Web Scraping Project

(python)

-Akhila V. Patil

Web scraping, also called web data mining or web harvesting, is the process of constructing an agent which can extract, parse, download and organize useful information from the web automatically.

In other words, instead of manually saving the data from websites, the web scraping software will automatically load and extract data from multiple websites as per our requirement. In this section, we are going to discuss about useful Python libraries for web scraping.

Uses of Web Scraping

The uses and reasons for using web scraping are as endless as the uses of the World Wide Web. Web scrapers can do anything like ordering online food, scanning online shopping website for you and buying ticket of a match the moment they are available etc. just like a human can do. Some of the important uses of web scraping are discussed here:

• E-commerce Websites: Web scrapers can collect the data specially related to the price of a specific product from various e-commerce websites for their comparison.

• Content Aggregators: Web scraping is used widely by content aggregators like news aggregators and job aggregators for providing updated data to their users.

• Marketing and Sales Campaigns: Web scrapers can be used to get the data like emails, phone number etc. for sales and marketing campaigns.

• Search Engine Optimization (SEO): Web scraping is widely used by SEO tools like SEMRush, Majestic etc. to tell business how they rank for search keywords that matter to them.

• Data for Machine Learning Projects: Retrieval of data for machine learning projects depends upon web scraping.

Why Python for Web Scraping?

Python is a popular tool for implementing web scraping. Python programming language is also used for other useful projects related to cyber security, penetration testing as well as digital forensic applications. Using the base programming of Python, web scraping can be performed without using any other third-party tool. Python programming language is gaining huge popularity and the reasons that make Python a good fit for web scraping projects are as below:

* Syntax Simplicity

Python has the simplest structure when compared to other programming languages. This feature of Python makes the testing easier and a developer can focus more on programming.

* Inbuilt Modules

Another reason for using Python for web scraping is the inbuilt as well as external useful libraries it possesses. We can perform many implementations related to web scraping by using Python as the base for programming.

* Open Source Programming Language

Python has huge support from the community because it is an open-source programming language. Wide range of Applications Python can be used for various programming tasks ranging from small shell scripts to enterprise web applications.

Python Modules for web scraping

We can understand the working of a web scraper in simple steps as shown in the diagram given below.

Step 1: Downloading Contents from Web Pages In this step, a web scraper will download the requested contents from multiple web pages.

Step 2: Extracting Data The data on websites is HTML and mostly unstructured. Hence, in this step, web scraper will parse and extract structured data from the downloaded contents.

Step 3: Storing the Data Here, a web scraper will store and save the extracted data in any of the format like CSV, JSON or in database.

Step 4: Analyzing the Data After all these steps are successfully done, the web scraper will analyze the data thus obtained.

Scrapy is intended to scratch web content from locales that are made out of numerous pages of

comparative semantic structure. An open-source and community system for extricating the information

you need from sites. The framework is actualized as a Firefox browser extension and works in three

principal stages to scratch web information [1]. Initial, a client explores to a page that he would like to

scratch and creates a format for the substance that he might want from that page. Next, the client chooses

a lot of links that point to pages matching the substance layout characterized by the client. At last, the

client chooses a final output to information group and Scrapy slithers the connections determined by the

client and scratches content comparing to the client's template [1]. Scrapy is composed in Python and runs

on Linux, Windows, and Macintosh.

IV. PROPOSED MODEL / ARCHITECTURE / METHODOLOGY/ MODEL TOOL

The proposed work centers around dissecting the website pages (HTML code). Right now, have built up a

working model. By utilizing this procedure weblink change into visual blocks. A visual block is actually a

section of web page. The framework is programmed top-down and deals to recognize web content

structure. Basically, the block-based page content structure is obtained by using a python script in

BeautifulSoup in order to further save it as a CSV file. Simulation of experimental work shown below [3];

A. Installation of BeautifulSoup and Requests

B. Python scripting

C. Execution of the python code

D. Content structure construction

E. Saving it as a CSV file

V

Scrapy is intended to scratch web content from locales that are made out of numerous pages of

comparative semantic structure. An open-source and community system for extricating the information

you need from sites. The framework is actualized as a Firefox browser extension and works in three

principal stages to scratch web information [1]. Initial, a client explores to a page that he would like to

scratch and creates a format for the substance that he might want from that page. Next, the client chooses

a lot of links that point to pages matching the substance layout characterized by the client. At last, the

client chooses a final output to information group and Scrapy slithers the connections determined by the

client and scratches content comparing to the client's template [1]. Scrapy is composed in Python and runs

on Linux, Windows, and Macintosh.

IV. PROPOSED MODEL / ARCHITECTURE / METHODOLOGY/ MODEL TOOL

The proposed work centers around dissecting the website pages (HTML code). Right now, have built up a

working model. By utilizing this procedure weblink change into visual blocks. A visual block is actually a

section of web page. The framework is programmed top-down and deals to recognize web content

structure. Basically, the block-based page content structure is obtained by using a python script in

BeautifulSoup in order to further save it as a CSV file. Simulation of experimental work shown below [3];

A. Installation of BeautifulSoup and Requests

B. Python scripting

C. Execution of the python code

D. Content structure construction

E. Saving it as a CSV file

V

Scrapy is intended to scratch web content from locales that are made out of numerous pages of

comparative semantic structure. An open-source and community system for extricating the information

you need from sites. The framework is actualized as a Firefox browser extension and works in three

principal stages to scratch web information [1]. Initial, a client explores to a page that he would like to

scratch and creates a format for the substance that he might want from that page. Next, the client chooses

a lot of links that point to pages matching the substance layout characterized by the client. At last, the

client chooses a final output to information group and Scrapy slithers the connections determined by the

client and scratches content comparing to the client's template [1]. Scrapy is composed in Python and runs

on Linux, Windows, and Macintosh.

IV. PROPOSED MODEL / ARCHITECTURE / METHODOLOGY/ MODEL TOOL

The proposed work centers around dissecting the website pages (HTML code). Right now, have built up a

working model. By utilizing this procedure weblink change into visual blocks. A visual block is actually a

section of web page. The framework is programmed top-down and deals to recognize web content

structure. Basically, the block-based page content structure is obtained by using a python script in

BeautifulSoup in order to further save it as a CSV file. Simulation of experimental work shown below [3];

A. Installation of BeautifulSoup and Requests

B. Python scripting

C. Execution of the python code

D. Content structure construction

E. Saving it as a CSV file

V

TOOL

The proposed work centers around dissecting the website pages (HTML code). Right now, have built up a

working model. By utilizing this procedure weblink change into visual blocks. A visual block is actually a

section of web page. The framework is programmed top-down and deals to recognize web content

structure. Basically, the block-based page content structure is obtained by using a python script in

BeautifulSoup in order to further save it as a CSV file. Simulation of experimental work shown below [3];

A. Installation of BeautifulSoup and Requests

B. Python scripting

C. Execution of the python code

D. Content structure construction

E. Saving it as a CSV

**Step 1: Inspect Your Data Source**

Before you write any Python code, you need to get to know the website that you want to scrape. That should be your first step for any web scraping project you want to tackle. You’ll need to understand the site structure to extract the information that’s relevant for you. Start by opening [the site you want to scrape](https://realpython.github.io/fake-jobs/) with your favourite browser.

**Step 2: Scrape HTML Content from a Page**

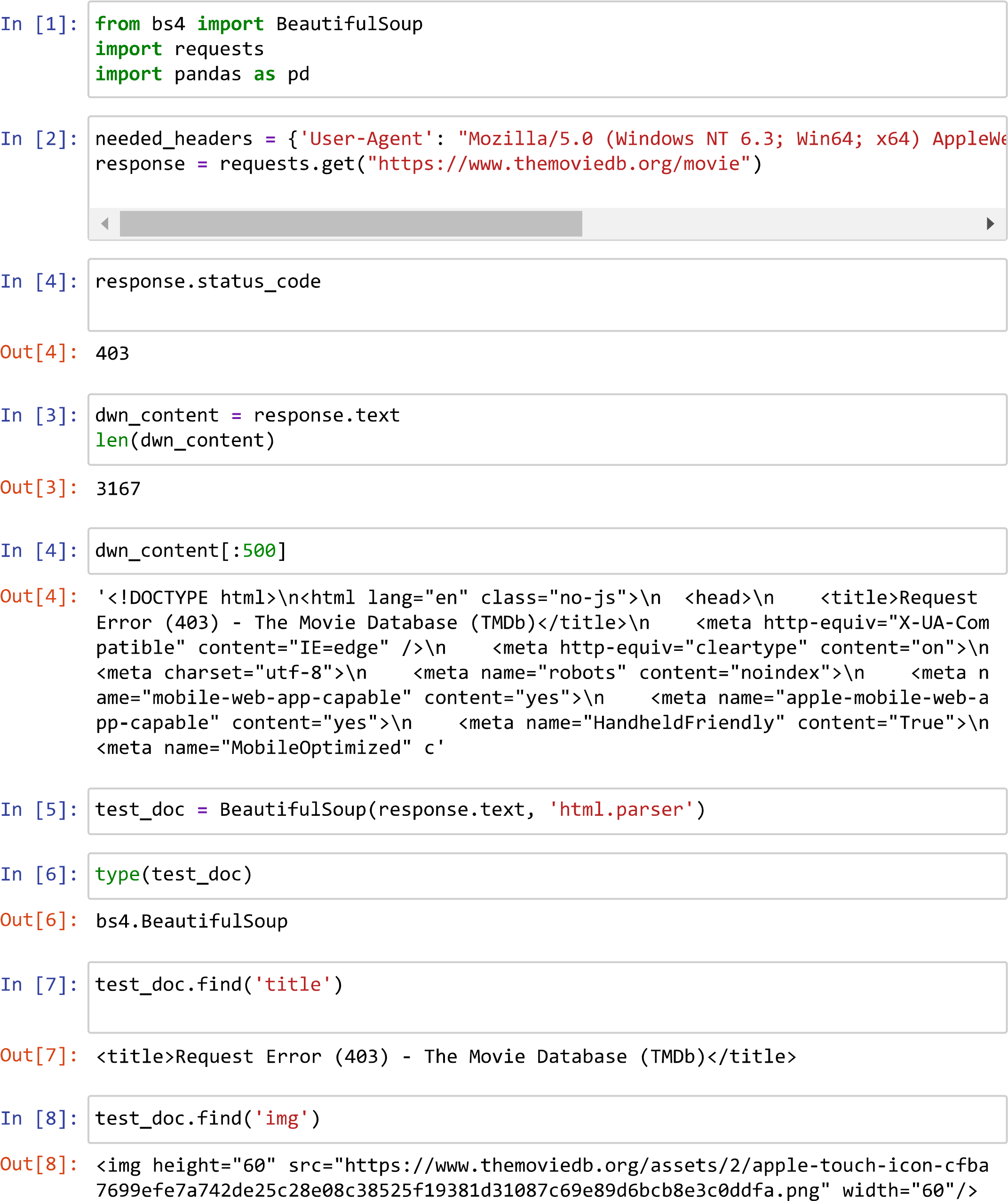
Now that you have an idea of what you’re working with, it’s time to start using Python. First, you’ll want to get the site’s HTML code into your Python script so that you can interact with it. For this task, you’ll use Python’s [requests](https://realpython.com/python-requests/) library.

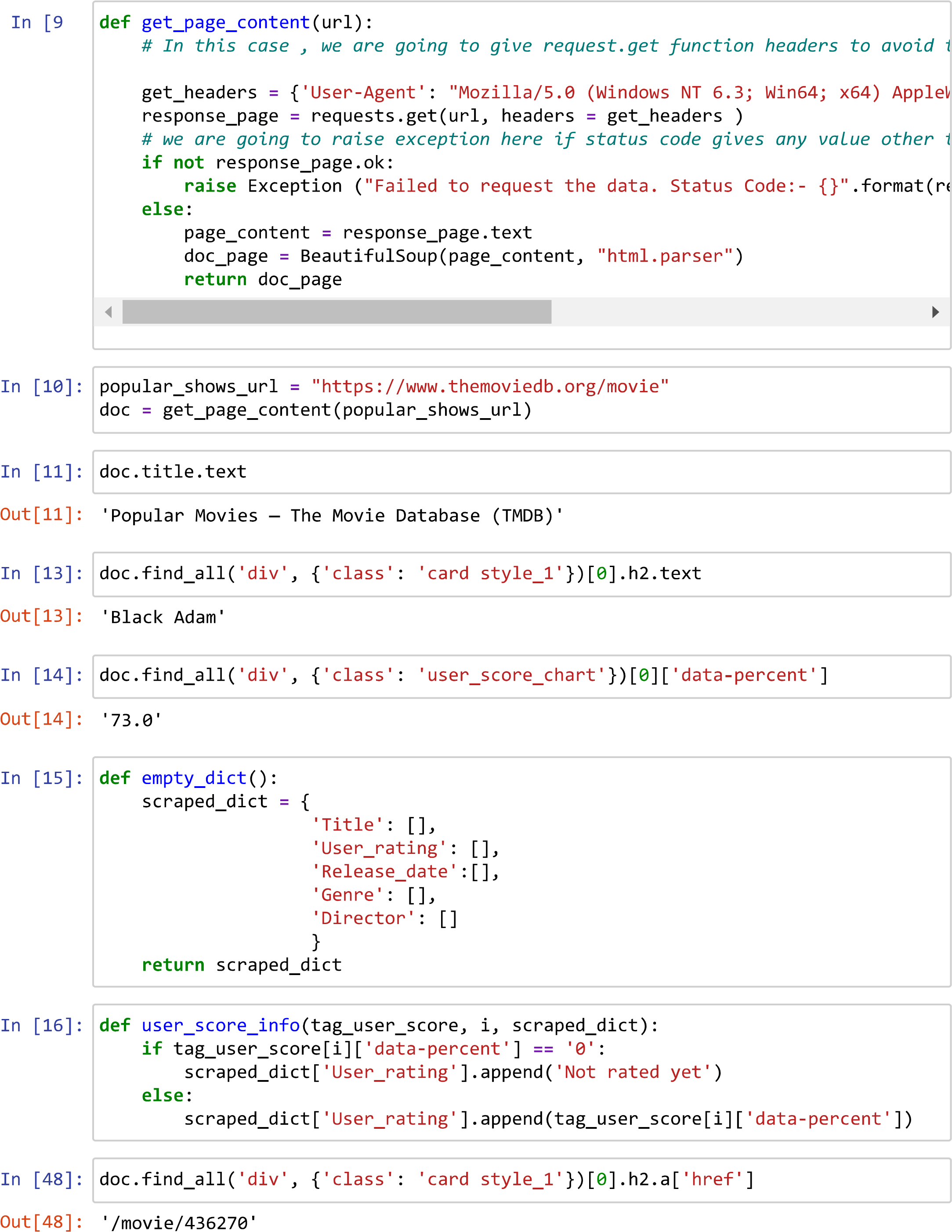
Requests is a simple python web scraping library. It is an efficient HTTP library used for accessing web pages. With the help of Requests, we can get the raw HTML of web pages which can then be parsed for retrieving the data.

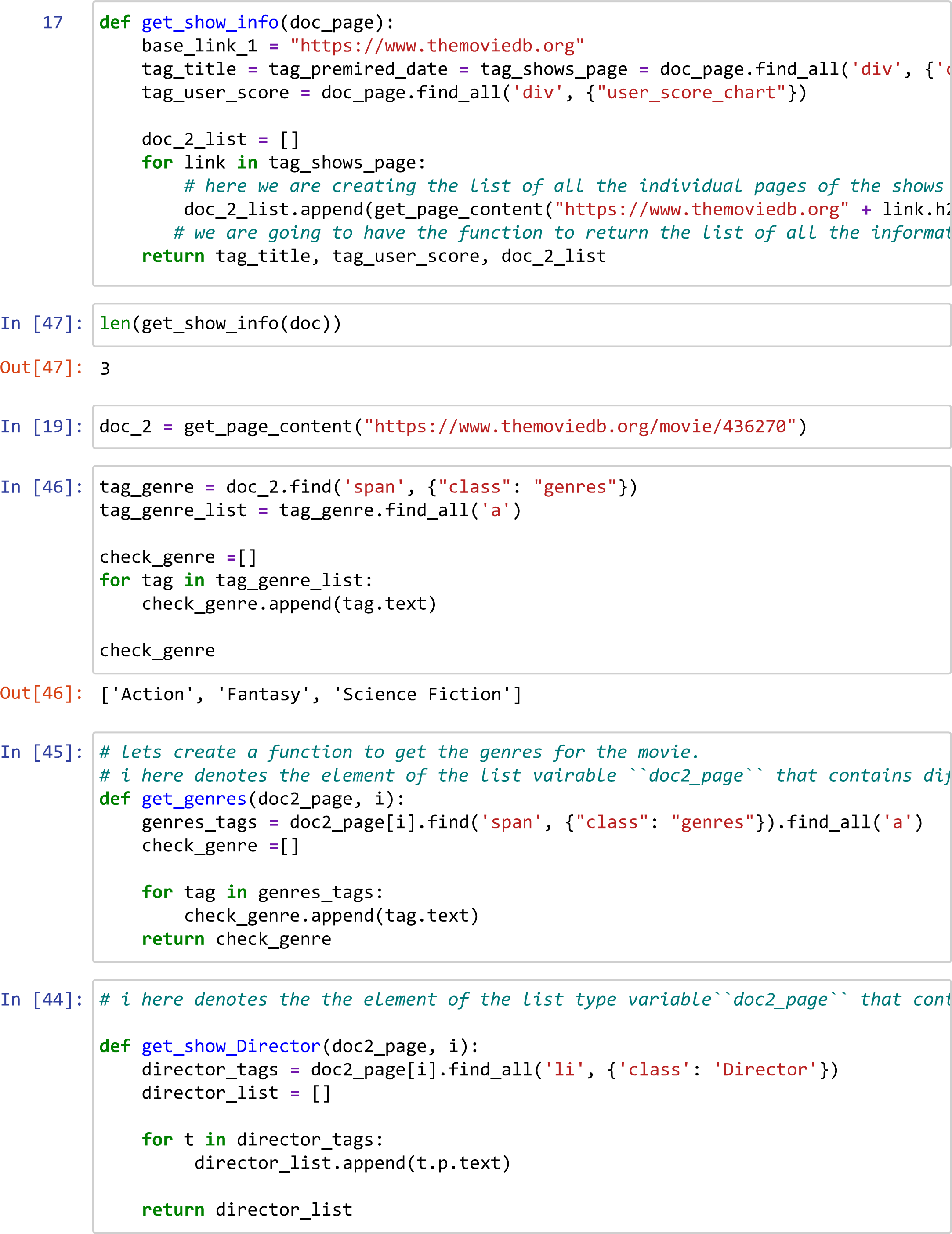
**Step 3: Parse HTML Code with Beautiful Soup**

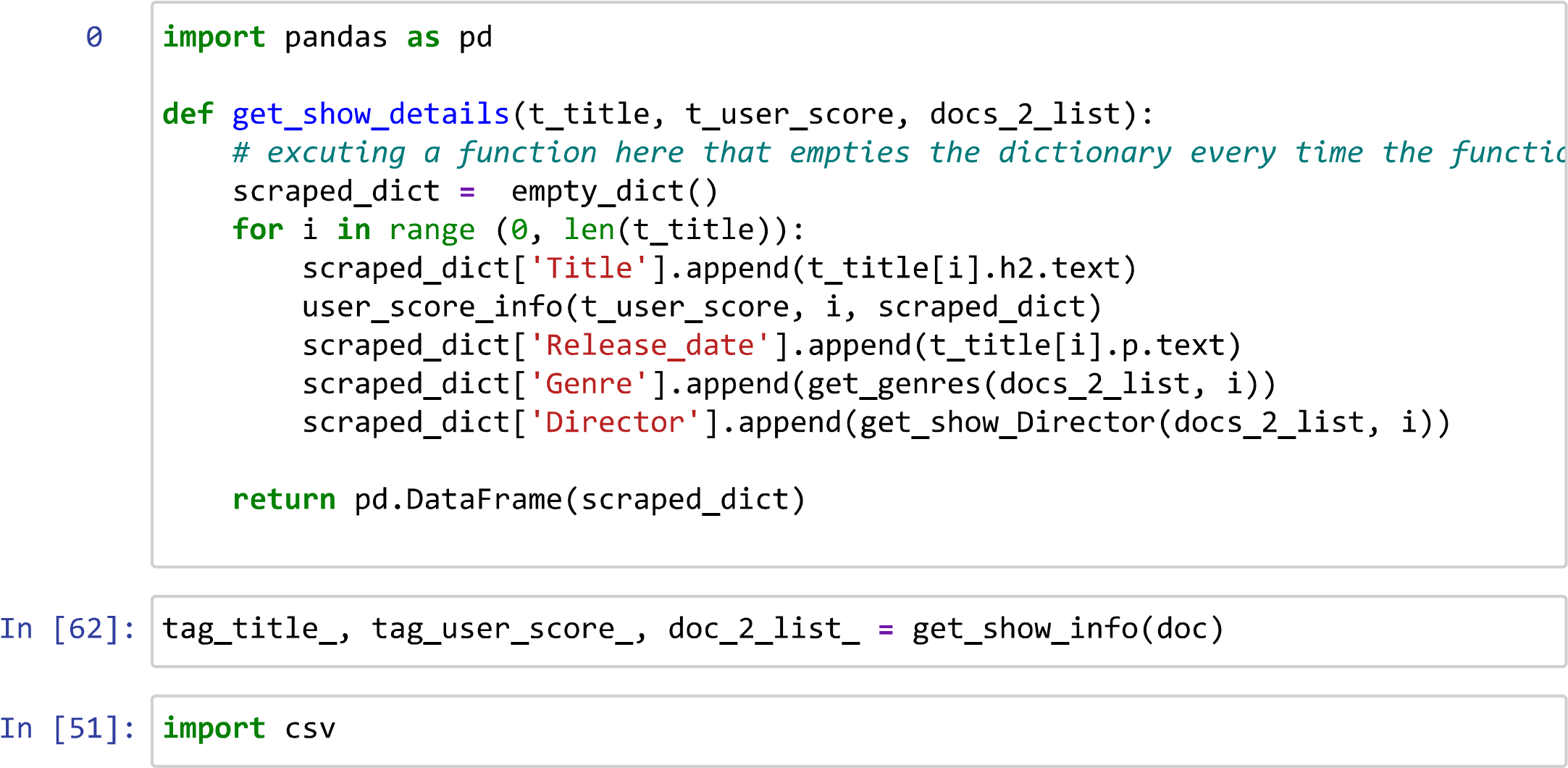
You’ve successfully scraped some HTML from the Internet, but when you look at it, it just seems like a huge mess. There are tons of HTML elements here and there, thousands of attributes scattered around—and wasn’t there some JavaScript mixed in as well? It’s time to parse this lengthy code response with the help of Python to make it more accessible and pick out the data you want.

[Beautiful Soup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/) is a Python library for **parsing structured data**. It allows you to interact with HTML in a similar way to how you interact with a web page using developer tools. The library exposes a couple of intuitive functions you can use to explore the HTML you received.

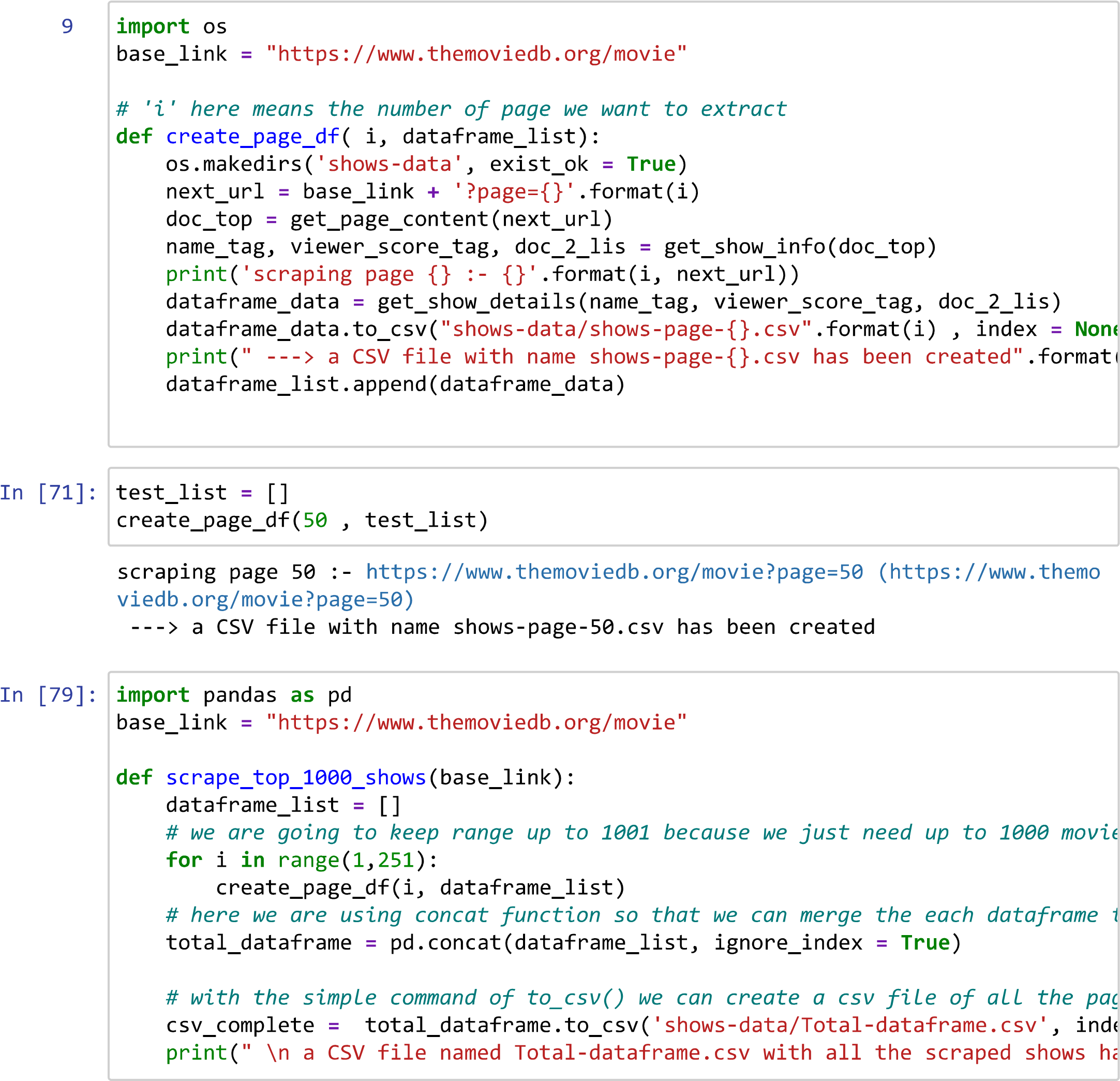
Below shown is scraping of tmdb website :-

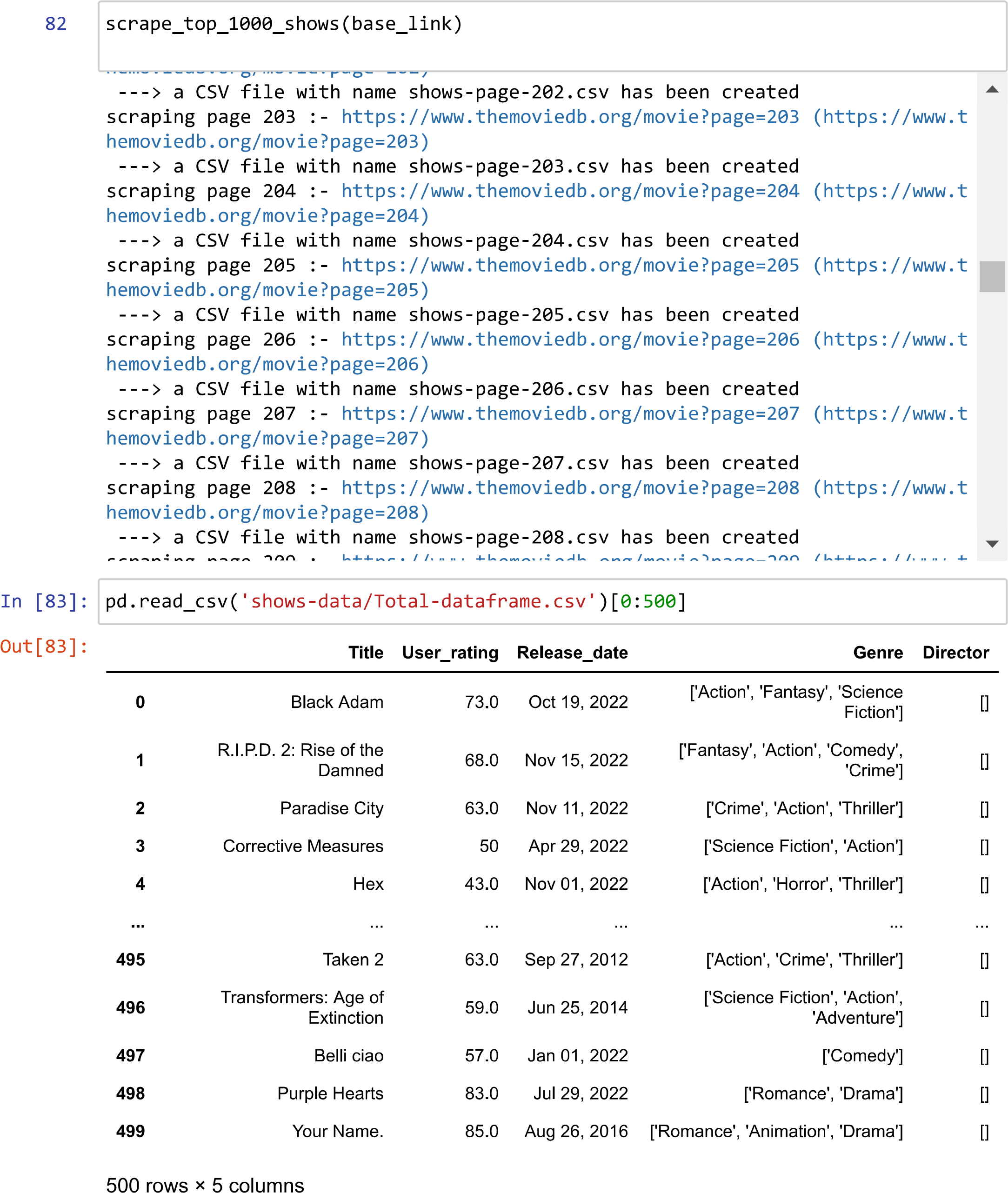














Conclusion:

Web scraping has become very important as the value of data is increasing day by day. There are many more advantages that web scraping brings to businesses.

Thank you