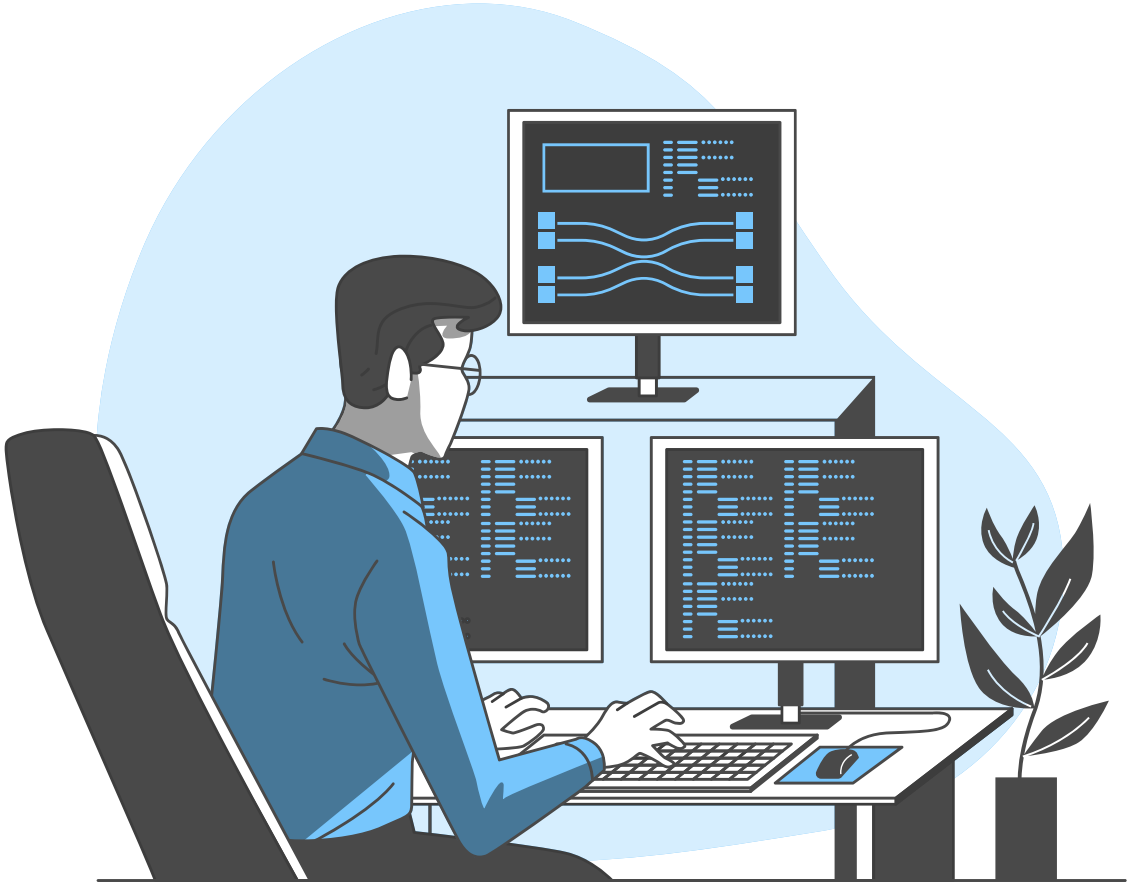


Web Scraping

by Alex

Agenda

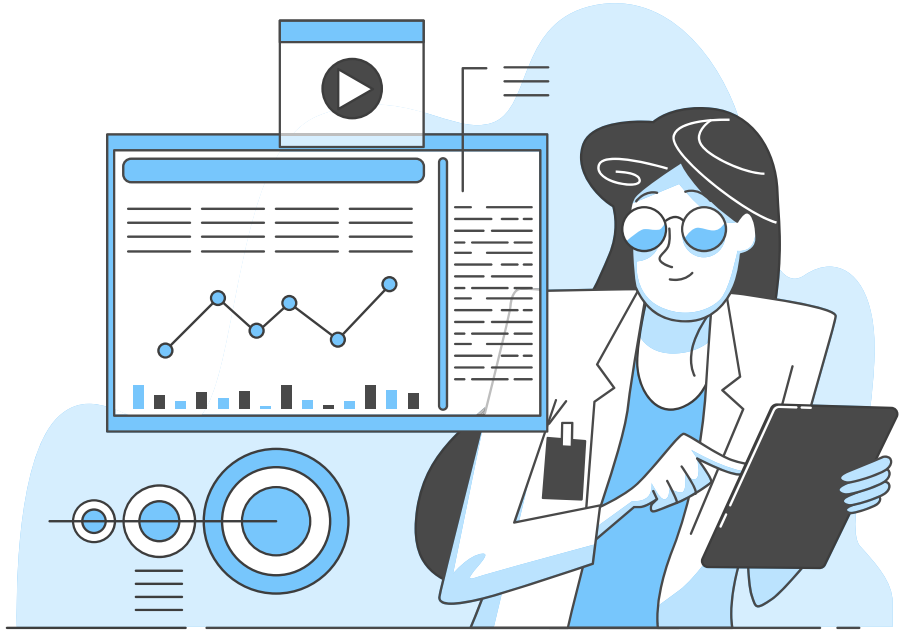
01 What is web scraping?

02 A bit about HTML and parsing

03 Web scraping with Python

04 More advanced uses

API and cloud scraping



01

What is web scraping?

What is web scraping?

Data extraction from the web

Use a HTTP
HyperText Transfer Protocol

Web crawling and web fetching

Webpages are written in HTML/XTML

Web scraping in a nutshell

You

Generate
Request intention

Absolute magic

Transfer request

The data

Explaining the magic

TCP (Transmission Control Protocol) Handshake

Establishing a server connection:

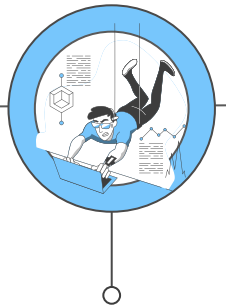
- Generate request to server (SYN)
- Server reception, followed by implicit key exchange if admitted (SYN - ACK)
- Key referral acknowledgement on user's end (ACK)
- End connection (FIN) or interrupt the session (RST)

Operating while connected

Server 1

Server 2

Server 3

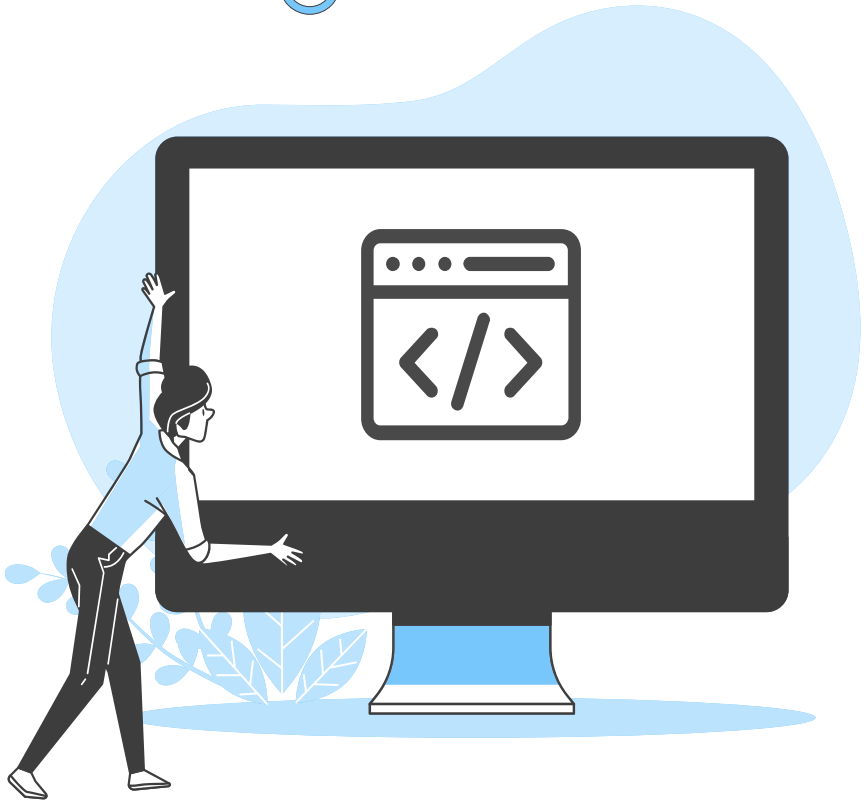


Server 4

“Plz no more”

02
HTML and parsing

About HTML



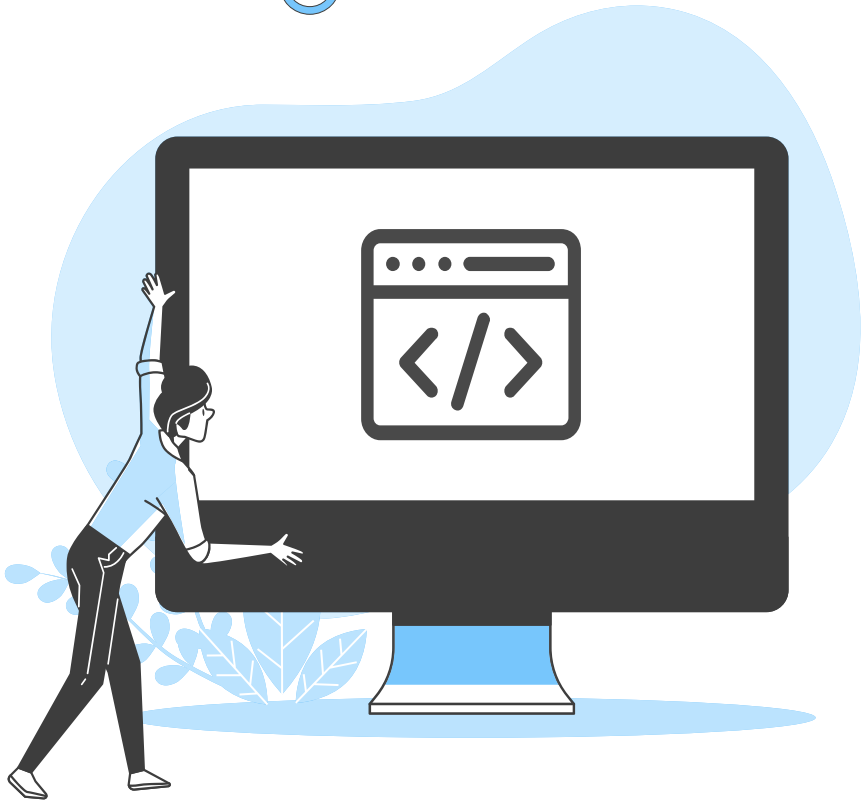
HTML is a language created by Tim-Bernes Lee in 1989 in which most websites are written.

Hypertext Markup Language (HTML) is:

- a Markup, not a Programming language
- built on two things: Tags and Attributes

Hypertext = formats links as well as text, for interaction

HTML Tags



- Mark the start of an HTML element
- Opened using <>, closed using </>
  Example: *<h1> ..... </h1>*

  1. Must be closed
  2. Must be closed in the same order as they were opened. (think nested sets)
     Example:
     <h1><h2><h3>....</h3></h2></h1>

- Contain limited information

HTML Attributes

- Unlike tags, they contain additional details/information about the element.
- Live inside tags, determine properties of elements
- Privileged type of attribute: class → Used to link multiple tags
- HTML writing supports various editors: Sublime text 3, Notepad, Notepad++, Komodo Edit etc.

Parsing

- Telling Python you're expecting something
- Similar to RegEx metacharacters
- Use an integrated parser from your libraries

Agenda

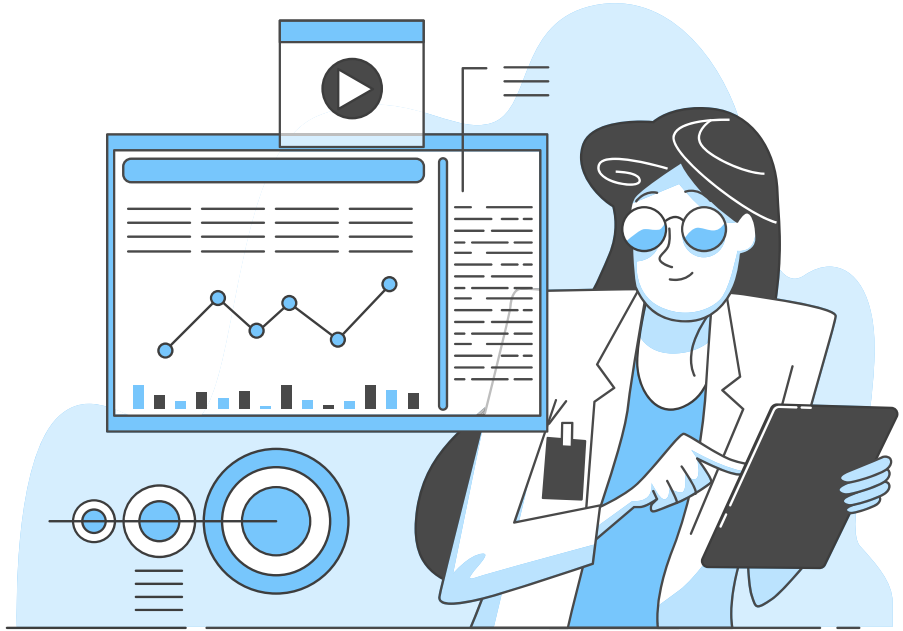
01 ~~What is web scraping?~~

02 ~~A bit about HTML and parsing~~

03 Web scraping with Python

04 More advanced uses

API and cloud scraping



03

Web scraping in Python

Web scraping in Python

4 ways

- LXML
- Python request
- Scrapy
- Selenium

Web scraping in Python

LXML

- Speedy parsing of both XML and HTML formats
- Retain the native Python interface while scraping

Libraries: libxml2 and libxslt

Python requests

- Generates direct response from server without needing to decode the content
- Has a HTTP scraping functionality in its own, but can also be used solely for connecting

Libraries: requests

Web scraping in Python

SCRAPY

- Open-source web crawler, highly available
- Uses classes called spiders for the fetching, has an built-in request manager

Libraries: scrapy

SELENIUM

- Has its own API called Selenium WebDriver
- Provides the largest range of functionalities, being the preferred choice for more elaborate scraping tasks

Libraries: selenium (needs setup for API)

And...

Gorgeous, splendid, sublime

Beautiful Soup

- The library we'll focus in depth on
- Allows for very easy interpretation of scraped pages using the pure Python interface
- Understandable structure

Libraries: bs4 (and we'll use requests for server connection)

In conclusion

Python

Environment/Facilitator: provides interface for request

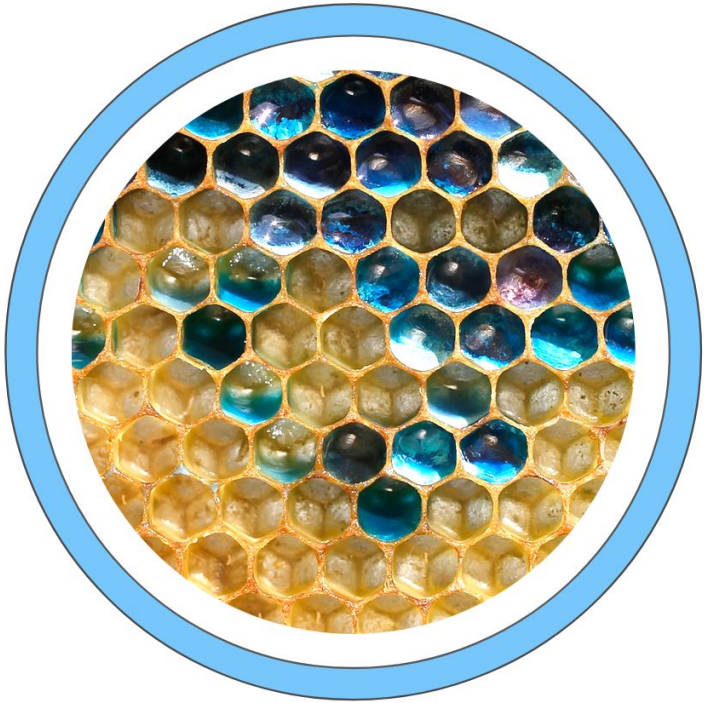
Beautiful Soup

Framework: handles the requests

04

More advanced uses

Application Programming Interface

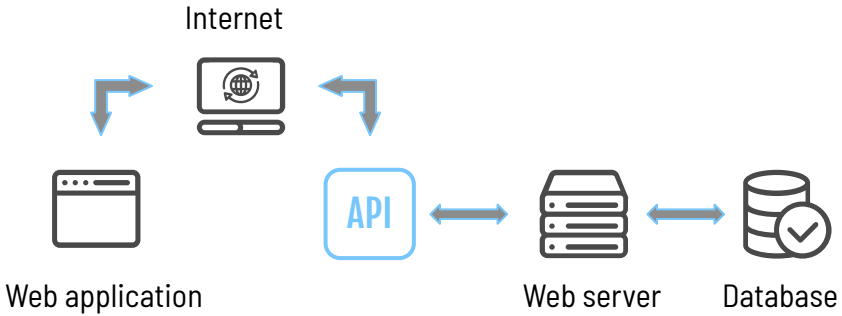


API

The protocols defining how the software is built and implemented

- Allows communication across services/products *without* disclosing information regarding how they're built.
- Have their own management system to handle requests in the form of "contracts"

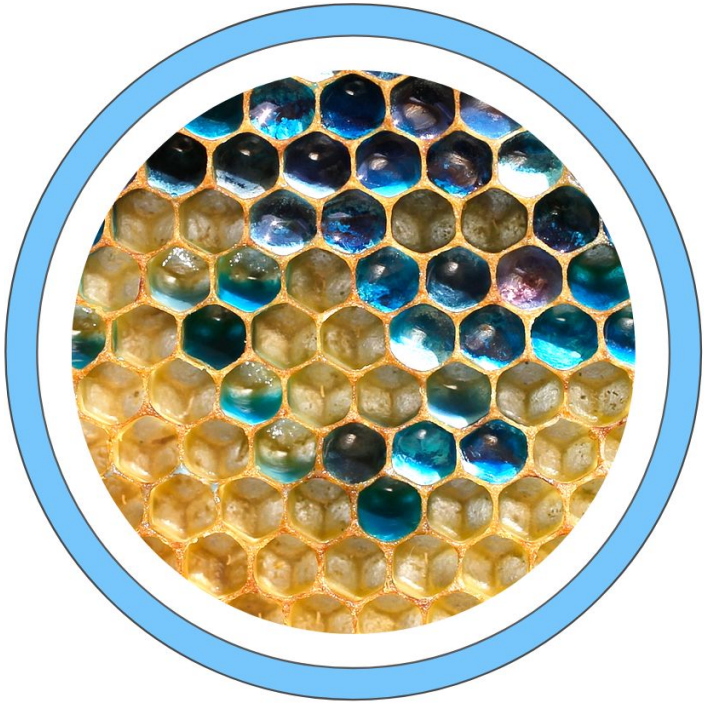
Application Programming Interface



- API's eliminate the need to find workarounds to communicate to the server (eg: scraping).
- Chain websites use them to communicate among each other more easily, but they can also be implemented for user interaction.
- Used as a response to how easy interacting with web pages was - which led to buff overflow attacks and bots.

Watch this:
https://www.youtube.com/watch?v=BxV14h0kFs0

Application Programming Interface



- In modern days, big webpages (Twitter, Facebook) retain subscription-based API's and handle requests more securely
- Smaller websites may still have simple APIs implemented and easy to use
- Python interacts with APIs very similarly to scraping, using the requests library
- Results are returned in a JSON (JavaScript Object Notation) format

Is scraping just a work around to APIs not being available?

Yes... and no

Cloud scraping

Not all websites will like your IP accessing their pages thousands of times per hour...

There are two ways to deal with this problem:

- Legal solutions
- Illegal solutions

Cloud scraping

Legal solutions

- Solution 1: Use a trusted, constant IP
- Solution 2: Outsource your requests from a cloud environment
- Solution 3: Use multiple IPs associated with a cloud service
  Eg: Scrapestack, ScraperAPI, ScrapingBee or Octoparse

Cloud scraping

Illegal solutions

They are the the reason why large services discourage scraping on their sites

- Solution 4: IP spoofing or masking (a VPN doesn't typically work)
- Solution 5: Botting

When to use Cloud scraping

You should use Cloud scraping in the following cases:

- When you need streamlined data (Velocity)
- When you need sizeable data from a single set of sources (Volume)
- When the process is time consuming and you need more computational power.

Thanks!

Do you have any questions?