Week 4 Exercise - Python webscraper

Task：Build a simple webscraper that scrapes a set of documents from the internet and summarises them using gensim.

If you manage to achieve this, extract keywords from all the different documents and see if any are more popular than others.

Search for documents that contain those keywords using Python and then summarise those documents too.

Describe：

The process ：

get all of the data from inside a table that was displayed on a web page, the code would be written to go through these steps in sequence:

1．Request the content (source code) of a specific URL from the server

2．Download the content that is returned

3．Identify the elements of the page that are part of the table i want

4．Extract and (if necessary) reformat those elements into a dataset we can analyze or use required.

The first thing I'll do is download the web page. I can use the Python request library to download the page.

The request library will issue a GET request to the Web server, which will download the HTML content for us for a given Web page. GET is just one of several different types of requests you can make using requests.

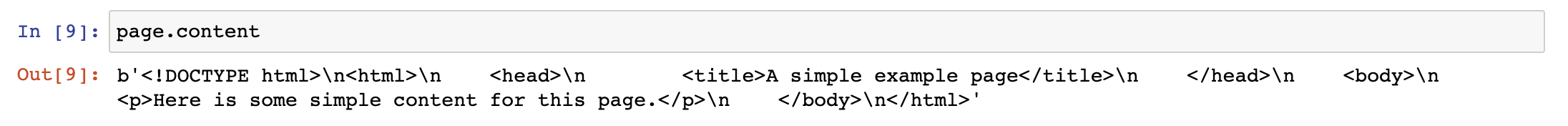
First import the request library, then use the request download page. Obtain method:

After running request, I get a Response object. This object has a status\_code property, which indicates if the page was downloaded successfully:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

print out the HTML content of the page usi ng the content property:



Parse the page using BeautifulSoup

I have now downloaded an HTML document.

I can use the beautifulSoup library to parse the document and extract the text from the p tag.

First, I must import the library and create an instance of the BeautifulSoup class to parse the document，then I printed out the HTML content of the page, formatted nicely, using the prettify method on the BeautifulSoup object.

图形用户界面, 文本, 应用程序, 电子邮件

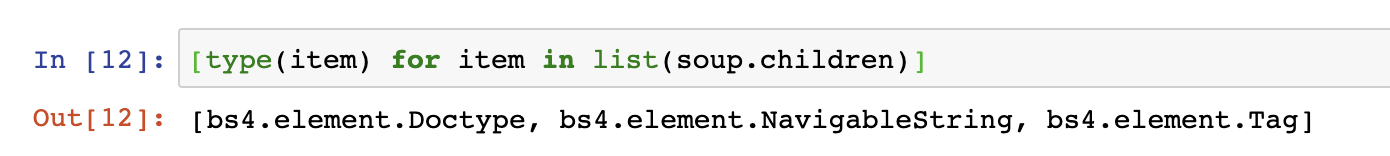
描述已自动生成

Since all the tags are nested, I can move through the structure one layer at a time. You can start by selecting all the elements at the top of the page using the soup's child attributes. Children returns a list generator, so I need to call the list function on it:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

The above tells that there are two tags at the top level of the page — the initial <!DOCTYPE html> tag, and the <html> tag. There is a newline character (n) in the list as well. Let’s see what the type of each element in the list is:



Tthat all of the items are BeautifulSoup objects:

The first is a Doctype object, which contains information about the type of the document.

The second is a NavigableString, which represents text found in the HTML document.

The final item is a Tag object, which contains other nested tags.

The most important object type, and the one we’ll deal with most often, is the Tag object.

The Tag object allows us to navigate through an HTML document, and extract other tags and text. You can learn more about the various BeautifulSoup objects here.

Now select the html tag and its children by taking the third item in the list:



Each item in the list returned by the children property is also a BeautifulSoup object, so I can also call the children method on html.

Now, I can find the children inside the html tag

图形用户界面, 文本, 应用程序, 电子邮件

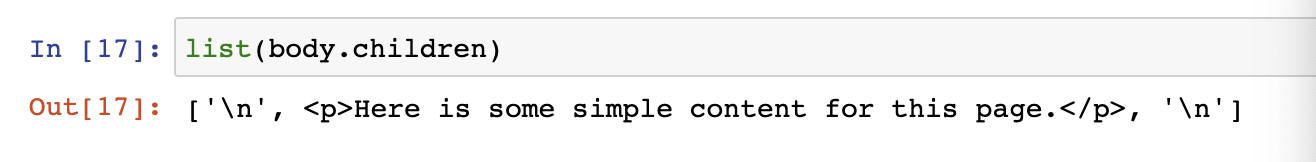
描述已自动生成

I want to extract the text inside the p tag, so I’ll dive into the body:

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成$

Now, i can get the p tag by finding the children of the body tag:

$

I can now isolate the P. tag:



Once I isolated the tag, i can use the get\_text method to extract all of the text inside the tag:

图形用户界面, 文本

描述已自动生成

If i want to extract a single tag, i can instead use the find\_all method, which will find all the instances of a tag on a page.

图形用户界面, 文本, 应用程序

描述已自动生成

Note that find\_all returns a list, so I ’ll have to loop through, or use list indexing, it to extract text:

文本

中度可信度描述已自动生成

use the find method, which will return a single BeautifulSoup object:

图形用户界面, 文本

描述已自动生成

First download the page and create a BeautifulSoup object:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

use the find\_all method to search for items by class or by id. In the below example, I ’ll search for any p tag that has the class outer-text:

文本

描述已自动生成

earch for elements by id:

文本

描述已自动生成

BeautifulSoup objects support searching a page via CSS selectors using the select method. We can use CSS selectors to find all the p tags in our page that are inside of a div like this: 文本

描述已自动生成

• Download the web page containing the forecast.

• Create a BeautifulSoup class to parse the page.

• Find the div with id seven-day-forecast, and assign to seven\_day

• Inside seven\_day, find each individual forecast item.

• Extract and print the first forecast item.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Extracting information from the page

• The name of the forecast item — in this case, Tonight.

• The description of the conditions — this is stored in the title property of img.

• A short description of the conditions — in this case, Mostly Clear.

• The temperature low — in this case, 49 degrees.

Extract the name of the forecast item, the short description, and the temperature first, since they’re all similar:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

extract the title attribute from the img tag. To do this, we just treat the Beautiful Soup object like a dictionary, and pass in the attribute we want as a key:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Extracting all the information from the page

• Select all items with the class period-name inside an item with the class tombstone-container in seven\_day.

• Use a list comprehension to call the get\_text method on each BeautifulSoup object.

图形用户界面, 文本, 应用程序

描述已自动生成

Apply the same technique to get the other three fields:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Combining our data into a Pandas Dataframe

Each dictionary key will become a column in the DataFrame, and each list will become the values in the column:

图形用户界面, 文本, 应用程序

描述已自动生成

summarises them using gensim.

图片包含 文本

描述已自动生成