Web Scrapping

Janvier 2025

Cédric Dangeard
cedric.dangeard@orange.com

Programme

- Rappel sur le web
- Beautifull Soup
- Selenium
- sqlite3
- Scrapy

Une page web est généralement constitué à l'aide des trois langages çi-dessous:

- **HTML** (HyperText Markup Langage) :
  Défini le "fond" de la page : sa structure et son contenu.

- **CSS** (Cascading Style Sheets) :
  Défini la forme, le style et la position de chaque bloc HTML
- **Javascript** :
  Le langage de programation qui est interprété et exécuté par le navigateur du client

```
<!DOCTYPE html>
<html>
    <head>
        <title>Une petite page HTML</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Un titre de niveau 1</h1>
        <p class = 'enRouge'>
            Un premier petit paragraphe avec une classe associé
        </p>
        <h2>Un titre de niveau 2</h2>
        < id='uiid_paragraph'>
            Un autre paragraphe avec un identifiant
        </p>
    </body>
</html>
```

```css
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}
h2 {
  color: blue;
}
p {
  color: black;
}
.enRouge{
  color: red;
}
#uuid_paragraph{
  font-size: 42px;
  font-style: italic;
}
```

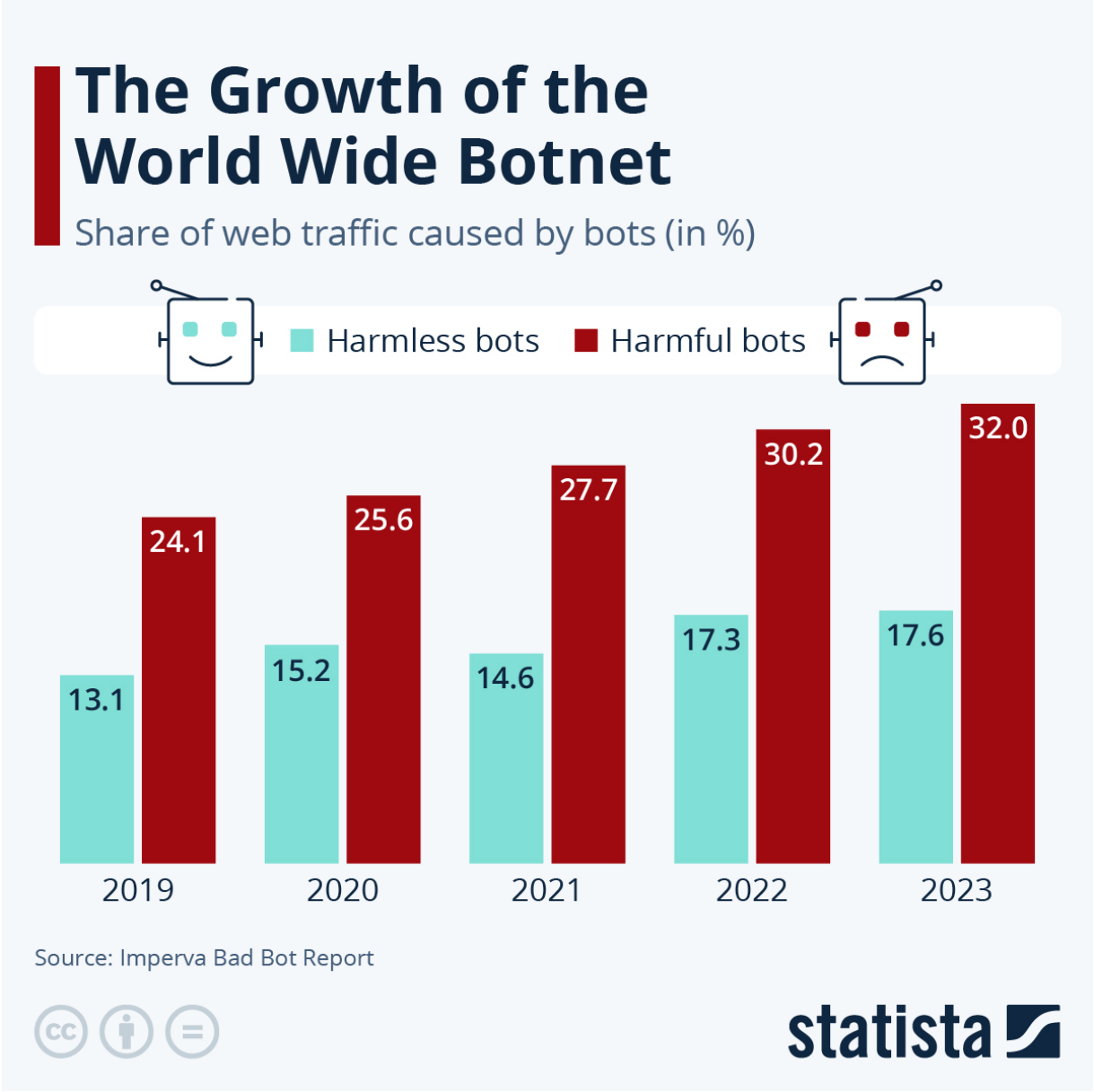
```
<h2>What Can JavaScript Do?</h2>
<p>JavaScript can show hidden HTML elements.</p>
<p id="demo" style="display:none">Hello JavaScript!</p>
<script>
function myFunction() {
  if (document.getElementById('demo').style.display=='none')
    document.getElementById('demo').style.display='block';
  else
    document.getElementById('demo').style.display='none';
}
</script>
<button type="button"
onclick="myFunction()">
Click Me!
</button>
```

WebScrapping

- Extraction automatique de données d'une page web

Web Crawling

- Parcourir les pages web
- Récupérer les liens
- Suivre les liens



Le WebScrapping est-il légal?

- Oui
- Cependant :
  - Attention aux usages et aux données personnelles
  - Privilégier les API quand elles existent
  - Faire attention aux Conditions Générales d'Utilisation

Exemple de CGU
Legifrance

ElementTree & lxml

- Bibliothèques python pour extraire des données de fichiers XML
- elementTree est inclus dans la librairie standard
- lxml est une extension plus rapide et plus complète

Documentation ElementTree
Documentation lxml

Beautifull Soup

- Bibliothèque Python pour extraire des données de fichiers HTML et XML
- Permet de naviguer dans le document, de rechercher des éléments, de les extraire
- Cas d'usage:
  - Récupérer des éléments d'une page web
  - Itérer sur quelques pages similaires
- Installation : `pip install beautifulsoup4`

Exemples

```
from bs4 import BeautifulSoup
import requests

url = 'https://www.lemonde.fr/'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Affiche le titre de la page dans sa balise
print(soup.title)

# Affiche le titre de la page
print(soup.title.text)

# Affiche tous les liens de la page
print(soup.find_all('a'))
```

- `find_all()` : Recherche tous les éléments correspondant à un critère
- `find()` : Recherche le premier élément correspondant à un critère
- `get_text()` : Récupère le texte d'un élément
- `get()` : Récupère la valeur d'un attribut
- `select()` : Recherche des éléments avec un sélecteur CSS
- `parent` : Elément parent
- `children` : Liste des enfants
- `next_sibling` : Elément suivant

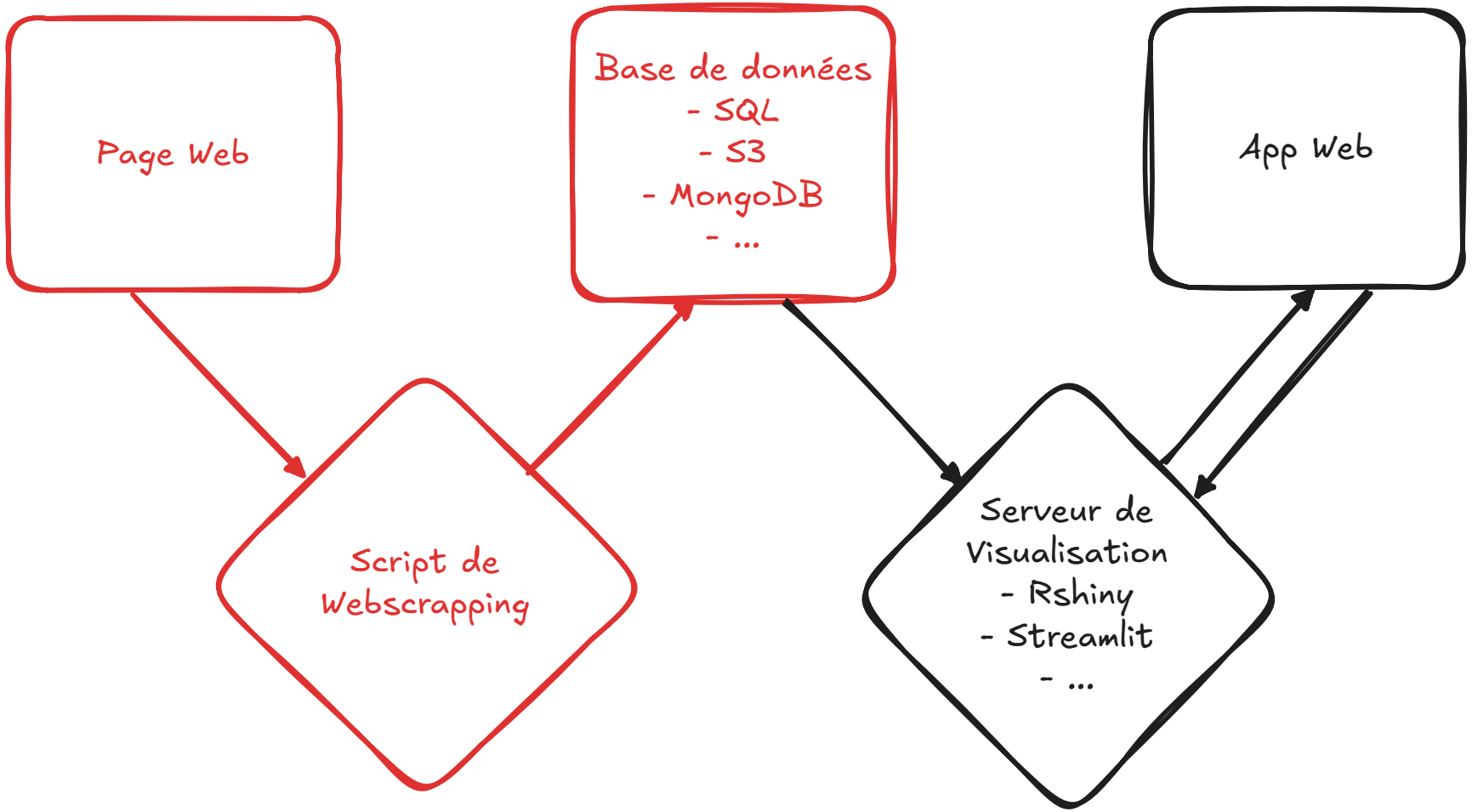
sqlite3

- Bibliothèque Python basé sur le moteur de base de données SQLite
- Permet de stocker les données extraites dans une base de données simple et légère
- stocke les données dans un fichier unique
- Installation : `pip install sqlite3`

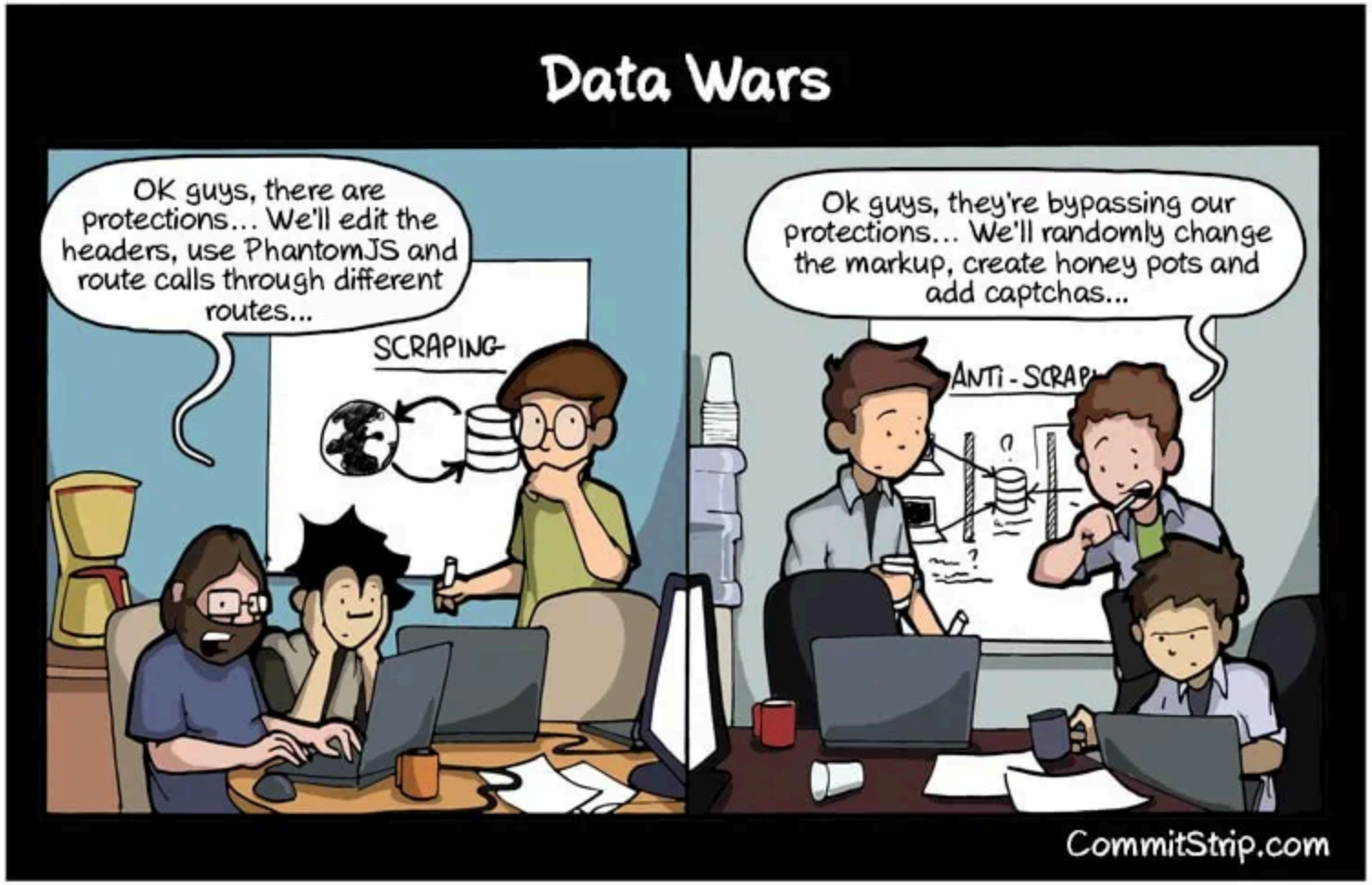
Exemple

```
import sqlite3

conn = sqlite3.connect('example.db')
c = conn.cursor()

c.execute('''CREATE TABLE stocks''')
c.execute("""
INSERT
INTO stocks
VALUES ('2006-01-05','BUY','RHAT',100,35.14)
""")
conn.commit()
conn.close()
```

Exemple d'architecture



TP : Web Scrapping



Selenium et Playwright

- Bibliothèques Python pour automatiser des actions dans un navigateur
- S'appuient sur un navigateur pour exécuter le code JavaScript
- Permet de simuler des actions de l'utilisateur
- Installation Selenium : `pip install selenium`
- Installation Playwright :
  - `pip install pytest-playwright`
  - `playwright install`

```python
from selenium import webdriver

driver = webdriver.Chrome()
driver.get("https://www.lemonde.fr/")
print(driver.title)
driver.quit()
```

```python
from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    browser = p.chromium.launch()
    page = browser.new_page()
    page.goto('https://www.lemonde.fr/')
    print(page.title())
    browser.close()
```

TP : Scrapping avec Selenium

Scrapy

- Framework Python pour extraire des données de pages web
- Permet de créer des spiders pour extraire des données
- Automatise la navigation et l'extraction de données
- Cas d'usage:
  - Récupérer des données sur plusieurs pages
  - Récupérer des données sur des sites plus complexes
- Installation : `pip install scrapy`
- Documentation

Exemple

```
scrapy startproject myproject
cd myproject
scrapy genspider example example.com
```