Data hunting and gathering (part 2)

Sesión 2: Creando Nuestra Propia API Web - Scraping

En esta sesión, nos adentramos en el mundo del web scraping, enfocándonos en técnicas y herramientas avanzadas para extraer datos de páginas web dinámicas.

Objetivos de Aprendizaje

- **Entender HTML y CSS**: Comprender los fundamentos de la estructura HTML y el estilo CSS para navegar efectivamente por las páginas web.
- **Selectores XPath**: Aprender a usar selectores XPath para localizar y extraer contenido específico de las páginas web.
- **Scraping de Contenido Dinámico con Selenium**: Entender cómo raspar contenido generado dinámicamente, el cual las herramientas de scraping estándar no siempre pueden manejar.

Bibliotecas Adicionales de Python

Necesitarás instalar estas bibliotecas de Python para las tareas de scraping:

- **lxml**: Una poderosa biblioteca para procesar XML y HTML en Python.
  - Instalar vía pip: `pip install lxml`
- **selenium**: Una herramienta automatizada de navegador web para probar aplicaciones web, también útil para tareas complejas de scraping.
  - Instalar vía pip: `pip install selenium`

Nota

Asegúrate de que todo el software y las bibliotecas estén instalados antes de la sesión. Esto te permitirá participar activamente en los ejercicios y seguir los ejemplos de scraping.

1. "Creando Tu Propia API": Web Scraping

Entendiendo el Web Scraping

El web scraping se vuelve esencial cuando los datos están disponibles en la web pero no son accesibles a través de una API, o la API existente carece de ciertas funcionalidades o tiene términos de servicio restrictivos. En tales escenarios, el **Web Scraping** es la técnica que permite la extracción automatizada de estos datos, replicando el acceso que tendría visualmente un humano.

¿Por Qué Web Scraping?

- **Accesibilidad de Datos**: A veces, la única forma de acceder a ciertos datos es directamente desde las páginas web donde se muestran.
- **Flexibilidad**: El web scraping permite personalizar la extracción de datos para necesidades específicas, eludiendo las limitaciones de las APIs existentes.

Preparándose para el Web Scraping: Entendiendo la Estructura de las Páginas Web

Antes de adentrarse en el scraping, es crucial tener un conocimiento básico de la estructura de las páginas web y cómo se almacenan y presentan los datos. Esta sesión cubre:

Páginas Estáticas Básicas de HTML y CSS

- **HTML (Lenguaje de Marcado de Hipertexto)**: El lenguaje de marcado estándar utilizado para crear páginas web. Entender HTML es clave para identificar los datos que quieres raspar.
- **CSS (Hojas de Estilo en Cascada)**: Se utiliza para describir la presentación de un documento escrito en HTML. Conocer CSS ayuda a localizar elementos específicos en una página.

HTML Dinámico

- **Ejemplo Básico de JavaScript Usando JQuery**: Los sitios web a menudo usan JavaScript para cargar datos dinámicamente. Entender cómo funciona esto es crucial para raspar datos de dichas páginas dinámicas.

1.1 Fundamentos de HTML + CSS 101

Entendiendo los Fundamentos de las Páginas Web

Las páginas web más fundamentales están construidas usando HTML y CSS. Estas tecnologías cumplen dos propósitos primarios: **HTML (Lenguaje de Marcado de Hipertexto)** estructura y almacena el contenido, siendo el objetivo principal para el web scraping, mientras que **CSS (Hojas de Estilo en Cascada)** formatea y estiliza el contenido, destacando elementos visuales como fuentes, colores, bordes y diseño.

HTML: La Estructura de la Web

HTML es un lenguaje de marcado típicamente renderizado por navegadores web. Usa 'etiquetas' para definir elementos en una página web. Un formato de etiqueta típico incluye un nombre de etiqueta, atributos (si los hay) y el contenido entre etiquetas de apertura y cierre.

Componentes Clave de un Archivo HTML

- **Declaración DOCTYPE**:
  - Comienza con `<!DOCTYPE html>`, indicando el uso de HTML5.
  - Versiones anteriores de HTML tenían diferentes DOCTYPEs.
- **Etiqueta HTML**:
  - La etiqueta `html` (y su etiqueta de cierre `/html`) encierra todo el contenido de la página web.

- **Cabeza y Cuerpo**:
  - La sección `head` a menudo incluye la etiqueta `title`, definiendo el nombre de la página web, enlaces a hojas de estilo CSS y archivos JavaScript para comportamiento dinámico.
  - El `body` contiene el contenido visible de la página web.
- **Elementos Comunes de HTML**:
  - **Encabezados y Párrafos**: Usa `h#` (donde # es un número) para encabezados y `p` para párrafos.
  - **Hipervínculos**: Definidos con el atributo `href` en etiquetas `a` (ancla).
  - **Imágenes**: Incrustadas usando etiquetas `img` con el atributo `src`. Nota: `img` se cierra por sí misma.

Ejercicio: Construir una Página Web Básica en HTML

Pongamos en práctica tu conocimiento de HTML:

- Crea un archivo llamado 'ejemplo.html' en tu editor de texto favorito.
- Construye una página web básica en HTML que contenga elementos como etiquetas `title`, `h1`, `p`, `img` y `a`. Recuerda que casi todas las etiquetas necesitan cerrarse con una `/tag`.

Este ejercicio tiene como objetivo familiarizarte con la estructura básica de HTML y cómo varios elementos se unen para formar una página web.

Si eres perezoso, ve a la carpeta de archivos y haz doble clic en "ejemplo.html". Puedes verificar el código html ejecutando la siguiente línea.

```
# %%html

# <!-- Start of the HTML head section -->
# <head>
#     <!-- Title of the webpage -->
#     <title>
#         Basic knowledge for web scraping.
#     </title>
# </head>
# <!-- Start of the HTML body section -->
# <body>
#     <!-- Header 1 indicating the subject of the content -->
#     <h1>About HTML
#     </h1>
#     <!-- Paragraph explaining what HTML is and providing a link for
further information -->
#     <p>Html (Hypertext markdown language) is the basic language to
provide contents in the web. It is a tagged language. You can check
more about it in <a
href="http://www.w3.org/community/webed/wiki/HTML">World Wide Web
Consortium.</a></p>
```

```
#     <!-- Paragraph indicating that one of the following images is
clickable -->
#     <p> One of the following rubberduckies is clickable
#     </p>
#     <!-- Image of a rubber ducky; this one is not clickable -->
#     <p>
#         <img src = "files/rubberduck.jpg"/>

#         <!-- Clickable image (hyperlinked) of a rubber ducky -->
#         <a href="http://www.pinterest.com/misscannabliss/rubber-
duck-mania/"><img src = "files/rubberduck.jpg"/></a>
#     </p>
# </body>
```

Entendiendo Páginas Estáticas HTML Antiguas vs. Actuales

Páginas Estáticas HTML al Estilo Antiguo

Las páginas HTML antiguas a menudo dependían de tablas y listas para estructurar el contenido.

- **Listas**:
  - **Listas Ordenadas (`ol`)**: Se utilizan para crear listas donde el orden importa, con cada elemento representado por `li` (list item).
  - **Listas Desordenadas (`ul`)**: Se utilizan para listas donde el orden no es importante, nuevamente usando `li` para cada elemento.
- **Tablas**:
  - La etiqueta `table` se utiliza para crear una tabla.
  - Cada fila de la tabla se marca con una etiqueta `tr`.
  - Las columnas de la tabla se definen por elementos `td` (table data) dentro de cada fila.
  - Las tablas pueden incluir un encabezado (`thead`) y un cuerpo (`tbody`).
  - Los elementos `th` se utilizan de manera similar a `td` pero para encabezados.
  - Para extender una celda a través de múltiples columnas, se utiliza `colspan` con el número de celdas a cubrir.

Páginas Estáticas HTML Actuales

Las páginas HTML modernas se centran más en el uso de contenedores y CSS para el diseño y estilo.

- **Divisiones (`div`)**:
  - La etiqueta `div` señala una división y se utiliza para definir un bloque de contenido. Es un contenedor versátil utilizado en el diseño web moderno.
- **Span (`span`)**:
  - La etiqueta `span` se utiliza para resaltar o estilizar una parte específica de un bloque de contenido. Es un contenedor en línea y se utiliza a menudo para modificaciones a pequeña escala de texto u otros elementos.

Tanto los estilos antiguos como los actuales de HTML tienen sus usos, pero las prácticas modernas favorecen el uso de `div` y `span` junto con CSS para un diseño más flexible y adaptable.

Entendiendo CSS para el Web Scraping

CSS, que significa Hojas de Estilo en Cascada, es un lenguaje de hojas de estilo utilizado para describir la presentación y formato de documentos HTML. En el web scraping, entender CSS es crucial para navegar y extraer datos de las páginas web eficazmente.

¿Qué es CSS?

- **CSS** es un lenguaje diseñado para estilizar el contenido de archivos HTML. Mediante CSS, los desarrolladores web definen cómo deben aparecer varios elementos HTML en una página web.
- El término **"cascada"** se refiere a la prioridad que se da a ciertas reglas de estilo sobre otras más genéricas. Esta jerarquía es un aspecto fundamental de CSS.

El Papel de CSS en el Web Scraping

- **Separación de Preocupaciones**: CSS permite una clara separación entre la estructura de HTML (contenido) y el estilo de la página web (apariencia). Esta separación hace que las páginas web sean más fáciles de diseñar y mantener, y también más fáciles de raspar.
- **Selectores y Propiedades**:
  - **Selectores** son patrones utilizados para seleccionar el(los) elemento(s) que se desea estilizar, o en el caso del scraping, los elementos que se desean extraer.
  - **Propiedades** son los aspectos de los elementos que se desea estilizar, como color, fuente, ancho, altura y más.
- **Orden de Cascada**:
  - Los estilos se aplican en orden de especificidad, con selectores más específicos anulando los más generales. Los estilos en línea (directamente dentro de un elemento HTML) tienen la mayor especificidad.

Ejemplo en Web Scraping

Considera una página web con el siguiente HTML y CSS:

```
<!-- HTML Example -->
<div class="product-description">
    <p>Awesome product</p>
</div>
```

Now, css example:

```
/* CSS Example */
.product-description p {
    color: blue;
}
```

En este ejemplo, el CSS se dirige a un elemento `p` dentro de un `div` de la clase `product-description` y cambia el color de su texto a azul. Entender cómo se aplica esta regla de CSS ayuda en el scraping de datos de manera precisa.

Conclusión

Para el web scraping, CSS no se trata solo de entender la estética de la página web; se trata de comprender de manera integral la estructura de la página web. Este entendimiento es crucial para una extracción de datos efectiva.

Ejemplo en Python: Web Scraping Usando Selectores CSS

Este ejemplo demuestra cómo raspar una página web (el blog oficial de Python) y extraer contenido específico utilizando selectores CSS en Python. Utilizamos las bibliotecas `requests` y `BeautifulSoup` para lograr esto.

Explicación del Script

1. **Import Libraries**:
   - `requests` for sending HTTP requests.
   - `BeautifulSoup` from `bs4` for parsing HTML content.

```python import requests from bs4 import BeautifulSoup
```

```
#pip3 install requests beautifulsoup4

import requests
from bs4 import BeautifulSoup

# URL del sitio web que queremos extraer datos
# En este caso, estamos apuntando a la página de blogs de Python.org
url = "https://www.python.org/blogs/"

# Enviar una solicitud GET a la URL especificada
# Esta solicitud obtiene el contenido HTML de la página web
response = requests.get(url)

# Analizar el contenido HTML de la página
# 'BeautifulSoup' es una biblioteca de Python para analizar documentos
HTML
# Crea un árbol de análisis a partir del código fuente de la página
que se puede usar para extraer datos fácilmente
soup = BeautifulSoup(response.text, 'html.parser')

# # what is soup if you print it:
# soup

# Usar un selector CSS para extraer elementos específicos
# Aquí seleccionamos todos los elementos 'h2' (comúnmente utilizados
para títulos o encabezados en HTML)
# 'select' es un método que encuentra todas las instancias de una
etiqueta con la ruta CSS especificada
```

```
titles = soup.select('h2')

# Iterar a través de los títulos extraídos y mostrarlos
# 'get_text()' extrae la parte de texto del elemento HTML, y 'strip()' elimina espacios en blanco al inicio y al final
for title in titles:
    print(title.get_text().strip())

# Este script imprime todo el contenido textual de las etiquetas 'h2' encontradas en la página de blogs de Python
# Es un ejemplo de cómo extraer y mostrar partes específicas de una página web
```

```
Welcome to the Python Insider
Latest News
Python Insider Subscriptions
Copyright
```

Ejercicio de Web Scraping: Extracción de Titulares de Noticias de Tecnología de la BBC

Objetivo

Escribir un script en Python para raspar titulares de la sección de noticias de tecnología de la BBC y categorizarlos basándose en palabras clave.

Detalles de la Tarea

1. **Sitio Web para Raspar**:
   - Dirigirse a la sección 'Tecnología' de la BBC: Noticias de Tecnología de la BBC.
2. **Requisito de Scraping**:
   - Raspar los titulares principales de la página, que se encuentran típicamente en etiquetas `h3` o en una clase específica.
3. **Categorización**:
   - Categorizar los titulares basándose en palabras clave predefinidas como 'Apple', 'Microsoft', 'Google', etc.
   - Contar el número de titulares que caen en cada categoría.
4. **Salida**:
   - Imprimir cada titular junto con su respectiva categoría.
   - Resumir con el recuento de titulares en cada categoría.

```
import requests
from bs4 import BeautifulSoup

# URL de la sección de noticias de tecnología de la BBC
url = "https://www.bbc.co.uk/news/technology"

# Enviar una solicitud GET y analizar el contenido HTML
response = requests.get(url)
```

```
soup = BeautifulSoup(response.text, 'html.parser')

# Definir categorías y palabras clave asociadas
# Estas palabras clave ayudarán a clasificar los titulares en
diferentes categorías
categorias = {
    'Apple': ['Apple', 'iPhone', 'iPad'],
    'Microsoft': ['Microsoft', 'Windows', 'Bill Gates'],
    'Google': ['Google', 'Android', 'Alphabet']
    # Agregar más categorías según sea necesario
}

# Función para determinar la categoría de un titular
def categorizar_titular(titular):
    """
    Clasifica un titular basándose en palabras clave.
    Si se encuentra una palabra clave en el titular, devuelve el
nombre de la categoría.
    De lo contrario, devuelve 'Otros'.
    """
    for categoria, palabras_clave in categorias.items():
        for palabra in palabras_clave:
            if palabra.lower() in titular.lower():
                return categoria
    return 'Otros'

# Raspar y procesar los titulares
# Buscar etiquetas 'h3' u otras etiquetas relevantes que contengan
titulares
titulares = soup.find_all('h3')
titulares_categorizados = {}

for etiqueta_titular in titulares:
    titular = etiqueta_titular.get_text().strip()
    categoria = categorizar_titular(titular)
    if categoria not in titulares_categorizados:
        titulares_categorizados[categoria] = []
    titulares_categorizados[categoria].append(titular)

# Imprimir cada titular con su categoría correspondiente
for categoria, lista_titulares in titulares_categorizados.items():
    print(f"Categoría: {categoria}\n")
    for titular in lista_titulares:
        print(f" - {titular}")
    print("\n")
```

```
Categoría: Otros

 - 06:00Logan Paul accused of misleading fans over crypto investments,
published at 06:00Logan Paul accused of misleading fans over crypto
```

investments
 - 04:27Who has joined Trump's team so far?, published at 04:27Who has joined Trump's team so far?
 - 02:03No plans to join Bluesky yet, Starmer says, published at 02:03No plans to join Bluesky yet, Starmer says
 - 22:48 19 NovemberTrump joins Musk to watch Starship test launch. Video, 00:01:03, published at 22:48 19 NovemberTrump joins Musk to watch Starship test launch
 - 19:00 19 NovemberTeenager 'picked on by teachers' before death - inquest, published at 19:00 19 NovemberTeenager 'picked on by teachers' before death - inquest
 - 16:34 19 NovemberAI cameras catch drivers on phone or without belts, published at 16:34 19 NovemberAI cameras catch drivers on phone or without belts
 - 15:19 19 NovemberInstagram testing tool that resets all recommended posts , published at 15:19 19 NovemberInstagram testing tool that resets all recommended posts
 - 11:45 19 NovemberSexual offence rumours online unfounded - police, published at 11:45 19 NovemberSexual offence rumours online unfounded - police
 - 08:49 19 NovemberMan who binned Bitcoin says it's now worth Â£500m. Video, 00:00:36, published at 08:49 19 NovemberMan who binned Bitcoin says it's now worth Â£500m
 - 08:39 19 NovemberMan says his binned Bitcoin fortune now worth Â£500m, published at 08:39 19 NovemberMan says his binned Bitcoin fortune now worth Â£500m
 - 01:15 19 NovemberBritish Airways says 'tech issue' resolved after delays, published at 01:15 19 NovemberBritish Airways says 'tech issue' resolved after delays
 - 00:41 19 November'We booked a Â£99 stargazing break - all we got was an empty field', published at 00:41 19 November'We booked a Â£99 stargazing break - all we got was an empty field'
 - 00:00 19 NovemberChocolate makers stoke boom for Indian cocoa beans, published at 00:00 19 NovemberChocolate makers stoke boom for Indian cocoa beans
 - 17:55 18 NovemberThe Game Awards 2024: Astro Bot and Final Fantasy lead nominations , published at 17:55 18 NovemberThe Game Awards 2024: Astro Bot and Final Fantasy lead nominations
 - 17:27 18 NovemberMum 'not told about outbursts' before daughter's death, published at 17:27 18 NovemberMum 'not told about outbursts' before daughter's death
 - 16:29 18 NovemberRoblox to ban young children from messaging others, published at 16:29 18 NovemberRoblox to ban young children from messaging others
 - 12:33 18 NovemberStop terrorising children with sextortion, say parents, published at 12:33 18 NovemberStop terrorising children with sextortion, say parents
 - 06:28 18 NovemberDigital hub opens to train teachers in new tech, published at 06:28 18 NovemberDigital hub opens to train teachers in

```
new tech
 - 06:21 18 November'Mobile phones take away children's childhoods', published at 06:21 18 November'Mobile phones take away children's childhoods'
 - 00:13 18 November'You are under digital arrest': Inside a scam looting millions from Indians, published at 00:13 18 November'You are under digital arrest': Inside a scam looting millions from Indians
 - 16:03 17 NovemberBrazil first lady uses expletive against Elon Musk at G20 event, published at 16:03 17 NovemberBrazil first lady uses expletive against Elon Musk at G20 event
 - 07:50 17 Novemberâ€Do not petâ€: Why are robot dogs patrolling Mar-A-Lago?, published at 07:50 17 Novemberâ€Do not petâ€: Why are robot dogs patrolling Mar-A-Lago?


Categoría: Google

 - 13:05 19 NovemberGoogle reacts angrily to report it will have to sell Chrome, published at 13:05 19 NovemberGoogle reacts angrily to report it will have to sell Chrome



# Imprimir el conteo de titulares en cada categoría
print("\nConteo de Titulares por Categoría:")
for categoria, lista_titulares in titulares_categorizados.items():
    print(f"{categoria}: {len(lista_titulares)}")


Conteo de Titulares por Categoría:
Otros: 22
Google: 1
```

1.3 Seleccionando Elementos con XPath

XPath, o XML Path Language, es una herramienta versátil y robusta para navegar y seleccionar elementos dentro de documentos HTML. Aunque las bibliotecas Beautiful Soup y requests son comúnmente usadas para el web scraping, XPath ofrece un enfoque único y poderoso para extraer datos de páginas web.

¿Qué es XPath?

XPath fue diseñado originalmente para navegar documentos XML, pero es igualmente aplicable a HTML, que comparte una similitud estructural con XML. XPath te permite especificar la ubicación precisa de elementos o datos dentro de un documento HTML usando una sintaxis concisa y expresiva.

Diferenciadores Clave:

Aquí hay algunos diferenciadores clave que distinguen a XPath de otros enfoques de web scraping:

1. **Selección Granular**: XPath proporciona un control granular sobre la selección de elementos. A diferencia de Beautiful Soup, que a menudo requiere múltiples iteraciones y filtrado, XPath te permite apuntar directamente a elementos basados en sus atributos, etiquetas o posiciones dentro del documento.

2. **Navegación Jerárquica**: XPath sobresale en navegar la estructura jerárquica de documentos HTML. Te permite atravesar el árbol del documento, moviéndote hacia arriba, abajo o a través de ramas con facilidad.

3. **Consultas Precisas**: Con XPath, puedes crear consultas precisas para extraer datos específicos. Por ejemplo, puedes apuntar a elementos con atributos específicos, como seleccionar todos los elementos `<a>` con una clase particular o localizar elementos dentro de elementos padres específicos.

4. **Extracción de Texto**: La función `text()` de XPath simplifica la extracción del contenido de texto de los elementos. Esto es particularmente útil para raspar datos de texto, como titulares, párrafos o descripciones de productos.

Cómo Usar XPath:

Para utilizar XPath para el web scraping, típicamente sigues estos pasos:

1. **Enviar una Solicitud HTTP**: Usa una biblioteca como requests para enviar una solicitud GET HTTP a la página web que deseas raspar. Esto recupera el contenido HTML de la página.

2. **Parsear el HTML**: Una vez que tienes el contenido HTML, pásalo por una biblioteca como lxml o lxml.html. Este paso construye una representación estructurada de la página web que puedes navegar con XPath.

3. **Construir Expresiones XPath**: Formula expresiones XPath que apunten a los elementos o datos específicos que deseas extraer. Las expresiones XPath pueden variar en complejidad, permitiéndote adaptarte a diferentes estructuras de páginas web.

4. **Aplicar Expresiones XPath**: Aplica tus expresiones XPath al documento HTML parseado para seleccionar los elementos o datos deseados. Este proceso filtra efectivamente el contenido HTML para capturar solo lo que necesitas.

5. **Recuperar y Procesar Datos**: Recupera los elementos o datos seleccionados usando las consultas XPath y procésalos según sea necesario para tu tarea de scraping.

En resumen, XPath es una herramienta poderosa para el web scraping que ofrece una selección precisa y eficiente de elementos dentro de documentos HTML. Mientras que bibliotecas como Beautiful Soup y requests son valiosas, XPath proporciona una capa adicional de control y flexibilidad, haciéndolo una elección valiosa para proyectos de scraping avanzados.

Entendiendo la Sintaxis de XPath

- **Ruta Absoluta (`/`)**:
  - Usar una barra inclinada indica una ruta absoluta desde el elemento raíz.
  - Ejemplo: `xpath('/html/body/p')` selecciona todos los elementos de párrafo (`<p>`) directamente bajo el `<body>` dentro del elemento raíz `<html>`.
- **Ruta Relativa (`//`)**:
  - Las dobles barras inclinadas indican una ruta relativa, lo que significa que la selección puede comenzar en cualquier lugar de la jerarquía del documento.
  - Ejemplo: `xpath('//a/div')` encuentra todos los elementos `<div>` que son descendientes de etiquetas `<a>`, independientemente de su ubicación específica en el documento.
- **Comodines (`*`)**:
  - El asterisco actúa como un comodín, representando cualquier elemento.
  - Ejemplo: `xpath('//a/div/*')` selecciona todos los elementos que son hijos de etiquetas `<div>` bajo etiquetas `<a>`, en cualquier lugar del documento.
  - Otro ejemplo: `xpath('/*/*/div')` encuentra elementos `<div>` que están en el segundo nivel de la jerarquía desde la raíz.
- **Seleccionando Elementos Específicos (Usando Corchetes)**:
  - Si una selección devuelve múltiples elementos, puedes especificar cuál seleccionar usando corchetes.
  - Ejemplo: `xpath('//a/div[1]')` selecciona el primer `<div>` en el conjunto; `xpath('//a/div[last()]')` selecciona el último `<div>`.

Trabajando con Atributos

- **Seleccionando Atributos (`@`)**:
  - El símbolo `@` se utiliza para trabajar con atributos de elementos.
  - Ejemplo: `xpath('//@name')` selecciona todos los atributos llamados 'name' en el documento.
  - Para seleccionar elementos `<div>` con un atributo 'name': `xpath('//div[@name]')`.
  - Para seleccionar elementos `<div>` sin ningún atributo: `xpath('//div[not(@*)]')`.
  - Para encontrar elementos `<div>` con un valor de atributo 'name' específico: `xpath('//div[@name="chachiname"]')`.

Utilizando Funciones Incorporadas

- XPath viene con varias funciones incorporadas para ayudar en la selección de elementos.
  - `contains()`: Selecciona elementos que contienen una subcadena específica. Ejemplo: `xpath('//*[contains(name(),'iv')]')`.

  - `count()`: Se utiliza para selección condicional basada en el conteo de hijos. Ejemplo: `xpath('//*[count(div)=2]')`.

Combinando Rutas y Seleccionando Parientes

- **Combinando Rutas (`|`)**:
  - Usa el símbolo de tubería para combinar rutas, funcionando como un operador OR.
  - Ejemplo: `xpath('/div/p|/div/a')` selecciona elementos que coinciden con `div/p` o `div/a`.
- **Seleccionando Parientes**:
  - Puedes referirte a varios aspectos relacionales como padre, ancestros, hijos o descendientes.
  - Ejemplo: `xpath('//div/div/parent::*')` selecciona los elementos padres de las rutas `div/div`.

Entender XPath es esencial para el web scraping efectivo, ya que permite un direccionamiento y extracción precisos de datos basados en la estructura de una página web.

```
from lxml import html
import requests

# URL del sitio web que queremos raspar
# Para este ejemplo, usamos la sección de tecnología de la BBC
url = "https://www.bbc.co.uk/news/technology"

# Enviar una solicitud GET a la URL para obtener el contenido HTML de
la página web
response = requests.get(url)

# Analizar el contenido HTML de la página
# El método html.fromstring construye un documento HTML lxml a partir
del texto de respuesta
tree = html.fromstring(response.content)

# Usar XPath para seleccionar elementos específicos
# En este caso, intentaremos extraer el contenido textual de la página
# La expresión XPath utilizada aquí captura todo el contenido de texto
dentro de la estructura HTML
# Nota: La XPath exacta puede variar según la estructura del HTML de
la página
titulares = tree.xpath('//text()')

# Imprimir cada titular extraído
# La función text() en XPath extrae el contenido de texto de los
elementos seleccionados
# for titular in titulares:
#     print(titular.strip())

# Este script imprime todo el contenido textual de las etiquetas <h3>
```

```
encontradas en la página de tecnología de la BBC
# Demuestra cómo usar XPath para extraer información específica de una página web
```

2.0. Comenzando con Selenium

Selenium es una herramienta poderosa utilizada principalmente para automatizar navegadores web. Se utiliza ampliamente en áreas como el web scraping, pruebas automatizadas y automatización de tareas administrativas basadas en web.

Introducción a Selenium Sin Geckodriver

Tradicionalmente, Selenium funciona en conjunto con un controlador específico para cada navegador, como geckodriver para Firefox o chromedriver para Chrome. Sin embargo, desarrollos recientes han permitido que ciertos navegadores sean controlados directamente por Selenium sin la necesidad de un controlador adicional:

- **Chrome**: Versiones recientes de Google Chrome pueden ser controladas directamente por Selenium a través del Protocolo Chrome DevTools. Esto simplifica el proceso de configuración ya que no necesitas descargar y configurar chromedriver por separado.
- **Microsoft Edge**: Similar a Chrome, el navegador Edge (versión Chromium) también puede ser automatizado directamente usando Selenium con sus capacidades de controlador integradas.

Este enfoque de usar Selenium sin un controlador adicional agiliza las tareas de automatización del navegador, haciéndolo más accesible y fácil de configurar, especialmente para principiantes y aquellos que buscan configurar rápidamente interacciones automatizadas del navegador.

2.1 Conceptos Básicos de Selenium WebDriver

Entendiendo WebDriver

WebDriver es un componente clave del conjunto de herramientas Selenium. Actúa como una interfaz para interactuar con el navegador web, permitiéndote controlarlo programáticamente. WebDriver puede realizar operaciones como abrir páginas web, hacer clic en botones, ingresar texto en formularios y extraer datos de páginas web.

Funciones Clave de WebDriver

- **Abrir una Página Web**: WebDriver puede navegar a una URL específica.
- **Localizar Elementos**: Puede encontrar elementos en una página web basados en sus atributos (como ID, nombre, XPath).
- **Interactuar con Elementos**: WebDriver puede simular acciones como hacer clic en botones, escribir texto y enviar formularios.

Interactuando con Elementos Web

Puedes localizar e interactuar con elementos en una página web usando varios métodos proporcionados por WebDriver. La elección del método depende de los atributos de los elementos HTML que estás apuntando.

- **find_element_by_id**: Localiza un elemento por su ID único.
- **find_element_by_name**: Encuentra un elemento por su atributo de nombre.
- **find_element_by_xpath**: Utiliza consultas XPath para localizar elementos, proporcionando una forma poderosa de navegar el DOM.

```
### Selenium WebDriver Python Examples
#### Example 1: Opening a Web Page

# Este ejemplo muestra cómo abrir una página web utilizando Selenium
WebDriver.

from selenium import webdriver

# Inicializar el WebDriver de Chrome
# Se crea una instancia del controlador de Chrome con opciones
predeterminadas
options = webdriver.ChromeOptions()
driver = webdriver.Chrome(options=options)

# Abrir una página web
# Aquí, se abre la página oficial de Python
driver.get("https://www.python.org")

# Cerrar el navegador
driver.quit()
```

Ejemplo 2: Web Scraping de Salarios de Jugadores de la NBA

Este ejemplo demuestra cómo raspar datos de salarios de jugadores de la NBA de un sitio web usando Selenium en Python. Es una ilustración práctica de cómo Selenium puede ser utilizado para automatizar la navegación web y extraer datos específicos de páginas web. El script navega a través de diferentes páginas para cada temporada de la NBA, recopila los nombres de los jugadores y sus correspondientes salarios, y organiza estos datos en un DataFrame de pandas para cada año desde 1990 hasta 2018. Este es un ejemplo útil para aprender cómo manejar elementos web, extraer texto y manejar datos usando pandas en Python.

```
# Importando las bibliotecas necesarias
from selenium import webdriver  # Usado para automatizar la
interacción con navegadores web
from selenium.webdriver.common.by import By  # Ayuda a localizar
elementos en páginas web
import pandas as pd  # Biblioteca Pandas para manipulación y análisis
de datos
```

```
# Creando un DataFrame vacío con columnas especificadas
# Este DataFrame se usará para almacenar los datos extraídos
df = pd.DataFrame(columns=['Player', 'Salary', 'Year'])

# Inicializando el WebDriver de Chrome
# Esto abre una ventana del navegador Chrome para realizar el scraping
web
driver = webdriver.Chrome()

# Iterando a través de los años 2017 a 2018
for yr in range(2017, 2019):
    # Construyendo la URL para cada año agregando el rango de años a
la URL base
    page_num = str(yr) + '-' + str(yr + 1) + '/'
    url = 'https://hoopshype.com/salaries/players/' + page_num
    driver.get(url)  # Navegando a la URL construida en el navegador

    # Encontrando todos los elementos de nombre de jugador en la
página usando su XPATH
    # XPATH es una sintaxis usada para navegar a través de elementos y
atributos en un documento XML
    # players = driver.find_elements(By.XPATH, '//td[@class="name"]')
    players_container = driver.find_element(By.XPATH,
'//div[@class="pinned"]')
    players = players_container.find_elements(By.XPATH,
'.//td[@class="name"]/a[@href]')

    # De manera similar, encontrando todos los elementos de salario en
la página usando su XPATH
    # salaries = driver.find_elements(By.XPATH, '//td[@class="hh-
salaries-sorted"]')
    salaries_table = driver.find_element(By.XPATH,
'//table[contains(@class, "hh-salaries-ranking-table")]')
    salaries = salaries_table.find_elements(By.XPATH,
'.//td[@class="hh-salaries-sorted"]')

    # Extrayendo el texto de cada elemento de jugador y almacenándolo
en una lista
    players_list = [player.text for player in players]

    # Extrayendo el texto de cada elemento de salario y almacenándolo
en una lista
    salaries_list = [salary.text for salary in salaries]

    # Emparejando el nombre de cada jugador con su salario y año
usando la función zip
    data_tuples = list(zip(players_list[1:], salaries_list[1:]))

    # Creando un DataFrame temporal para el año actual
    # Este DataFrame contiene los nombres de los jugadores, sus
```

```
salarios y el año
    temp_df = pd.DataFrame(data_tuples, columns=['Player', 'Salary'])
    temp_df['Year'] = yr

    # Agregando el DataFrame temporal al DataFrame principal
    # ignore_index=True se usa para garantizar que el índice continúe
correctamente en el DataFrame principal
    #df = df.append(temp_df, ignore_index=True)
    # Concatenando el DataFrame temporal al DataFrame principal
    df = pd.concat([df, temp_df], ignore_index=True)

# Cerrando el WebDriver después de completar el scraping
# Esto es importante para liberar recursos y evitar posibles fugas de
memoria
driver.close()

df
```

	Player	Salary	Year
0	LeBron James	$34,682,550	2017
1	Paul Millsap	$33,285,709	2017
2	Gordon Hayward	$30,769,231	2017
3	Blake Griffin	$29,727,900	2017
4	Kyle Lowry	$29,512,900	2017
...	...	...	...
1155	Michael Frazier	$47,370	2018
1156	Tahjere McCall	$47,370	2018
1157	Mitchell Creek	$47,370	2018
1158	Isaac Humphries	$47,370	2018
1159	Jordan Sibert	$47,370	2018

[1160 rows x 3 columns]

Business challenge: **Análisis del Mercado de Alquileres en Barcelona: Un Proyecto de Web Scraping y Visualización de Datos**

Objetivo:

El objetivo es desarrollar un web scraper basado en Python para extraer datos de propiedades de alquiler de Idealista para diferentes barrios de Barcelona. Estos datos serán analizados usando Power BI para descubrir insights sobre el mercado de alquileres de la ciudad.

Alcance:

- **Web Scraping**: Extraer puntos clave de datos como precios de alquiler, tamaño de la propiedad, número de habitaciones y ubicaciones de barrios de Idealista.

- **Análisis y Visualización de Datos**: Analizar los datos raspados para identificar tendencias y patrones, luego visualizar estos hallazgos usando Power BI.

Pasos y Aplicación de la Metodología Ágil:

1. Iniciación y Planificación del Proyecto (Sprint 0)

- **Configuración del Equipo**: Formar equipos multifuncionales con roles como Scrum Master, Propietario del Producto y Analistas de Datos.
- **Recolección de Requisitos**: Definir los puntos específicos de datos a ser raspados de Idealista.
- **Selección de Herramientas**: Elegir herramientas apropiadas para el web scraping (por ejemplo, Python con bibliotecas como BeautifulSoup, Selenium) y para la visualización de datos (Power BI).
- **Creación del Backlog**: Crear un backlog de producto que comprenda historias de usuario (por ejemplo, "Como analista de datos, quiero raspar precios de alquiler para poder analizar el alquiler promedio en cada barrio").

2. Ejecución del Sprint

- **Planificación del Sprint**: Desglosar el backlog en tareas más pequeñas y manejables para ser completadas en cada sprint (por ejemplo, configurar el entorno de scraping, diseñar el modelo de datos, etc.).
- **Reuniones Diarias**: Realizar breves reuniones diarias para discutir el progreso, obstáculos y próximos pasos.
- **Desarrollo y Pruebas**: Realizar desarrollo iterativo, con pruebas regulares para asegurar la precisión y fiabilidad de los datos.
- **Revisión del Sprint**: Al final de cada sprint, revisar el trabajo completado y demostrar la funcionalidad.
- **Retrospectiva del Sprint**: Reflexionar sobre el proceso del sprint para identificar mejoras para el próximo sprint.

3. Fase de Web Scraping

- Implementar scripts de web scraping para extraer los datos requeridos de Idealista.
- Asegurar el cumplimiento de las políticas de web scraping de Idealista y consideraciones legales.

4. Análisis y Visualización de Datos

- Limpiar y preprocesar los datos raspados para el análisis.
- Usar Power BI para crear dashboards interactivos y visualizaciones que resalten aspectos clave del mercado de alquileres en Barcelona.

5. Revisión Final y Presentación

- Compilar los hallazgos e insights en un informe comprensivo.
- Presentar el análisis de datos y visualizaciones a los stakeholders o en un entorno de clase.

Entregables:

- Código fuente para el script de web scraping.
- Archivos de dashboard de Power BI.
- Informe final detallando la metodología, hallazgos e insights.

Resultados de Aprendizaje:

- Aplicación práctica de web scraping y análisis de datos.
- Experiencia en el uso de metodología Ágil para la gestión de proyectos.
- Mejora de la colaboración y habilidades de trabajo en equipo.
- Competencia en el uso de Python para la recolección de datos y Power BI para la visualización de datos.

Respuestas

```
import requests
from bs4 import BeautifulSoup

# URL de la sección de noticias de tecnología de la BBC
url = "https://www.bbc.co.uk/news/technology"

# Enviar una solicitud GET y analizar el contenido HTML
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Definir categorías y palabras clave asociadas
categories = {
    'Apple': ['Apple', 'iPhone', 'iPad'],
    'Microsoft': ['Microsoft', 'Windows', 'Bill Gates'],
    'Google': ['Google', 'Android', 'Alphabet'],
    'Other': []  # Agregar una categoría para titulares que no
coincidan con ninguna palabra clave
}

# Función para determinar la categoría de un titular
def categorize_headline(headline):
    # Iterar sobre cada categoría y sus palabras clave
    for category, keywords in categories.items():
        # Buscar si alguna palabra clave está en el titular
        for keyword in keywords:
            if keyword.lower() in headline.lower():  # Usar .lower()
para hacer la búsqueda insensible a mayúsculas/minúsculas
                return category
    return 'Other'  # Si no se encuentra ninguna palabra clave,
asignar la categoría 'Other'

# Raspar y procesar los titulares
# Buscar todas las etiquetas 'h3' que contienen los titulares
headlines = soup.find_all('h3')
```

```python
# Crear un diccionario para contar los titulares en cada categoría
category_counts = {category: 0 for category in categories.keys()}

# Procesar cada titular extraído
for h in headlines:
    headline_text = h.get_text().strip()  # Extraer el texto del titular y eliminar los espacios extra
    category = categorize_headline(headline_text)  # Determinar la categoría del titular
    category_counts[category] += 1  # Incrementar el contador de la categoría correspondiente
    print(f"Titular: {headline_text}\nCategoría: {category}\n")
```

```
Titular: 06:00Logan Paul accused of misleading fans over crypto investments, published at 06:00Logan Paul accused of misleading fans over crypto investments
Categoría: Other

Titular: 04:27Who has joined Trump's team so far?, published at 04:27Who has joined Trump's team so far?
Categoría: Other

Titular: 02:03No plans to join Bluesky yet, Starmer says, published at 02:03No plans to join Bluesky yet, Starmer says
Categoría: Other

Titular: 22:48 19 NovemberTrump joins Musk to watch Starship test launch. Video, 00:01:03, published at 22:48 19 NovemberTrump joins Musk to watch Starship test launch
Categoría: Other

Titular: 19:00 19 NovemberTeenager 'picked on by teachers' before death - inquest, published at 19:00 19 NovemberTeenager 'picked on by teachers' before death - inquest
Categoría: Other

Titular: 16:34 19 NovemberAI cameras catch drivers on phone or without belts, published at 16:34 19 NovemberAI cameras catch drivers on phone or without belts
Categoría: Other

Titular: 15:19 19 NovemberInstagram testing tool that resets all recommended posts , published at 15:19 19 NovemberInstagram testing tool that resets all recommended posts
Categoría: Other

Titular: 13:05 19 NovemberGoogle reacts angrily to report it will have to sell Chrome, published at 13:05 19 NovemberGoogle reacts angrily to report it will have to sell Chrome
Categoría: Google
```

Titular: 11:45 19 NovemberSexual offence rumours online unfounded - police, published at 11:45 19 NovemberSexual offence rumours online unfounded - police
Categoría: Other

Titular: 08:49 19 NovemberMan who binned Bitcoin says it's now worth Â£500m. Video, 00:00:36, published at 08:49 19 NovemberMan who binned Bitcoin says it's now worth Â£500m
Categoría: Other

Titular: 08:39 19 NovemberMan says his binned Bitcoin fortune now worth Â£500m, published at 08:39 19 NovemberMan says his binned Bitcoin fortune now worth Â£500m
Categoría: Other

Titular: 01:15 19 NovemberBritish Airways says 'tech issue' resolved after delays, published at 01:15 19 NovemberBritish Airways says 'tech issue' resolved after delays
Categoría: Other

Titular: 00:41 19 November'We booked a Â£99 stargazing break - all we got was an empty field', published at 00:41 19 November'We booked a Â£99 stargazing break - all we got was an empty field'
Categoría: Other

Titular: 00:00 19 NovemberChocolate makers stoke boom for Indian cocoa beans, published at 00:00 19 NovemberChocolate makers stoke boom for Indian cocoa beans
Categoría: Other

Titular: 17:55 18 NovemberThe Game Awards 2024: Astro Bot and Final Fantasy lead nominations , published at 17:55 18 NovemberThe Game Awards 2024: Astro Bot and Final Fantasy lead nominations
Categoría: Other

Titular: 17:27 18 NovemberMum 'not told about outbursts' before daughter's death, published at 17:27 18 NovemberMum 'not told about outbursts' before daughter's death
Categoría: Other

Titular: 16:29 18 NovemberRoblox to ban young children from messaging others, published at 16:29 18 NovemberRoblox to ban young children from messaging others
Categoría: Other

Titular: 12:33 18 NovemberStop terrorising children with sextortion, say parents, published at 12:33 18 NovemberStop terrorising children with sextortion, say parents
Categoría: Other

```
Titular: 06:28 18 NovemberDigital hub opens to train teachers in new tech, published at 06:28 18 NovemberDigital hub opens to train teachers in new tech
Categoría: Other

Titular: 06:21 18 November'Mobile phones take away children's childhoods', published at 06:21 18 November'Mobile phones take away children's childhoods'
Categoría: Other

Titular: 00:13 18 November'You are under digital arrest': Inside a scam looting millions from Indians, published at 00:13 18 November'You are under digital arrest': Inside a scam looting millions from Indians
Categoría: Other

Titular: 16:03 17 NovemberBrazil first lady uses expletive against Elon Musk at G20 event, published at 16:03 17 NovemberBrazil first lady uses expletive against Elon Musk at G20 event
Categoría: Other

Titular: 07:50 17 Novemberâ€Do not petâ€: Why are robot dogs patrolling Mar-A-Lago?, published at 07:50 17 Novemberâ€Do not petâ€: Why are robot dogs patrolling Mar-A-Lago?
Categoría: Other


# Imprimir el recuento de titulares por categoría
print("Recuento de titulares por categoría:")
for category, count in category_counts.items():
    print(f"{category}: {count}")

Recuento de titulares por categoría:
Apple: 0
Microsoft: 0
Google: 1
Other: 22
```