

# Web Scraping



Gustavo Juantorena [[github.com/GEJ1](https://github.com/GEJ1)]

# Hoja de ruta

1. ¿Qué es el web scraping?
2. ¿Qué son las APIs?
3. ¿Cuándo conviene usar cada cosa?
4. Breve introducción a las tecnologías web.
5. Web Scraping con Python.



# ¿Qué es el web scraping?

***La práctica de recopilar datos mediante un programa automatizado que consulta un servidor web***

# ¿Qué es el web scraping?

*La práctica de recopilar datos mediante un programa automatizado que consulta un servidor web*

*El **web crawling** o indexación, se utiliza para indexar la información de la página mediante bots*

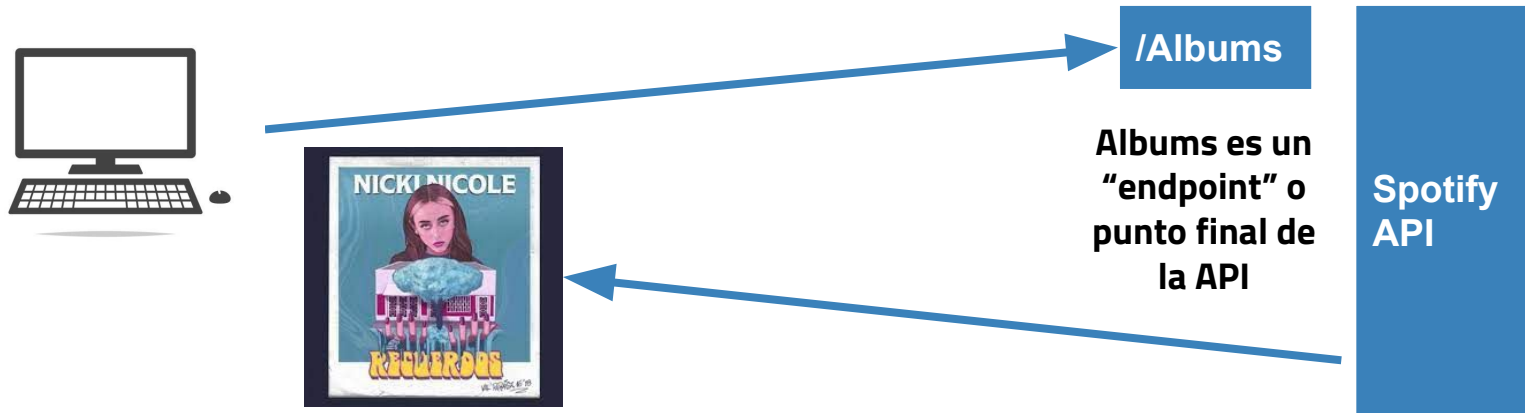
*Un bot rastrea un sitio web, recorre todas las páginas y todos los enlaces, hasta la última línea del sitio web*

# Antes de empezar ⚠️

1. ¿Lo que estoy haciendo es legal en mi país? ¿Puedo estar afectando al servidor?
2. ¿Es necesario utilizar web scraping o se podría haber resuelto de manera más sencilla?

# API (Interfaz de programación de aplicaciones)

Conjunto de funciones, métodos y protocolos de comunicación que ofrece un programa para ser utilizado por otro.



Nos provee el contenido con un determinado formato y especifica las limitaciones (ej: cantidad de pedidos al servidor).

# Ejemplos de APIs



**Developer Platform**

Products ▾Use cases ▾Docs ▾Community ▾

Updates ▾Support ▾

Documentation

Search the docs

Home > Twitter API

Getting started ▾TutorialsTools and librariesMigrateAPI reference index

Introducing the new Twitter API v2 Fundamentals ▾Tweets ▾Users ▾

## API reference index

This resource is the Twitter API-specific API reference index. If you are looking for a list of endpoints from the entire platform, including Twitter Ads API and Labs, please visit the [platform API Reference Index](#).

## Twitter API v2

### Tweets

#### Filtered stream

- GET /2/tweets/search/stream
- GET /2/tweets/search/stream/rules
- POST /2/tweets/search/stream/rules

#### Hide replies

- PUT /2/tweets/:id/hidden

#### Likes

- DELETE /2/users/:id/likes/tweet\_id
- GET /2/tweets/:id/liking\_users



**WIKIPEDIA**  
*The Free Encyclopedia*

## Wikipedia API

Wikipedia-API is easy to use Python wrapper for [Wikipedia's](#) API. It supports extracting texts, sections, links, categories, translations, etc from Wikipedia. Documentation provides code snippets for the most common use cases.

build passing docs passing test coverage 96% python v0.5.4 Py Versions Stars 255

### Installation

This package requires at least Python 3.4 to install because it's using IntEnum.

```
pip3 install wikipedia-api
```

### Usage

Goal of `Wikipedia-API` is to provide simple and easy to use API for retrieving informations from Wikipedia. Below are examples of common use cases.

### Importing

```
import wikipediaapi
```

### How To Get Single Page

Getting single page is straightforward. You have to initialize `Wikipedia` object and ask for page by its name. It's parameter language has be one of [supported languages](#).

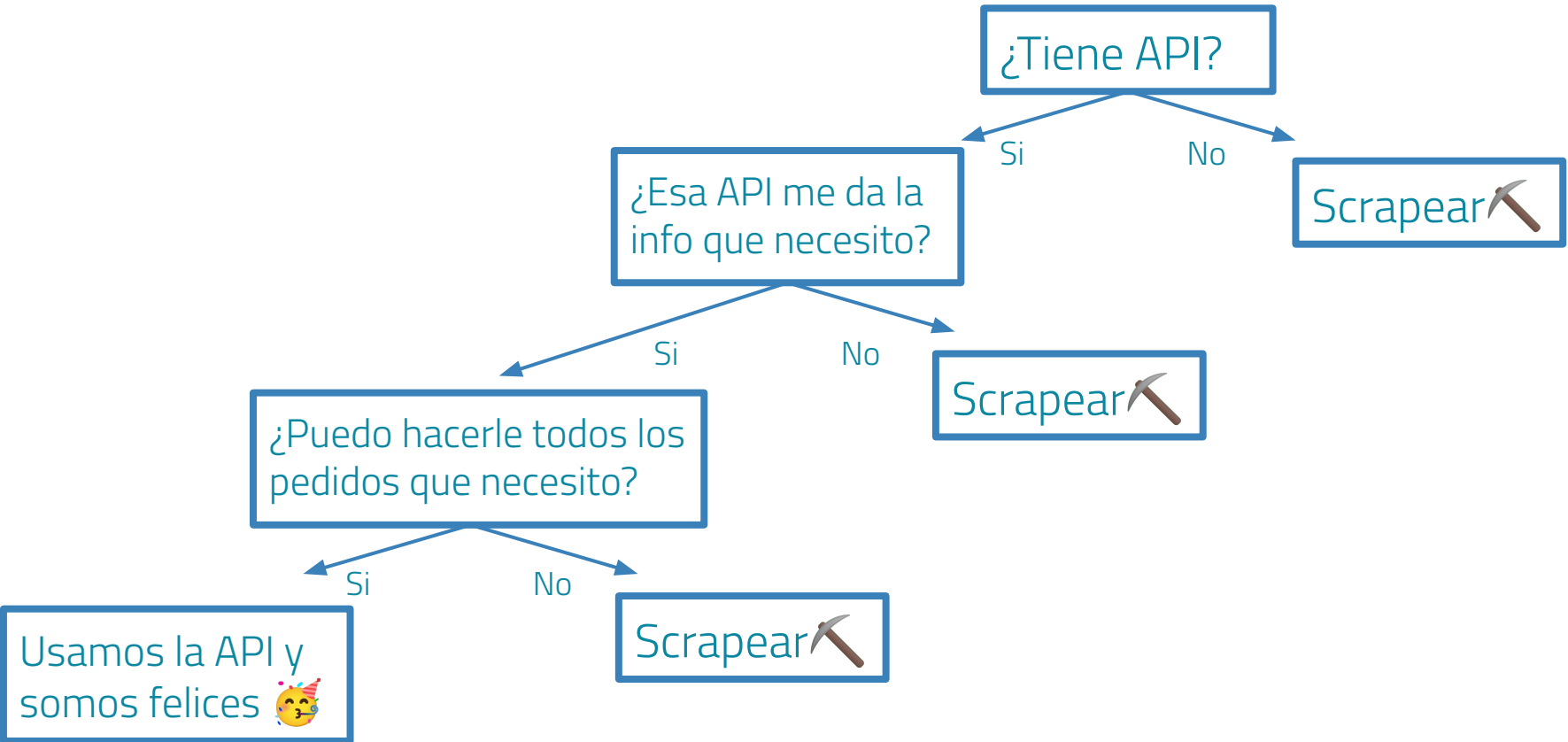
```
import wikipediaapi
wiki_wiki = wikipediaapi.Wikipedia('en')
page_py = wiki_wiki.page('Python_(programming_language)')
```

# API vs Web scraping (I)

Web scraping	API
Extraer información de un sitio web usando un programa informático.	Proveer acceso a los datos de una aplicación, sistema operativo u otro servicio.
Mismo objetivo: Acceder a los datos del sitio web	

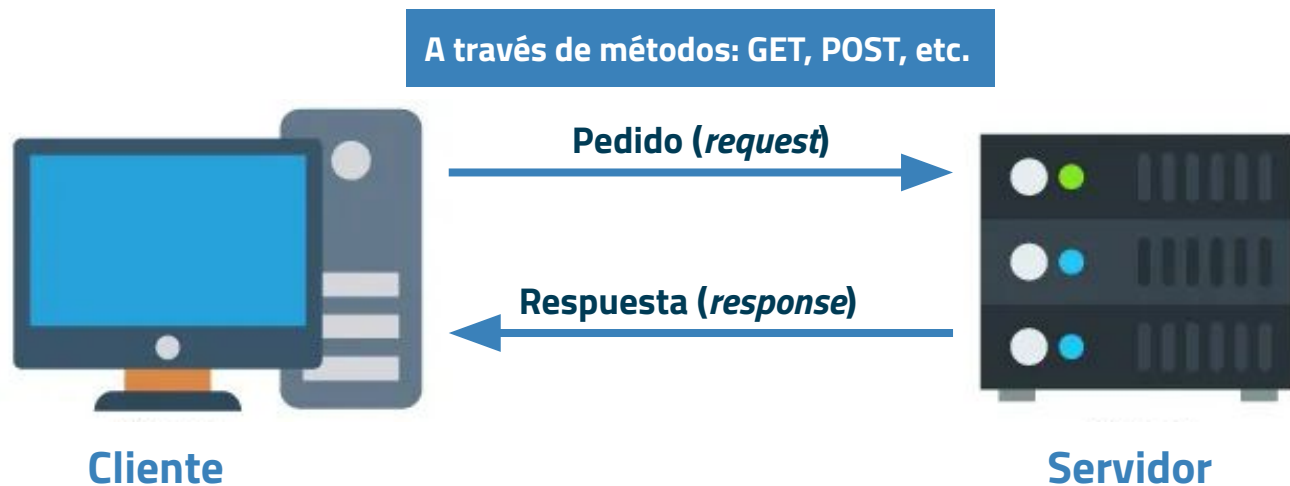


# API vs Web scraping (II)



# Conceptos básicos sobre la web (I)

- **HTTPS** (Protocolo seguro de transferencia de hipertexto): Protocolo de mensajería que permite a los navegadores web comunicarse con los servidores web (donde se almacenan los sitios web).



# Conceptos básicos sobre la web (II)

HTTP Status Codes		
Éxito	Errores del cliente	Errores del servidor
<b>Level 200 (Success)</b> 200 : OK 201 : Created 203 : Non-Authoritative Information 204 : No Content	<b>Level 400</b> 400 : Bad Request 401 : Unauthorized 403 : Forbidden 404 : Not Found 409 : Conflict <b>429: Too many requests</b>	<b>Level 500</b> 500 : Internal Server Error 503 : Service Unavailable 501 : Not Implemented 504 : Gateway Timeout 599 : Network timeout 502 : Bad Gateway

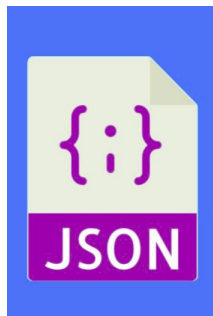
Fuente: <https://gabicuesta.blogspot.com/2019/01/http-status-codes.html>

# Conceptos básicos sobre la web (III)



comma-separated  
values

```
album, año, ranking  
The White Stripes, 1999, 2  
De Stijl, 2000, 3  
Lo mejor del amor, 1996, 1
```



JavaScript  
Object  
Notation

```
[  
  {  
    "album": "The White Stripes",  
    "año": 1999,  
    "ranking": 2  
  },  
  {  
    "album": "De Stijl",  
    "año": 2000,  
    "ranking": 3  
  },  
  {  
    "album": "Lo mejor del amor",  
    "año": 1996,  
    "ranking": 1  
  }  
]
```

<https://csvjson.com/csv2json>

# Conceptos básicos sobre la web (III)

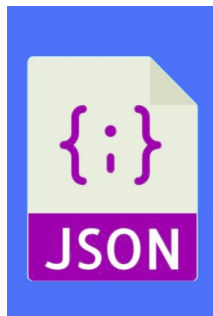


comma-separated  
values

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <element>
    <album>The White Stripes</album>
    <year number="true">1999</year>
    <ranking number="true">2</ranking>
  </element>
  <element>
    <album>De Stijl</album>
    <year number="true">2000</year>
    <ranking number="true">3</ranking>
  </element>
  <element>
    <album>Lo mejor del amor</album>
    <year number="true">1996</year>
    <ranking number="true">1</ranking>
  </element>
</root>
```

```
album, año, ranking
The White Stripes,
De Stijl, 2000, 3
Lo mejor del amor,
```



Javascript  
Object  
Notation

```
"The White Stripes",
1999,
": 2

"De Stijl",
2000,
": 3

"Lo mejor del amor",
1996,
": 1
```

<https://csvjson.com/csv2json>

# Conceptos básicos sobre la web (IV)

- **HTML, CSS y JavaScript son los tres lenguajes principales con los que está hecho la parte de la web que vemos.**



# Conceptos básicos sobre la web (IV)

- HTML, CSS y JavaScript son los 3 lenguajes principales con los que está hecho la parte de la web que vemos.



Estructura



Funcionalidad



Estilo



# HTML: Lenguaje de marcado de hipertexto

## CSS: Hojas de estilo en cascada

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi primer página</title>
  </head>
  <body>

    <h1>Título</h1>
    <h2 style='color:red;'> Subtitulo en rojo</h2>
    <p>Primer párrafo.</p>
    <hr>

    <h3>Gatito</h3>
    <img style='width: 100px;'
src='https://i.pinimg.com/originals/d4/8d/07/d48d074f8c
9ec8448612822295686754.jpg' >

  </body>
</html>
```

## Título

### Subtitulo en rojo

Primer párrafo.

---

### Gatito





# HTML: Lenguaje de marcado de hipertexto

## CSS: Hojas de estilo en cascada

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi primer página</title>
  </head>
  <body>

    <h1>Título</h1>
    <h2 style='color:red;'> Subtitulo en rojo</h2>
    <p>Primer párrafo.</p>
    <hr>

    <h3>Gatito</h3>
    <img style='width: 100px;'
src='https://i.pinimg.com/originals/d4/8d/07/d48d074f8c
9ec8448612822295686754.jpg' >

  </body>
</html>
```

HTML TAG

CSS

HTML ATTRIBUTE

## Título

### Subtitulo en rojo

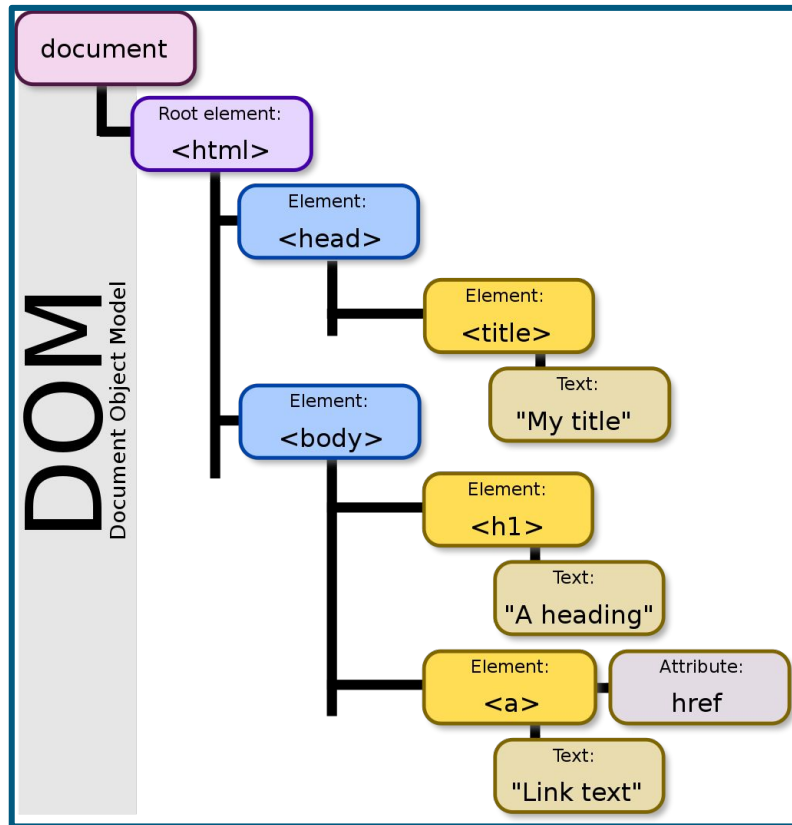
Primer párrafo.

### Gatito



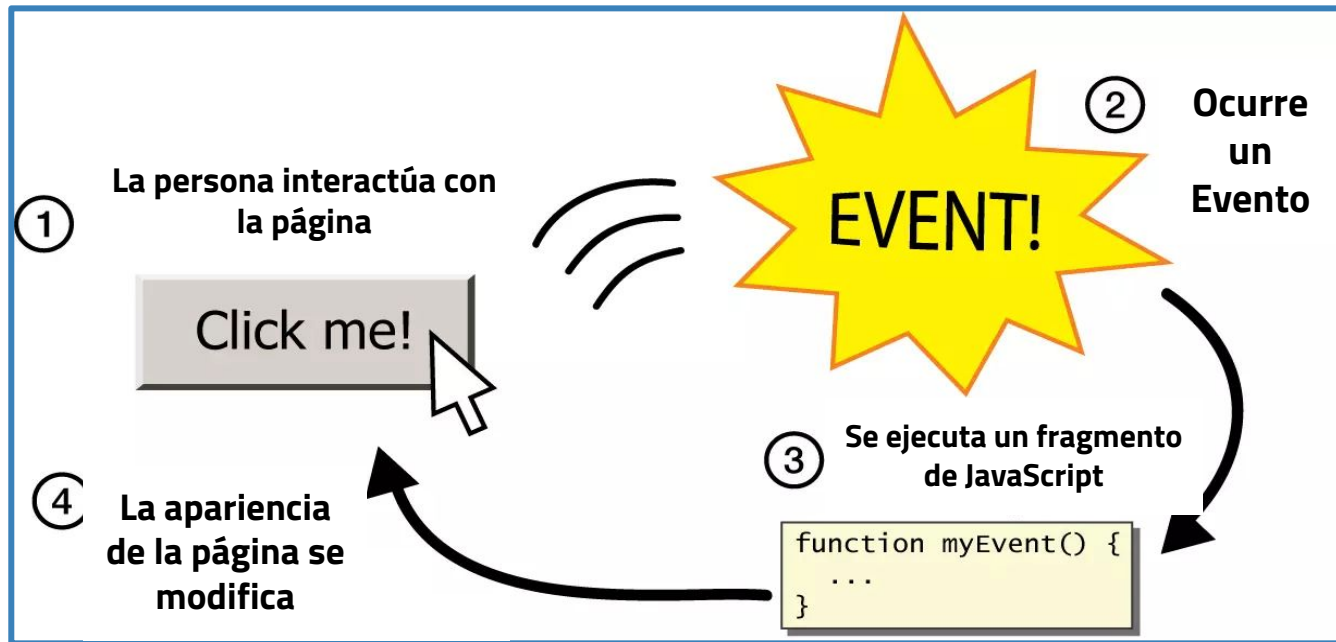
# DOM (Document Object Model)

Interfaz independiente del lenguaje que trata un documento XML o HTML como una estructura de árbol



# JavaScript

- Permite usar un paradigma de programación orientado a eventos (entre otros).
- Más del 97% de las páginas web lo usan para generar comportamientos interactivos.



# Uso de herramientas de desarrollador en el navegador

¡En todos los productos!

Aprovechá nuestras ofertas.

Comprar ahora

¿Por qué comprar con nosotros?

Truck icon, FREE, Target icon

Elements

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body data-new-gr-c-s-check-loaded="14.1101.0" data-gr-ext-installed>
    <div class="hero-area">...</div>
    <!-- why section -->
    <section class="why-section layout-padding">
      <div class="container">
        <div class="heading-container heading-center" id="about">
          <h2>¿Por qué comprar con nosotros?</h2>
          <div class="row">...</div>
        </div>
      </div>
    </section>
    <!-- end why section -->
  </body>
</html>
```

Styles

Computed Layout Event Listeners

Filter :hov .cls + - [4] ^

Console What's New x

Highlights from the Chrome 105 update

Step-by-step replay in the Recorder panel Set a

new

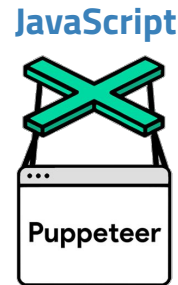
Practiquemos ...

¿¿¿Y Python???



# 🐍 Web scraping con Python

- Python no es la única tecnología con la cual podemos hacer scraping (ej: R, JavaScript, Java, C++, etc)

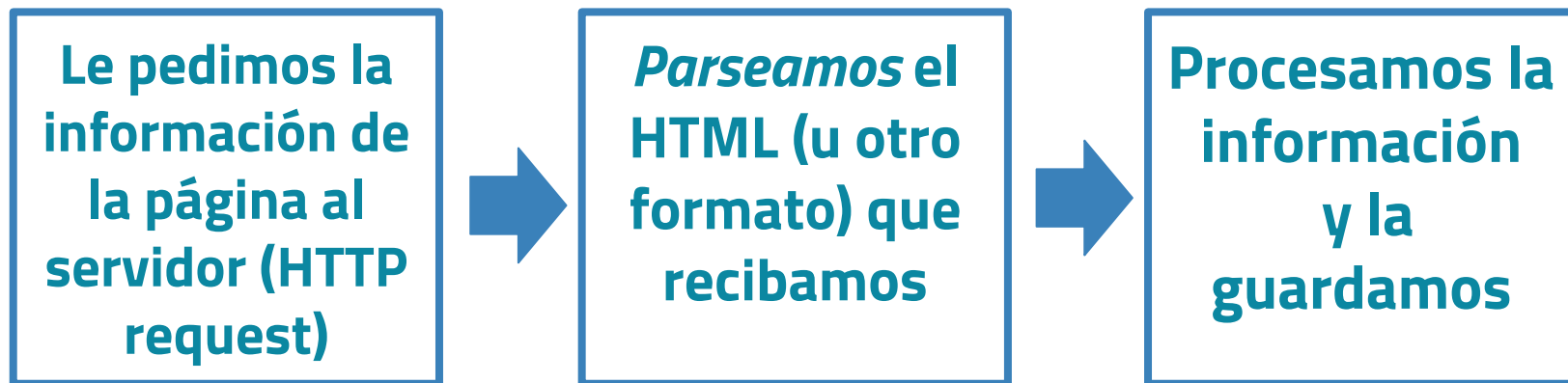


- Python nos puede ayudar en muchas cosas:
  - Pedidos HTTP (urllib, requests)
  - Parseo de la información (Beautiful Soup)
  - Automatización
  - Control de excepciones
  - etc

# “Parsear” la información

- Dividir un texto en sus componentes y describir sus roles sintácticos.
- El “parseo” de un documento HTML es básicamente tomar código HTML y extraer información relevante como el título de la página, párrafos, encabezados, enlaces, texto en negrita, etc.

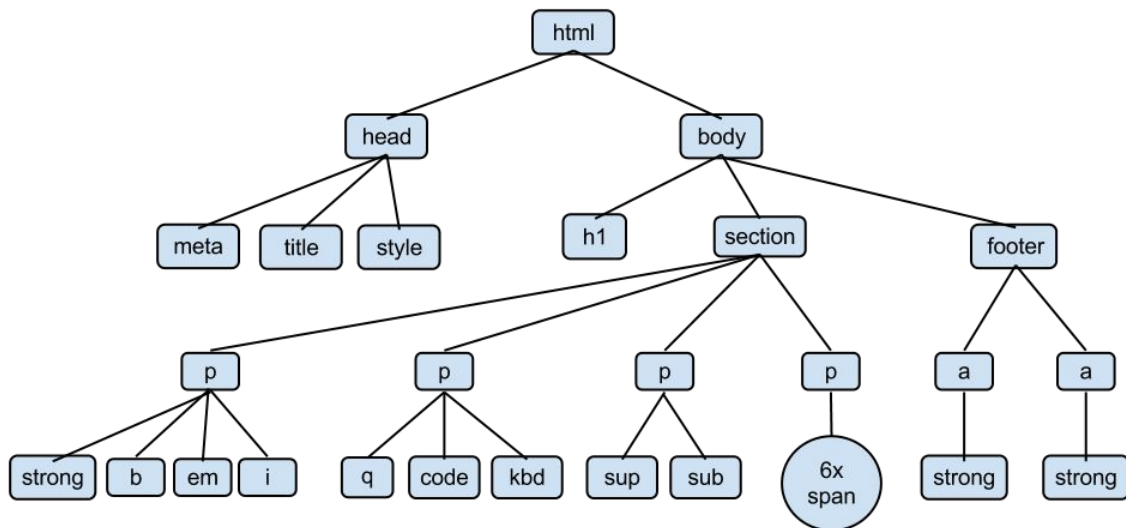
## Flujo de trabajo en *web scraping*





# Parseando con Python: BeautifulSoup

- BeautifulSoup es una librería de Python para web scraping
- Se usa para extraer los datos de archivos HTML y XML.
- Crea un árbol de análisis a partir del código fuente de la página



Fuente: <https://medium.com/miloo-project/python-simple-crawling-using-beautifulsoup-8247657c2de5>

**Ahora si: ¡Vamos a Colab!**

