Guion de Prácticas

Teoría de la Computación
3º grado en Ingeniería Matemática
Universidad CEU San Pablo
Juan Luis Sánchez Toural

Abril 2025

PROYECTO DE SOFTWARE: WEB-SCRAPING

Entrega obligatoria: Sí
Fecha límite de entrega:
6 de Mayo de 2025 (Entrega intermedia)
30 de Mayo de 2025 (Entrega final)
PESO: 30% de la nota final de la asignatura

Esta práctica es de entrega obligatoria. La sección 4 recoge instrucciones sobre qué elementos deben ser entregados y el procedimiento a seguir. La fecha límite para la entrega de la práctica está establecida en este enunciado. No se aceptarán entregas a partir de dicha fecha.

1. Introducción

El objetivo principal de este proyecto de programación es ilustrar la aplicación práctica de los conceptos teóricos revisados en la asignatura. Concretamente, se centrará en las características propias de los analizadores léxicos (definidos mediante autómatas finitos o expresiones regulares). Estos analizadores son utilizados, fundamentalmente, como una de las primeras etapas en los procesos de compilación. En contraposición, también se trabajará con otros analizadores más avanzados que permiten examinar textos estructurados. Como hilo conductor de ambas aproximaciones, se utilizarán las técnicas de Web-Scraping.

A continuación, en la sección 2 se establece el contexto y los requisitos que debe cumplir el proyecto software. La sección 3 especifica cuestiones propias de la evaluación de la práctica. Posteriormente, la sección 4 indica los elementos que deben entregarse y cómo debe realizarse el envío al profesor. Finalmente, la sección 5 establece los mecanismos por los que se resolverán las dudas correspondientes a esta práctica.

2. Enunciado

Web-Scraping es el término utilizado para aquellas técnicas que extraen datos de los sitios web. Habitualmente, se implementan en forma de programas conocidos como *bots* o *web crawlers*. Estos programas analizan las páginas web, recopilando la información para la que han sido diseñados y almacenándola, posteriormente, en bases de datos u hojas de cálculo para poder ser procesada.

Algunos de estos programas se utilizan, por ejemplo, para obtener y comparar precios de distintas fuentes (por ejemplo, comparar precios de un mismo producto en distintos comercios online), obtener los hiperenlaces contenidos en una página web (tal y como hacen los bots de Google para descubrir nuevos sitios), etcétera.

En este proyecto de programación, se implementará la lógica de un bot que simulará el comportamiento de *Google* cuando intenta descubrir nuevas páginas. Es decir, se utilizarán técnicas de *web-scraping* para obtener los hiperenlaces de una página web. También se implementará la lógica necesaria para obtener todas las imágenes contenidas en una página web.

Para la realización de la práctica, es necesario que:

Cree un repositorio privado en GitHub.
En el fichero README.md del repositorio indique su nombre completo y grado al que pertenece.
Gestione los permisos del repositorio GitHub para agregar al profesor de la asignatura como "colaborador" (el usuario gihub del profesor encargado de la práctica es "toural").
Tenga en cuenta que el uso de un repositorio privado en GitHub para gestionar esta práctica es obligatorio y se tendrá en cuenta todo el trabajo que quede reflejado en el repositorio. Por ello se recomienda que haga commits a menudo cada vez que introduzca algún cambio importante en el proyecto. En cualquier caso, se aconseja realizar commits única y exclusivamente si el software es estable.

2.1. Parte 1: Utilización de un analizador léxico de páginas web

Una de las técnicas de web-scraping más utilizadas es el análisis (o parseado) del código HTML de una página web.

En esta parte de la práctica, el alumno hará uso de un analizador léxico. Estos analizadores reciben un fichero de texto y lo leen carácter a carácter; guiados por expresiones regulares (o autómatas finitos), convierten grupos de estos caracteres en unidades léxicas. Por ejemplo, en el caso del analizador léxico del lenguaje Python, cada vez que reconoce que el carácter 'I' es seguido por el caracter 'F', genera un token (o unidad léxica) "TOKEN_IF".

En un compilador, los tokens generados por los analizadores léxicos, sirven como entrada del analizador sintáctico que, en base a una gramática, verifica si esas unidades léxicas están bien organizadas sintácticamente. Por ejemplo, el analizador sintáctico de Python, al recibir el

token "TOKEN_IF", determinará cuáles de los tokens que vienen después, forman parte de una expresión If-then-else, y comprobará que esa expresión está bien formada (esto es, que se ajusta a la gramática definida).

2.1.1 PLY (Python Lex-Yacc)

Para implementar el proyecto de web scraping con PLY (lex/yacc), se requieren los siguientes elementos y conocimientos:

Python instalado

Versión 3.x recomendada (por ejemplo, 3.7 en adelante).

PLY (Python Lex-Yacc) instalado

PLY no se incluye por defecto en la mayoría de distribuciones de Python, de forma que deberá instalarlo.
Se puede instalar con `pip install ply`

Conocimientos básicos de Python

Manejo de módulos y paquetes.

Uso de estructuras de datos

Manejo de listas, diccionarios, etc.

Conocimientos de PLY (o de parsing en general)

Saber definir tokens (reglas léxicas) en el archivo del lexer (lexer.py).

Entender cómo se definen las reglas de la gramática en parser.py

Conocer la secuencia lexer => parser para alimentar el contenido HTML y obtener el árbol o el resultado del análisis sintáctico.

Archivo(s) HTML o URL(s) para analizar

Un archivo de prueba local donde se pueda observar cómo se reconocen las etiquetas <a> y <img>.

(Opcional) Conocimientos de HTML

Puesto que HTML es un lenguaje con muchos matices, es muy útil tener una base sobre su estructura y peculiaridades. Esto ayuda a entender por qué pueden aparecer casos que la gramática definida no maneje (atributos sin comillas, comentarios, etiquetas anidadas de forma incorrecta, etc.).

(Opcional) Manejo de excepciones y optimizaciones

PLY, por ser un parser genérico, no maneja automáticamente las ambigüedades o particularidades de HTML.

Será necesario extender y mejorar las reglas si el HTML real se desvía de la forma esperada o presenta un contenido HTML malformado.

2.1.2 Estructura general del proyecto

Una forma organizada de abordar el código sería la siguiente:

```
mi_scraper/
 ├── lexer.py
 ├── parser.py
 ├── main.py
 └── test_page.html
```

Módulo principal (main.py o similar)
Contiene la lógica de arranque del programa.

Carga el HTML.
Llama al lexer y parser para analizarlo.
Muestra los enlaces e imágenes extraídos.

Módulo lexer(por ejemplo lexer.py)
Contiene funciones responsables de:
Definir los tokens y las reglas léxicas (usando PLY).

Módulo parser (por ejemplo parser.py)
Contiene la lógica de:
Definir la gramática y las reglas de parsing (usando PLY)

Módulo de utilidades (opcional, utils.py)
Para funciones genéricas adicionales.

Almacenamiento
Guardar todo en fichero en disco duro.

Página HTML de ejemplo para probar
test_page.html

2.1.3 Tareas a realizar

Deberá implementar lexer.py para analizar un documento HTML (complejo y REAL) y:
- Obtener todas las URLs a las que hacen referencia los hiperenlaces que tenga.
- Obtener todas las URLs de las diferentes imágenes que están incrustadas en la página.
- Señalar si el documento HTML está bien balanceado. Es decir, que todas las etiquetas HTML que se abren también se cierran y, además, se cierran en el orden adecuado.

Además, deberá completar el código principal de la aplicación (main.py) para que, utilizando lexer.py, realice las siguientes tareas:
- Inicializar con un fichero que contenga un documento HTML que se corresponda con una página real (simple: https://www.example.com, media: https://www.python.org, compleja: https://www.wikipedia.org).
- Obtener todas las URLs de los hiperenlaces que tenga.
- Obtener todas las URLs de las imágenes que estén incrustadas.
- Guardar las URLs obtenidas en los puntos anteriores en un fichero en el disco duro.
- Mostrar por pantalla si el documento HTML está bien balanceado.

2.2. Parte 2: Análisis de la estructura DOM de una página Web

Otra de las técnicas de web-scraping consiste en, a partir de una página web, obtener el árbol DOM que la representa, y recorrerlo para obtener información acerca de los diferentes nodos y sus contenidos.

BeautifulSoup es un analizador de páginas web para **Python**. Esta librería permite construir, a partir de la URL (o de un archivo) de un documento HTML, un árbol DOM que represente su contenido. De este modo, una vez obtenido el árbol, se ofrece una API (https://www.crummy.com/software/BeautifulSoup/bs4/doc/) que permite recorrer los diferentes elementos HTML, realizar búsquedas directas, extraer información, etcétera. Para comprender cómo se usa esta librería, se recomienda consultar su documentación oficial y guías de uso (comúnmente conocidas como *cookbooks*).

2.2.2. Tareas a realizar

Deberá preparar un código para que, haciendo uso del árbol DOM, sea capaz de analizar un documento HTML y:

- Obtener todas las URLs a las que hacen referencia los hiperenlaces que tenga.
- Obtener todas las URLs de las imágenes que están incrustadas en el documento.
- Obtener estadísticas de uso de varias etiquetas HTML.

Además, deberá completar el código para que realice las siguientes tareas:

- Inicializar **BeautifulSoup** con una dirección URL que apunte a un documento HTML que se corresponda con una página real y suficientemente compleja (simple: https://www.example.com, media: https://www.python.org, compleja: https://www.wikipedia.org).

- Obtener todas las URLs de los hiperenlaces del documento.
- Obtener todas las URLs de las imágenes incrustadas en el documento.
- Guardar las URLs en un fichero de texto plano con cada URL en una linea.
- Mostrar por pantalla las estadísticas de uso de las siguientes etiquetas HTML: a, img,br, div, li, ul, p, span, table, td y tr.

En ambas partes de la práctica, tenga en cuenta que el código fuente que genere deberá estar debidamente documentado y estructurado y reflejará buenas prácticas de programación como:

- Uso de un README con instrucciones de ejecución.
- Comentarios en el código.
- Organización en módulos (lexer.py, parser.py, etc.).
- Realización de commits atómicos en GitHub.
- Etc.

Por otra parte, deberá asegurarse de que su código supera unos tests de pruebas que deberán ser definidos, de tal manera que asegure que su implementación funciona correctamente.

Por último, una vez tenga implementados los dos apartados de la práctica (analizadores basados en PLY y BeautifulSoup), si ambos funcionan correctamente deberían obtenerla misma salida. En este sentido, tendrá especial relevancia el test que comprueba la correspondencia entre ambos (denominado CrossTest). Asegúrese de que su software pasa este conjunto de pruebas.

3. Evaluación

Para superar la práctica, la práctica se evaluará en relación a:
- Organización, estructuración y calidad del código fuente.
- Trabajo continuado y detallado que el alumno haya realizado a lo largo de la práctica. Para verificar este apartado se revisarán todos los commits realizados en el repositorio de GitHub (que es obligatorio utilizar para el seguimiento de la misma).
Se tendrá en cuenta: la calidad de los commits y la realización de manera progresiva de la práctica. Es decir, el trabajo en el repositorio debe reflejar que el alumno ha estado trabajando en la práctica de manera continuada.
- Funcionamiento ajustado a los requisitos establecidos.
- Que el código del alumno pase los tests de prueba. Además, se valorará la amplitud y variedad de los test de pruebas escritos por el alumno (que deberán demostrar el correcto funcionamiento de todo el código).
- Claridad del código y adecuación a las normas de estilo del lenguaje de programación Python.

- Trabajo realizado en la entrega intermedia (ver apartado 4.1).
- Memoria y calidad de la misma (presencia, estructuración de la información, claridad en la exposición,. . .).

4. Entrega

Tenga en cuenta que este proyecto de programación tiene dos fechas de entrega. Por una parte, la primera es una entrega intermedia cuyo objetivo es valorar el trabajo realizado por el alumno hasta ese momento. Esta entrega debería incluir el trabajo descrito en el apartado 2.1. Por otra parte, la segunda entrega incluye todo el trabajo solicitado en este enunciado.

4.1. Entrega INTERMEDIA

La entrega intermedia se corresponde con el desarrollo de las especificaciones establecidas en el apartado 2.1.
Como resultado de esta parte del proyecto se deberá entregar un único fichero zip que incluya: una pequeña memoria y toda la carpeta del proyecto desarrollado.
En el repositorio, marque esta entrega realizando un commit denominado “Entrega intermedia” (asegúrese de que el commit quede registrado en el repositorio remoto).
La confección de la memoria y el envío del fichero se realizarán siguiendo estas instrucciones:
La memoria incluirá una portada con el nombre del alumno y se ajustará a la siguiente estructura:

- Análisis y descripción del proyecto:
  Resumen de uso de PLY.
  Grafo del autómata que ilustre el funcionamiento del analizador programado.
  Decisiones de diseño tomadas hasta el momento y cómo se han implementado.
- La memoria se presentará en formato PDF.

El fichero se entregará a través de la actividad establecida a tal efecto en el Campus Virtual de la asignatura. No se admitirán entregas por otra vías (por ejemplo, por correo electrónico).
Recuerde que la entrega de la práctica deberá realizarse dentro del período de tiempo indicado. Fuera de ese plazo no se aceptará ninguna entrega.

4.2. Entrega FINAL

La entrega final se corresponde con el desarrollo completo del proyecto, tal y como se describe en este enunciado.
Como resultado de esta parte del proyecto se deberá entregar un único fichero zip que incluya: una memoria completa y toda la carpeta del proyecto desarrollado.
En el repositorio, marque esta entrega realizando un commit denominado “Entrega final (asegúrese de que el commit quede registrado en el repositorio remoto).
La confección de la memoria y el envío del fichero se realizarán siguiendo estas instrucciones:
La memoria incluirá una portada con el nombre del alumno y se ajustará a la siguiente estructura:

- Análisis y descripción del proyecto:
  - Introducción a la práctica.
  - Parte 1: PLY
    - Resumen del uso que ha realizado de PLY.
    - Grafo del autómata que ilustre el funcionamiento del analizador programado.
    - Decisiones de diseño tomadas en la práctica y cómo se han implementado.

  - Descripción de cada una de las pruebas de tests que el alumno ha definido: por qué ha realizado cada una de las pruebas (qué deseaba comprobar), por qué cree que el conjunto de pruebas demuestran que su software es correcto.

- Parte 2: BeautifulSoup
  - Resumen del uso que ha realizado de BeautifulSoup.
  - Decisiones de diseño tomadas en la práctica y cómo se han implementado.
  - Descripción de cada una de las pruebas de tests que el alumno ha definido: por qué ha realizado cada una de las pruebas (qué deseaba comprobar), por qué cree que el conjunto de pruebas demuestran que su software es correcto.

- Completitud de los tests definidos por el alumno. En este apartado deberá incluir una captura de pantalla donde se vea, claramente, que el software del alumno ha pasado todos los tests que ha definido el alumno.

- Conclusiones (que incluirán una valoración del tiempo dedicado a la práctica).

- La memoria se presentará en formato PDF.

El fichero se entregará a través de la actividad establecida a tal efecto en el Campus Virtual de la asignatura. No se admitirá la entrega por otras vías.

Recuerde que la entrega de la práctica deberá realizarse dentro del período de tiempo indicado, en caso contrario, se evaluará con cero puntos.

5. Tutorías

Se recomienda que el alumno establezca reuniones periódicas con el profesor para asegurarse de que está evolucionando correctamente en el desarrollo de la práctica. Las consultas y dudas que surjan en el desarrollo de esta práctica, se resolverán durante las horas de tutorías establecidas. Para ello, es recomendable concertar, previamente, una cita con el profesor a través de su dirección de correo electrónico (juan.sancheztoural@ceu.es).