Webscraping w/ Python

Webscraping & API

Sarra Ben Yahia | José Ángel García Sánchez

Introduction

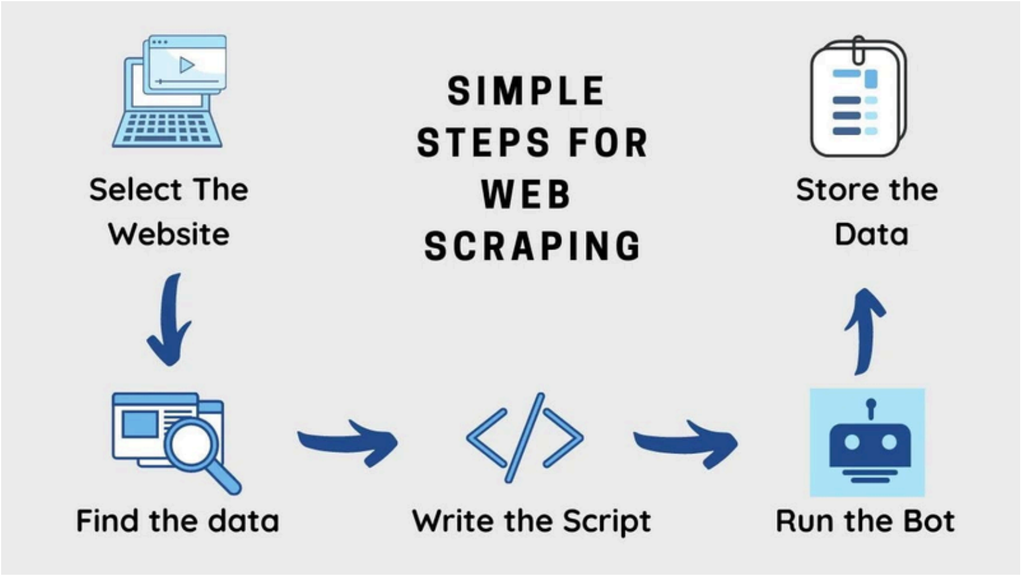
Definition

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites.

Web scraping software may directly access the World Wide Web using the **Hypertext Transfer Protocol (HTTP)** or a **web browser**.

The term typically refers to **automated processes implemented using a bot or web crawler**. It is a form of copying in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis.

Example



Webscraping with Python

Several Python frameworks webscrap

BeautifulSoup Request/BeautifulSoup

Scrapy

Selenium

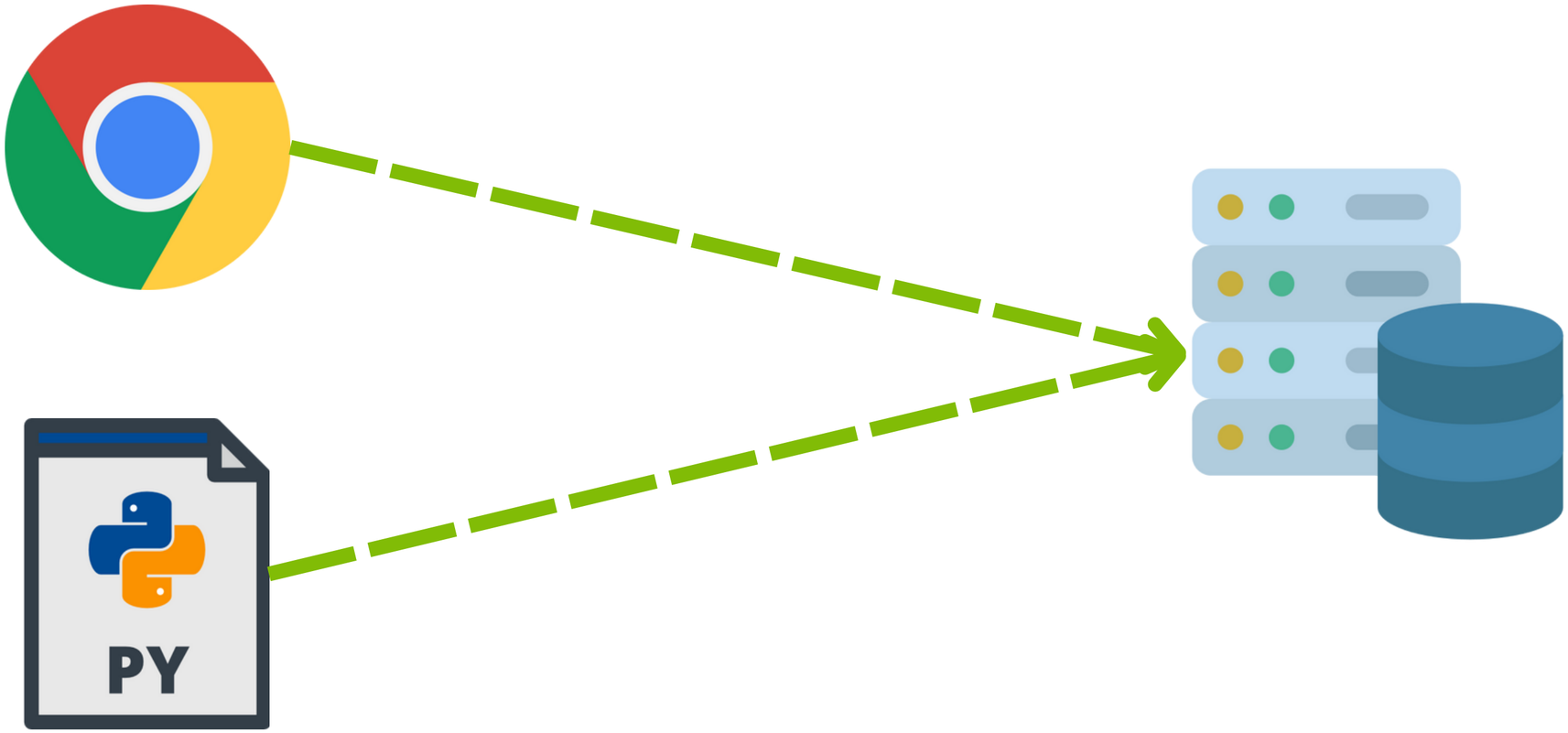
Request and BeautifulSoup

DEMO IN PYTHON : BOOKS

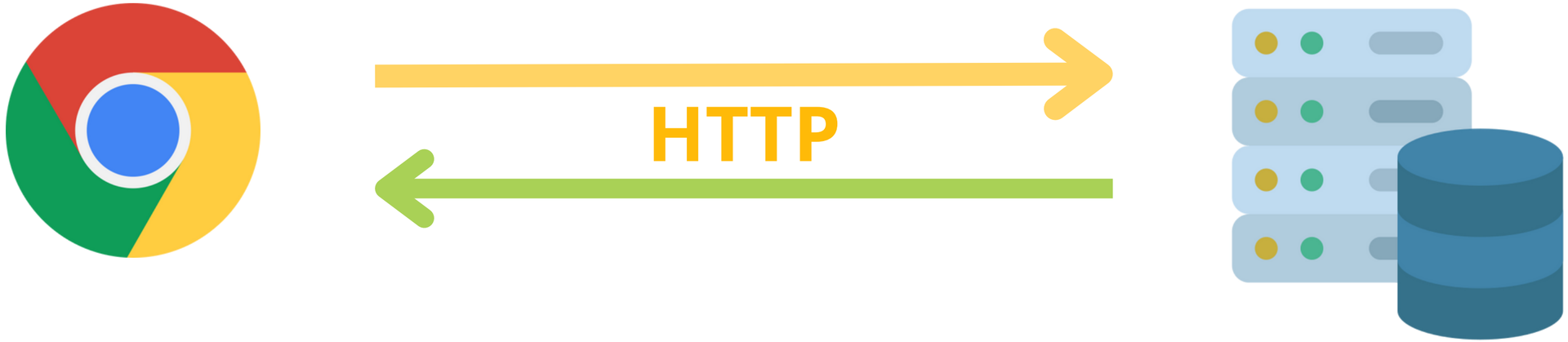
HTTPS://WWW.SCRAPETHISSITE.COM/PAGES/SIMPLE/

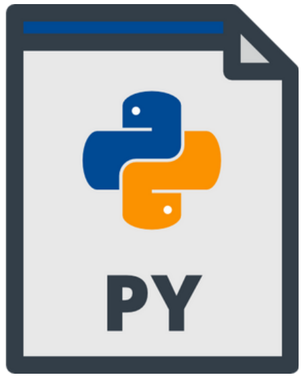
SCRAPE THIS WEBSITE AND SAVE RESULT IN EXCEL

Scraping Protections



Scraping Protections

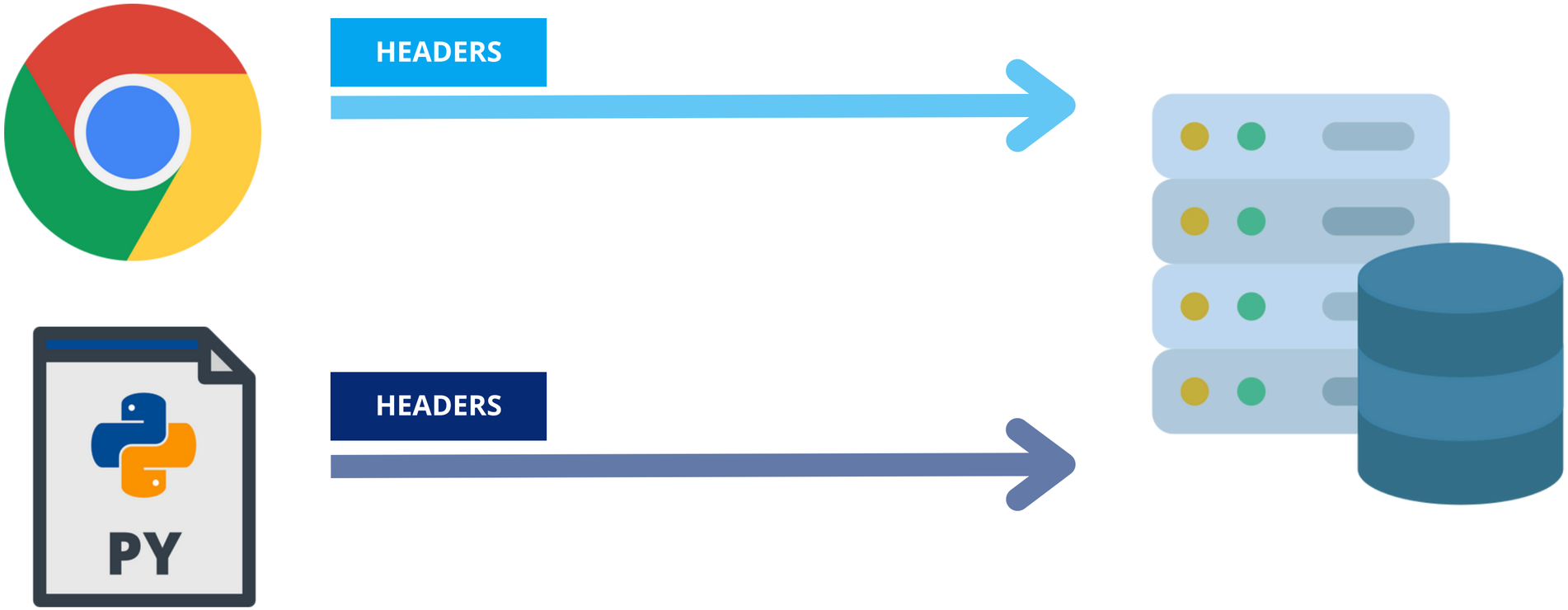




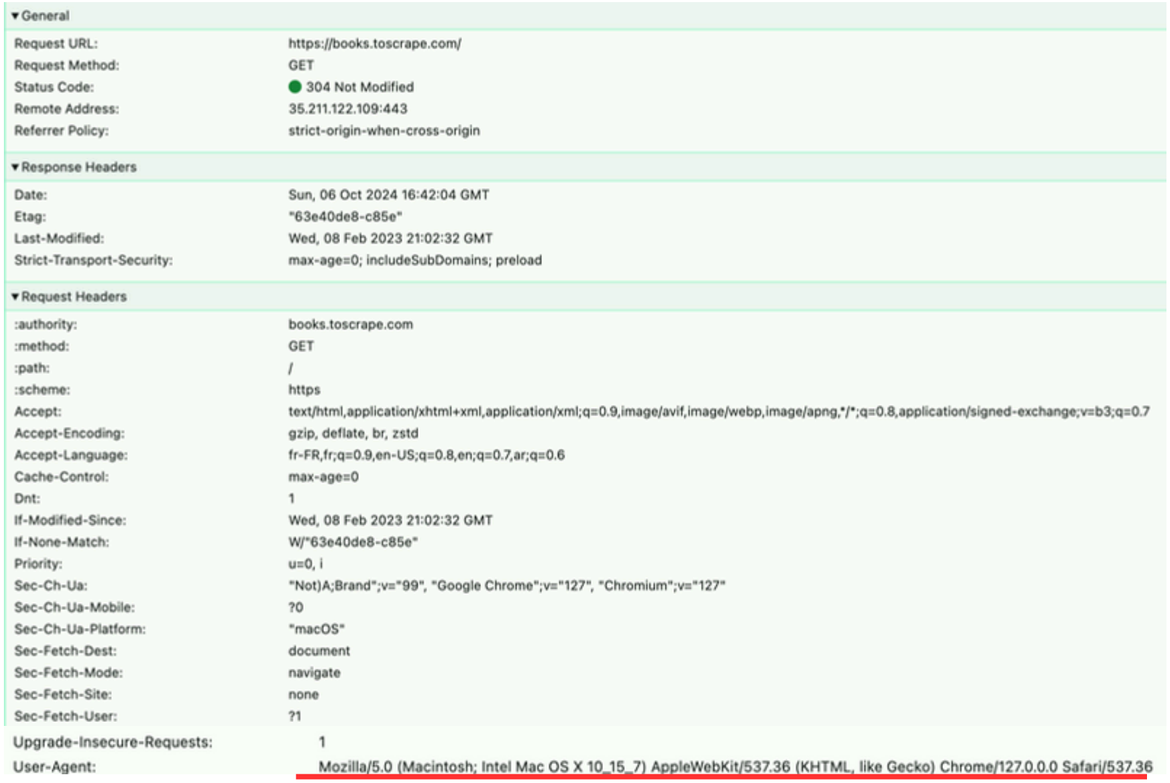
Principal Scraping Protections:

- **HTTP Headers**
- **IP/Proxys**
- **Javascript**

HTTP Headers



HTTP Headers example



▾ General

Request URL:	https://books.toscrape.com/
Request Method:	GET
Status Code:	304 Not Modified
Remote Address:	35.211.122.109:443
Referrer Policy:	strict-origin-when-cross-origin

▾ Response Headers

Date:	Sun, 06 Oct 2024 16:42:04 GMT
Etag:	"63e40de8-c85e"
Last-Modified:	Wed, 08 Feb 2023 21:02:32 GMT
Strict-Transport-Security:	max-age=0; includeSubDomains; preload

▾ Request Headers

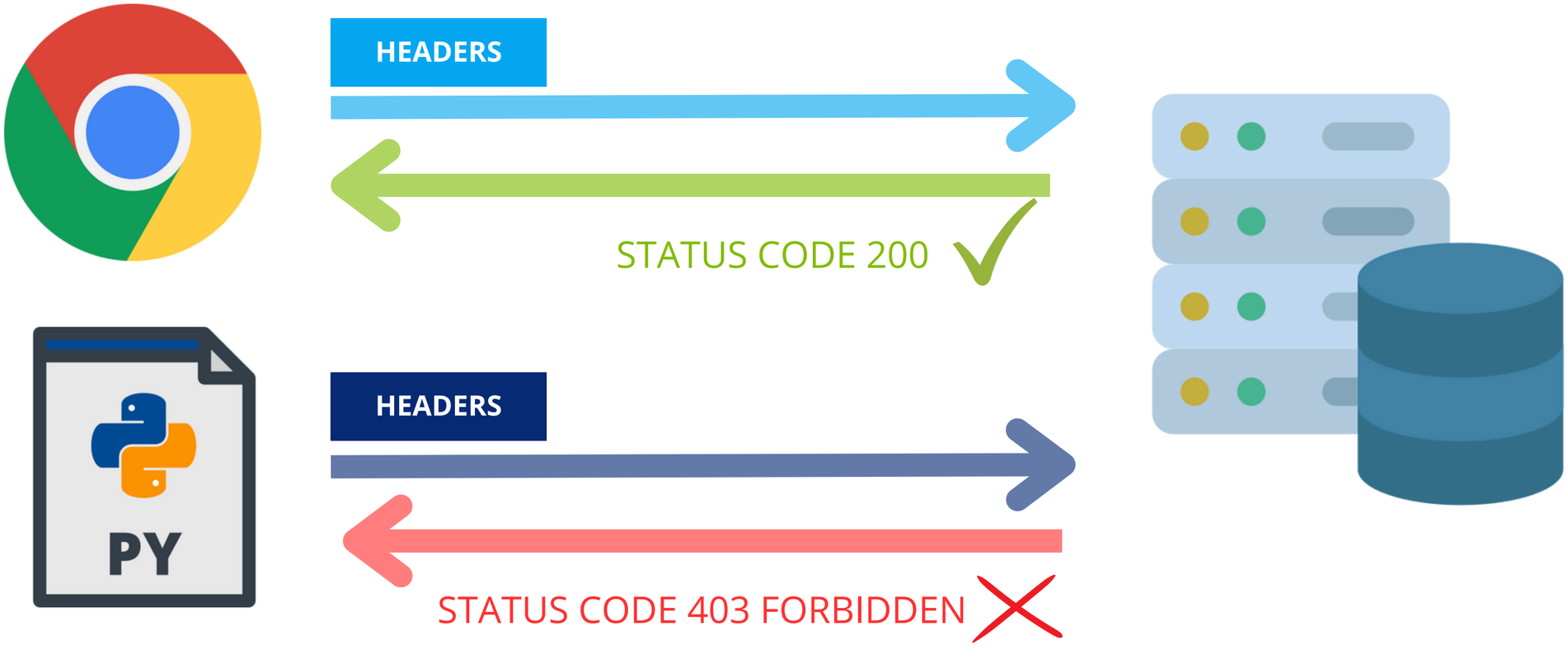
:authority:	books.toscrape.com
:method:	GET
:path:	/
:scheme:	https
Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding:	gzip, deflate, br, zstd
Accept-Language:	fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7,ar;q=0.6
Cache-Control:	max-age=0
Dnt:	1
If-Modified-Since:	Wed, 08 Feb 2023 21:02:32 GMT
If-None-Match:	W/"63e40de8-c85e"
Priority:	u=0, i
Sec-Ch-Ua:	"Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
Sec-Ch-Ua-Mobile:	?0
Sec-Ch-Ua-Platform:	"macOS"
Sec-Fetch-Dest:	document
Sec-Fetch-Mode:	navigate
Sec-Fetch-Site:	none
Sec-Fetch-User:	?1
Upgrade-Insecure-Requests:	1
User-Agent:	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36

Navigator

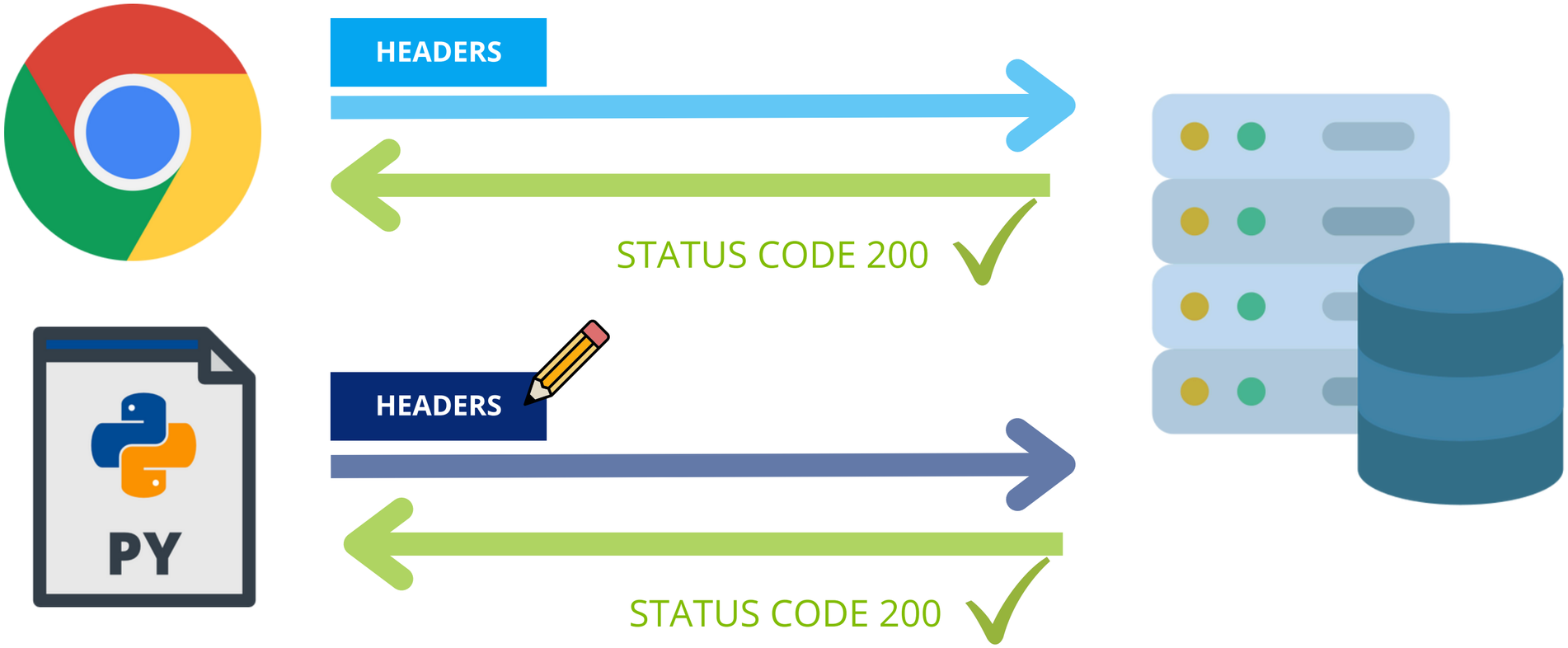
Script

```
_store: OrderedDict([('user-agent', ('User-Agent', 'pytho
  special variables
  function variables
  'user-agent': ('User-Agent', 'python-requests/2.28.1')
  'accept-encoding': ('Accept-Encoding', 'gzip, deflate')
  'accept': ('Accept', '*/*')
  'connection': ('Connection', 'keep-alive')
  len(): 4
```

HTTP Headers



HTTP Headers



Scraping w/ User-Agents

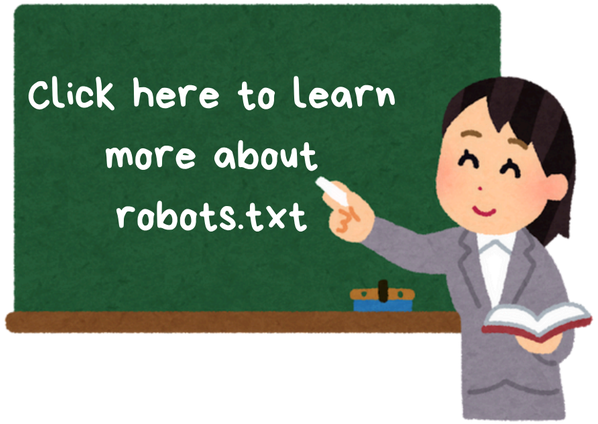
DEMO IN PYTHON : IMDB

```
er-agent: *
allow: /OnThisDay
allow: /ads/
allow: /ap/
allow: /mymovies/
allow: /r/
allow: /register
allow: /registration/
allow: /search/name-text
allow: /search/title-text
allow: /find
allow: /find$
allow: /find/
allow: /tvschedule
allow: /updates
allow: /watch/_ajax/option
allow: /_json/video/mon
allow: /_json/getAdsForMediaViewer/
allow: /list/ls*/_ajax
allow: /list/ls*/export
allow: /*/*/rg*/mediaviewer/rm*/tr
allow: /*/rg*/mediaviewer/rm*/tr
allow: /*/mediaviewer/*/tr
allow: /title/tt*/mediaviewer/rm*/tr
allow: /name/nm*/mediaviewer/rm*/tr
allow: /gallery/rg*/mediaviewer/rm*/tr
allow: /tr/
allow: /title/tt*/watchoptions
allow: /search/title/?title_type=feature,tv_mov                    ies,documentary,short,video,tv_short&release_date=,2020-12-
*lists=%21ls538187658,%21ls539867036,%21ls538186                    &sort=num_votes,asc&aft
allow: /name/nm*/filmotype/*
allow: /user/ur*/ratings
allow: /user/ur*/checkins
allow: /_json/*

er-agent: Baiduspider
allow: /list/*
allow: /user/*

er-agent: bingbot
```

In the robots.txt file of IMDB, we see that there are restrictions for crawlers



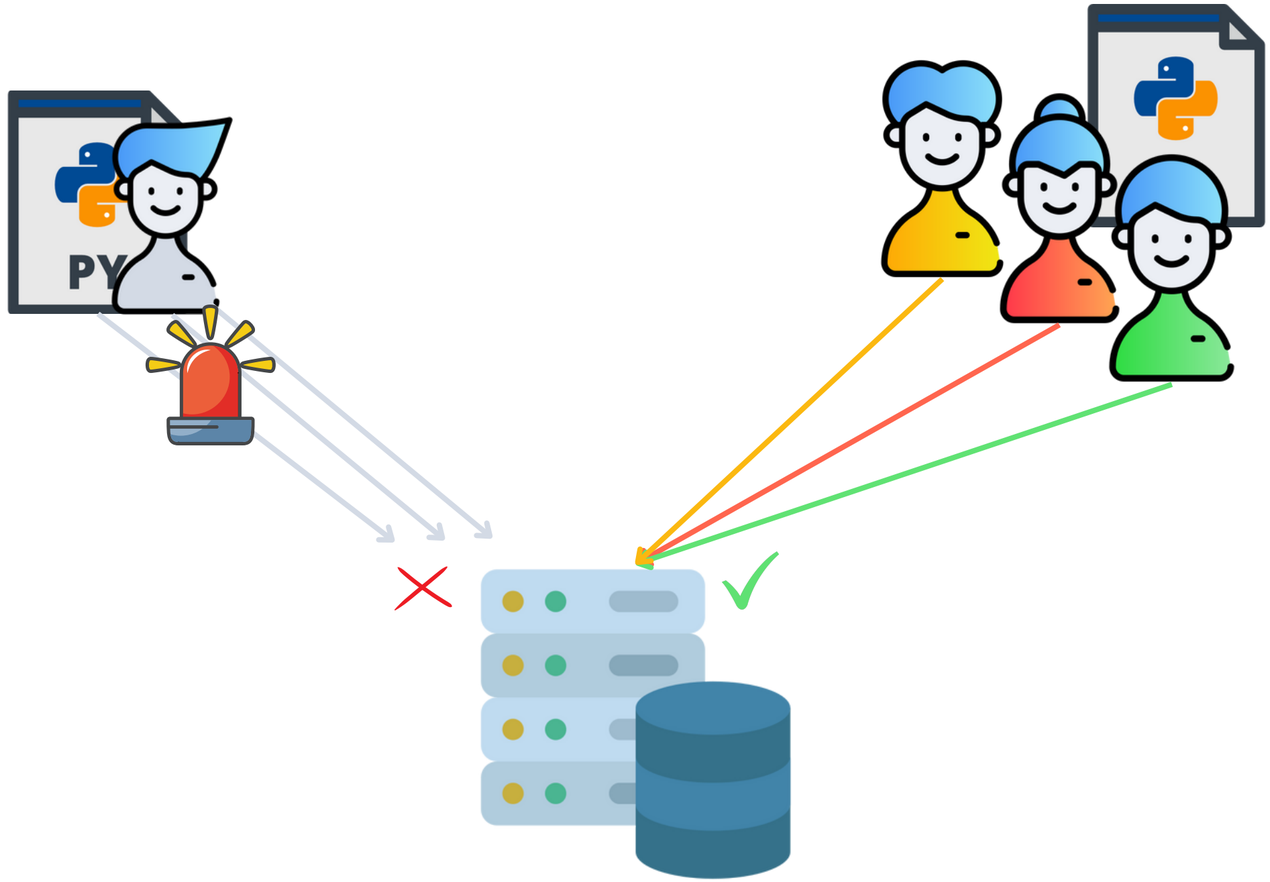
User-Agents Rotation

Why ?

Repeated requests from a **single** user-agent **can trigger server alerts.**

Rotating user-agents makes requests **appear to come from different sources.**

This helps **delay or avoid detection** by server security measures.



User-Agents Rotation

DEMO IN PYTHON : IMDB

HTTPS://WWW.SCRAPETHISSITE.COM/PAGES/ADVANCED/?GOTCHA=HEADERS

SCRAPE THIS WEBSITE BY SPOOFING HEADERS

IP/Proxys limitations

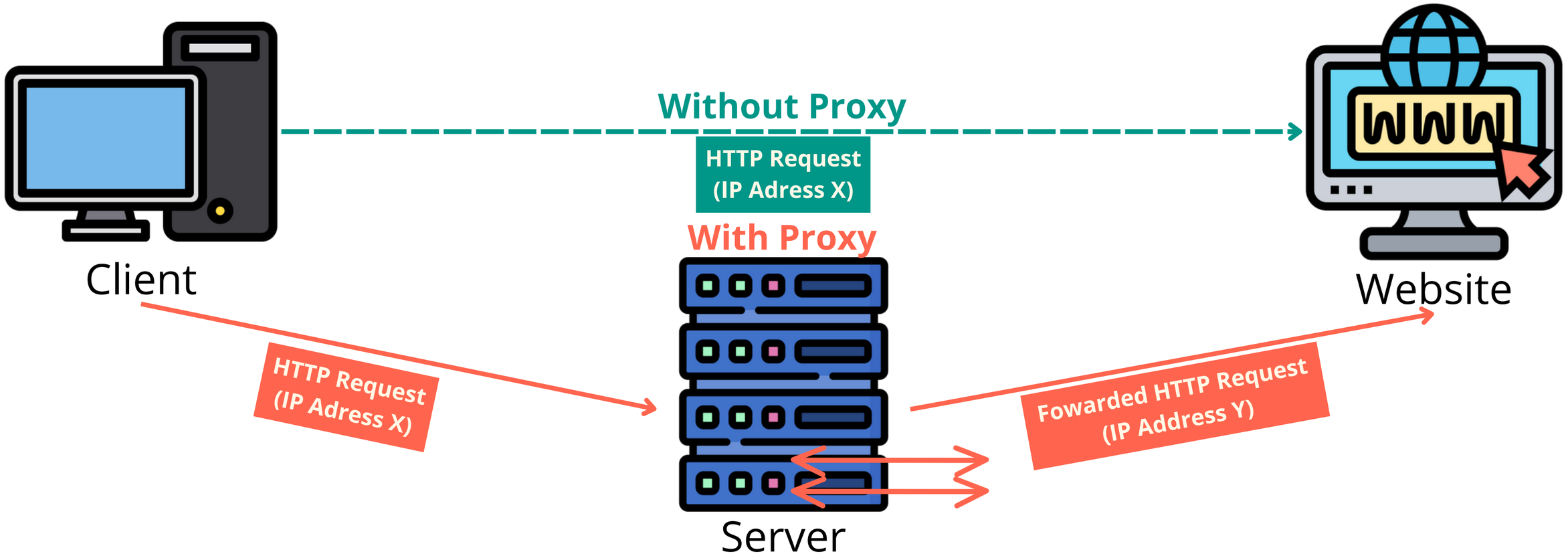
An **Internet Protocol address** (IP address) is a numerical label such as 192.0.2.1 that is assigned to a device connected to a computer network that uses the Internet Protocol for communication.

IP addresses serve two main functions: **network interface identification**, and **location addressing**.

Click to check your IP adress

IP/Proxys limitations

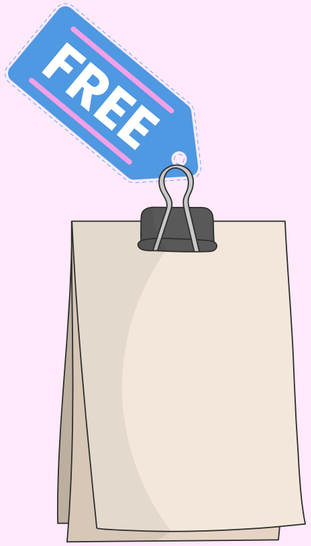
A proxy is an **intermediary server** that **forwards requests and responses** between a client and the internet, providing various functions such as **anonymity and content filtering**.



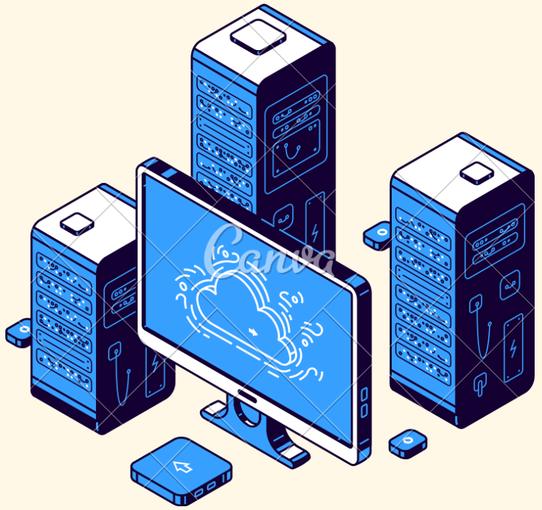
How to get a list of proxies ?

option1

option2

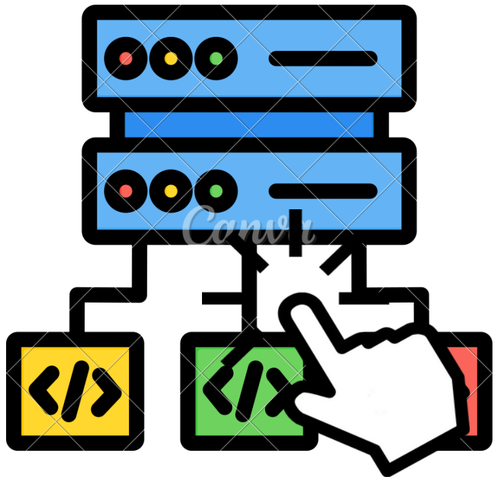


option3



IP adress rotation

DEMO IN PYTHON

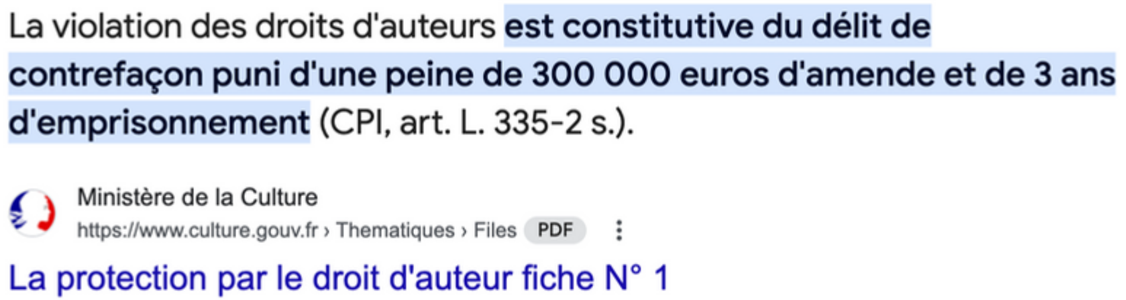


Legality: Do’s and Don’ts

- **Terms of Service (ToS):** Scraping may violate a website's ToS, leading to potential legal consequences.
- **Public vs. Private Data:** Scraping public data is usually more acceptable, but private/protected data can lead to legal issues.
- **Copyright:** Scraping copyrighted content without permission can infringe intellectual property rights.
- **GDPR (Europe):** Scraping personal data must comply with GDPR regulations.
- **Court Rulings:** Legal cases have led to mixed results; public data scraping has sometimes been ruled legal (e.g., HiQ Labs v. LinkedIn).

Good Practice:

- Always check and follow a **website’s ToS.**
- Respect **robots.txt files** and avoid scraping **private or protected data.**
- Ensure **compliance with GDPR** or other privacy laws **if scraping personal data.**
- Use **scraping ethically and avoid overloading websites** to prevent harm or misuse.



End of 1st lesson

Introduction to scrapy

- What is **Scrapy**? → A **fast, open-source web crawling framework** for Python.
- **Key Features:**
  - **Efficient**: Handles large-scale scraping with built-in mechanisms for speed and data integrity.
  - **Robust**: Built-in support for handling retries, redirects, and handling of cookies.
  - **Asynchronous**: Uses Twisted for non-blocking requests, making it capable of scraping multiple pages concurrently.

- Core Concepts:
  - **Spiders**: Define how to crawl and extract data from websites.
  - **Items**: Represent the data scraped from the pages.
  - **Pipelines**: Post-process and store scraped data (e.g., cleaning, saving to databases).
  - **Middlewares**: Customize request and response processing (e.g., handling proxies, user agents).
  - **Settings**: Control the behavior of Scrapy (e.g., request delays, concurrency limits).

Introduction to scrapy

DEMO IN PYTHON : SCRAPY PROJECT

Structure of scrapy projects

Script	Function	Example of use
scrapy.cfg	Global configuration of the project	Define the default settings module, configure deployment
settings.py	Project configuration parameters	Set bot name, configure robots.txt obedience, set delays between requests
items.py	Definition of data structures for scraped items	Create classes for products with fields like title, price, description
pipelines.py	Processing of extracted data	Clean data, convert types (e.g., string to float for prices), store in a database
middlewares.py	Customization of request/response behavior	User-Agent rotation, proxy management, modification of HTTP headers
spiders/ (folder)	Contains spiders that define how to scrape sites	Create a spider to extract data from an e-commerce site, define navigation and extraction rules

HTTPS://WWW.SCRAPETHISSITE.COM/PAGES/AJAX-JAVASCRIPT/

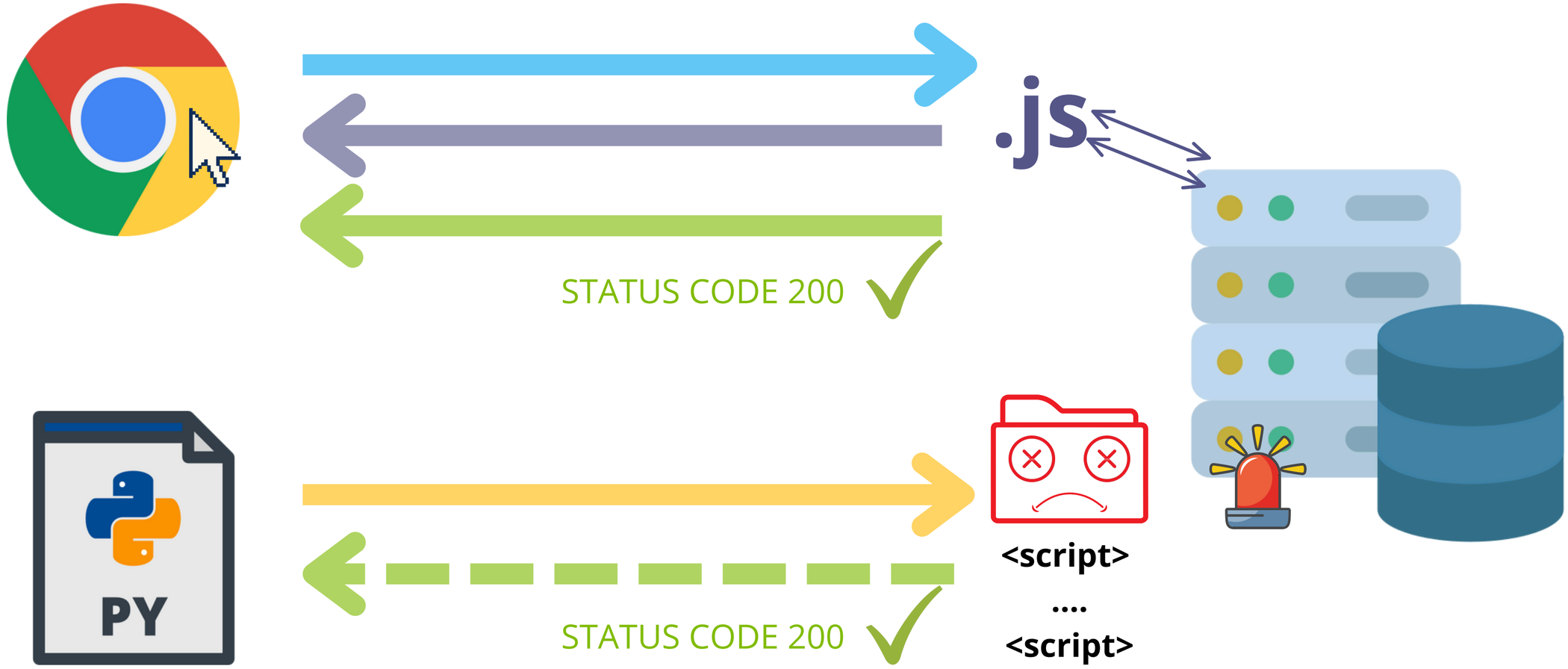
SCRAPE THIS WEBSITE

Principal terminal commands you'll need

Command	Description
scrapy startproject <nom_projet>	Create a new scrapy project
scrapy genspider <nom_spider> <domaine>	Create a new spider
scrapy crawl <nom_spider>	Execute a spider
scrapy shell <url>	Open an interactive Scrapy console to do some tests
scrapy list	List all the available spiders in your project
scrapy fetch <url>	Fetch the HTML of a page and print it in terminal
scrapy view <url>	Open the page in the navigator as Scrapy sees it
scrapy parse <url>	Parse the HTML page and print results
scrapy version	Print Scrapy's verison
scrapy bench	Small benchmark to evaluate perfomances

End of 2nd lesson

Javascript problems

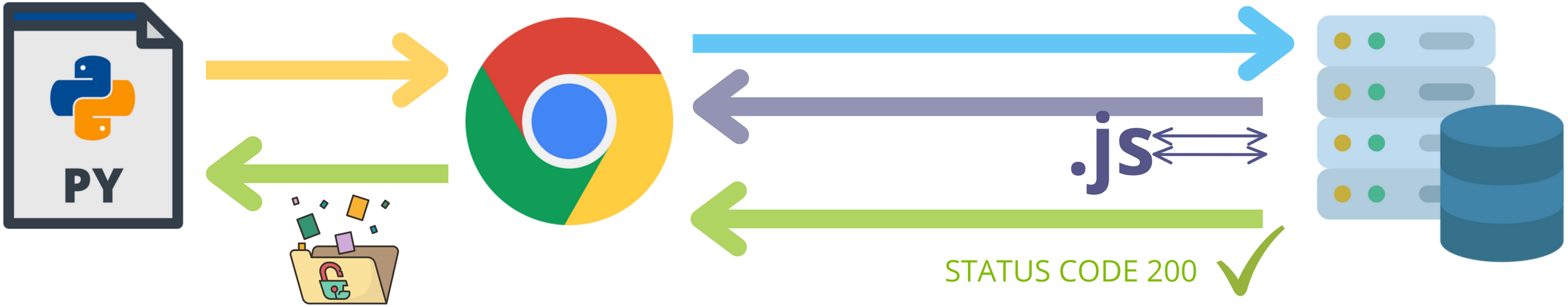


Introduction to selenium

- **What is Selenium?** → A powerful **web browser automation framework** that **simulates real user interactions**.
- Key Features:
  - **Browser Automation:** Controls web browsers programmatically across all major browsers.
  - **Dynamic Content:** Handles JavaScript-rendered content and dynamic page updates.
  - **Cross-Platform:** Works on multiple operating systems and browsers.
  - **Language Support:** Available in multiple programming languages (Python, Java, C#, etc.).

- **Core Concepts:**
  - **WebDriver**: The core component that controls browser actions and interactions.
  - **Elements**: Represent HTML elements on the page that can be interacted with.
  - **Locators**: Methods to find elements (ID, CSS, XPath, etc.).
  - **Waits**: Mechanisms to handle dynamic content loading:
  - **Implicit Waits**: Global timeout for finding elements
  - **Explicit Waits**: Conditional waiting for specific elements/states

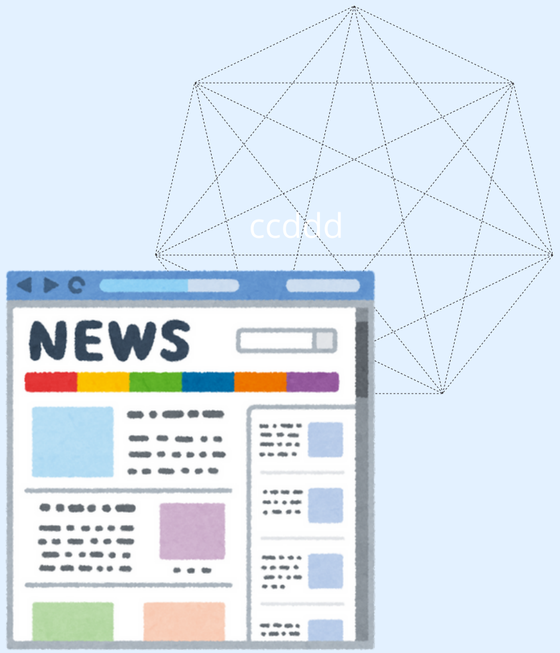
Selenium against JS problems



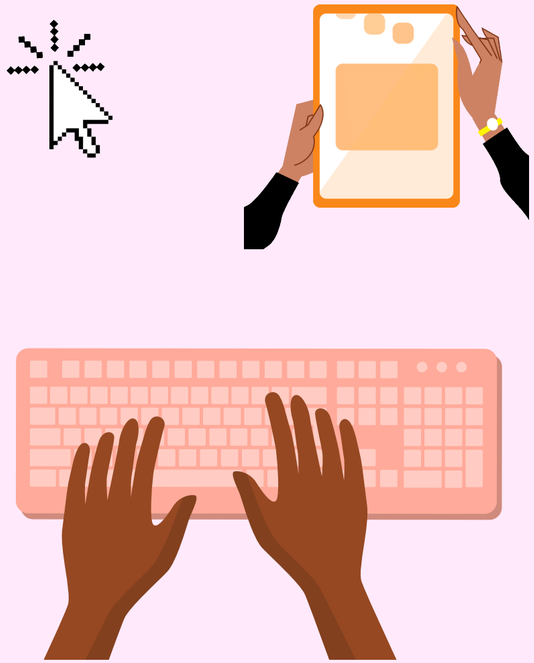
- **Executes JavaScript**, loading dynamic content.
- Captures full content **after JS is rendered**.
- Mimics **browser behavior**, avoiding bot detection.
- **Simulates user interactions** (mouse, scroll, clicks).

What can you make a webdriver do automatically?

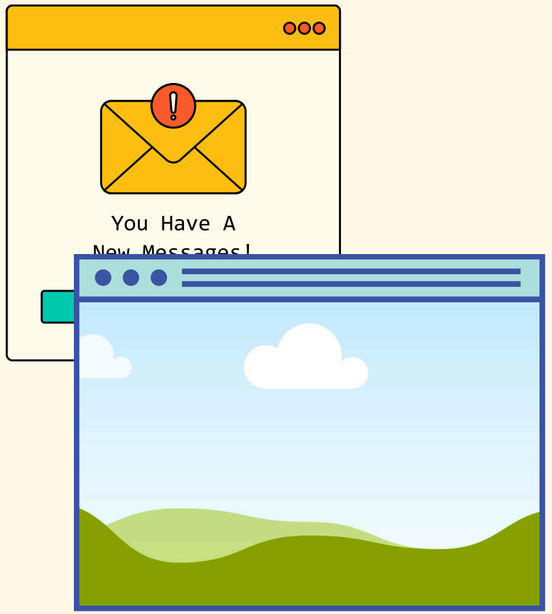
Open URLs, refresh, back, forward...



Click, type, scroll ...



Handle alerts, frames, windows...



And others...

Selenium Introduction

DEMO IN PYTHON

HTTPS://GITHUB.COM/SARRABENYAHIA/TUTO-WEBSCRAPING

WEBSCRAP THE README OF THE OUR REPO

End of lesson!

Sources

Github repository :

GitHub

benyahiasarra9@gmail.com

Complementary course material:

be curious!