IBM Developer
SKILLS NETWORK

Web Scraping Lab

Estimated time needed: **30** minutes

Objectives

After completing this lab you will be able to:

Table of Contents

Estimated time needed: **25 min**

For this lab, we are going to be using Python and several Python libraries. Some of these libraries might be installed in your lab environment or in SN Labs. Others may need to be installed by you. The cells below will install these libraries when executed.

In [1]:
```
!pip install bs4
#!pip install requests
```

```
Collecting bs4
  Downloading https://files.pythonhosted.org/packages/10/ed/7e8b97591f6
f456174139ec089c769f89a94a1a4025fe967691de971f314/bs4-0.0.1.tar.gz
Collecting beautifulsoup4 (from bs4)
  Downloading https://files.pythonhosted.org/packages/d1/41/e6495bd7d37
81cee623ce23ea6ac73282a373088fcd0ddc809a047b18eae/beautifulsoup4-4.9.3-
py3-none-any.whl (115kB)
    |████████████████████████████████| 122kB 24.5MB/s eta 0:00:01
Collecting soupsieve>1.2; python_version >= "3.0" (from beautifulsoup4-
>bs4)
  Downloading https://files.pythonhosted.org/packages/36/69/d82d04022f0
2733bf9a72bc3b96332d360c0c5307096d76f6bb7489f7e57/soupsieve-2.2.1-py3-n
one-any.whl
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/a0/b0/b2/4f80
b9456b87abedbc0bf2d52235414c3467d8889be38dd472
Successfully built bs4
Installing collected packages: soupsieve, beautifulsoup4, bs4
Successfully installed beautifulsoup4-4.9.3 bs4-0.0.1 soupsieve-2.2.1
```

Import the required modules and functions

In [2]:
```python
from bs4 import BeautifulSoup # this module helps in web scrapping.
import requests  # this module helps us to download a web page
```

Beautiful Soup Objects

Beautiful Soup is a Python library for pulling data out of HTML and XML files, we will focus on HTML files. This is accomplished by representing the HTML as a set of objects with methods used to parse the HTML. We can navigate the HTML as a tree and/or filter out what we are looking for.

Consider the following HTML:

In [3]:
```
%%html
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h3><b id='boldest'>Lebron James</b></h3>
<p> Salary: $ 92,000,000 </p>
```

```
<h3> Stephen Curry</h3>
<p> Salary: $85,000, 000 </p>
<h3> Kevin Durant </h3>
<p> Salary: $73,200, 000</p>
</body>
</html>
```

Lebron James

Salary: $ 92,000,000

Stephen Curry

Salary: $85,000, 000

Kevin Durant

Salary: $73,200, 000

We can store it as a string in the variable HTML:

In [4]:
```
html="<!DOCTYPE html><html><head><title>Page Title</title></head><body>
<h3><b id='boldest'>Lebron James</b></h3><p> Salary: $ 92,000,000 </p><
h3> Stephen Curry</h3><p> Salary: $85,000, 000 </p><h3> Kevin Durant </
h3><p> Salary: $73,200, 000</p></body></html>"
```

To parse a document, pass it into the `BeautifulSoup` constructor, the `BeautifulSoup` object, which represents the document as a nested data structure:

In [5]:
```
soup = BeautifulSoup(html, 'html5lib')
```

First, the document is converted to Unicode, (similar to ASCII), and HTML entities are converted to Unicode characters. Beautiful Soup transforms a complex HTML document into a complex tree of Python objects. The `BeautifulSoup` object can create other types of objects. In this lab, we will cover `BeautifulSoup` and `Tag` objects that for the purposes of this lab are identical, and `NavigableString` objects.

We can use the method `prettify()` to display the HTML in the nested structure:

In [6]:
```
print(soup.prettify())
```

```
<!DOCTYPE html>
<html>
 <head>
  <title>
   Page Title
  </title>
 </head>
 <body>
  <h3>
   <b id="boldest">
    Lebron James
   </b>
  </h3>
  <p>
   Salary: $ 92,000,000
  </p>
  <h3>
   Stephen Curry
  </h3>
  <p>
   Salary: $85,000, 000
  </p>
  <h3>
   Kevin Durant
  </h3>
  <p>
   Salary: $73,200, 000
  </p>
 </body>
</html>
```

Tags

Let's say we want the title of the page and the name of the top paid player we can use the `Tag`. The `Tag` object corresponds to an HTML tag in the original document, for example, the tag title.

In [7]:
```
tag_object=soup.title
print("tag object:",tag_object)
```

```
tag object: <title>Page Title</title>
```

we can see the tag type `bs4.element.Tag`

In [8]:
```
print("tag object type:",type(tag_object))
```

```
tag object type: <class 'bs4.element.Tag'>
```

If there is more than one `Tag` with the same name, the first element with that `Tag` name is called, this corresponds to the most paid player:

In [9]:
```
tag_object=soup.h3
tag_object
```

Out[9]:
```
<h3><b id="boldest">Lebron James</b></h3>
```

Enclosed in the bold attribute `b`, it helps to use the tree representation. We can navigate down the tree using the child attribute to get the name.

Children, Parents, and Siblings

As stated above the `Tag` object is a tree of objects we can access the child of the tag or navigate down the branch as follows:

In [10]:
```
tag_child =tag_object.b
tag_child
```
Out[10]:
```
<b id="boldest">Lebron James</b>
```

You can access the parent with the `parent`

In [11]:
```
parent_tag=tag_child.parent
parent_tag
```
Out[11]:
```
<h3><b id="boldest">Lebron James</b></h3>
```

this is identical to

In [ ]:
```
tag_object
```

`tag_object` parent is the `body` element.

In [ ]:
```
tag_object.parent
```

`tag_object` sibling is the `paragraph` element

In [13]:
```
sibling_1=tag_object.next_sibling
sibling_1
```
Out[13]:
```
<p> Salary: $ 92,000,000 </p>
```

`sibling_2` is the `header` element which is also a sibling of both `sibling_1` and `tag_object`

In [14]:
```
sibling_2=sibling_1.next_sibling
sibling_2
```

```
Out[14]: <h3> Stephen Curry</h3>
```

Exercise: `next_sibling`

Using the object `sibling_2` and the method `next_sibling` to find the salary of Stephen Curry:

```
In [17]: sibling_3=sibling_2.next_sibling
         sibling_3
```

```
Out[17]: <p> Salary: $85,000, 000 </p>
```

▶ Click here for the solution

HTML Attributes

If the tag has attributes, the tag `id="boldest"` has an attribute `id` whose value is `boldest`. You can access a tag's attributes by treating the tag like a dictionary:

```
In [18]: tag_child['id']
```

```
Out[18]: 'boldest'
```

You can access that dictionary directly as `attrs`:

```
In [19]: tag_child.attrs
```

```
Out[19]: {'id': 'boldest'}
```

You can also work with Multi-valued attribute check out [1] for more.

We can also obtain the content if the attribute of the `tag` using the Python `get()` method.

```
In [20]: tag_child.get('id')
```

```
Out[20]: 'boldest'
```

Navigable String

A string corresponds to a bit of text or content within a tag. Beautiful Soup uses the `NavigableString` class to contain this text. In our HTML we can obtain the name of the first player by extracting the sting of the `Tag` object `tag_child` as follows:

```
In [21]: tag_string=tag_child.string
         tag_string
```

```
Out[21]: 'Lebron James'
```

we can verify the type is Navigable String

```
In [22]: type(tag_string)
```

```
Out[22]: bs4.element.NavigableString
```

A NavigableString is just like a Python string or Unicode string, to be more precise. The main difference is that it also supports some `BeautifulSoup` features. We can covert it to sting object in Python:

```
In [23]: unicode_string = str(tag_string)
         unicode_string
```

```
Out[23]: 'Lebron James'
```

Filter

Filters allow you to find complex patterns, the simplest filter is a string. In this section we will pass a string to a different filter method and Beautiful Soup will perform a match against that exact string. Consider the following HTML of rocket launchs:

In [24]:
```
%%html
<table>
  <tr>
    <td id='flight' >Flight No</td>
    <td>Launch site</td>
    <td>Payload mass</td>
   </tr>
  <tr>
    <td>1</td>
    <td><a href='https://en.wikipedia.org/wiki/Florida'>Florida</a></td
>
    <td>300 kg</td>
  </tr>
  <tr>
    <td>2</td>
    <td><a href='https://en.wikipedia.org/wiki/Texas'>Texas</a></td>
    <td>94 kg</td>
  </tr>
  <tr>
    <td>3</td>
    <td><a href='https://en.wikipedia.org/wiki/Florida'>Florida<a> </td
>
    <td>80 kg</td>
  </tr>
</table>
```

Flight No	Launch site	Payload mass
1	Florida	300 kg
2	Texas	94 kg
3	Florida	80 kg

We can store it as a string in the variable `table`:

```
In [25]: table="<table><tr><td id='flight'>Flight No</td><td>Launch site</td> <td>Payload mass</td></tr><tr> <td>1</td><td><a href='https://en.wikipedia.org/wiki/Florida'>Florida<a></td><td>300 kg</td></tr><tr><td>2</td><td><a href='https://en.wikipedia.org/wiki/Texas'>Texas</a></td><td>94 kg</td></tr><tr><td>3</td><td><a href='https://en.wikipedia.org/wiki/Florida'>Florida<a> </td><td>80 kg</td></tr></table>"
```

```
In [27]: table_bs = BeautifulSoup(table, 'html5lib')
```

find All

The `find_all()` method looks through a tag's descendants and retrieves all descendants that match your filters.

The Method signature for `find_all(name, attrs, recursive, string, limit, **kwargs)`

Name

When we set the `name` parameter to a tag name, the method will extract all the tags with that name and its children.

In [28]:

```
table_rows=table_bs.find_all('tr')
table_rows
```

Out[28]:

```
[<tr><td id="flight">Flight No</td><td>Launch site</td> <td>Payload mass</td></tr
>,
 <tr> <td>1</td><td><a href="https://en.wikipedia.org/wiki/Florida">Florida</a><a
></a></td><td>300 kg</td></tr>,
 <tr><td>2</td><td><a href="https://en.wikipedia.org/wiki/Texas">Texas</a></td><t
d>94 kg</td></tr>,
 <tr><td>3</td><td><a href="https://en.wikipedia.org/wiki/Florida">Florida</a><a>
</a></td><td>80 kg</td></tr>]
```

The result is a Python Iterable just like a list, each element is a `tag` object:

In [1]:

```
first_row =table_rows[0]
first_row
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-8e8694bddf7e> in <module>
```

```
----> 1 first_row =table_rows[0]
      2 first_row

NameError: name 'table_rows' is not defined
```

The type is `tag`

In [30]:
```
print(type(first_row))
```

```
<class 'bs4.element.Tag'>
```

we can obtain the child

In [ ]:
```
first_row.td
```

If we iterate through the list, each element corresponds to a row in the table:

In [2]:

```
for i,row in enumerate(table_rows):
    print("row",i,"is",row)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-24eeda46f1a7> in <module>
----> 1 for i,row in enumerate(table_rows):
      2     print("row",i,"is",row)
      3 

NameError: name 'table_rows' is not defined
```

As `row` is a `cell` object, we can apply the method `find_all` to it and extract table cells in the object `cells` using the tag `td`, this is all the children with the name `td`. The result is a list, each element corresponds to a cell and is a `Tag` object, we can iterate through this list as well. We can extract the content using the `string` attribute.

In [ ]:

```
for i,row in enumerate(table_rows):
    print("row",i)
    cells=row.find_all('td')
    for j,cell in enumerate(cells):
        print('colunm',j,"cell",cell)
```

If we use a list we can match against any item in that list.

In [ ]:

```
list_input=table_bs .find_all(name=["tr", "td"])
list_input
```

Attributes

If the argument is not recognized it will be turned into a filter on the tag’s attributes. For example the `id` argument, Beautiful Soup will filter against each tag’s `id` attribute. For example, the first `td` elements have a value of `id` of `flight`, therefore we can filter based on that `id` value.

In [ ]:

```
table_bs.find_all(id="flight")
```

We can find all the elements that have links to the Florida Wikipedia page:

In [ ]:

```
list_input=table_bs.find_all(href="https://en.wikipedia.org/wiki/Florida")
list_input
```

If we set the `href` attribute to True, regardless of what the value is, the code finds all tags with `href` value:

In [ ]:

```
table_bs.find_all(href=True)
```

There are other methods for dealing with attributes and other related methods; Check out the following link

Exercise: `find_all`

Using the logic above, find all the elements without `href` value

In [ ]:

```

```

▶ Click here for the solution

Using the soup object `soup`, find the element with the `id` attribute content set to `"boldest"`.

In [ ]:

```

```

▶ Click here for the solution

string

With string you can search for strings instead of tags, where we find all the elments with Florida:

In [ ]:

```
table_bs.find_all(string="Florida")
```

find

The `find_all()` method scans the entire document looking for results, it’s if you are looking for one element you can use the `find()` method to find the first element in the document. Consider the following two table:

In [3]:

```
%%html
<h3>Rocket Launch </h3>

<p>
<table class='rocket'>
  <tr>
    <td>Flight No</td>
    <td>Launch site</td>
    <td>Payload mass</td>
  </tr>
  <tr>
    <td>1</td>
    <td>Florida</td>
    <td>300 kg</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Texas</td>
    <td>94 kg</td>
  </tr>
  <tr>
    <td>3</td>
    <td>Florida </td>
    <td>80 kg</td>
  </tr>
</table>
</p>
<p>

<h3>Pizza Party  </h3>


<table class='pizza'>
  <tr>
    <td>Pizza Place</td>
    <td>Orders</td>
    <td>Slices </td>
   </tr>
  <tr>
    <td>Domino's Pizza</td>
    <td>10</td>
    <td>100</td>
  </tr>
  <tr>
    <td>Little Caesars</td>
    <td>12</td>
    <td >144 </td>
  </tr>
  <tr>
    <td>Papa John's </td>
    <td>15 </td>
    <td>165</td>
  </tr>
```

Rocket Launch

Flight No	Launch site	Payload mass
1	Florida	300 kg
2	Texas	94 kg
3	Florida	80 kg

Pizza Party

We store the HTML as a Python string and assign `two_tables`:

In [ ]:

```
two_tables="<h3>Rocket Launch </h3><p><table class='rocket'><tr><td>Flight No</td><td>Launch site</td> <td>Payload mass</td></tr><tr><td>1</td><td>Florida</td><td>300 kg</td></tr><tr><td>2</td><td>Texas</td><td>94 kg</td></tr><tr><td>3</td><td>Florida </td><td>80 kg</td></tr></table></p><p><h3>Pizza Party  </h3><table class='pizza'><tr><td>Pizza Place</td><td>Orders</td> <td>Slices </td></tr><tr><td>Domino's Pizza</td><td>10</td><td>100</td></tr><tr><td>Little Caesars</td><td>12</td><td >144 </td></tr><tr><td>Papa John's </td><td>15 </td><td>165</td></tr>"
```

We create a `BeautifulSoup` object `two_tables_bs`

In [ ]:
```
two_tables_bs= BeautifulSoup(two_tables, 'html.parser')
```

We can find the first table using the tag name table

In [ ]:
```
two_tables_bs.find("table")
```

We can filter on the class attribute to find the second table, but because class is a keyword in Python, we add an underscore.

In [ ]:
```
two_tables_bs.find("table",class_='pizza')
```

Downloading And Scraping The Contents Of A Web Page

We Download the contents of the web page:

In [32]:
```
url = "http://www.ibm.com"
```

We use `get` to download the contents of the webpage in text format and store in a variable called `data`:

In [33]:
```
data  = requests.get(url).text
```

We create a `BeautifulSoup` object using the `BeautifulSoup` constructor

In [34]:
```
soup = BeautifulSoup(data,"html5lib")  # create a soup object usi
ng the variable 'data'
```

Scrape all links

In [35]:
```
for link in soup.find_all('a',href=True):  # in html anchor/link
 is represented by the tag <a>

    print(link.get('href'))
```

```
#main-content
http://www.ibm.com
https://www.ibm.com/security/ransomware?lnk=ushpv18l1
https://www.ibm.com/cloud/hybrid?lnk=ushpv18f1
https://www.ibm.com/services/talent-management/hr-outsourcing?lnk
=ushpv18f2
https://www.ibm.com/events/event/pages/ibm/vh7rknmb/1581037797007
001PJAd.html?lnk=ushpv18f3
https://www.ibm.com/thought-leadership/institute-business-value/r
eport/virtual-enterprise?lnk=ushpv18f4
https://www.ibm.com/products/offers-and-discounts?link=ushpv18t5&
lnk2=trial_mktpl_MPDISC
https://www.ibm.com/cloud/openshift/get-started?lnk=ushpv18t1&lnk
2=trial_RedHatOpenShift&psrc=none&pexp=def
https://www.ibm.com/security/identity-access-management/cloud-ide
ntity?lnk=ushpv18t2&lnk2=trial_Verify&psrc=none&pexp=def
https://www.ibm.com/products/planning-analytics?lnk=ushpv18t3&lnk
2=trial_PlanningAnalytics&psrc=none&pexp=def
https://www.ibm.com/cloud/instana?lnk=ushpv18t4&lnk2=trial_Instan
a&psrc=none&pexp=def
https://www.ibm.com/search?lnk=ushpv18srch&locale=en-us&q=
https://www.ibm.com/products?lnk=ushpv18p1&lnk2=trial_mktpl&psrc=
none&pexp=def
https://developer.ibm.com/depmodels/cloud/?lnk=ushpv18ct16
```

```
https://developer.ibm.com/technologies/artificial-intelligence?ln
k=ushpv18ct19
https://www.ibm.com/demos/?lnk=ushpv18ct12
https://developer.ibm.com/?lnk=ushpv18ct9
https://www.ibm.com/docs/en?lnk=ushpv18ct14
https://www.redbooks.ibm.com/?lnk=ushpv18ct10
https://www.ibm.com/support/home/?lnk=ushpv18ct11
https://www.ibm.com/training/?lnk=ushpv18ct15
https://www.ibm.com/cloud/hybrid?lnk=ushpv18ct20
https://www.ibm.com/cloud/learn/public-cloud?lnk=ushpv18ct17
https://www.ibm.com/cloud/redhat?lnk=ushpv18ct13
https://www.ibm.com/artificial-intelligence?lnk=ushpv18ct3
https://www.ibm.com/quantum-computing?lnk=ushpv18ct18
https://www.ibm.com/cloud/learn/kubernetes?lnk=ushpv18ct8
https://www.ibm.com/products/spss-statistics?lnk=ushpv18ct7
https://www.ibm.com/blockchain?lnk=ushpv18ct1
https://www-03.ibm.com/employment/technicaltalent/developer/?lnk=
ushpv18ct2
https://www.ibm.com/cloud/automation?lnk=ushpv18ct21
https://www.ibm.com/search?lnk=ushpv18srch&locale=en-us&q=
https://www.ibm.com/products?lnk=ushpv18p1&lnk2=trial_mktpl&psrc=
none&pexp=def
https://www.ibm.com/cloud/hybrid?lnk=ushpv18pt14&bv=true
https://www.ibm.com/watson?lnk=ushpv18pt17&bv=true
https://www.ibm.com/us-en/products/categories?technologyTopics[0]
[0]=cat.topic:Blockchain&isIBMOffering[0]=true&lnk=ushpv18pt4&bv=
true
https://www.ibm.com/us-en/products/category/technology/analytics?
lnk=ushpv18pt1&bv=true
https://www.ibm.com/financing?lnk=ushpv18pt3&bv=true
https://www.ibm.com/cloud/public?lnk=ushpv18pt15&bv=true
https://www.ibm.com/garage?lnk=ushpv18pt13&bv=true
https://www.ibm.com/thought-leadership/institute-business-value/?
lnk=ushpv18pt12&bv=true
https://www.ibm.com/us-en/products/category/technology/security?l
nk=ushpv18pt9&bv=true
https://www.ibm.com/quantum-computing?lnk=ushpv18pt16&bv=true
https://www.ibm.com/cloud/hybrid?lnk=ushpv18ct20
https://www.ibm.com/cloud/public?lnk=ushpv18ct17
https://www.ibm.com/cloud/redhat?lnk=ushpv18ct13
https://www.ibm.com/artificial-intelligence?lnk=ushpv18ct3
https://www.ibm.com/quantum-computing?lnk=ushpv18ct18
https://www.ibm.com/cloud/learn/kubernetes?lnk=ushpv18ct8
https://www.ibm.com/products/spss-statistics?lnk=ushpv18ct7
https://www.ibm.com/blockchain?lnk=ushpv18ct1
```

```
https://www-03.ibm.com/employment/technicaltalent/developer/?lnk=
ushpv18ct2
https://www.ibm.com/
```

Scrape all images Tags

In [36]:

```
for link in soup.find_all('img'):# in html image is represented b
y the tag <img>
    print(link)
    print(link.get('src'))
```

```
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iMTA1NSIgaGVpZ2h0PSI1MjcuNSIgeG1s
bnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4="
style="max-width:100%;display:block;margin:0;border:none;padding:
0"/>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iMTA1NSIgaGVpZ2h0PSI1Mjc
uNSIgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIx
LjEiLz4=
<img alt="leadspace mobile image" class="ibm-resize" decoding="as
ync" src="https://1.dam.s81c.com/public/content/dam/worldwide-con
tent/homepage/ul/g/41/85/20210524-ls-ransomware-25915-720x360.jp
g" style="position:absolute;top:0;left:0;bottom:0;right:0;box-siz
ing:border-box;padding:0;border:none;margin:auto;display:block;wi
dth:0;height:0;min-width:100%;max-width:100%;min-height:100%;max-
height:100%"/>
https://1.dam.s81c.com/public/content/dam/worldwide-content/homep
age/ul/g/41/85/20210524-ls-ransomware-25915-720x360.jpg
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjMyMCIgeG1sbnM9
Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4=" sty
le="max-width:100%;display:block;margin:0;border:none;padding:0"/
>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjMyMCI
```

geG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEi
Lz4=
<img alt="The hybrid cloud advantage" class="ibm-resize ibm-ab-im
age featured-image" decoding="async" src="https://1.dam.s81c.com/
public/content/dam/worldwide-content/homepage/ul/g/d2/9d/20210524
-f-hybrid-cloud-open-shift-25911.jpg" style="position:absolute;to
p:0;left:0;bottom:0;right:0;box-sizing:border-box;padding:0;borde
r:none;margin:auto;display:block;width:0;height:0;min-width:100%;
max-width:100%;min-height:100%;max-height:100%"/>
https://1.dam.s81c.com/public/content/dam/worldwide-content/homep
age/ul/g/d2/9d/20210524-f-hybrid-cloud-open-shift-25911.jpg
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjMyMCIgeG1sbnM9
Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4=" sty
le="max-width:100%;display:block;margin:0;border:none;padding:0"/
>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjMyMCI
geG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEi
Lz4=
<img alt="This is what modern HR looks like" class="ibm-resize ib
m-ab-image featured-image" decoding="async" src="https://1.dam.s8
1c.com/public/content/dam/worldwide-content/homepage/ul/g/1d/65/2
0210524-f-hr-oursourcing-25898.jpg" style="position:absolute;top:
0;left:0;bottom:0;right:0;box-sizing:border-box;padding:0;border:
none;margin:auto;display:block;width:0;height:0;min-width:100%;ma
x-width:100%;min-height:100%;max-height:100%"/>
https://1.dam.s81c.com/public/content/dam/worldwide-content/homep
age/ul/g/1d/65/20210524-f-hr-oursourcing-25898.jpg
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjMyMCIgeG1sbnM9
Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4=" sty
le="max-width:100%;display:block;margin:0;border:none;padding:0"/
>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjMyMCI
geG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEi
Lz4=
<img alt="IBM Health Forum, June&amp;nbsp;10" class="ibm-resize i
bm-ab-image featured-image" decoding="async" src="https://1.dam.s
81c.com/public/content/dam/worldwide-content/homepage/ul/g/15/19/
20210524-f-watson-health-forum-c-25877-444x320.jpg" style="positi
on:absolute;top:0;left:0;bottom:0;right:0;box-sizing:border-box;p
adding:0;border:none;margin:auto;display:block;width:0;height:0;m
in-width:100%;max-width:100%;min-height:100%;max-height:100%"/>
https://1.dam.s81c.com/public/content/dam/worldwide-content/homep
age/ul/g/15/19/20210524-f-watson-health-forum-c-25877-444x320.jpg

```
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjMyMCIgeG1sbnM9
Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4=" sty
le="max-width:100%;display:block;margin:0;border:none;padding:0"/
>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjMyMCI
geG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEi
Lz4=
<img alt="The Virtual Enterprise is here" class="ibm-resize ibm-a
b-image featured-image" decoding="async" src="https://1.dam.s81c.
com/public/content/dam/worldwide-content/homepage/ul/g/27/c9/2021
0524-ibv-virtual-enterprise-d-25840-444x320.jpg" style="position:
absolute;top:0;left:0;bottom:0;right:0;box-sizing:border-box;padd
ing:0;border:none;margin:auto;display:block;width:0;height:0;min-
width:100%;max-width:100%;min-height:100%;max-height:100%"/>
https://1.dam.s81c.com/public/content/dam/worldwide-content/homep
age/ul/g/27/c9/20210524-ibv-virtual-enterprise-d-25840-444x320.jp
g
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjI2MCIgeG1sbnM9
Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4=" sty
le="max-width:100%;display:block;margin:0;border:none;padding:0"/
>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjI2MCI
geG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEi
Lz4=
<img alt="Red Hat OpenShift on IBM&amp;nbsp;Cloud" class="ibm-res
ize ibm-ab-image trials-image" decoding="async" src="data:image/g
if;base64,R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAABAAEAAAIBRAA
7" style="position:absolute;top:0;left:0;bottom:0;right:0;box-siz
ing:border-box;padding:0;border:none;margin:auto;display:block;wi
dth:0;height:0;min-width:100%;max-width:100%;min-height:100%;max-
height:100%"/>
data:image/gif;base64,R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAA
BAAEAAAIBRAA7
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjI2MCIgeG1sbnM9
Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4=" sty
le="max-width:100%;display:block;margin:0;border:none;padding:0"/
>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjI2MCI
geG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEi
Lz4=
<img alt="IBM Security Verify" class="ibm-resize ibm-ab-image tri
als-image" decoding="async" src="data:image/gif;base64,R0lGODlhAQ
```

ABAIAAAAAAAP///yH5BAEAAAAALAAAAAABAAEAAAIBRAA7" style="position:a
bsolute;top:0;left:0;bottom:0;right:0;box-sizing:border-box;paddi
ng:0;border:none;margin:auto;display:block;width:0;height:0;min-w
idth:100%;max-width:100%;min-height:100%;max-height:100%"/>
data:image/gif;base64,R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAA
BAAEAAAIBRAA7
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjI2MCIgeG1sbnM9
Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4=" sty
le="max-width:100%;display:block;margin:0;border:none;padding:0"/
>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjI2MCI
geG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEi
Lz4=
<img alt="IBM Planning Analytics" class="ibm-resize ibm-ab-image
trials-image" decoding="async" src="data:image/gif;base64,R0lGODl
hAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAABAAEAAAIBRAA7" style="positio
n:absolute;top:0;left:0;bottom:0;right:0;box-sizing:border-box;pa
dding:0;border:none;margin:auto;display:block;width:0;height:0;mi
n-width:100%;max-width:100%;min-height:100%;max-height:100%"/>
data:image/gif;base64,R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAA
BAAEAAAIBRAA7
<img alt="" aria-hidden="true" role="presentation" src="data:imag
e/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjI2MCIgeG1sbnM9
Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEiLz4=" sty
le="max-width:100%;display:block;margin:0;border:none;padding:0"/
>
data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNDQwIiBoZWlnaHQ9IjI2MCI
geG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB2ZXJzaW9uPSIxLjEi
Lz4=
<img alt="IBM Observability by Instana" class="ibm-resize ibm-ab-
image trials-image" decoding="async" src="data:image/gif;base64,R
0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAABAAEAAAIBRAA7" style="p
osition:absolute;top:0;left:0;bottom:0;right:0;box-sizing:border-
box;padding:0;border:none;margin:auto;display:block;width:0;heigh
t:0;min-width:100%;max-width:100%;min-height:100%;max-height:10
0%"/>
data:image/gif;base64,R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAA
BAAEAAAIBRAA7

Scrape data from HTML tables

In [37]:
```
#The below url contains an html table with data about colors and
 color codes.
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdoma
in.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/HTMLColorCodes.
html"
```

Before proceeding to scrape a web site, you need to examine the contents, and the way data is organized on the website. Open the above url in your browser and check how many rows and columns are there in the color table.

In [38]:
```
# get the contents of the webpage in text format and store in a v
ariable called data
data  = requests.get(url).text
```

In [39]:
```
soup = BeautifulSoup(data,"html5lib")
```

In [40]:
```
#find a html table in the web page
table = soup.find('table') # in html table is represented by the
 tag <table>
```

In [41]:

```
#Get all rows from the table
for row in table.find_all('tr'): # in html table row is represent
ed by the tag <tr>
    # Get all columns in each row.
    cols = row.find_all('td') # in html a column is represented b
y the tag <td>
    color_name = cols[2].string # store the value in column 3 as
 color_name
    color_code = cols[3].string # store the value in column 4 as
 color_code
    print("{}--->{}".format(color_name,color_code))
```

```
Color Name--->None
lightsalmon--->#FFA07A
salmon--->#FA8072
darksalmon--->#E9967A
lightcoral--->#F08080
coral--->#FF7F50
tomato--->#FF6347
orangered--->#FF4500
gold--->#FFD700
orange--->#FFA500
darkorange--->#FF8C00
lightyellow--->#FFFFE0
lemonchiffon--->#FFFACD
papayawhip--->#FFEFD5
moccasin--->#FFE4B5
peachpuff--->#FFDAB9
palegoldenrod--->#EEE8AA
khaki--->#F0E68C
darkkhaki--->#BDB76B
yellow--->#FFFF00
lawngreen--->#7CFC00
chartreuse--->#7FFF00
limegreen--->#32CD32
lime--->#00FF00
```

```
forestgreen--->#228B22
green--->#008000
powderblue--->#B0E0E6
lightblue--->#ADD8E6
lightskyblue--->#87CEFA
skyblue--->#87CEEB
deepskyblue--->#00BFFF
lightsteelblue--->#B0C4DE
dodgerblue--->#1E90FF
```

Scrape data from HTML tables into a DataFrame using BeautifulSoup and Pandas

In [42]:

```
import pandas as pd
```

In [43]:

```
#The below url contains html tables with data about world populat
ion.
url = "https://en.wikipedia.org/wiki/World_population"
```

Before proceeding to scrape a web site, you need to examine the contents, and the way data is organized on the website. Open the above url in your browser and check the tables on the webpage.

In [44]:

```
# get the contents of the webpage in text format and store in a v
ariable called data
data  = requests.get(url).text
```

In [45]:

```
soup = BeautifulSoup(data,"html5lib")
```

In [46]:

```
#find all html tables in the web page
tables = soup.find_all('table') # in html table is represented by
the tag <table>
```

In [47]:

```
# we can see how many tables were found by checking the length of
the tables list
len(tables)
```

Out[47]:

```
26
```

Assume that we are looking for the `10 most densly populated countries` table, we can look through the tables list and find the right one we are look for based on the data in each table or we can search for the table name if it is in the table but this option might not always work.

In [48]:
```
for index,table in enumerate(tables):
    if ("10 most densely populated countries" in str(table)):
        table_index = index
print(table_index)
```

```
5
```

See if you can locate the table name of the table, `10 most densly populated countries` , below.

In [49]:
```
print(tables[table_index].prettify())
```

```
<table class="wikitable sortable" style="text-align:right">
 <caption>
  10 most densely populated countries
  <small>
   (with population above 5 million)
  </small>
 </caption>
 <tbody>
  <tr>
   <th>
    Rank
   </th>
```

```
    <th>
     Country
    </th>
    <th>
     Population
    </th>
    <th>
     Area
     <br/>
     <small>
      (km
      <sup>
       2
      </sup>
      )
     </small>
    </th>
    <th>
     Density
     <br/>
     <small>
      (pop/km
      <sup>
       2
      </sup>
      )
     </small>
    </th>
   </tr>
   <tr>
    <td>
     1
    </td>
    <td align="left">
     <span class="flagicon">
      <img alt="" class="thumbborder" data-file-height="2880" data
-file-width="4320" decoding="async" height="15" src="//upload.wik
imedia.org/wikipedia/commons/thumb/4/48/Flag_of_Singapore.svg/23p
x-Flag_of_Singapore.svg.png" srcset="//upload.wikimedia.org/wikip
edia/commons/thumb/4/48/Flag_of_Singapore.svg/35px-Flag_of_Singap
ore.svg.png 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/
4/48/Flag_of_Singapore.svg/45px-Flag_of_Singapore.svg.png 2x" wid
th="23"/>
     </span>
     <a href="/wiki/Singapore" title="Singapore">
    Singapore
   </a>
  </td>
  <td>
   5,704,000
  </td>
  <td>
   710
  </td>
  <td>
   8,033
  </td>
 </tr>
 <tr>
  <td>
   2
  </td>
  <td align="left">
   <span class="flagicon">
    <img alt="" class="thumbborder" data-file-height="600" data-
file-width="1000" decoding="async" height="14" src="//upload.wiki
media.org/wikipedia/commons/thumb/f/f9/Flag_of_Bangladesh.svg/23p
x-Flag_of_Bangladesh.svg.png" srcset="//upload.wikimedia.org/wiki
pedia/commons/thumb/f/f9/Flag_of_Bangladesh.svg/35px-Flag_of_Bang
ladesh.svg.png 1.5x, //upload.wikimedia.org/wikipedia/commons/thu
mb/f/f9/Flag_of_Bangladesh.svg/46px-Flag_of_Bangladesh.svg.png 2
x" width="23"/>
   </span>
   <a href="/wiki/Bangladesh" title="Bangladesh">
    Bangladesh
   </a>
  </td>
  <td>
   170,740,000
  </td>
  <td>
   143,998
  </td>
  <td>
   1,186
  </td>
 </tr>
 <tr>
  <td>
   3
   </td>
   <td align="left">
    <span class="flagicon">
     <img alt="" class="thumbborder" data-file-height="600" data-
file-width="900" decoding="async" height="15" src="//upload.wikim
edia.org/wikipedia/commons/thumb/5/59/Flag_of_Lebanon.svg/23px-Fl
ag_of_Lebanon.svg.png" srcset="//upload.wikimedia.org/wikipedia/c
ommons/thumb/5/59/Flag_of_Lebanon.svg/35px-Flag_of_Lebanon.svg.pn
g 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/5/59/Flag_
of_Lebanon.svg/45px-Flag_of_Lebanon.svg.png 2x" width="23"/>
    </span>
    <a href="/wiki/Lebanon" title="Lebanon">
     Lebanon
    </a>
   </td>
   <td>
    6,856,000
   </td>
   <td>
    10,452
   </td>
   <td>
    656
   </td>
  </tr>
  <tr>
   <td>
    4
   </td>
   <td align="left">
    <span class="flagicon">
     <img alt="" class="thumbborder" data-file-height="600" data-
file-width="900" decoding="async" height="15" src="//upload.wikim
edia.org/wikipedia/commons/thumb/7/72/Flag_of_the_Republic_of_Chi
na.svg/23px-Flag_of_the_Republic_of_China.svg.png" srcset="//uplo
ad.wikimedia.org/wikipedia/commons/thumb/7/72/Flag_of_the_Republi
c_of_China.svg/35px-Flag_of_the_Republic_of_China.svg.png 1.5x,
//upload.wikimedia.org/wikipedia/commons/thumb/7/72/Flag_of_the_R
epublic_of_China.svg/45px-Flag_of_the_Republic_of_China.svg.png 2
x" width="23"/>
    </span>
    <a href="/wiki/Taiwan" title="Taiwan">
     Taiwan
    </a>
   </td>
   <td>
    23,604,000
   </td>
   <td>
    36,193
   </td>
   <td>
    652
   </td>
  </tr>
  <tr>
   <td>
    5
   </td>
   <td align="left">
    <span class="flagicon">
     <img alt="" class="thumbborder" data-file-height="600" data-
file-width="900" decoding="async" height="15" src="//upload.wikim
edia.org/wikipedia/commons/thumb/0/09/Flag_of_South_Korea.svg/23p
x-Flag_of_South_Korea.svg.png" srcset="//upload.wikimedia.org/wik
ipedia/commons/thumb/0/09/Flag_of_South_Korea.svg/35px-Flag_of_So
uth_Korea.svg.png 1.5x, //upload.wikimedia.org/wikipedia/commons/
thumb/0/09/Flag_of_South_Korea.svg/45px-Flag_of_South_Korea.svg.p
ng 2x" width="23"/>
    </span>
    <a href="/wiki/South_Korea" title="South Korea">
     South Korea
    </a>
   </td>
   <td>
    51,781,000
   </td>
   <td>
    99,538
   </td>
   <td>
    520
   </td>
  </tr>
  <tr>
   <td>
    6
   </td>
   <td align="left">
    <span class="flagicon">
```

```
     <img alt="" class="thumbborder" data-file-height="720" data-
file-width="1080" decoding="async" height="15" src="//upload.wiki
media.org/wikipedia/commons/thumb/1/17/Flag_of_Rwanda.svg/23px-Fl
ag_of_Rwanda.svg.png" srcset="//upload.wikimedia.org/wikipedia/co
mmons/thumb/1/17/Flag_of_Rwanda.svg/35px-Flag_of_Rwanda.svg.png
1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/1/17/Flag_of
_Rwanda.svg/45px-Flag_of_Rwanda.svg.png 2x" width="23"/>
    </span>
    <a href="/wiki/Rwanda" title="Rwanda">
     Rwanda
    </a>
   </td>
   <td>
    12,374,000
   </td>
   <td>
    26,338
   </td>
   <td>
    470
   </td>
  </tr>
  <tr>
   <td>
    7
   </td>
   <td align="left">
    <span class="flagicon">
     <img alt="" class="thumbborder" data-file-height="600" data-
file-width="1000" decoding="async" height="14" src="//upload.wiki
media.org/wikipedia/commons/thumb/5/56/Flag_of_Haiti.svg/23px-Fla
g_of_Haiti.svg.png" srcset="//upload.wikimedia.org/wikipedia/comm
ons/thumb/5/56/Flag_of_Haiti.svg/35px-Flag_of_Haiti.svg.png 1.5x,
//upload.wikimedia.org/wikipedia/commons/thumb/5/56/Flag_of_Hait
i.svg/46px-Flag_of_Haiti.svg.png 2x" width="23"/>
    </span>
    <a href="/wiki/Haiti" title="Haiti">
     Haiti
    </a>
   </td>
   <td>
    11,578,000
   </td>
   <td>
    27,065
   </td>
   <td>
    428
   </td>
  </tr>
  <tr>
   <td>
    8
   </td>
   <td align="left">
    <span class="flagicon">
     <img alt="" class="thumbborder" data-file-height="600" data-
file-width="900" decoding="async" height="15" src="//upload.wikim
edia.org/wikipedia/commons/thumb/2/20/Flag_of_the_Netherlands.sv
g/23px-Flag_of_the_Netherlands.svg.png" srcset="//upload.wikimedi
a.org/wikipedia/commons/thumb/2/20/Flag_of_the_Netherlands.svg/35
px-Flag_of_the_Netherlands.svg.png 1.5x, //upload.wikimedia.org/w
ikipedia/commons/thumb/2/20/Flag_of_the_Netherlands.svg/45px-Flag
_of_the_Netherlands.svg.png 2x" width="23"/>
    </span>
    <a href="/wiki/Netherlands" title="Netherlands">
     Netherlands
    </a>
   </td>
   <td>
    17,600,000
   </td>
   <td>
    41,526
   </td>
   <td>
    424
   </td>
  </tr>
  <tr>
   <td>
    9
   </td>
   <td align="left">
    <span class="flagicon">
     <img alt="" class="thumbborder" data-file-height="800" data-
file-width="1100" decoding="async" height="15" src="//upload.wiki
media.org/wikipedia/commons/thumb/d/d4/Flag_of_Israel.svg/21px-Fl
ag_of_Israel.svg.png" srcset="//upload.wikimedia.org/wikipedia/co
mmons/thumb/d/d4/Flag_of_Israel.svg/32px-Flag_of_Israel.svg.png
```

```
1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/d/d4/Flag_of
_Israel.svg/41px-Flag_of_Israel.svg.png 2x" width="21"/>
    </span>
    <a href="/wiki/Israel" title="Israel">
     Israel
    </a>
   </td>
   <td>
    9,350,000
   </td>
   <td>
    22,072
   </td>
   <td>
    424
   </td>
  </tr>
  <tr>
   <td>
    10
   </td>
   <td align="left">
    <span class="flagicon">
     <img alt="" class="thumbborder" data-file-height="900" data-
file-width="1350" decoding="async" height="15" src="//upload.wiki
media.org/wikipedia/en/thumb/4/41/Flag_of_India.svg/23px-Flag_of_
India.svg.png" srcset="//upload.wikimedia.org/wikipedia/en/thumb/
4/41/Flag_of_India.svg/35px-Flag_of_India.svg.png 1.5x, //upload.
wikimedia.org/wikipedia/en/thumb/4/41/Flag_of_India.svg/45px-Flag
_of_India.svg.png 2x" width="23"/>
    </span>
    <a href="/wiki/India" title="India">
     India
    </a>
   </td>
   <td>
    1,377,490,000
   </td>
   <td>
    3,287,240
   </td>
   <td>
    419
   </td>
  </tr>
```

```
 </tbody>
</table>
```

In [50]:

```
population_data = pd.DataFrame(columns=["Rank", "Country", "Popul
ation", "Area", "Density"])

for row in tables[table_index].tbody.find_all("tr"):
    col = row.find_all("td")
    if (col != []):
        rank = col[0].text
        country = col[1].text
        population = col[2].text.strip()
        area = col[3].text.strip()
        density = col[4].text.strip()
        population_data = population_data.append({"Rank":rank, "C
ountry":country, "Population":population, "Area":area, "Density":
density}, ignore_index=True)

population_data
```

Out[50]:

Pizza Place	Orders	Slices
Domino's Pizza	10	100
Little Caesars	12	144
Papa John's	15	165

	Rank	Country	Population	Area	Density
0	1	Singapore	5,704,000	710	8,033
1	2	Bangladesh	170,740,000	143,998	1,186

	Rank	Country	Population	Area	Density
2	3	Lebanon	6,856,000	10,452	656
3	4	Taiwan	23,604,000	36,193	652
4	5	South Korea	51,781,000	99,538	520
5	6	Rwanda	12,374,000	26,338	470
6	7	Haiti	11,578,000	27,065	428
7	8	Netherlands	17,600,000	41,526	424
8	9	Israel	9,350,000	22,072	424
9	10	India	1,377,490,000	3,287,240	419

Scrape data from HTML tables into a DataFrame using BeautifulSoup and read_html

Using the same `url`, `data`, `soup`, and `tables` object as in the last section we can use the `read_html` function to create a DataFrame.

Remember the table we need is located in `tables[table_index]`

We can now use the `pandas` function `read_html` and give it the string version of the table as well as the `flavor` which is the parsing engine `bs4`.

In [ ]:

```
pd.read_html(str(tables[5]), flavor='bs4')
```

The function `read_html` always returns a list of DataFrames so we must pick the one we want out of the list.

In [ ]:

```
population_data_read_html = pd.read_html(str(tables[5]), flavor='bs4')[0]

population_data_read_html
```

Scrape data from HTML tables into a DataFrame using read_html

We can also use the `read_html` function to directly get DataFrames from a `url`.

In [ ]:

```
dataframe_list = pd.read_html(url, flavor='bs4')
```

We can see there are 25 DataFrames just like when we used `find_all` on the `soup` object.

In [ ]:

```
len(dataframe_list)
```

Finally we can pick the DataFrame we need out of the list.

In [ ]:

```
dataframe_list[5]
```

We can also use the `match` parameter to select the specific table we want. If the table contains a string matching the text it will be read.

In [ ]:
```
pd.read_html(url, match="10 most densely populated countries", flavor='bs4')[0]
```

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-10-17	0.1	Joseph Santarcangelo	Created initial version of the lab