Web Scrapping

Ariet Michal

EIP POSGRADOS / HACKING

INTRODUCCIÓN

En esta actividad se desarrollaron dos scripts independientes para realizar tareas de web scraping y automatización de navegadores utilizando Python. El primer script utiliza las librerías requests y BeautifulSoup para interactuar con Wikipedia, permitiendo realizar búsquedas automatizadas y extraer el primer párrafo relevante de los artículos encontrados. Este enfoque demuestra cómo trabajar con contenido estático mediante el análisis directo del HTML de una página.

El segundo script emplea Selenium para automatizar búsquedas en YouTube. A través de la simulación de interacciones con el navegador, como la entrada de texto en formularios y la extracción de títulos de videos, se evidencia cómo manejar sitios dinámicos que requieren interacción con elementos de la interfaz.

Ambos ejercicios cumplen el objetivo de explorar herramientas de scraping y automatización, aplicando diferentes técnicas según las características de los sitios web.

REALIZAR UN SCRIPT CON LOS MÓDULOS DE REQUESTS Y BEAUTIFULSOUP PARA REALIZAR UN EJERCICIO DE RECOLECCIÓN DE INFORMACIÓN

Importación de librerías y configuración inicial

En esta sección, se preparan las herramientas y configuraciones básicas necesarias para realizar solicitudes HTTP a Wikipedia, analizar las respuestas y visualizar los datos extraídos. Este es el punto de partida para el script de web scraping.

Importación de módulos:

- requests: Para realizar solicitudes HTTP y obtener contenido web.
- BeautifulSoup: De la biblioteca bs4, permite analizar y extraer datos de contenido HTML o XML.
- matplotlib.pyplot: Para realizar visualizaciones gráficas a partir de los datos recolectados.

Definición de constantes:

- TIMEOUT: Define el tiempo máximo de espera (en segundos) para las solicitudes HTTP antes de que se produzca un error. En este caso, es de 10 segundos.
- BASE_WIKIPEDIA: Especifica la URL base del sitio web de Wikipedia en español.
- SEARCH_URL: Construye la URL de búsqueda en Wikipedia, usando la constante BASE_WIKIPEDIA.

```python
import requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
from typing import List, Dict
from pathlib import Path
import json

# Constantes
TIMEOUT = 10
BASE_WIKIPEDIA = "https://es.wikipedia.org"
SEARCH_URL = f"{BASE_WIKIPEDIA}/w/index.php"
```

Funciones principales del Script

En esta sección se define la función buscar_en_wikipedia, que realiza una consulta a la API de Wikipedia para buscar un término y obtener la URL del primer resultado relevante. Se configuran los parámetros de búsqueda, se hace la solicitud HTTP con un tiempo de espera definido, y se utiliza BeautifulSoup para analizar el HTML y extraer el enlace del primer resultado. Si no se encuentran resultados, se intenta obtener una redirección canónica. Finalmente, la función devuelve la URL del artículo o una cadena vacía si no hay resultados.

```python
def buscar_en_wikipedia(termino: str) -> str:
    """
    Dado un término de búsqueda, realiza una consulta en la versión en inglés de Wikipedia
    y devuelve la URL del primer resultado (o cadena vacía si no hay resultados).
    """
    params = {
        "search": termino,
        "ns0": 1  # buscar solo en artículos (namespace 0)
    }

    try:
        resp = requests.get(SEARCH_URL, params=params, timeout=TIMEOUT)
        resp.raise_for_status()
    except requests.RequestException as e:
        print(f"Error de conexión o de estado HTTP en la búsqueda: {e}")
        return ""

    soup = BeautifulSoup(resp.text, "html.parser")

    # 1) Intentar obtener la lista de resultados (en div.mw-search-result-heading)
    resultado = soup.select_one("div.mw-search-result-heading a")
    if not resultado:
        # Puede que no haya resultados o que Wikipedia te redirija a un artículo directamente
        link_canonical = soup.select_one("link[rel='canonical']")
        if link_canonical and link_canonical.has_attr("href"):
            return link_canonical["href"].strip()  # URL canónica del artículo
        else:
            return ""  # Sin resultados ni redirección clara

    # 2) Construir la URL con el href del primer resultado
    href = resultado.get("href", "")
    if href.startswith("/wiki/"):
        return BASE_WIKIPEDIA + href
    return ""
```

Luego se presenta la función obtener_primer_parrafo, que toma una URL de Wikipedia y busca el primer párrafo significativo del artículo. Si la página es de desambiguación (detectada por texto o estructura), sigue el enlace más relevante y vuelve a llamar a sí misma recursivamente. Si no es una página de desambiguación, selecciona los párrafos del contenido y retorna el primero que contenga texto útil. La función también maneja errores de conexión y URLs no válidas, devolviendo mensajes claros en esos casos.

```python
def obtener_primer_parrafo(url_articulo: str) -> str:
    """
    Dada la URL de un artículo de Wikipedia, devuelve el primer párrafo no vacío de su contenido.

    - Si la página es de desambiguación (o incluye "may refer to"),
      busca el primer enlace <a href="/wiki/..." y visita esa página.
    - Devuelve un mensaje de error si no encuentra nada útil.
    """
    if not url_articulo:
        return "No se encontró un enlace válido al artículo."

    try:
        resp = requests.get(url_articulo, timeout=TIMEOUT)
        resp.raise_for_status()
    except requests.RequestException as e:
        return f"Error de conexión al obtener el artículo: {e}"

    # Chequear si es una página de desambiguación
    # 1) Si el texto incluye "may refer to", o
    # 2) si la URL contiene "disambiguation"
    if ("may refer to" in resp.text.lower()) or ("disambiguation" in resp.url.lower()):
        print("Página de desambiguación detectada. Intentando seguir el primer enlace relevante...")
        soup = BeautifulSoup(resp.text, "html.parser")
        # Normalmente los enlaces de desambiguación están en listas <ul><li><a>
        primer_enlace = soup.select_one("#mw-content-text ul li a[href^='/wiki/']")
        if primer_enlace:
            new_link = BASE_WIKIPEDIA + primer_enlace["href"]
            # Volvemos a llamar a la misma función para obtener el primer párrafo del nuevo enlace
            return obtener_primer_parrafo(new_link)
        else:
            return "Página de desambiguación sin enlaces válidos."
```

A continuación, la función descripcion_wikipedia, que actúa como la función principal del script. Combina las dos funciones anteriores (buscar_en_wikipedia y obtener_primer_parrafo) para realizar todo el flujo: buscar un término, obtener la URL del primer resultado y extraer el primer párrafo del artículo. Si no se encuentra contenido para el término proporcionado, se devuelve un mensaje indicando que no hay resultados. Esta función coordina todo el proceso de interacción con Wikipedia y garantiza un resultado final claro.

```
    # Si no es desambiguación, extraemos el primer párrafo
    soup = BeautifulSoup(resp.text, "html.parser")
    parrafos = soup.select("div#mw-content-text p")
    for p in parrafos:
        texto = p.get_text().strip()
        if texto:
            return texto

    return "No se encontró un párrafo con contenido en el artículo."


def descripcion_wikipedia(termino: str) -> str:
    """
    Función principal que:
    1. Busca el término en la página de resultados de Wikipedia (buscar_en_wikipedia).
    2. Obtiene la URL del primer resultado.
    3. Devuelve el primer párrafo útil de esa página (obtener_primer_parrafo),
       resolviendo automáticamente páginas de desambiguación.
    """
    url_final = buscar_en_wikipedia(termino)
    if not url_final:
        return f"No se encontraron resultados para '{termino}' en Wikipedia."

    return obtener_primer_parrafo(url_final)
```

Ejecución del script y resultados

Esta parte del código muestra cómo se utiliza la función descripcion_wikipedia para buscar y extraer descripciones de varios términos de Wikipedia. Los pasos realizados son los siguientes:

Definición de términos de búsqueda:

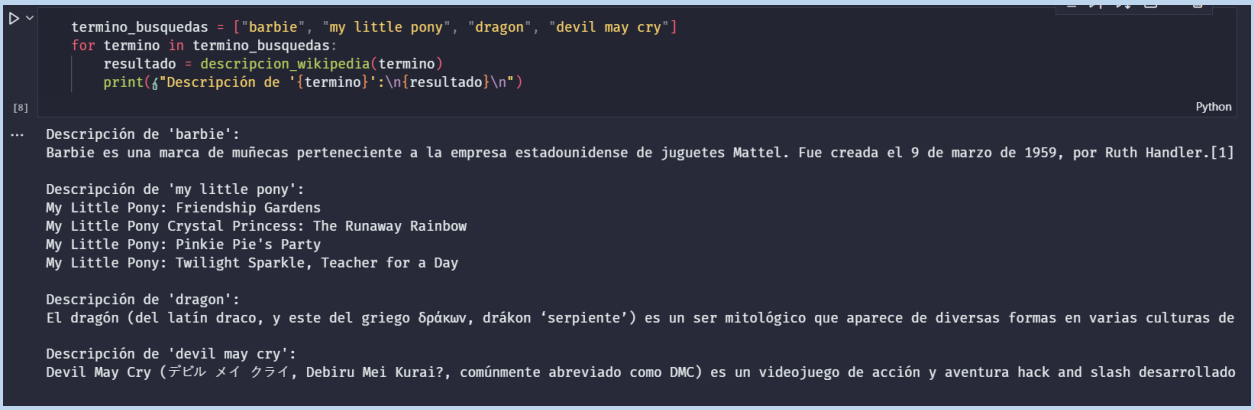
- Se crea una lista de términos (termino_busquedas) que incluye palabras clave como "barbie", "my little pony", "dragon", y "devil may cry".

Bucle para procesar cada término:

- Se itera sobre cada término de la lista, llamando a la función descripcion_wikipedia para obtener la descripción correspondiente.
- El resultado de cada término se imprime en formato legible, precedido por la etiqueta "Descripción de '{término}':".

Resultados generados:

- Para cada término, el script devuelve el primer párrafo útil del artículo correspondiente en Wikipedia.
- Si no se encuentra contenido, retornará un mensaje indicando la ausencia de resultados.



```
termino_busquedas = ["barbie", "my little pony", "dragon", "devil may cry"]
for termino in termino_busquedas:
    resultado = descripcion_wikipedia(termino)
    print(f"Descripción de '{termino}':\n{resultado}\n")
```

[8] Python

```
Descripción de 'barbie':
Barbie es una marca de muñecas perteneciente a la empresa estadounidense de juguetes Mattel. Fue creada el 9 de marzo de 1959, por Ruth Handler.[1]

Descripción de 'my little pony':
My Little Pony: Friendship Gardens
My Little Pony Crystal Princess: The Runaway Rainbow
My Little Pony: Pinkie Pie's Party
My Little Pony: Twilight Sparkle, Teacher for a Day

Descripción de 'dragon':
El dragón (del latín draco, y este del griego δράκων, drákon ‘serpiente’) es un ser mitológico que aparece de diversas formas en varias culturas de

Descripción de 'devil may cry':
Devil May Cry (デビル メイ クライ, Debiru Mei Kurai?, comúnmente abreviado como DMC) es un videojuego de acción y aventura hack and slash desarrollado
```

REALIZAR UN SCRIPT CON SELENIUM PARA REALIZAR UN EJERCICIO DE AUTOMATIZACIÓN DEL NAVEGADOR

Automatización de búsqueda en YouTube

1. Inicializar el navegador con webdriver.Chrome() y abrir la página principal de YouTube (driver.get()).
2. Localizar la barra de búsqueda usando su atributo name (search_query).
3. Escribir el término de búsqueda y simular la pulsación de Enter (Keys.ENTER).

Extracción de títulos de los videos

4. Identificar los elementos de los títulos de los videos mediante su atributo ID (video-title).
5. Extraer el texto de cada título y almacenarlo en una lista.
6. Cerrar el navegador con driver.quit() al finalizar el proceso.

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

def buscar_en_youtube(termino):
    """
    Abre YouTube en un navegador automatizado, realiza la búsqueda del término
    y extrae los títulos de los primeros resultados.

    :param termino: (str) Término que queremos buscar en YouTube.
    :return: (list) Lista con los títulos de los primeros videos encontrados.
    """
    # Inicializamos el driver (en este ejemplo, Chrome)
    # Asegúrate de que el driver (chromedriver) esté instalado y en el PATH.
    driver = webdriver.Chrome()
```

```
    try:
        driver.get("https://www.youtube.com/")
        time.sleep(2)  # Esperamos para que cargue el sitio

        # Si aparece un banner de cookies, podrías cerrarlo con un .click() si es necesario

        # Localizamos la barra de búsqueda por NAME
        barra_busqueda = driver.find_element(By.NAME, "search_query")
        # Escribimos el término y presionamos Enter
        barra_busqueda.send_keys(termino)
        barra_busqueda.send_keys(Keys.ENTER)
        time.sleep(3)  # Esperamos a que aparezcan los resultados

        # Buscamos los elementos con ID "video-title"
        titulos_elementos = driver.find_elements(By.ID, "video-title")

        # Obtenemos el texto de cada elemento
        titulos = [titulo.text for titulo in titulos_elementos]

        return titulos
    finally:
        # Cerramos el navegador al final
        driver.quit()
```

Resultados de la búsqueda automatizada en YouTube

En esta parte del código se demuestra el uso de la función buscar_en_youtube para buscar un término específico en YouTube y mostrar los títulos de los videos encontrados.

Definición del término de búsqueda:

- Se asigna el término "canciones de rock" a la variable termino_busqueda_youtube, indicando lo que se desea buscar en YouTube.

Ejecución de la función:

- La función buscar_en_youtube se llama con el término definido, y los títulos de los videos encontrados se almacenan en la lista resultados.

Visualización de los resultados:

- Se imprime un encabezado que indica el término buscado.

- A través de un bucle for, se enumeran los títulos de los videos, mostrando su índice y el texto correspondiente.

```
# Ejemplo de uso:
termino_busqueda_youtube = "canciones de rock"  # Cambia a lo que desees buscar
resultados = buscar_en_youtube(termino_busqueda_youtube)

print(f"Resultados de la búsqueda para '{termino_busqueda_youtube}':")
for idx, titulo in enumerate(resultados, start=1):
    print(f"{idx}. {titulo}")
```

[10]

```
Resultados de la búsqueda para 'canciones de rock':
1. Nirvana, Led Zeppelin, Bon Jovi, Aerosmith, U2, ACDC 🤘 Classic Rock Songs 70s 80s 90s Full Album
2. Classic Rock Songs 70s 80s 90s Full Album - Queen, Eagles, Pink Floyd, Def Leppard, Bon Jovi, ACDC
3. Enrique Bunbury, Caifanes, Enanitos Verdes, Maná, Soda Estereo Rock en Espanol de los 80 y 90
4. Metallica, Queen, Nirvana, Guns N Roses, Bon Jovi, ACDC 🔥 Best Classic Rock Songs 70s 80s 90s
5. Kiss - I Was Made For Lovin' You
6. AC/DC - Back In Black (Official 4K Video)
7. Queen - We Will Rock You (Official Video)
8. 👉 MIX ROCK en ESPAÑOL de los 80 y 90 🎵🎵🎵 CLÁSICOS DE LOS 80 & 90 Dj Suarez PUCALLPA
9. 25/12 NUEVO ÁLBUM DE BUEN PUNTO 💿✨ #conlamusicaatodaspartes #nuevodisco #musicachilena #music
10. Tatiana - Navidad Rock
11. 100 CANCIONES ICONICAS DEL POP ROCK (1960-2023)
12. SALVA 1 CANCION de ROCK #2🤦‍♂️🎶 🤟 ¿Cuál es tu Canción Favorita?
13. Rock para Viajar ✈️ (Soda Stereo, Enanitos Verdes, Prisioneros, Git, Virus, Vilma Palma, Fito Paez)
14. Soda Stereo - De Música Ligera (Official Video)
15. Guns N' Roses - Sweet Child O' Mine (Official Music Video)
16. Nirvana - Smells Like Teen Spirit (Official Music Video)
17. Bon Jovi - It's My Life (Official Music Video)
18. Mägo de Oz- Fiesta pagana 2.0 (Videoclip oficial)
19. Enganchado Rock Nacional Argentino #1
20. Gorillaz - Clint Eastwood (Official Video)
```

CONCLUSIÓN

Los ejercicios realizados muestran cómo las herramientas de Python pueden ser utilizadas para automatizar tareas de recolección de información en la web. Por un lado, el script de Wikipedia destaca por su simplicidad y eficiencia para trabajar con páginas de contenido estático, mientras que el script de YouTube resalta la versatilidad de Selenium al interactuar con sitios dinámicos que requieren interacción con un navegador.

Estos ejercicios ofrecen una introducción práctica a técnicas fundamentales de web scraping y automatización, reforzando el entendimiento de cómo aplicar diferentes enfoques según las necesidades específicas del caso. Además, destacan la importancia de realizar estas actividades de manera ética y respetando los términos de uso de los sitios web. La actividad, en conjunto, permitió fortalecer habilidades prácticas y teóricas en el uso de herramientas de scraping y automatización con Python.

GITHUB

https://github.com/Zhanya001/task_haking
