

Web scraping

(Scrapy)

Présenté par :
Maryame Laouina



Plan



- Définition NLP et ses domaines d'application
- La relation entre web scraping et nlp
- Les outils web scraping
- Scrapy et ses avantages
- Comparaison du Scrapy avec d'autres outils web scraping
- L'architecture du scrapy
- Les selecteurs CSS et XPATH
- Stockage des données
- Les conditions d'utilisation d'un site web
- défit courant et les solutions proposés
- Atelier

NLP(Natural Language Processing)



Le Traitement Automatique du Langage Naturel est un domaine de l'intelligence artificielle (IA) qui se concentre sur l'interaction entre les ordinateurs et le langage humain. Il vise à permettre aux machines de comprendre, d'interpréter et de générer du langage humain de manière naturelle.

Sa mission principale est de permettre aux machines de comprendre le langage humain sous différentes formes, y compris le texte écrit et la parole. Cela inclut la capacité à extraire des informations significatives à partir de documents textuels, de courriels, des réseaux sociaux ...

Les exemples d'application de NLP

Machine
translation



Microsoft
Translator

NER(Name entity
recognition)



spaCy
OPEN NLP™



Chatbots / voice
assistants



Siri



Hey Cortana



ChatGPT

Sentiment Analysis

- Textblob
- Microsoft Azure Text Analytics
- VADER
- Google Cloud Natural Language API
- Amazon Comprehend

AUTRES

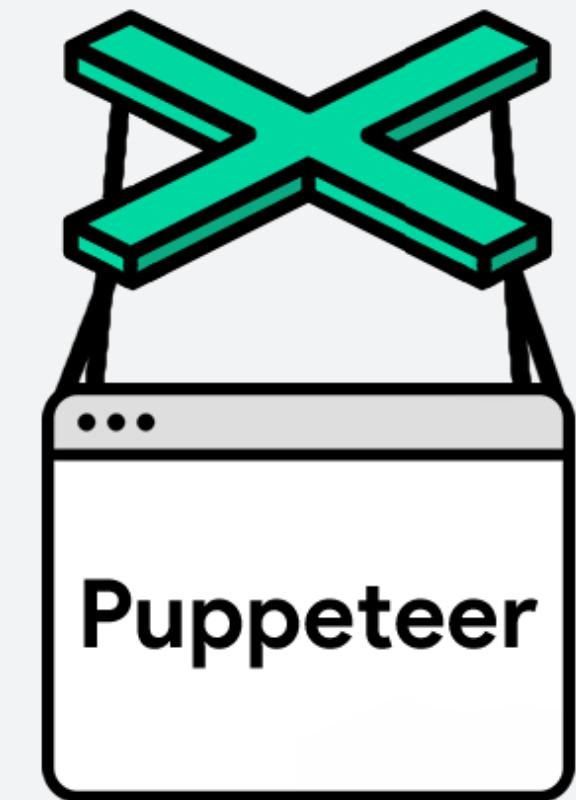
- Recherche d'informations
- Résumé automatique
- Analyse de texte

Relation entre web scraping et NLP



- 1) collecte des données :** Le web scraping consiste à extraire des données à partir de sites web.
- 2) Prétraitement des Données :** Avant d'appliquer nlp, les données collectées doivent être prétraitées pour rendre les données exploitables.
- 3) Extraction et l'analyse de Texte :** nlp peut être utilisé pour extraire et analyser le texte. Cela peut inclure l'analyse de sentiments, l'extraction d'entités nommées ...
- 4) Recherche d'Informations :** Le web scraping collecte des informations brutes, tandis que le NLP recherche des données spécifiques ou génère des analyses à partir de ces informations.
- 5) Surveillance des Médias Sociaux :** Le web scraping collecte des données textuelles sur les médias sociaux, tandis que le NLP surveille les opinions, les tendances et les sentiments des utilisateurs sur ces plateformes.

Les outils web scraping



Framework Scrapy

[Scrapy](#)

Download | Documentation | Resources | Community | Commercial Support | FAQ | Fork on GitHub



Scrapy

An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

Maintained by [Zyte](#) (formerly Scrapinghub) and [many other contributors](#)

[pypi v2.11.0](#) [wheel yes](#) [coverage 89%](#) [Anaconda.org 2.11.0](#)

Install the latest version of Scrapy

 **Scrapy 2.11.0**

`$ pip install scrapy`

[PyPI](#) [Conda](#) [Release Notes](#)



Scrapy

Scrapy et la notion du spider

- Un spider est un programme informatique conçu pour explorer des sites web, collecter des informations à partir de ces sites et les extraire. Il fonctionne en naviguant sur le web, en visitant différentes pages et en récupérant des données spécifiques selon les instructions programmées.

```
(nlp) C:\Users\HP\Desktop\nlp\scrapetuto>scrapy crawl NomSpider
```



Amazon.com: Last 30 days: Books > Departments > Books > Advanced Search > New Releases > Amazon Charts > Best Sellers & More > The New York Times® Best Sellers > Children's Books > Textbooks > Textbook Rentals > Sell Us Your Books > Best Books of the Month > Kindle eBooks

1-12 of over 70,000 results for Books : Last 30 days

Show results for

New Releases

Books

BECOMING

MICHELLE OBAMA

BECOMING Nov 13, 2018 by Michelle Obama

Eligible for Shipping to India

Hardcover \$19.50 \$32.50

More Buying Choices \$13.99 (58 used & new offers)

Kindle Edition \$7.01

Get it TODAY, Nov 27

Audible Audiobook \$0.00

Free with Audible trial

Other Formats: Paperback, Audio CD

Homebody: A Guide to Creating Spaces You Never Want to Leave Nov 6, 2018 by Joanna Gaines

Eligible for Shipping to India

Hardcover \$20.43 \$40.00

More Buying Choices \$18.00 (102 used & new offers)

Kindle Edition

```
Scrap E:\PythonProjects\Scrap
  Project
    - tutorial
      - spiders
        - __init__.py
        - amazon_spider.py
        - quotes_spider.py
        - __init__.py
        - items.py
        - middlewares.py
        - pipelines.py
        - settings.py
      - items.py
      - settings.py
      - items.json
      - scrapy.cfg
      - venv library root
      - list.txt
      - External Libraries
      - Scratches and Consoles
```

```
product_name": ["Becoming", "Homebody: A Guide to Creating Spaces You Never Want to Leave", "Diary of a Wimpy Kid"],  
"product_price": ["19", "22", "20", "19", "6", "5", "8", "1", "21", "13", "16", "15", "35", "11", "12", "14"]
```

```
Scrap E:\PythonProjects\Scrap
  Project
    - tutorial
      - spiders
        - __init__.py
        - amazon_spider.py
        - quotes_spider.py
        - __init__.py
        - items.py
        - middlewares.py
        - pipelines.py
        - settings.py
      - items.py
      - settings.py
      - items.json
      - scrapy.cfg
      - venv library root
      - list.txt
      - External Libraries
      - Scratches and Consoles
```

```
import scrapy
from ..items import AmazonItem

class AmazonSpider(scrapy.Spider):
    name = "amazon"
    start_urls = [
        'https://www.amazon.com/Books-Last-30-days/s?&page=1&rh=n%3A283155%2Cp_n_publication_date%3A1250226011&bbn=283155&ie=UTF8&qid=1543315240&ajr=3'
    ]

    def parse(self, response):
        items = AmazonItem()
        product_name = response.css('h2.s-access-title::text').extract()
        product_price = response.css('span.sx-price-whole::text').extract()
        items['product_name'] = product_name
```

Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

```
(venv) E:\PythonProjects\Scrap>cd tutorial
(venv) E:\PythonProjects\Scrap\tutorial>scrapy crawl amazon -o items.json
```

Pourquoi Scrapy ?



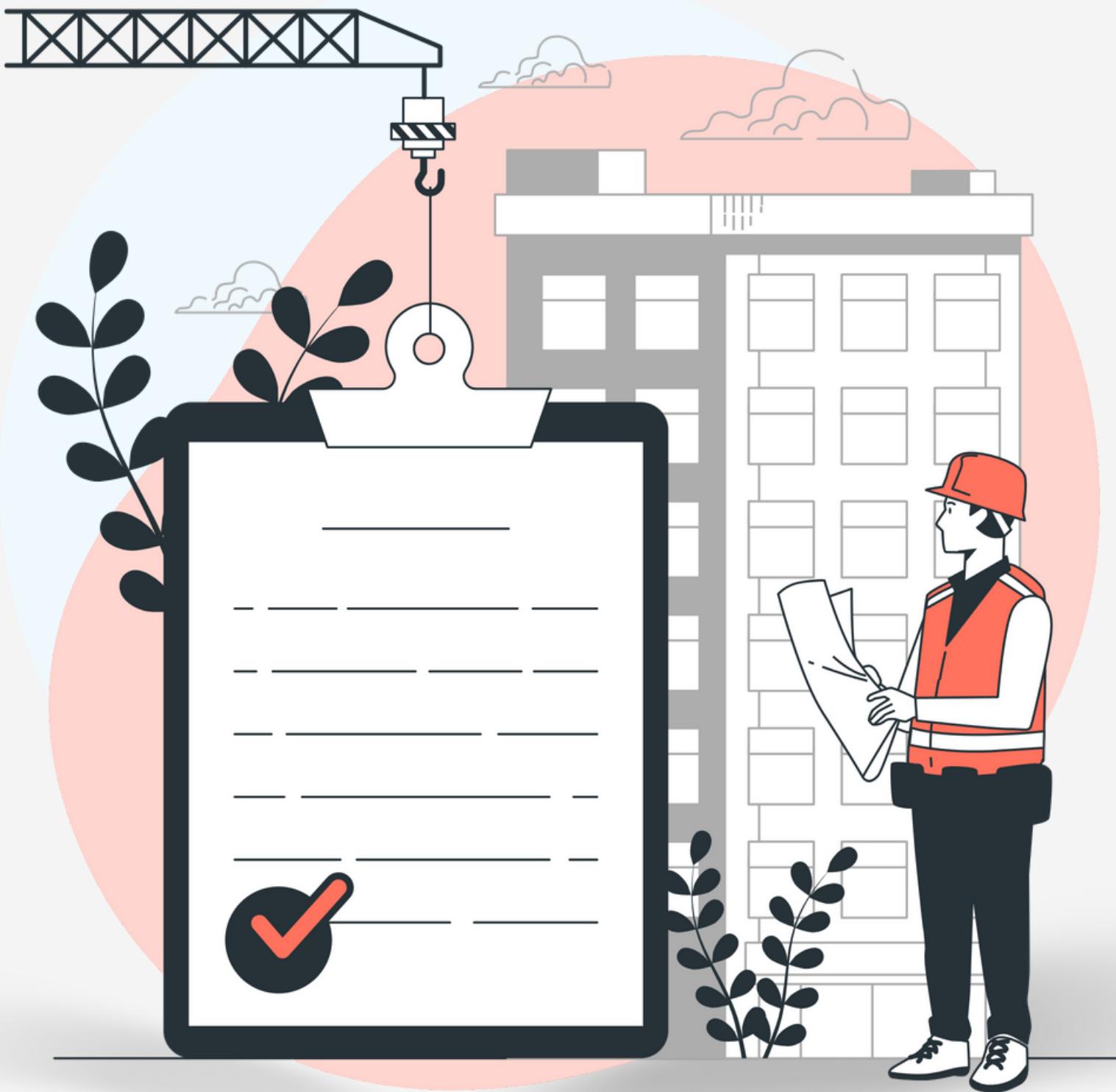
- Open source
- Puissant et Flexible
- Fonctionnalités Intégrées
- une performance élevé
- Extensibilité



Comparaison du scrapy avec d'autres framework

- **Scrapy** est conçu spécifiquement pour le web scraping, tandis que **Beautiful Soup** conçu pour l'analyse de documents HTML et XML. Beautiful Soup est souvent utilisé en combinaison avec d'autres bibliothèques pour effectuer du web scraping.
- **Scrapy** offre une architecture plus complète pour gérer l'ensemble du processus de scraping, tandis que **Requests-HTML** se concentre principalement sur le téléchargement de pages et l'analyse de leur contenu.
- **Scrapy** est spécifique à Python, tandis que **Puppeteer** est basé sur JavaScript. Le choix entre les deux dépendra de la préférence de langage et de la complexité du site web à scraper.
- **Scrapy** est principalement axé sur l'extraction de données à partir de pages web, tandis que **Selenium** est plus orienté vers l'interaction avec des sites web en utilisant un navigateur.

Architectures Scrapy



Les sélecteurs

Naviguer dans la structure des sources HTML et XML pour en extraire des informations précises.

XPath

XPath est un langage de requête utilisé pour naviguer dans les documents XML.

Sélectionner tous les paragraphes d'un document.

```
response.xpath('//p')
```

Sélectionner le texte du premier paragraphe.

```
response.xpath('//p[1]/text()').get()
```

Sélectionner les liens avec un certain attribut.

```
response.xpath('//a[@class="special"]/href').getall()
```

Les sélecteurs

Naviguer dans la structure des sources HTML et XML pour en extraire des informations précises.

CSS

Les sélecteurs CSS sont basés sur la syntaxe de sélection utilisée dans les feuilles de style CSS.

Sélectionner tous les paragraphes d'un document

```
response.css('p')
```

Sélectionner le texte du premier paragraphe.

```
response.css('p::text').get()
```

Sélectionner les liens avec un certain attribut.

```
response.css('a.special::attr(href)').getAll()
```

Items

- Utilisés pour structurer les données scrapées.
- Similaire aux dictionnaires Python.
- Structuration des données et leur validation.
- Organiser et standardiser la manière dont les données sont collectées et traitées dans Scrapy.

```
import scrapy

class MonItem(scrapy.Item):
    titre = scrapy.Field()
    prix = scrapy.Field()
    description = scrapy.Field()
    lien = scrapy.Field()
```

Items

- Un item une classe , avec des champs déclarés comme des attributs de classe.
- scrapy.Field() un marqueur pour définir un champ dans un item.

```
import scrapy

class MonItem(scrapy.Item):
    titre = scrapy.Field()
    prix = scrapy.Field()
    description = scrapy.Field()
    lien = scrapy.Field()
```

Pipline

- Traitement post-extraction des données , filtre, validation et stockage .
- Nettoyage des données .
- Validation d'item .
- Élimination des doublons .
- Stockage des items.
- Manipulation d'images .

```
class BookPipeline:  
  
    # Initialisation de la base de données lors de l'ouverture du spider  
    def open_spider(self, spider):  
        self.conn = sqlite3.connect('books.db')  
        self.curs = self.conn.cursor()  
        self.curs.execute('''CREATE TABLE IF NOT EXISTS books (titre TEXT, auteur TEXT, prix TEXT)''')  
        self.conn.commit()  
  
    # Traitement de chaque item  
    def process_item(self, item, spider):  
        # Nettoyage simple : supprimer les espaces excédentaires  
        item['titre'] = item['titre'].strip()  
        item['auteur'] = item['auteur'].strip()  
        item['prix'] = item['prix'].strip()  
  
        # Insertion dans la base de données  
        self.curs.execute('''INSERT INTO books (titre, auteur, prix) VALUES (?, ?, ?)''',  
                         (item['titre'], item['auteur'], item['prix']))  
        self.conn.commit()  
  
        return item  
  
    # Fermeture de la base de données lors de la fermeture du spider  
    def close_spider(self, spider):  
        self.conn.close()
```

Stockage des Données Collectées



1 : Stockage simple : Fichiers

Scrapy peut stocker les données directement dans des fichiers. Voici les formats les plus courants :

JSON :

```
scrapy crawl mon_spider -o output.json
```

JSON Lines :

```
scrapy crawl mon_spider -o output.jl
```

CSV :

```
scrapy crawl mon_spider -o output.csv
```

XML :

```
scrapy crawl mon_spider -o output.xml
```

2. Bases de données :

Préférable pour des projets plus grands ou pour des données qui nécessitentl un accès structuré, des requêtes complexes ou une intégration avec d'autres applications.

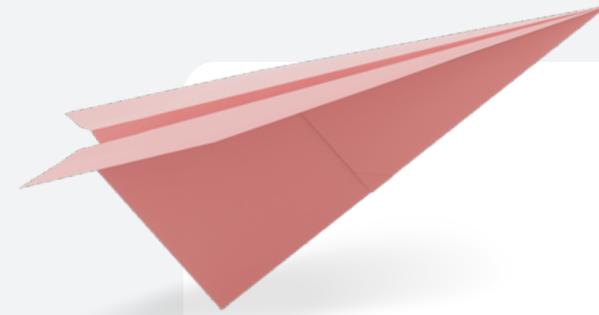
```
import sqlite3

class SQLitePipeline:
    def open_spider(self, spider):
        # Établir la connexion à la base de données lors de l'ouverture du sp
        self.conn = sqlite3.connect('books.db')
        self.curs = self.conn.cursor()
        self.curs.execute('''
            CREATE TABLE IF NOT EXISTS books (
                titre TEXT,
                auteur TEXT,
                prix TEXT
            )
        ''')
        self.conn.commit()
```

3. Stockage dans le cloud :

Avec l'émergence des services cloud, vous pouvez également stocker les données directement dans des solutions comme Amazon S3, Google Cloud Storage ou d'autres plateformes de stockage en cloud.

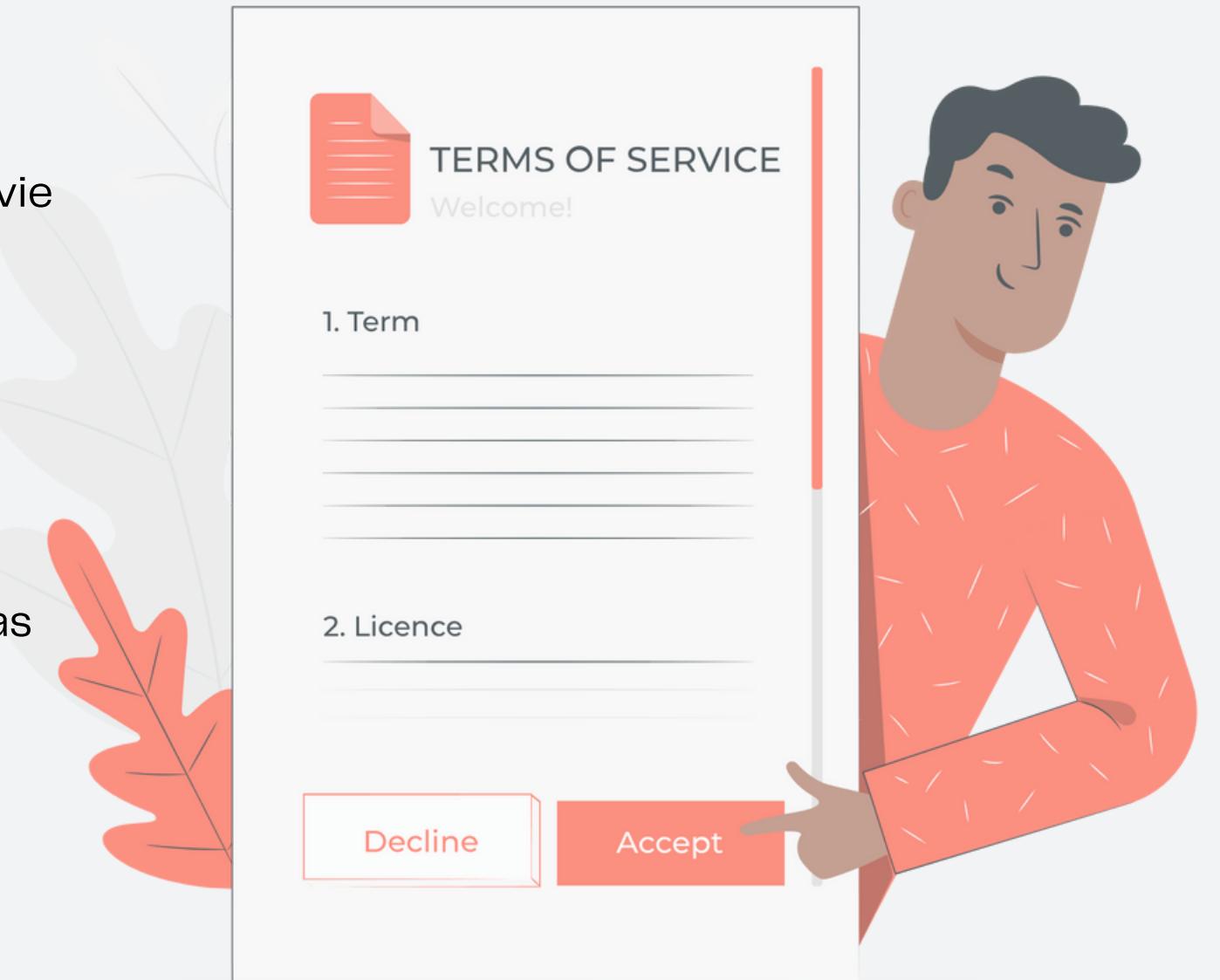




Condition d'utilisation scapy



- **Respectez robots.txt** : Assurez-vous de ne pas accéder aux sections interdites.
- **Fréquence des requêtes** : Utilisez des délais entre les requêtes.
- **Données personnelles** : Respecter les réglementations en matière de protection de la vie privée, telles que le RGPD en Europe.
- **Termes d'utilisation** : Certains sites ont des clauses qui interdisent explicitement le scraping.
- **Utilisation éthique** : Toujours utiliser Scrapy de manière responsable, en veillant à ne pas causer de préjudice ou de perturbations aux sites web cibles.



Limitations, défis lors de l'extraction et solutions :

Restrictions d'accès:

Défi : Blocage via robots.txt.

Solution : Respectez-le ou demandez l'autorisation d'accéder

Blocs IP :

Défi : IP bloquée après trop de requêtes.

Solution : Utilisez des proxies et introduisez des délais.

CAPTCHAs :

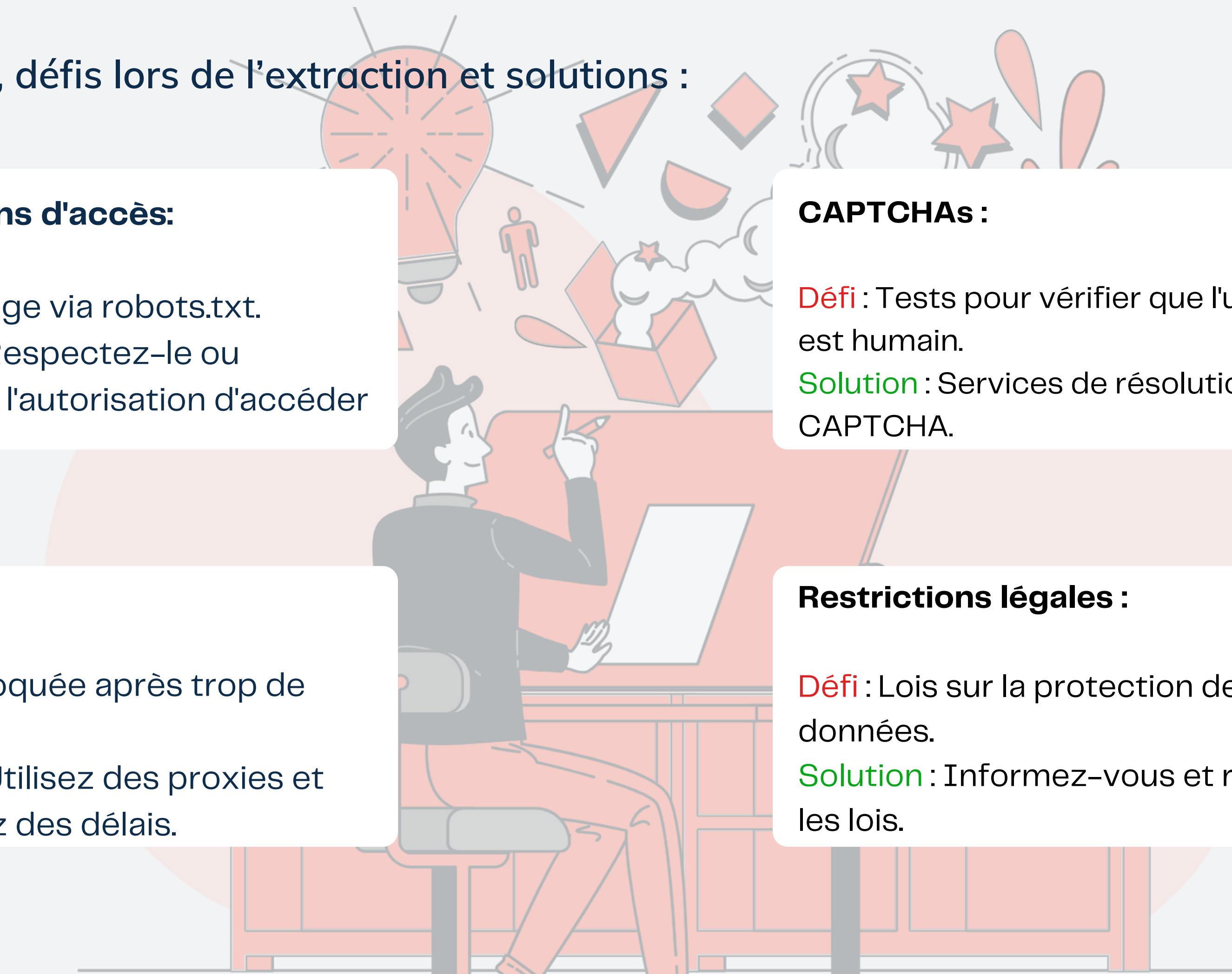
Défi : Tests pour vérifier que l'utilisateur est humain.

Solution : Services de résolution CAPTCHA.

Restrictions légales :

Défi : Lois sur la protection des données.

Solution : Informez-vous et respectez les lois.



Atelier

