Python para tu portfolio

Web Scraper

Martin Ozuna

21 de Abril del 2023

Introducción

1. Importa las bibliotecas necesarias para el scraping web, el análisis de datos y la conexión a una base de datos MySQL.
2. Se conecta a una base de datos MySQL utilizando las credenciales proporcionadas.
3. Realiza una solicitud GET a la página web que contiene la tabla de posiciones de la liga profesional de fútbol, utilizando la biblioteca requests.
4. Utiliza la biblioteca BeautifulSoup para analizar el HTML de la página y extraer los datos de interés, en este caso los nombres de los equipos y los puntos que han acumulado en la temporada.
5. Crea un DataFrame de pandas a partir de los datos extraídos, con los nombres de los equipos y los puntos.
6. Recorre el DataFrame y ejecuta una consulta INSERT en la base de datos MySQL para cada fila del DataFrame, insertando el nombre del equipo y los puntos acumulados en la tabla de clasificación.
7. Guarda los cambios en la base de datos y muestra el número de filas insertadas.
8. Exporta el DataFrame a un archivo CSV para su posterior uso.

Consejo: necesitarás las credenciales de tu workbench mysql para poder guardar los datos en una base de datos.

Instrucciones

1. Identifica el sitio web del que quieres extraer información y asegúrate de que está permitido según las políticas de uso del sitio. Yo en este caso utilizare la web de TyC Sports.
2. Abre el terminal e instala las librerías necesarias: requests, beautifulsoup4, pandas y mysql-connector-python.

   Para instalar la librería requests, ejecuta el siguiente comando:

   ```
   pip install requests
   ```

   Para instalar la librería beautifulsoup4, ejecuta el siguiente comando:

   ```
   pip install beautifulsoup4
   ```

   Para instalar la librería pandas, ejecuta el siguiente comando:

   ```
   pip install pandas
   ```

   Para instalar la librería mysql-connector-python, ejecuta el siguiente comando:

   ```
   pip install mysql-connector-python
   ```

3. Utiliza la librería Requests para hacer una solicitud HTTP a la página web y obtener el contenido HTML de la página:

   ```
   # web url de donde se hace el get de datos
   url = 'https://www.ejemplo.com'

   r = requests.get(url)
   soup = BeautifulSoup(r.text, 'html.parser')
   ```

4. Utiliza la librería Beautiful Soup para parsear el contenido HTML y extraer la información que necesitas. Por ejemplo, si quieres extraer todos los enlaces de la página:

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

# Equipos

teams = soup.find_all('td', class_='equipo')

equipos = list()
```

En este ejemplo, este código utiliza “beautifulsoup4” para buscar todos los elementos HTML que tengan una etiqueta td y una clase equipo en la página web analizada y los almacena en una lista llamada equipos.

- Si quieres almacenar la información extraída en una base de datos, puedes utilizar una librería de bases de datos como SQLite o PostgreSQL.

  Por ejemplo, para utilizar SQLite, primero debes importar la librería:

```
import mysql.connector
```

- Luego, deberás realizar la conexión:

```
# Conectar con la base de datos
mydb = mysql.connector.connect(
  host="localhost",
  user="tu_usuario",
  password="tu_contraseña",
  database="nombre_de_tu_base_de_datos"
)
```

  Recuerda reemplazar: tu_usuario, tu_contraseña, nombre_de_tu_ base_de_datos, con tus credenciales de MySQL Workbench.

5. En esta parte del código se está creando un objeto de tipo DataFrame de pandas que contiene dos columnas: 'Nombre' y 'Puntos'. Los valores de la columna 'Nombre' son los elementos de la lista 'equipos', mientras que los valores de la columna 'Puntos' son los

elementos de la lista 'puntos'. Además, se está especificando un índice numérico para el DataFrame usando la función 'range' de Python. El índice va del número 1 al número 10, que se corresponden con los 10 primeros equipos de la tabla de posiciones.

```
df = pd.DataFrame({'Nombre': equipos, 'Puntos': puntos}, index=list(range(1,11)))
```

6. Deberás crear un cursor de MySQL. Esta línea de código crea un cursor de base de datos para realizar operaciones en la base de datos.

   Un cursor es un objeto que permite ejecutar comandos SQL y recuperar datos de la base de datos. Puedes usarlo para ejecutar consultas, insertar, actualizar o eliminar registros en una tabla, entre otras cosas.

```
mycursor = mydb.cursor()
```

7. A partir de este punto usarás MySQL workbench para crear la base de datos y las tablas correspondientes que utilizarás, para guardar tu información.

```
1    USE tabla_futbol;
2
3 •  CREATE TABLE clasificacion(
4        nombre VARCHAR(40),
5        puntos INT
6    );
7
8 •  SELECT * FROM clasificacion;
```

8. Esta parte del código ejecuta la inserción de los datos del DataFrame df en la tabla clasificación de la base de datos mydb utilizando MySQL Connector/Python.

   Primero, se crea un cursor para ejecutar sentencias SQL. Luego, se itera sobre los datos del DataFrame y se ejecuta una sentencia SQL de inserción para cada fila. La variable nombre contiene el valor de la columna Nombre de la fila actual, y la variable puntos contiene el valor de la columna Puntos. La sentencia SQL de inserción se construye con estos valores utilizando placeholders %. Luego, se ejecuta la sentencia SQL utilizando el método execute() del cursor, pasando como argumentos la sentencia SQL y los valores para los placeholders. Una vez que se han ejecutado todas las sentencias SQL de inserción, se confirman los cambios en la base de datos utilizando el método commit() de

la conexión a la base de datos. Finalmente, se cierra la conexión a la base de datos utilizando el método close()

```
# Cursor para ejecutar sentencias SQL
mycursor = mydb.cursor()

# Iterar sobre los datos del DataFrame y ejecutar la sentencia SQL de inserción
for index, row in df.iterrows():
    nombre = row['Nombre']
    puntos = row['Puntos']
    sql = "INSERT INTO clasificacion (nombre, puntos) VALUES (%s, %s)"
    val = (nombre, puntos)
    mycursor.execute(sql, val)

# Confirmar los cambios
mydb.commit()

# Cerrar la conexión a la base de datos
mydb.close()
```

9. Y por último para finalizar, La función print(mycursor.rowcount, "filas insertadas.") imprime el número de filas que se han insertado en la tabla clasificacion mediante la ejecución de la consulta mycursor.executemany(sql, val).

   mycursor.rowcount es un atributo que devuelve el número de filas afectadas por la última operación ejecutada. En este caso, el número de filas afectadas corresponde al número de filas que se han insertado en la tabla clasificacion.

   La última línea df.to_csv('clasificacion.csv', index=False) guarda los datos obtenidos del web scraping en un archivo csv llamado clasificacion.csv sin incluir el índice de filas en el archivo.

```
# Imprimir el número de filas insertadas
print(mycursor.rowcount, "filas insertadas.")

df.to_csv('clasificacion.csv', index=False)
```

Consejo: Recuerda que el código de un web scraper puede ser bastante complejo dependiendo de lo que quieras extraer, así que asegúrate de leer la documentación de las librerías y comprender bien el código que estás escribiendo. Además, respeta siempre las políticas de uso del sitio web que estás raspando para evitar problemas legales.

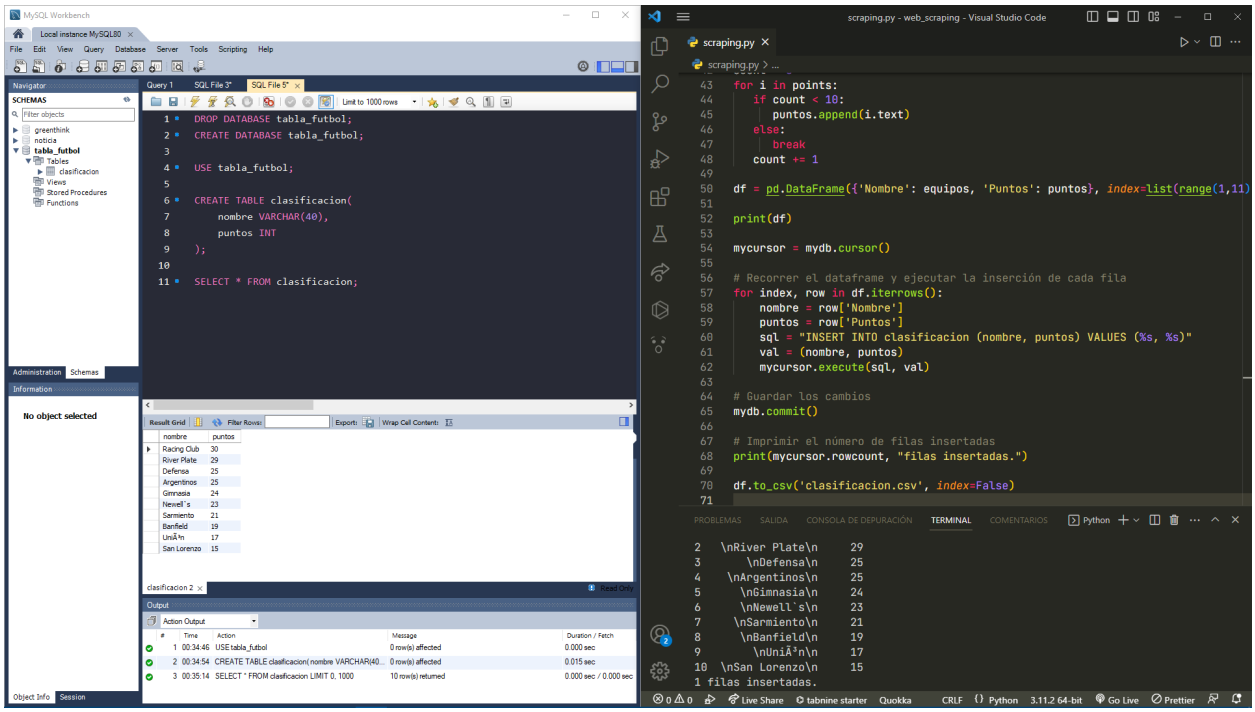
Explicación

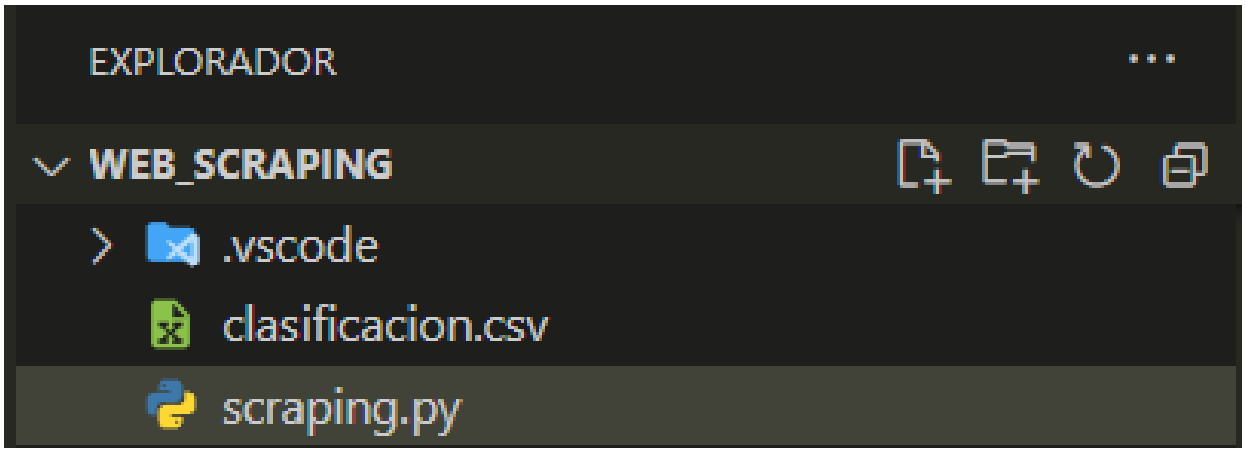
Este código realiza las siguientes tareas:

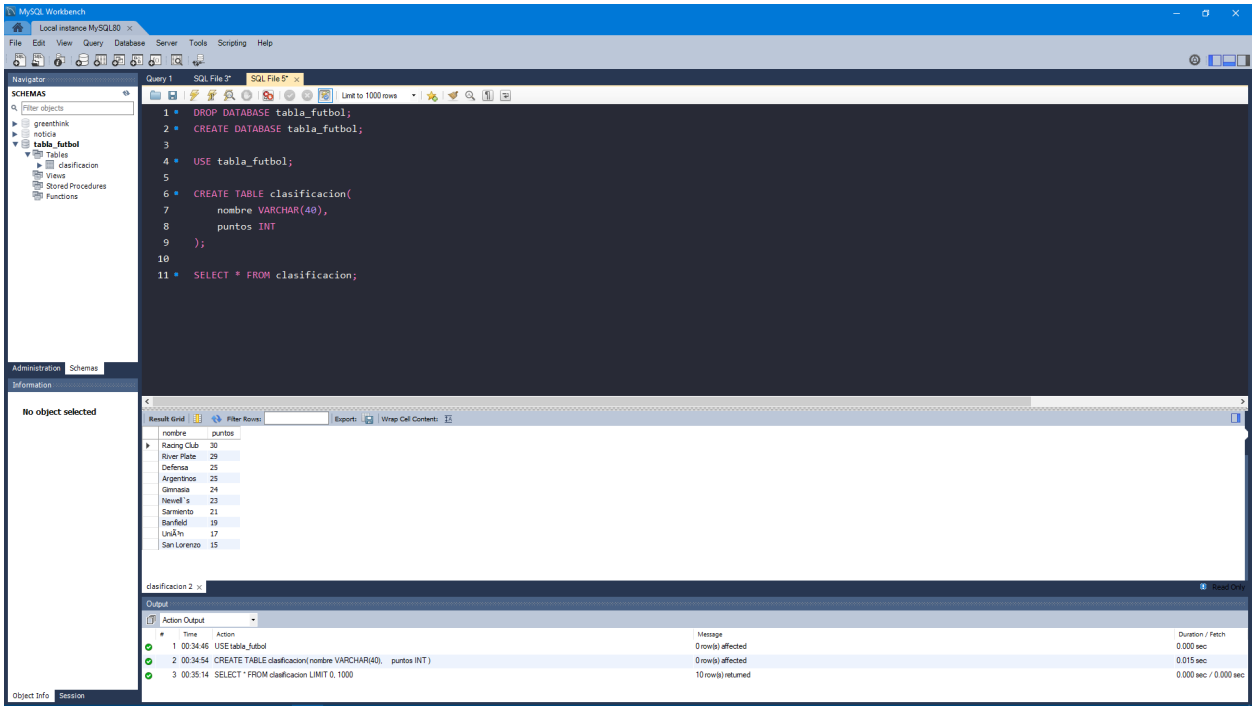
1. Importa las bibliotecas necesarias para el scraping web, el análisis de datos y la conexión a una base de datos MySQL.
2. Se conecta a una base de datos MySQL utilizando las credenciales proporcionadas.
3. Realiza una solicitud GET a la página web que contiene la tabla de posiciones de la liga profesional de fútbol, utilizando la biblioteca requests.
4. Utiliza la biblioteca BeautifulSoup para analizar el HTML de la página y extraer los datos de interés, en este caso los nombres de los equipos y los puntos que han acumulado en la temporada.
5. Crea un DataFrame de pandas a partir de los datos extraídos, con los nombres de los equipos y los puntos.
6. Recorre el DataFrame y ejecuta una consulta INSERT en la base de datos MySQL para cada fila del DataFrame, insertando el nombre del equipo y los puntos acumulados en la tabla de clasificación.
7. Guarda los cambios en la base de datos y muestra el número de filas insertadas.
8. Exporta el DataFrame a un archivo CSV para su posterior uso.

En resumen, este código realiza scraping web para extraer información de una tabla de clasificación de la liga de fútbol, analiza los datos y los almacena en una base de datos MySQL y en un archivo CSV.

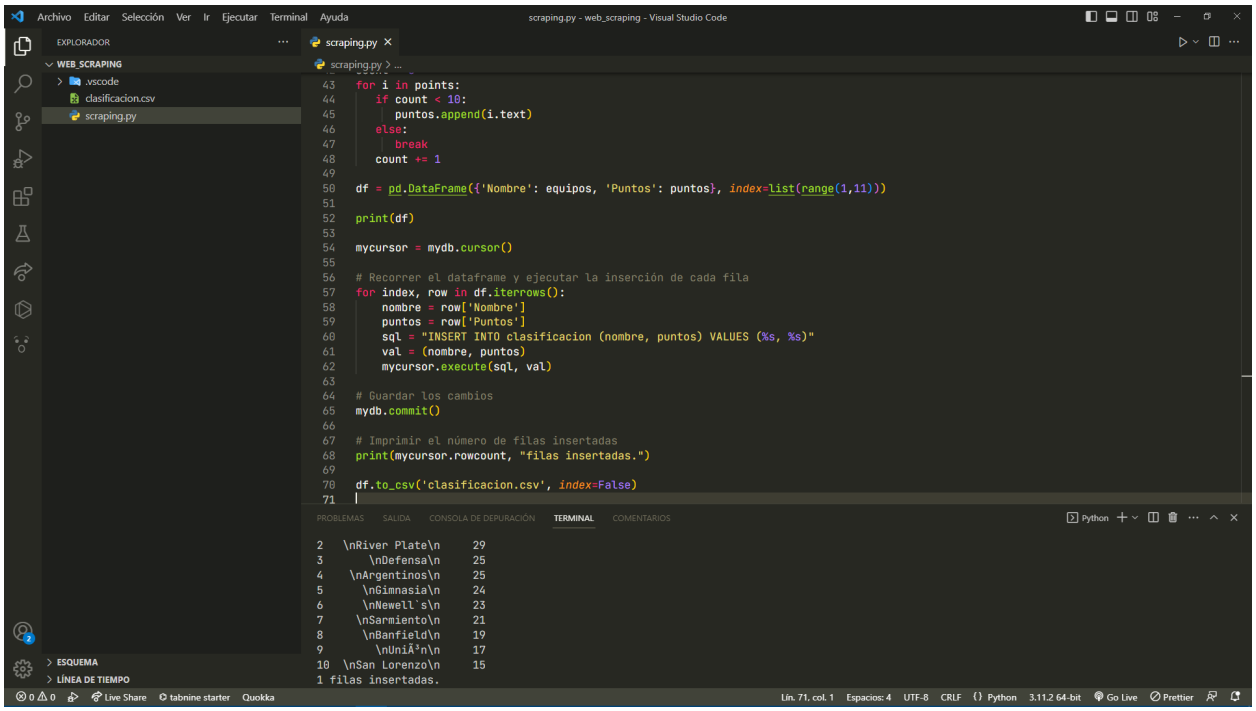
Resultados:


DROP DATABASE tabla_futbol;
CREATE DATABASE tabla_futbol;
USE tabla_futbol;
CREATE TABLE clasificacion(
nombre VARCHAR(40),
puntos INT
);
SELECT * FROM clasificacion;
No object selected
scraping.py - web_scraping - Visual Studio Code
for i in points:
if count < 10:
puntos.append(i.text)
else:
break
count += 1
df = pd.DataFrame({'Nombre': equipos, 'Puntos': puntos}, index=list(range(1,11)))
print(df)
mycursor = mydb.cursor()
# Recorrer el dataframe y ejecutar la inserción de cada fila
for index, row in df.iterrows():
nombre = row['Nombre']
puntos = row['Puntos']
sql = "INSERT INTO clasificacion (nombre, puntos) VALUES (%s, %s)"
val = (nombre, puntos)
mycursor.execute(sql, val)
# Guardar los cambios
mydb.commit()
# Imprimir el número de filas insertadas
print(mycursor.rowcount, "filas insertadas.")
df.to_csv('clasificacion.csv', index=False)
2 \nRiver Plate\n 29
3 \nDefensa\n 25
4 \nArgentinos\n 25
5 \nGimnasia\n 24
6 \nNewell`s\n 23
7 \nSarmiento\n 21
8 \nBanfield\n 19
9 \nUniÃ³n\n 17
10 \nSan Lorenzo\n 15
1 filas insertadas.


EXPLORADOR
WEB_SCRAPING
.vscode
clasificacion.csv
scraping.py



nombre	puntos
Racing Club	30
River Plate	29
Defensa	25
Argentinos	25
Gimnasia	24
Newell`s	23
Sarmiento	21
Banfield	19
UniÃ³n	17
San Lorenzo	15




```
                Nombre Puntos
1    \nRacing Club\n      30
2    \nRiver Plate\n      29
3        \nDefensa\n      25
4     \nArgentinos\n      25
5       \nGimnasia\n      24
6       \nNewell`s\n      23
7     \nSarmiento\n      21
8       \nBanfield\n      19
9         \nUniÃ³n\n\n      17
10   \nSan Lorenzo\n      15
1 filas insertadas.
```