Extracción de datos con Python

Ejemplo 1:

El objetivo es obtener información de datos sobre la población del mundo utilizando la técnica de scraping o raspado. Los datos están disponibles en el sitio de Worldometer. Es un sitio de código abierto dirigido por un equipo internacional de desarrolladores e investigadores voluntarios, cuyo objetivo es poner las estadísticas globales a disposición de un amplio público en todo el mundo.

a. Ejecutar cada una de las celdas de ejemplo.
b. observar y revisar cada resultado.

In [ ]:
```
import csv
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

Crear una variable URL en formato string (texto) que contendrá el enlace a la página.

In [ ]:
```
url = "https://www.worldometers.info/world-population/population-by-country/"
```

Obtener los datos, utilizando la función requests.get()

In [ ]:
```
req = requests.get(url)
```

Con la clase BeautifulSoup(), extraer el código HTML de la página. En el argumento de esta función se selecciona el objeto .text

In [ ]:
```
soup = BeautifulSoup(req.text)
```

En los datos almacenados en la variable, buscar el código HTML con la palabra clave "table" y usar la función .find_all()

In [ ]:
```
data = soup.find_all("table")[0]
```

Leer el código HTML con el comando .read_html(str()), luego recuperar el primer y único elemento del objeto (el array)

In [ ]:
```
df_population = pd.read_html(str(data))[0]
```

Emitir los primeros elementos, utilizando la función .head() del DataFrame

In [ ]:
```
df_population.head()
```

Exportar los datos a un .csv

In [ ]:
```
export_csv = df_population.to_csv(r"archivo.csv", index = None, header = True)
```

Ejemplo 2:

El objetivo es hacer un raspado o scraping de citas de una página web y guardarlos en un archivo. Luego se leen y se emiten traducidas.

a. Ejecutar el código a continuación.
b. Revisar que se haya creado el archivo.
c. Inspeccionar el archivo creado.
d. Ejecutar el código siguiente.
e. Inspeccionar el resultado.
f. Quitar del texto las repeticiones de la comilla doble usando expresiones regulares.
g. Investigar como guardar la última información obtenida en un archivo.

In [ ]:

```python
# ejecutar
import requests
from bs4 import BeautifulSoup
import csv

def scrape_page(soup, quotes):
    # recuperando todo el elemento HTML <div> en la página
    quote_elements = soup.find_all('div', class_='quote')

     # iterando sobre la lista de elementos de cotización
     # para extraer los datos de interés y almacenarlos
     # entre comillas
    for quote_element in quote_elements:
        # extrayendo el texto de la cita
        text = quote_element.find('span', class_='text').text
        # extrayendo el autor de la cita
        author = quote_element.find('small', class_='author').text

        # extrayendo la etiqueta <a> elementos HTML relacionados con la cita
        tag_elements = quote_element.find('div', class_='tags').find_all('a', class_='tag')

        # almacenar la lista de cadenas de etiquetas en una lista
        tags = []
        for tag_element in tag_elements:
            tags.append(tag_element.text)

        # agregando un diccionario que contiene los datos
        # en un nuevo formato en la lista
        quotes.append(
            {
                'text': text,
                'author': author,
                'tags': ', '.join(tags)  # fusionar las etiquetas en una cadena "A, B, ..., Z"
            }
        )
```

In [ ]:

```
# la URL de la página de inicio del sitio web de destino
base_url = 'https://quotes.toscrape.com'

# definir el encabezado User-Agent para usar en la solicitud GET a continuación
headers = {
'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 \
    (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36'
}

# recuperar la página web de destino
page = requests.get(base_url, headers=headers)

# analizando la página web de destino con Beautiful Soup
soup = BeautifulSoup(page.text, 'html.parser')

# inicializando la variable que contendrá
# la lista de todos los datos
quotes = []

# raspando la página de inicio
scrape_page(soup, quotes)

# obteniendo el elemento "Siguiente →" HTML
next_li_element = soup.find('li', class_='next')

# si hay una página siguiente para raspar
while next_li_element is not None:
    next_page_relative_url = next_li_element.find('a', href=True)['href']

    # obteniendo la nueva página
    page = requests.get(base_url + next_page_relative_url, headers=headers)

    # analizando la nueva página
    soup = BeautifulSoup(page.text, 'html.parser')

    # raspando la nueva página
    scrape_page(soup, quotes)

    # buscando el elemento "Siguiente →" HTML  en la nueva página
    next_li_element = soup.find('li', class_='next')

# leer el archivo "citas.csv" y crearlo
# si no está presente
csv_file = open('citas.csv', 'w', encoding='utf-8', newline='')

# inicializando el objeto escritor para insertar datos
# en el archivo CSV
writer = csv.writer(csv_file)

# escribiendo el encabezado del archivo CSV
writer.writerow(['Text', 'Author', 'Tags'])

# escribiendo cada fila del CSV
for quote in quotes:
    writer.writerow(quote.values())

# finalizar la operación y liberar los recursos
csv_file.close()
```

TextBlob es una biblioteca de Python para procesar datos textuales. Proporciona una API sencilla para el procesamiento del lenguaje natural (PLN).

In [ ]:

```
import textblob
from textblob import TextBlob
```

In [ ]:

```
file= open('citas.csv', encoding="utf8")
dataset= file.read()
trad=TextBlob(dataset)
traducido=trad.translate(from_lang='en',to='es')
traducido
```