https://books.toscrape.com/`
Objectif :

— Cartographier toute l'arborescence des catégories

— Pour chaque catégorie, calculer :

  — Nombre total de livres

  — Prix moyen, minimum et maximum moyenne pondérée

— Générer un classement des catégories

Challenge :

— Créer un arbre hiérarchique des catégories

— Détecter les catégories sous-représentées

— Implémenter une recherche full-text

— Exporter en format arborescent (JSON nested)

Exercice 6 : Scraper résilient

Site : `http://books.toscrape.com/`
Objectif :

— Implémenter un système de retry intelligent

— Gérer les timeouts adaptatifs

— Logger détaillé des succès/échecs

— Sauvegarde automatique de la progression

Challenge :

— Reprise sur interruption

— Limitation automatique du débit

— Détection de blocage IP

— Rapport de performance temps réel

Exercice 7 : Pipeline de data cleaning

Site : `https://books.toscrape.com/`
Objectif :

— Nettoyer et standardiser tous les champs

— Valider la cohérence des données

— Corriger les encodages problématiques

— Générer un rapport de qualité

Challenge :

— Détection automatique d'anomalies

— Imputation des valeurs manquantes

— Validation croisée des données

— Métriques de qualité automatisées

Exercice 8 : Scraping multi-sources

Sites :

— `https://books.toscrape.com/`

— `http://quotes.toscrape.com/`

— `https://realpython.github.io/fake-jobs/`

Objectif :

— Architecture modulaire pour multiples sources

— Unification du format de sortie

— Gestion des schémas différents

— Comparaison des performances

Challenge :

— System de plugins dynamique

— Configuration externe (YAML/JSON)

— Parallel processing des sources

— Agrégation intelligente des données

Exercice 9 : Authentification et sessions

Site : `http://quotes.toscrape.com/login`

Objectif :

— Gérer l'authentification complète

— Maintenir la session active

— Accéder au contenu protégé

— Gérer les cookies et tokens

Challenge :

— Automatisation du login/logout

— Refresh automatique de session

— Gestion des CAPTCHAs simples

— Sécurisation des credentials

Conclusion

Ce cours couvre l'ensemble des techniques avancées de web scraping avec BeautifulSoup. Les exercices proposés sont progressifs et couvrent des scénarios réels de complexité croissante.

Recommandations

- **Toujours tester localement** avant de scaler
- **Respecter les limites** techniques et légales
- **Documenter son code** pour le maintenir
- **Monitorer les performances** et les erreurs
- **Sauvegarder régulièrement** les données intermédiaires