

# Web Scrapping con Python

Ing. Federico Bertola | Ing. Germán Parisi



@SecLabSis  
UTN FRC



Ing. Germán Parisi

Miembro del equipo de seguridad en LabSis -  
UTN FRC y Co-fundador de Pabex.

Docente en la cátedra de sistemas  
operativos en UTN FRC.

@GochiParisi



Ing. Federico Bertola

Miembro del equipo de seguridad en LabSis -  
UTN FRC y Co-fundador de Pabex.

@FedeJBertola

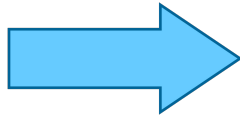
**Pabex**  
Desarrollo de software

# Agenda

- Web Scraping en OSINT.
- Técnicas de Web Scraping.
- Demos.
- Dificultades.
- Web Scraping en la darknet.
- Scraping en videos.
- Conclusiones.

# Web Scrapping

Proceso para recopilar información desde un sitio web, obteniendo los datos en una estructura que nos facilite el posterior procesamiento.




# Web Scraping vs Web Spider/Crawler

- Web Scraping tiene un target específico para obtener la información deseada, por lo que su implementación depende de la estructura y tecnología del sitio web objetivo.
- Web Spider/Crawler: su principal objetivo es recorrer todos los enlaces de un sitio e indexar su contenido. Continúa aplicando el mismo procedimiento en las páginas encontradas.

# Web Scraping en OSINT

OSINT (Open Source Intelligence): Inteligencia de fuentes abiertas.

Etapas:

- Planificación.
- Selección de fuentes.
- **Obtención de datos.** ← **Web Scraping**
- Procesamiento.
- Análisis.
- Reporte.

# Técnicas de Web Scraping

Las técnicas dependen de la tecnología utilizada en el sitio web objetivo.

Principalmente, podemos dividirlo en técnicas para:

- Sitios web estáticos: Al hacer una petición HTTP se obtiene todo el documento html con la información.
- Sitios webs dinámicos: La información es cargada a través de la ejecución de JavaScript (por ejemplo con Angular o mediante peticiones AJAX).

# Técnicas de Web Scraping - Sitios web estáticos

Decimos que una página es estática, cuando al realizar la solicitud obtenemos todo el documento html con toda la información. No es necesario la ejecución de JavaScript para la visualización del sitio.

```
<html>
<head>
  <title>Html puro</title>
</head>
<body>
  <h1>
    En el HTML se encuentra todo el contenido que necesitamos extraer.
  </h1>
</body>
</html>
```



# Técnicas de Web Scraping - Sitios web estáticos

Lógica:

1. Hacer la petición tal cual hace el navegador:  
`response = requests.get('https://sitio.com/busqueda/?q=JuanPerez')`
2. Parsear el HTML:  
`bs = BeautifulSoup(response.text, 'html.parser')`
3. Buscar las etiquetas que contengan los datos que deseamos sacar:  
`spans = bs.find_all('span', class_='phone-number')`
4. Estructurar los datos de una forma conveniente:  
`phone_numbers = []`  
`for span in spans:`  
    `phone_number = span.string`  
    `phone_numbers.append(phone_number)`

# Demo



Nombre Dirección Teléfono DDN-DDI

Juan Perez

¿Provincia? ¿Localidad?

Buscar

PA Empresas

Industria

H Hoteles

Restaurantes

Mapas

Páginas Blancas > Juan Perez en Argentina

### Juan Perez en Argentina | 1029 resultados (Mostrar resultados en mapa)

Refinar por: [Provincia / Localidad](#) ▾

Acciones Múltiples



Seleccionar Acción ▾



Perez Juan A



B Mitre Lonquimay - Ciudad de Buenos Aires (CP:1039)

Ver teléfono

Compartir

Perez Juan P



Sánchez Ramos San Juan (CP:5400)

Ver teléfono

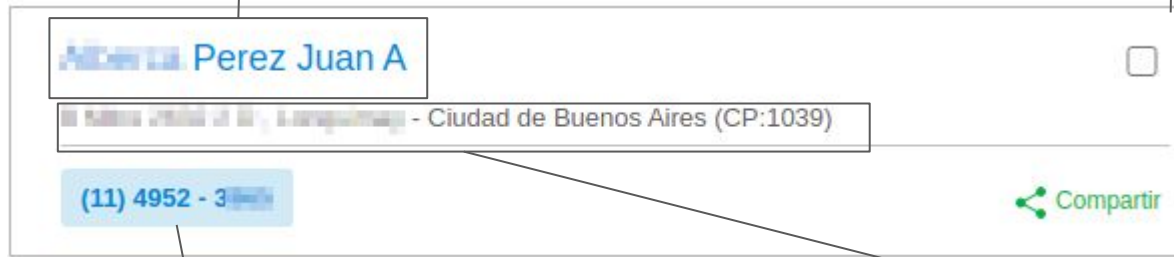
Compartir

Perez Juan M



```
<li class="m-results-business m-results-subscriber">
....
</li>
```

```
<h3 class="m-results-business--name"> .... Perez Juan A </h3>
```



```
<div class="m-button--results-business--icon
m-button--results-business--see-phone"
onclick="clickDirTel(TipoClick.CLIC_DIR_TEL,
541149523___)">
</div>
```

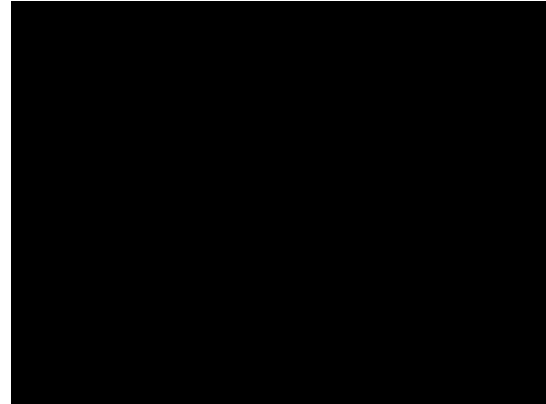
```
<div class="m-results-business--address">
  <span>Calle y nro</span>
  <span>Localidad</span>
  <span>Código postal</span>
</div>
```

```
url = "http://www.paginasblancas.com.ar/persona/%s" % name
response = requests.get(url, headers=headers)
bs = BeautifulSoup(response.text, "html.parser")

lis = bs.find_all("li", class_="m-results-business m-results-subscriber")
for li in lis:
    # Name
    h3_name = li.find("h3", class_="m-results-business--name")
    a_name = h3_name.find("a")
    name = a_name.string

    # Address
    div_address = li.find("div", class_="m-results-business--address")
    spans_address = div_address.find_all("span")
    street_address = spans_address[0].string.strip()
    street_address = re.sub(' +', ' ', street_address)
    locality_address = spans_address[1].string.strip()
    postal_code_address = spans_address[2].string.strip()

    # Phone number
    div_number = li.find("div", class_="m-button--results-business--icon m-button--results-bu...")
    onclick = div_number.attrs['onclick']
    match_number = re.search(r'\d+', onclick)
    phone_number = match_number.group() if match_number.group() else None
```



# Técnicas de Web Scraping - Sitios dinámicos

Cuando el contenido de la página no es obtenido directamente en el HTML de la página, sino que es renderizado mediante JavaScript, ya sea por alguna petición AJAX o algún framework como Angular, tenemos principalmente las siguientes opciones:

- Utilizar un entorno que ejecute JavaScript, por ejemplo con Selenium ejecutar el navegador y así sacar los datos que necesitemos.
- Otra alternativa más recomendada, es analizar qué peticiones hace el sitio web y nosotros ejecutarlas de la misma manera.


# Login


Algo no tan relacionado a OSINT... pero ¿qué pasa si para acceder a los datos primero tenemos un login?


```
data = {  
    "usuario": "juanlopez",  
    "password": "12345678",  
    "remember-me": True  
}
```

```
session = requests.session()  
response = session.post("https://ejemplo.com/login/", data)
```

# Login





**Mi Claro**

¡Nueva forma de ingreso!

E-mail o Número de Línea

Contraseña

[MOSTRAR](#)

☒ Mantenerme conectado

[Olvidé mi Contraseña](#)

Ingresá

[¿No tenés usuario? Registrate](#)

[¿Necesitás ayuda?](#)

Al ingresar estás aceptando los

[Términos y Condiciones](#)



▼ General

**Request URL:** [https://miclaro.claro.com.ar/web/guest/bienvenido?p\\_p\\_id=58&p\\_p\\_lifecycle=1&p\\_p\\_state=normal&p\\_p\\_mode=view&p\\_p\\_col\\_id=column-1&p\\_p\\_col\\_count=2&sav\\_eLastPath=0&\\_58\\_doActionAfterLogin=false&\\_58\\_struts\\_action=%2Flogin%2Flogin](https://miclaro.claro.com.ar/web/guest/bienvenido?p_p_id=58&p_p_lifecycle=1&p_p_state=normal&p_p_mode=view&p_p_col_id=column-1&p_p_col_count=2&sav_eLastPath=0&_58_doActionAfterLogin=false&_58_struts_action=%2Flogin%2Flogin)

**Request Method:** POST

**Status Code:** 🟡 302 Movido temporalmente

**Remote Address:** 170.51.242.17:443

**Referrer Policy:** no-referrer-when-downgrade

▶ Response Headers (12)

▶ Request Headers (17)

▶ Query String Parameters (9)

▼ Form Data [view source](#) [view URL encoded](#)

**\_login\_usuario:** [fernando.lopez@claro.com.ar](mailto:fernando.lopez@claro.com.ar)

**\_login\_password:** [\[REDACTED\]](#)

**remember-me:** true

▼ Response Headers [view source](#)

**Content-Encoding:** gzip

**Content-Length:** 0

**Date:** Mon, 14 Sep 2020 04:08:43 GMT

**Location:** <https://miclaro.claro.com.ar/web/guest/home>

**Server:** Apache-Coyote/1.1

**Set-Cookie:** remember-me=Y3ZIVzRUamhiZzBzB3NTEG1SZ1JEUT090k9rNmYvTXFiSXlYSU0zZFdmcdJJoTXc9PQ; Expires=Mon, 09-Nov-2020 04:08:43 GMT; Path=/; HttpOnly

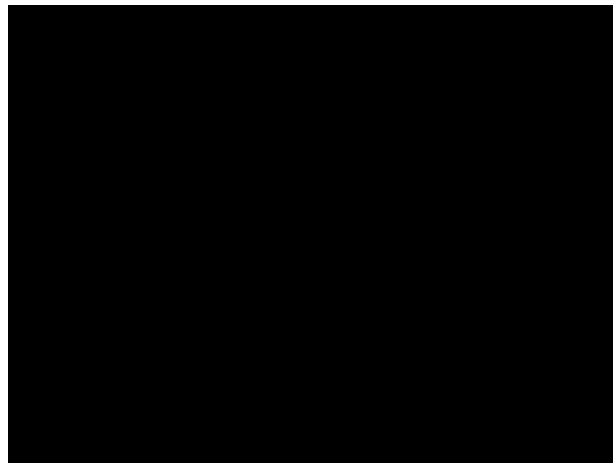
**Set-Cookie:** sso=24ddc4e4-aa09-4161-a5b7-91d3f5170c8d; Domain=.claro.com.ar; Path=/; HttpOnly

**Set-Cookie:** JSESSIONID=F6FFED18D43F9B0C9D9C7B87C43F3B1; Path=/; HttpOnly

# Login

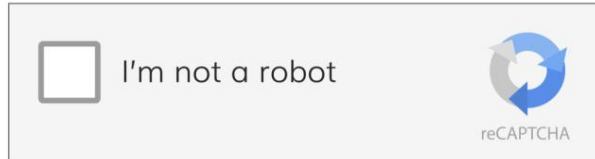
```
data = {  
    "_login_usuario": USUARIO_CLARO,  
    "_login_password": CLAVE_CLARO,  
    "remember-me": True  
}
```

```
session = requests.session()  
session.get("https://miclaro.claro.com.ar/web/guest/bienvenido")  
URL = "https://miclaro.claro.com.ar/web/guest/bienvenido?p_p_id=58&p_p_lifecyc...."  
response = session.post(URL, data)
```



# Dificultades

- Captchas.



- Términos y condiciones.

In addition, you agree not to use the Service or the Site to:

- harvest or collect email addresses or other contact information of other users from the Service or the Site by electronic or other means for the purposes of sending unsolicited emails or other unsolicited communications;
- use the Service or the Site in any unlawful manner or in any other manner that could damage, disable, overburden or impair the Site;
- use automated scripts to collect information from or otherwise interact with the Service or the Site;

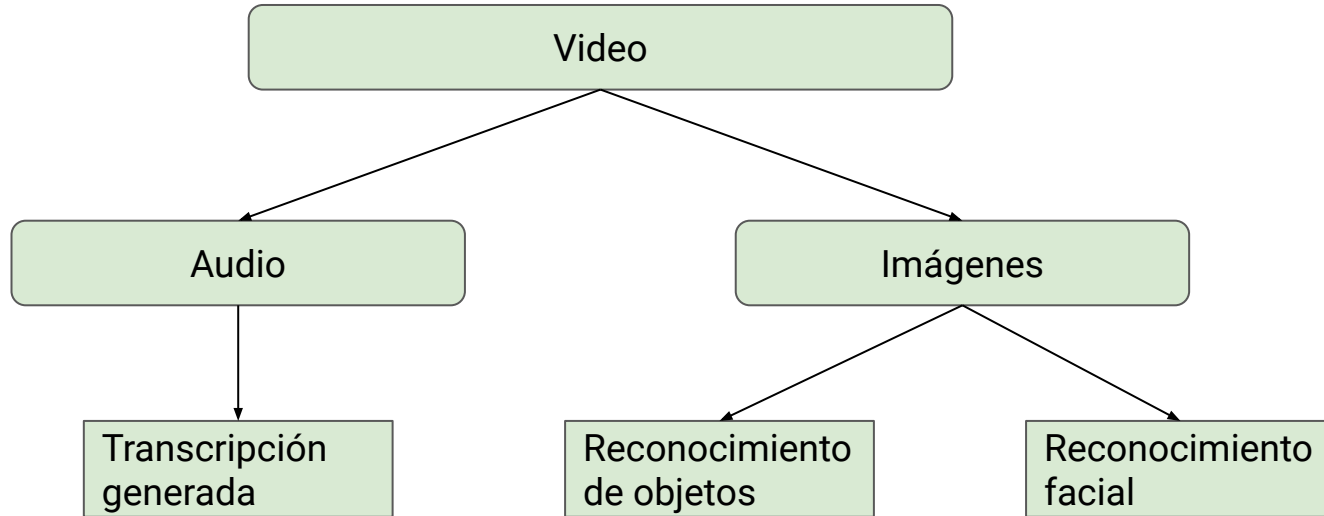
# Web Scraping en la Darknet (tor)

```
session = requests.session()
```

```
session.proxies = {'http': 'socks5h://127.0.0.1:9050', 'https':  
'socks5h://127.0.0.1:9050'}
```

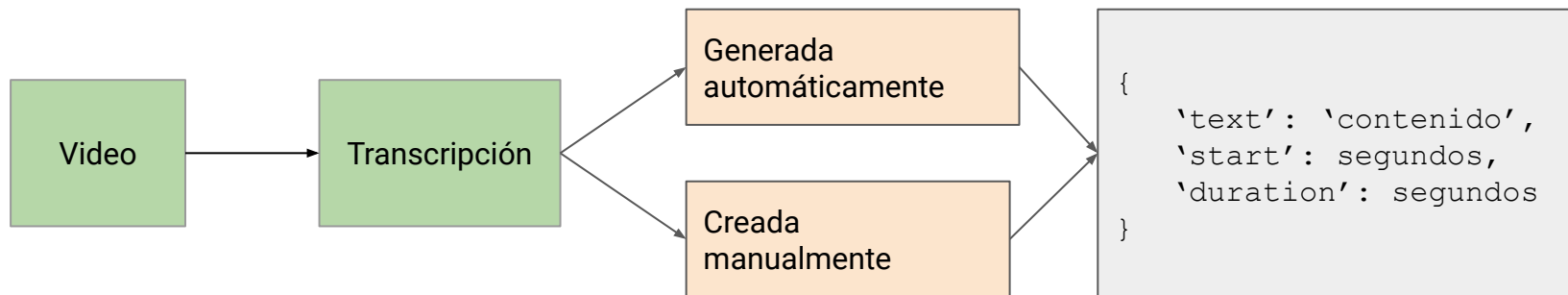
```
response = session.get(url_onion)
```

# Scraping en un Video



# Scraping en videos (YouTube)

Obtener información de la **transcripción** de un video de YouTube.



La API captions de YouTube permite obtener la transcripción de cualquier video:

<https://developers.google.com/youtube/v3/docs/captions>

Con Python se puede utilizar la librería:

<https://pypi.org/project/youtube-transcript-api/>

Demo

# Código fuente de demo

```
from youtube_transcript_api import YouTubeTranscriptApi

video_id="1zxKZx7we4s"

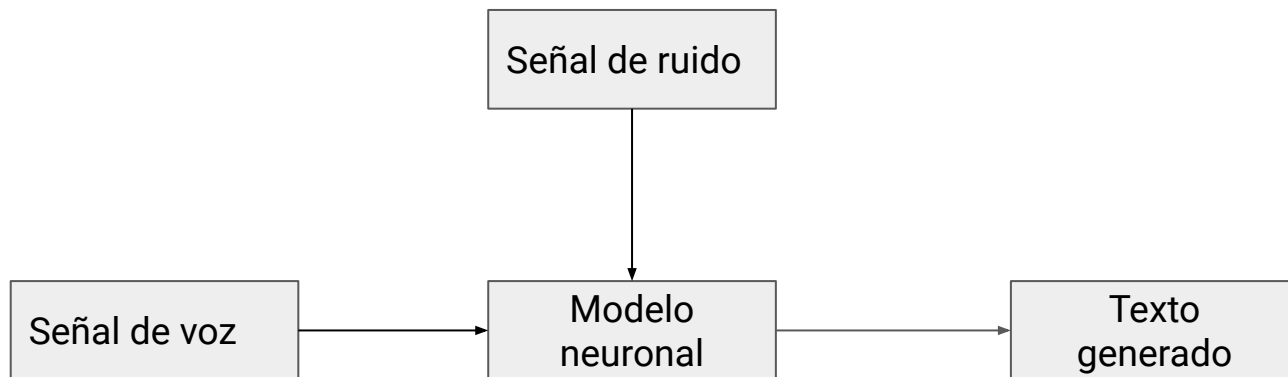
transcript_list = YouTubeTranscriptApi.get_transcript(video_id,
languages=['es'])

for transcript in transcript_list:

    print(transcript)
```



# Reconocimiento de voz (Speech Recognition)



Librería para Python:  
<https://pypi.org/project/SpeechRecognition/>

# Reconocimiento de imágenes en video

¡Problema!

¿Cada cuánto debo realizar una **muestra**?

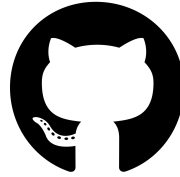
Intentemos responder

1. Un video está compuesto por una o más escenas.
2. En una escena existen objetos y personajes.
3. Sería muy interesante detectar el cambio de **escena** para tomar una **muestra**.

# Conclusiones

1. Cada página web tiene datos cuya extracción puede variar de acuerdo a la tecnología implementada.
2. El captcha es un problema para el scraping (¿Se puede romper?).
3. Para hacer scraping en la darknet es básicamente de la misma forma que en la web normal.
4. La web ya no es sólo texto plano. Hay mucho contenido multimedia abierto y hay que aprender a extraer esa información.

# ¡Gracias!



<https://github.com/Pabex/webscraping>