Proyecto final 1 – Araña rastreadora

Antes de empezar con la práctica.

¿Qué es web scraping?

Es una técnica en la cual extraemos datos de páginas web de internet de una forma automática.

En Python, existen herramientas que nos ayudan a hacer web scraping, son Scrapy, Selenium y BeautifulSoup.

En esta práctica vamos a utilizar BeautifulSoup, por tanto vamos a ver que necesitamos saber.

Objeto BeautifulSoup

Acepta 2 argumentos, el primero es la página web y el segundo el parser que queremos utilizar, los parser pueden ser html.parser, lxml y html5lib.

Objetos

Hay cuatro objetos python que necesitamos:

- Tag: Es una etiqueta Html del documento, y utilizaremos tag.name para acceder al nombre. También accederemos a atributos mediante tag['id'] o el diccionario de atributos completo tag.attrs.
- NavigableString: El texto que contiene una etiqueta será de este tipo.
- BautifulSoup:
- Comment: Este objeto contendrá los comentarios de la página web.

Por otro lado utilizaremos el paquete requests para acceder al contenido de una web de internet.

Realiza la búsqueda de los productos de consum mediante web scraping:

1. Recuerda que en la url de consum, la búsqueda de productos se realiza mediante la url "https://tienda.consum.es/es/s/'producto_a_buscar'" siendo 'producto_a_buscar' lo que necesitas buscar en el supermercado.
2. En esta plataforma los espacios los tenemos que escapar por %20.
3. Recuerda ir a la web para encontrar las etiquetas, clases o id que podemos utilizar para realizar la lectura de los productos y su precio.

```
import requests
from bs4 import BeautifulSoup

busqueda = 'cerveza turia'

web = requests.get('https://tienda.consum.es/es/s/+busqueda.replace(" ","%
20"))
soup = BeautifulSoup(web.text,"lxml")
```

Ejercicio 1. Realiza un recorrido por todos los artículos de este supermercado.

Existen plataformas que tienen una protección para que no podamos realizar web scraping, pero existe una alternativa para emular que somos un navegador.

```
# Ejecutar para poder acceder a webs que tienen protección y necesitan emu
lar un navegador
# Cada 12 horas en colab se pierde la instalación de este driver
!pip install selenium
!apt-get update # to update ubuntu to correctly run apt install
!apt install chromium-chromedriver

import sys
sys.path.insert(0,'/usr/lib/chromium-browser/chromedriver')
```

Realiza la búsqueda de los productos de Carrefour mediante web scraping:

1. Recuerda que en la url de Carrefour, la búsqueda de productos se realiza mediante la url por Get mediante la variable q,"https://www.carrefour.es/?q='producto_a_buscar'" siendo 'producto_a_buscar' lo que necesitas buscar en el supermercado.
2. En esta plataforma los espacios los tenemos que escapar por +.
3. Recuerda ir a la web para encontrar las etiquetas, clases o id que podemos utilizar para realizar la lectura de los productos y su precio.

Como Carrefour tiene una protección debemos utilizar la emulación del navegador.

```
from selenium import webdriver
from bs4 import BeautifulSoup
options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
wd = webdriver.Chrome('chromedriver',options=options)

busqueda = 'cerveza turia'

wd.get("https://www.carrefour.es/?q="+busqueda.replace(" ","+"))
```

Ejercicio 2. Realiza un recorrido por todos los artículos de este supermercado, imprime el título de la página.

Estamos interesados en ver la relación de precios entre un supermercado y otro de este producto:

busqueda = 'chocolate milka'

Utilizaremos esta función para encontrar la similitud entre la descripción de un artículo y otro:

```
from difflib import SequenceMatcher

def similar(a, b):
    return SequenceMatcher(None, a, b).ratio()
```

Ejercicio 3. Utilizando los resultados de los ejercicios anteriores, compara los dos listados de artículos y almacena en un diccionario todos los productos referenciados a cada supermercado, almacenando en un mismo índice los

productos que sean similares en un 66%. Aunque no tengan similar en un supermercado agrégalos también al diccionario.

Es recomendable almacenar el porcentaje de la similitud en el diccionario.

Muestra el diccionario en pantalla.

Ejemplo de resultado del diccionario:

{'consum': {'Chocolate con Leche 270 Gr': '1,85 €'}, 'carrefour': {'Chocolate con leche Milka 270 g.': '2,10 €'}, 'porcentaje': 0.7931034482758621}

{'consum': {'-': '-'}, 'carrefour': {'Galletas con chocolate Choco Biscuits Milka 150 g.': '1,91 €'}}

{'consum': {'Chocolate Triple 90 Gr': '1,29 €'}, 'carrefour': {'-': '-'}}

Proyecto final 2 – Panel de administración web dinámico

Se necesita mediante Flask una web para poder crear, listar y eliminar usuarios. Podríamos decir una gestión de datos de usuarios.

Para instalar Flask y tener un acceso externo en colab ejecutaremos:

```
!pip install flask-ngrok
```

Para poder visualizar la web remotamente desde colab, importaremos run_with_ngrok como en este ejemplo:

```
from flask_ngrok import run_with_ngrok
from flask import Flask,request, url_for
app = Flask(__name__)
run_with_ngrok(app)  # Ruta externa
@app.route("/")
def home():
    return "<h1>Ejecutamos Flask en Colab externamente</h1>"

app.run()
```

Ejercicio 1. Crea un formulario web, que permita añadir usuarios mediante Post y los almacene en una lista de objetos usuario y los coloque en un fichero de formato JSON.

Cada usuario debe contener:

- Email
- Nombre
- Apellidos
- Contraseña
- Teléfono
- Edad

Un ejemplo de formulario:

```
<form action="" method="post">
                  <label for="email">Email: </label>
                  <input required="required" type="text" id="email" name="
email" /><br>
                  <input type="submit" id="send-
signup" name="signup" value="Registrar" />
</form>
```

Crea 10 usuarios desde el formulario, y comprueba en el fichero que se han almacenado.

Ejercicio 2. Crea una web donde se muestre una tabla con todos los usuarios que estén creados en el fichero del ejercicio anterior.

Recuerda:

- Utilizar from types import SimpleNamespace para poder convertir el .json en una lista de objetos Usuario.
- json.loads('string--json', object_hook=lambda d: SimpleNamespace(**d))

Ejercicio 3. Crea un menú para acceder a las dos páginas web anteriores, y además, añade la funcionalidad de editar y borrar un elemento de la tabla del ejercicio anterior mediante GET.

No olvides poner en cada página un botón para volver a la página del menú.

Proyecto final 3 – Dashboard científico generado con Python y variables dinámicas

Se necesita desde un fichero .csv (accede al Campus online), extraer varios indicadores:

*Recuerda subir el fichero a la carpeta de trabajo, se eliminará automáticamente el fichero subido, pasado un tiempo.

Ejercicio 1. Almacena los datos del csv en un fichero txt en formato JSON, agrupando los datos por día de la semana y por provincias, mostrar los acumulado de defunciones, los nuevos casos de covid , hospitalizados y uci (num_def,new_cases,num_hosp,num_uci).

Utilizaremos el módulo csv para facilitar la lectura del fichero.

```
import csv

results = []
with open('example.csv') as File:
   reader = csv.DictReader(File)
   for row in reader:
       results.append(row)
   print results
```

Ejercicio 2. Realiza 4 gráficas de los datos almacenados en el fichero del ejercicio anterior, realiza un menú con un bucle infinito para acceder a cada una de las gráficas.

```
# Menú
int(input("""¿Qué gráfica quieres visualizar?
            1. Defunciones
            2. Casos
            3. Hospitalizados
            4. UCI
            5. Salir
        """))
```

Ejercicio 3. Extrae del fichero generado en el ejercicio 1 que provincia contiene más defunciones, más casos, más hospitalizados y más en uci. Muestra el resultado en 4 gráficas de queso. Y realiza un menú como el ejercicio anterior. Después de mostrar la gráfica indica que provincia tiene el máximo de la variable a mostrar y cual la mínima.