Collect Data from Web

```
import pandas as pd
```

Wiki

Some websites maintains structured data, which is easy to read

```
table = pd.read_html('https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nomir
```

```
for i in table:
  print(type(i))
```

```
    <class 'pandas.core.frame.DataFrame'>
    <class 'pandas.core.frame.DataFrame'>
    <class 'pandas.core.frame.DataFrame'>
    <class 'pandas.core.frame.DataFrame'>
    <class 'pandas.core.frame.DataFrame'>
    <class 'pandas.core.frame.DataFrame'>
    <class 'pandas.core.frame.DataFrame'>
```

```
for i in table:
  print(i.columns)
```

```
    Int64Index([0], dtype='int64')
    Int64Index([0, 1, 2], dtype='int64')
    MultiIndex([( 'Country/Territory', 'Country/Territory'),
                (         'UN Region',         'UN Region'),
                (        'IMF[1][13]',          'Estimate'),
                (        'IMF[1][13]',              'Year'),
                (    'World Bank[14]',          'Estimate'),
                (    'World Bank[14]',              'Year'),
                ('United Nations[15]',          'Estimate'),
                ('United Nations[15]',              'Year')],
               )
    Index(['.mw-parser-output .navbar{display:inline;font-size:88%;font-weight:no
    Index(['vteLists of countries by GDP rankings', 'vteLists of countries by GDP
    Index(['vteEconomic classification of countries', 'vteEconomic classification
    Int64Index([0, 1], dtype='int64')
```

```
df = table[2]
df.head()
```

	Country/Territory	UN Region	IMF[1][13]		World Bank[14]		United Nations[15]	
	Country/Territory	UN Region	Estimate	Year	Estimate	Year	Estimate	Year
0	World	—	101560901	2022	96513077	2021	85328323	202
1	United States	Americas	25035164	2022	22996100	2021	20893746	202
2	China	Asia	18321197	[n 1]2022	17734063	[n 3]2021	14722801	[1]202
3	Japan	Asia	4300621	2022	4937422	2021	5057759	202

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217 entries, 0 to 216
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype 
---  ------                                  --------------  ----- 
 0   (Country/Territory, Country/Territory)  217 non-null    object
 1   (UN Region, UN Region)                  217 non-null    object
 2   (IMF[1][13], Estimate)                  217 non-null    object
 3   (IMF[1][13], Year)                      217 non-null    object
 4   (World Bank[14], Estimate)              217 non-null    object
 5   (World Bank[14], Year)                  217 non-null    object
 6   (United Nations[15], Estimate)          217 non-null    object
 7   (United Nations[15], Year)              217 non-null    object
dtypes: object(8)
memory usage: 13.7+ KB
```

Web Scraping

Some websites are semi-structured, which has metadata, such as labels, classes, etc, so we can look into their source code, and do web scraping.

> Note: You need to have a basic understanding of html, xml, in order to understand the source code and collect data from these websites.

> Note: Some websites prevent users from scraping or scraping rapidly.

The first thing we'll need to do to scrape a web page is to download the page. We can download pages using the Python `requests` library.

The requests library will make a `GET` request to a web server, which will download the HTML contents of a given web page for us. There are several different types of requests we can make using requests, of which `GET` is just one. If you want to learn more, check out our API tutorial.

Let's try downloading a simple sample website, https://dataquestio.github.io/web-scraping-pages/simple.html.

Download by requests

We'll need to first import the requests library, and then download the page using the requests.get method:

```
import requests

page = requests.get("https://dataquestio.github.io/web-scraping-pages/simple.html")
page
```

```
<Response [200]>
```

After running our request, we get a Response object. This object has a status_code property, which indicates if the page was downloaded successfully:

```
page.status_code
```

```
200
```

A status_code of 200 means that the page downloaded successfully. We won't fully dive into status codes here, but a status code starting with a 2 generally indicates success, and a code starting with a 4 or a 5 indicates an error.

We can print out the HTML content of the page using the content property:

```
page.content
```

```
b'<!DOCTYPE html>\n<html>\n    <head>\n        <title>A simple example
page</title>\n    </head>\n    <body>\n        <p>Here is some simple content
for this page.</p>\n    </body>\n</html>'
```

Parsing by BeautifulSoup

As you can see above, we now have downloaded an HTML document.

We can use the `BeautifulSoup` library to parse this document, and extract the text from the p tag.

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(page.content, 'html.parser')
```

We can now print out the HTML content of the page, formatted nicely, using the prettify method on the BeautifulSoup object.

```
print(soup.prettify())
```

```
<!DOCTYPE html>
<html>
 <head>
  <title>
   A simple example page
  </title>
 </head>
 <body>
  <p>
   Here is some simple content for this page.
  </p>
 </body>
</html>
```

This step isn't strictly necessary, and we won't always bother with it, but it can be helpful to look at prettified HTML to make the structure of the and where tags are nested easier to see.

Finding Tags

Finding all instances of a tag at once What we did above was useful for figuring out how to navigate a page, but it took a lot of commands to do something fairly simple. If we want to extract a single tag, we can instead use the find_all method, which will find all the instances of a tag on a page.

if we are looking for the title, we can look for `<title>` tag

```
soup.find_all('title')
```

```
[<title>A simple example page</title>]
```

```
for t in soup.find_all('title'):
  print(t.get_text())
```

```
A simple example page
```

If we are looking for text, we can look for `<p>` tag

```
for t in soup.find_all('p'):
  print(t.get_text())
```

```
Here is some simple content for this page.
```

If you instead only want to find the first instance of a tag, you can use the find method, which will return a single BeautifulSoup object:

```
soup.find('p').get_text()
```

```
'Here is some simple content for this page.'
```

Searching for tags by class and id:

Classes and ids are used by CSS to determine which HTML elements to apply certain styles to. But when we're scraping, we can also use them to specify the elements we want to scrape.

Let's try another page.

```
page = requests.get("https://dataquestio.github.io/web-scraping-pages/ids_and_class
soup = BeautifulSoup(page.content, 'html.parser')
soup
```

```
<html>
<head>
<title>A simple example page</title>
</head>
<body>
<div>
<p class="inner-text first-item" id="first">
                First paragraph.
            </p>
<p class="inner-text">
                Second paragraph.
            </p>
</div>
<p class="outer-text first-item" id="second">
<b>
                First outer paragraph.
            </b>
</p>
<p class="outer-text">
<b>
                Second outer paragraph.
            </b>
</p>
</body>
</html>
```

Now, we can use the find_all method to search for items by class or by id. In the below example, we'll search for any p tag that has the class outer-text:

```
soup.find_all('p', class_='outer-text')
```

```
[<p class="outer-text first-item" id="second">
 <b>
                First outer paragraph.
            </b>
 </p>, <p class="outer-text">
 <b>
                Second outer paragraph.
            </b>
 </p>]
```

In the below example, we'll look for any tag that has the class outer-text:

```
soup.find_all(class_="outer-text")
```

```
[<p class="outer-text first-item" id="second">
 <b>
                First outer paragraph.
            </b>
 </p>, <p class="outer-text">
 <b>
                Second outer paragraph.
            </b>
 </p>]
```

We can also search for elements by id:

```
soup.find_all(id="first")
```

```
[<p class="inner-text first-item" id="first">
                First paragraph.
            </p>]
```

Colab paid products - Cancel contracts here