Web Scraping: Infinite Possibilities ★

Prepared by: Wanjao Colins

In [218]:
```
#import libraries

from bs4 import BeautifulSoup
import requests
import pandas as pd
```

In [219]:
```
#connecting to website

url = 'http://books.toscrape.com/'
html_page = requests.get(url)
```

In [220]:
```
# Verifying the successful retrieval of the webpage

html_page.status_code == requests.codes.ok
```

Out[220]: True

In [221]:
```
# Creating a BeautifulSoup object to parse the HTML content and make it more

soup = BeautifulSoup(html_page.content, 'html.parser')
```

In [222]:
```
# Extracting book titles by locating 'h3' HTML tags in the parsed webpage co

titles = soup.find_all('h3')
titles
```

Out[222]:
```
[<h3><a href="catalogue/a-light-in-the-attic_1000/index.html" title="A Ligh
t in the Attic">A Light in the ...</a></h3>,
 <h3><a href="catalogue/tipping-the-velvet_999/index.html" title="Tipping t
he Velvet">Tipping the Velvet</a></h3>,
 <h3><a href="catalogue/soumission_998/index.html" title="Soumission">Soumi
ssion</a></h3>,
 <h3><a href="catalogue/sharp-objects_997/index.html" title="Sharp Object
s">Sharp Objects</a></h3>,
 <h3><a href="catalogue/sapiens-a-brief-history-of-humankind_996/index.htm
l" title="Sapiens: A Brief History of Humankind">Sapiens: A Brief History
...</a></h3>,
 <h3><a href="catalogue/the-requiem-red_995/index.html" title="The Requiem
Red">The Requiem Red</a></h3>,
 <h3><a href="catalogue/the-dirty-little-secrets-of-getting-your-dream-job_
994/index.html" title="The Dirty Little Secrets of Getting Your Dream Job">
The Dirty Little Secrets ...</a></h3>,
 <h3><a href="catalogue/the-coming-woman-a-novel-based-on-the-life-of-the-i
nfamous-feminist-victoria-woodhull_993/index.html" title="The Coming Woman:
A Novel Based on the Life of the Infamous Feminist, Victoria Woodhull">The
Coming Woman: A ...</a></h3>,
 <h3><a href="catalogue/the-boys-in-the-boat-nine-americans-and-their-epic-
quest-for-gold-at-the-1936-berlin-olympics_992/index.html" title="The Boys
in the Boat: Nine Americans and Their Epic Quest for Gold at the 1936 Berli
n Olympics">The Boys in the ...</a></h3>,
 <h3><a href="catalogue/the-black-maria_991/index.html" title="The Black Ma
ria">The Black Maria</a></h3>,
 <h3><a href="catalogue/starving-hearts-triangular-trade-trilogy-1_990/inde
x.html" title="Starving Hearts (Triangular Trade Trilogy, #1)">Starving Hea
rts (Triangular Trade ...</a></h3>,
 <h3><a href="catalogue/shakespeares-sonnets_989/index.html" title="Shakesp
eare's Sonnets">Shakespeare's Sonnets</a></h3>,
 <h3><a href="catalogue/set-me-free_988/index.html" title="Set Me Free">Set
Me Free</a></h3>,
 <h3><a href="catalogue/scott-pilgrims-precious-little-life-scott-pilgrim-1
_987/index.html" title="Scott Pilgrim's Precious Little Life (Scott Pilgrim
#1)">Scott Pilgrim's Precious Little ...</a></h3>,
 <h3><a href="catalogue/rip-it-up-and-start-again_986/index.html" title="Ri
p it Up and Start Again">Rip it Up and ...</a></h3>,
 <h3><a href="catalogue/our-band-could-be-your-life-scenes-from-the-america
n-indie-underground-1981-1991_985/index.html" title="Our Band Could Be Your
Life: Scenes from the American Indie Underground, 1981-1991">Our Band Could
Be ...</a></h3>,
 <h3><a href="catalogue/olio_984/index.html" title="Olio">Olio</a></h3>,
 <h3><a href="catalogue/mesaerion-the-best-science-fiction-stories-1800-184
9_983/index.html" title="Mesaerion: The Best Science Fiction Stories 1800-1
849">Mesaerion: The Best Science ...</a></h3>,
 <h3><a href="catalogue/libertarianism-for-beginners_982/index.html" title
="Libertarianism for Beginners">Libertarianism for Beginners</a></h3>,
 <h3><a href="catalogue/its-only-the-himalayas_981/index.html" title="It's
Only the Himalayas">It's Only the Himalayas</a></h3>]
```

In [223]:
```
# Extracting the title attribute from the first 'a' tag within the first 'h3
# providing a more specific detail about the book title.

titles[0].find('a').attrs['title']
```

Out[223]:
```
'A Light in the Attic'
```

In [224]:
```
# Iterating over the list of 'h3' tags to extract book titles using the 'tit

books_list = [title.find('a').attrs['title'] for title in soup.find_all('h3'
books_list
```

Out[224]:
```
['A Light in the Attic',
 'Tipping the Velvet',
 'Soumission',
 'Sharp Objects',
 'Sapiens: A Brief History of Humankind',
 'The Requiem Red',
 'The Dirty Little Secrets of Getting Your Dream Job',
 'The Coming Woman: A Novel Based on the Life of the Infamous Feminist, Vic
toria Woodhull',
 'The Boys in the Boat: Nine Americans and Their Epic Quest for Gold at the
1936 Berlin Olympics',
 'The Black Maria',
 'Starving Hearts (Triangular Trade Trilogy, #1)',
 "Shakespeare's Sonnets",
 'Set Me Free',
 "Scott Pilgrim's Precious Little Life (Scott Pilgrim #1)",
 'Rip it Up and Start Again',
 'Our Band Could Be Your Life: Scenes from the American Indie Underground,
1981-1991',
 'Olio',
 'Mesaerion: The Best Science Fiction Stories 1800-1849',
 'Libertarianism for Beginners',
 "It's Only the Himalayas"]
```

In [225]:
```
# Extracting book ratings by searching for 'p' tags with class containing 's
import re

# Define a regular expression to match the 'star-rating' class
regex = re.compile('star-rating(.*)')

# Find all 'p' tags with the specified class and extract the rating values
ratings = [rating.attrs['class'][1] for rating in soup.find_all('p', {"class
ratings
```

Out[225]:
```
['Three',
 'One',
 'One',
 'Four',
 'Five',
 'One',
 'Four',
 'Three',
 'Four',
 'One',
 'Two',
 'Four',
 'Five',
 'Five',
 'Five',
 'Three',
 'One',
 'One',
 'Two',
 'Two']
```

In [226]:
```
# Mapping textual star ratings to numerical values using a predefined dictio

star_dict = {'One': 1, 'Two': 2, 'Three': 3, 'Four': 4, "Five": 5}
star_rating = [star_dict[s] for s in ratings]
star_rating
```

Out[226]:
```
[3, 1, 1, 4, 5, 1, 4, 3, 4, 1, 2, 4, 5, 5, 5, 3, 1, 1, 2, 2]
```

In [227]:
```
# Extracting book prices from 'p' tags with the class 'price_color' in the p

soup.find_all('p', class_="price_color")
```

Out[227]:
```
[<p class="price_color">£51.77</p>,
 <p class="price_color">£53.74</p>,
 <p class="price_color">£50.10</p>,
 <p class="price_color">£47.82</p>,
 <p class="price_color">£54.23</p>,
 <p class="price_color">£22.65</p>,
 <p class="price_color">£33.34</p>,
 <p class="price_color">£17.93</p>,
 <p class="price_color">£22.60</p>,
 <p class="price_color">£52.15</p>,
 <p class="price_color">£13.99</p>,
 <p class="price_color">£20.66</p>,
 <p class="price_color">£17.46</p>,
 <p class="price_color">£52.29</p>,
 <p class="price_color">£35.02</p>,
 <p class="price_color">£57.25</p>,
 <p class="price_color">£23.88</p>,
 <p class="price_color">£37.59</p>,
 <p class="price_color">£51.33</p>,
 <p class="price_color">£45.17</p>]
```

In [228]:
```
#using the '.text' attribute to extract textual content within the HTML elem

price_list = [price.text for price in soup.find_all('p', class_="price_color
price_list
```

Out[228]:
```
['£51.77',
 '£53.74',
 '£50.10',
 '£47.82',
 '£54.23',
 '£22.65',
 '£33.34',
 '£17.93',
 '£22.60',
 '£52.15',
 '£13.99',
 '£20.66',
 '£17.46',
 '£52.29',
 '£35.02',
 '£57.25',
 '£23.88',
 '£37.59',
 '£51.33',
 '£45.17']
```

In [229]:
```
# Extracting book availability text from 'p' tags with the class 'instock av
# and using strip() to remove leading and trailing whitespaces for cleaner d
availablility = [avail.text.strip() for avail in soup.find_all('p', class_="
availablility
```

Out[229]:
```
['In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock',
 'In stock']
```

Scraping Images

In [230]:
```
# Extracting all image tags ('img') from the parsed webpage content.
images = soup.find_all('img')

# Previewing an example image by selecting the first image in the list.
ex_image = images[0]
ex_image
```

Out[230]:
```
<img alt="A Light in the Attic" class="thumbnail" src="media/cache/2c/da/2c
dad67c44b002e7ead0cc35693c0e8b.jpg"/>
```

In [231]:
```
# Selecting the URL of the image from the 'src' attribute of the example ima
image_url = ex_image.attrs['src']
image_url
```

Out[231]:
```
'media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg'
```

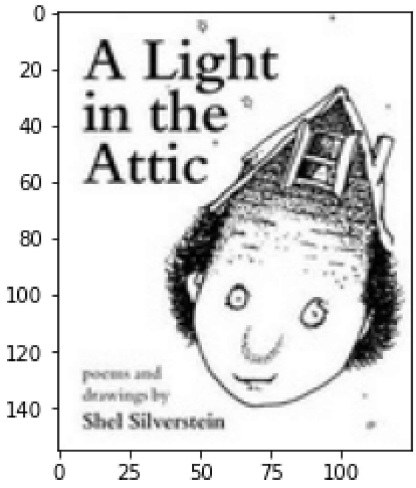
Saving Image Files Locally

In [232]:
```
import shutil

# Base URL for the book images on the website
url_base = "http://books.toscrape.com/"

# Extracting the image URL from the 'src' attribute of the example image
image_url = ex_image.attrs['src']

# Creating the full URL by combining the base URL and the image URL
full_url = url_base + image_url

# Sending a GET request to the image URL and streaming the content
r = requests.get(full_url, stream=True)

# Saving the image locally with a specified file path
with open('Images/book1cover.jpg', 'wb') as file:
    # Decode the content if needed
    r.raw.decode_content = True

    # Copying the raw content to the local file
    shutil.copyfileobj(r.raw, file)
```

Previewing an Individual Image File

In [233]:
```
# Importing necessary libraries for displaying images in Jupyter Notebook
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from IPython.display import Image, HTML

# Allowing inline rendering of matplotlib plots in Jupyter Notebook
%matplotlib inline
```

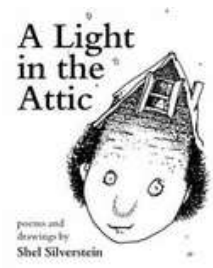
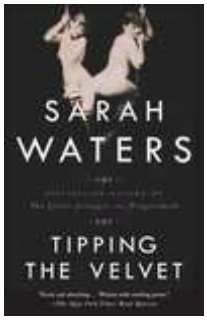
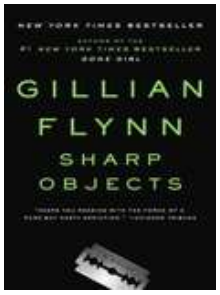
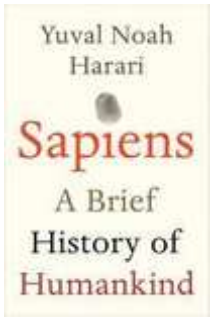
In [234]:
```
# Reading and displaying the image using matplotlib
img = mpimg.imread('Images/book1cover')

# Creating an image plot using imshow
imgplot = plt.imshow(img)

# Displaying the image plot
plt.show()
```

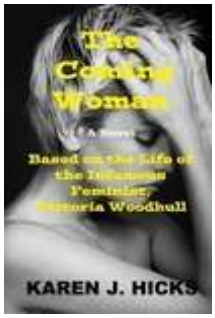
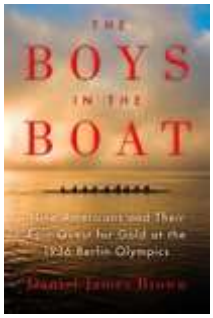


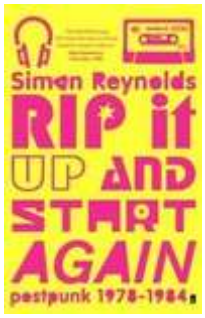
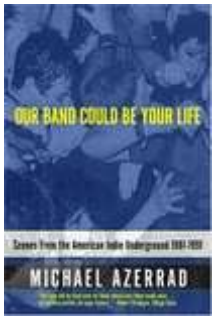
In [235]:
```python
# Looping over all the images in the webpage, downloading each image, and cr

import pandas as pd

# Initialize an empty list to store data for the DataFrame
data = []

# Iterate over the images on the webpage
for n, img in enumerate(images):
    # Define the base URL and extract the image URL
    url_base = "http://books.toscrape.com/"
    url_ext = img.attrs['src']

    # Create the full URL
    full_url = url_base + url_ext

    # Send a GET request to the image URL and stream the content
    r = requests.get(full_url, stream=True)

    # Define the file path to save the image
    path = 'images/book{}.jpg'.format(n + 1)

    # Extract the title and download the image if the status code is 200
    title = img.attrs['alt']
    if r.status_code == 200:
        with open(path, 'wb') as f:
            r.raw.decode_content = True
            shutil.copyfileobj(r.raw, f)
        # Create a row with title and image tag for the DataFrame
        row = [title, '<img src="{}">'.format(path)]
        data.append(row)

# Create a DataFrame from the collected data
df = pd.DataFrame(data)

# Print the number of rows in the DataFrame
print('Number of rows:', len(df))

# Rename the columns in the DataFrame
df.columns = ['title', 'cover']

# Display the DataFrame with images using HTML
HTML(df.to_html(escape=False))
```

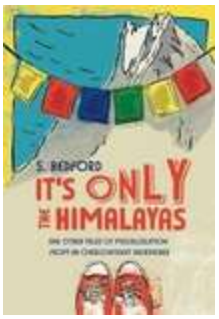
```
Number of rows: 20
```

Out[235]:

	title	cover
0	A Light in the Attic	
1	Tipping the Velvet	
2	Soumission	
3	Sharp Objects	
4	Sapiens: A Brief History of Humankind	
5	The Requiem Red	

	title	cover
6	The Dirty Little Secrets of Getting Your Dream Job	
7	The Coming Woman: A Novel Based on the Life of the Infamous Feminist, Victoria Woodhull	
8	The Boys in the Boat: Nine Americans and Their Epic Quest for Gold at the 1936 Berlin Olympics	
9	The Black Maria	
10	Starving Hearts (Triangular Trade Trilogy, #1)	
11	Shakespeare's Sonnets	

	title	cover
12	Set Me Free	
13	Scott Pilgrim's Precious Little Life (Scott Pilgrim #1)	
14	Rip it Up and Start Again	
15	Our Band Could Be Your Life: Scenes from the American Indie Underground, 1981-1991	
16	Olio	
17	Mesaerion: The Best Science Fiction Stories 1800-1849	

	title	cover
18	Libertarianism for Beginners	
19	It's Only the Himalayas	

In [236]:
```
# Appending additional columns to the existing DataFrame.

# Adding 'availability', 'rating', and 'price_list' columns to the DataFrame
df['availability'] = availablility
df['rating'] = ratings
df['price_list'] = price_list

# Displaying the DataFrame with the new columns
df
```

Out[236]:

	title	cover	availability	rating	price_list
0	A Light in the Attic	<img src="images/book1.jpg">	In stock	Three	£51.77
1	Tipping the Velvet	<img src="images/book2.jpg">	In stock	One	£53.74
2	Soumission	<img src="images/book3.jpg">	In stock	One	£50.10
3	Sharp Objects	<img src="images/book4.jpg">	In stock	Four	£47.82
4	Sapiens: A Brief History of Humankind	<img src="images/book5.jpg">	In stock	Five	£54.23
5	The Requiem Red	<img src="images/book6.jpg">	In stock	One	£22.65
6	The Dirty Little Secrets of Getting Your Dream...	<img src="images/book7.jpg">	In stock	Four	£33.34
7	The Coming Woman: A Novel Based on the Life of...	<img src="images/book8.jpg">	In stock	Three	£17.93
8	The Boys in the Boat: Nine Americans and Their...	<img src="images/book9.jpg">	In stock	Four	£22.60
9	The Black Maria	<img src="images/book10.jpg">	In stock	One	£52.15
10	Starving Hearts (Triangular Trade Trilogy, #1)	<img src="images/book11.jpg">	In stock	Two	£13.99
11	Shakespeare's Sonnets	<img src="images/book12.jpg">	In stock	Four	£20.66
12	Set Me Free	<img src="images/book13.jpg">	In stock	Five	£17.46
13	Scott Pilgrim's Precious Little Life (Scott Pi...	<img src="images/book14.jpg">	In stock	Five	£52.29
14	Rip it Up and Start Again	<img src="images/book15.jpg">	In stock	Five	£35.02
15	Our Band Could Be Your Life: Scenes from the A...	<img src="images/book16.jpg">	In stock	Three	£57.25
16	Olio	<img src="images/book17.jpg">	In stock	One	£23.88
17	Mesaerion: The Best Science Fiction Stories 18...	<img src="images/book18.jpg">	In stock	One	£37.59
18	Libertarianism for Beginners	<img src="images/book19.jpg">	In stock	Two	£51.33
19	It's Only the Himalayas	<img src="images/book20.jpg">	In stock	Two	£45.17

Scraping Data from a Wikipedia Page!

In [237]:
```
# Connecting to the Wikipedia page containing a list of cities and towns in

# Define the URL of the Wikipedia page
url = "https://en.wikipedia.org/wiki/List_of_cities_and_towns_in_Kenya_by_po

# Send a GET request to the URL and store the response in html_page variable
html_page = requests.get(url)

# Print the status code to check if the connection was successful
print("Status Code:", html_page.status_code)
```

```
Status Code: 200
```

In [238]:
```
# defining our soup object

soup = BeautifulSoup(html_page.text, "html.parser")
```

In [239]:
```
# Extracting the table with class 'wikitable sortable' from the parsed HTML

# Use BeautifulSoup to find the table with class 'wikitable sortable'
table = soup.find('table', class_='wikitable sortable')

# Print the table object
table
```

Out[239]:
```
<table class="wikitable sortable">
<tbody><tr>
<th style="color: black; background-color: #ABCDEF">№
</th>
<th style="color: black; background-color: #ABCDEF">City/ Town
</th>
<th style="color: black; background-color: #ABCDEF">Status
</th>
<th style="color: black; background-color: #ABCDEF">Population
</th>
<th style="color: black; background-color: #ABCDEF"><a href="/wiki/Counti
es_of_Kenya" title="Counties of Kenya">County</a>
</th>
<th style="color: yellow; background-color: #ABCDEF">Ref
</th></tr>
<tr bgcolor="#FFFFE0">
<td>1.</td>
<td align="left"><span typeof="mw:File"><a class="mw-file-description" hr
ef="/wiki/File:Coat_of_Arms_of_Nairobi.svg"><img class="mw-file-element"
```

In [240]:
```
# Extracting the headers (th elements) from the table.

# Use BeautifulSoup to find all 'th' elements within the table
table_headers = table.findAll('th')

# Print the list of table headers
table_headers
```

Out[240]:
```
[<th style="color: black; background-color: #ABCDEF">№
 </th>,
 <th style="color: black; background-color: #ABCDEF">City/ Town
 </th>,
 <th style="color: black; background-color: #ABCDEF">Status
 </th>,
 <th style="color: black; background-color: #ABCDEF">Population
 </th>,
 <th style="color: black; background-color: #ABCDEF"><a href="/wiki/Countie
s_of_Kenya" title="Counties of Kenya">County</a>
 </th>,
 <th style="color: yellow; background-color: #ABCDEF">Ref
 </th>]
```

In [241]:
```
# Cleaning up table headers by extracting the text content and removing lead

# Use list comprehension to extract the text content of each header and stri
cleaned_table_headers = [header.text.strip() for header in table_headers]

# Print the cleaned table headers
cleaned_table_headers
```

Out[241]:
```
['№', 'City/ Town', 'Status', 'Population', 'County', 'Ref']
```

In [242]:
```
# Creating an empty DataFrame with structured columns using the cleaned tabl

# Use pandas DataFrame to create an empty DataFrame with columns specified b
df = pd.DataFrame(columns=cleaned_table_headers)

# Print the empty DataFrame structure
df
```

Out[242]:

№	City/ Town	Status	Population	County	Ref

In [243]:
```
# Extracting table rows (excluding the header row) from the parsed HTML tabl

# Use BeautifulSoup to find all 'tr' elements within the table, excluding th
table_rows = table.find_all('tr')[1:]

# Printing first 5 table rows
table_rows[:5]
```

Out[243]:
```
[<tr bgcolor="#FFFFE0">
 <td>1.</td>
 <td align="left"><span typeof="mw:File"><a class="mw-file-description" hre
f="/wiki/File:Coat_of_Arms_of_Nairobi.svg"><img class="mw-file-element" dat
a-file-height="730" data-file-width="724" decoding="async" height="30" src
="//upload.wikimedia.org/wikipedia/commons/thumb/e/e7/Coat_of_Arms_of_Nairo
bi.svg/30px-Coat_of_Arms_of_Nairobi.svg.png" srcset="//upload.wikimedia.or
g/wikipedia/commons/thumb/e/e7/Coat_of_Arms_of_Nairobi.svg/45px-Coat_of_Arm
s_of_Nairobi.svg.png 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/
e/e7/Coat_of_Arms_of_Nairobi.svg/60px-Coat_of_Arms_of_Nairobi.svg.png 2x" w
idth="30"/></a></span> <span style="font-size:110%;"><a href="/wiki/Nairob
i" title="Nairobi">Nairobi</a></span></td>
 <td><span style="font-size:100%;">Capital</span></td>
 <td style="background-color:#E0E0E0;"><span data-sort-value="7006439707300
000000♠">4,397,073</span></td>
 <td><span typeof="mw:File"><a class="mw-file-description" href="/wiki/Fil
e:Coat_of_Arms_of_Nairobi.svg"><img class="mw-file-element" data-file-heigh
t="730" data-file-width="724" decoding="async" height="30" src="//upload.wi
kimedia.org/wikipedia/commons/thumb/e/e7/Coat_of_Arms_of_Nairobi.svg/30px-C
oat_of_Arms_of_Nairobi.svg.png" srcset="//upload.wikimedia.org/wikipedia/co
mmons/thumb/e/e7/Coat_of_Arms_of_Nairobi.svg/45px-Coat_of_Arms_of_Nairobi.s
vg.png 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/e/e7/Coat_of_Ar
ms_of_Nairobi.svg/60px-Coat_of_Arms_of_Nairobi.svg.png 2x" width="30"/></a>
</span> <span style="font-size:110%;"><a class="mw-redirect" href="/wiki/Na
irobi_County" title="Nairobi County">Nairobi</a></span></td>
 <td><sup class="reference" id="cite_ref-2"><a href="#cite_note-2">[2]</a>
</sup>
 </td></tr>,
 <tr bgcolor="#D0E7FF">
 <td>2.</td>
 <td align="left"><span typeof="mw:File"><a class="mw-file-description" hre
f="/wiki/File:Mombasa_County_Coat_of_Arms.png"><img class="mw-file-element"
data-file-height="520" data-file-width="450" decoding="async" height="35" s
rc="//upload.wikimedia.org/wikipedia/commons/thumb/1/1f/Mombasa_County_Coat
_of_Arms.png/30px-Mombasa_County_Coat_of_Arms.png" srcset="//upload.wikimed
ia.org/wikipedia/commons/thumb/1/1f/Mombasa_County_Coat_of_Arms.png/45px-Mo
mbasa_County_Coat_of_Arms.png 1.5x, //upload.wikimedia.org/wikipedia/common
s/thumb/1/1f/Mombasa_County_Coat_of_Arms.png/60px-Mombasa_County_Coat_of_Ar
ms.png 2x" width="30"/></a></span> <span style="font-size:110%;"><a href="/
wiki/Mombasa" title="Mombasa">Mombasa</a></span></td>
 <td><span style="font-size:100%;">City</span></td>
 <td style="background-color:#E0E0E0;"><span data-sort-value="7006120833300
000000♠">1,208,333</span></td>
 <td><span typeof="mw:File"><a class="mw-file-description" href="/wiki/Fil
e:Mombasa_County_Coat_of_Arms.png"><img class="mw-file-element" data-file-h
eight="520" data-file-width="450" decoding="async" height="35" src="//uploa
d.wikimedia.org/wikipedia/commons/thumb/1/1f/Mombasa_County_Coat_of_Arms.pn
g/30px-Mombasa_County_Coat_of_Arms.png" srcset="//upload.wikimedia.org/wiki
pedia/commons/thumb/1/1f/Mombasa_County_Coat_of_Arms.png/45px-Mombasa_Count
y_Coat_of_Arms.png 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/1/1
f/Mombasa_County_Coat_of_Arms.png/60px-Mombasa_County_Coat_of_Arms.png 2x"
width="30"/></a></span> <span style="font-size:110%;"><a href="/wiki/Mombas
a_County" title="Mombasa County">Mombasa</a></span></td>
 <td><sup class="reference" id="cite_ref-Kenya_Census_2019_3-0"><a href="#c
ite_note-Kenya_Census_2019-3">[3]</a></sup>
 </td></tr>,
 <tr bgcolor="#D0E7FF">
 <td>3.</td>
 <td align="left"><span style="font-size:110%;"><a href="/wiki/Nakuru" titl
e="Nakuru">Nakuru</a></span></td>
 <td><span style="font-size:100%;">City</span></td>
```

```
 <td style="background-color:#E0E0E0;"><span data-sort-value="7005570674000
000000♠">570,674</span></td>
 <td><span style="font-size:110%;"><a href="/wiki/Nakuru_County" title="Nak
uru County">Nakuru</a></span></td>
 <td><sup class="reference" id="cite_ref-Kenya_Census_2019_3-1"><a href="#c
ite_note-Kenya_Census_2019-3">[3]</a></sup>
 </td></tr>,
 <tr bgcolor="#E0CAF1">
 <td>4.</td>
 <td align="left"><span style="font-size:110%;"><a href="/wiki/Ruiru" title
="Ruiru">Ruiru</a></span></td>
 <td><span style="font-size:100%;">Municipality (Ruiru + Githurai)</span></
td>
 <td style="background-color:#E0E0E0;"><span data-sort-value="7005490120000
000000♠">490,120</span></td>
 <td><span style="font-size:110%;"><a href="/wiki/Kiambu_County" title="Kia
mbu County">Kiambu</a></span></td>
 <td><sup class="reference" id="cite_ref-Kenya_Census_2019_3-2"><a href="#c
ite_note-Kenya_Census_2019-3">[3]</a></sup>
 </td></tr>,
 <tr bgcolor="#E0CAF1">
 <td>5.</td>
 <td align="left"><span style="font-size:110%;"><a href="/wiki/Eldoret" tit
le="Eldoret">Eldoret</a></span></td>
 <td><span style="font-size:100%;">Municipality</span></td>
 <td style="background-color:#E0E0E0;"><span data-sort-value="7005475716000
000000♠">475,716</span></td>
 <td align="left"><span style="font-size:110%;"><a href="/wiki/Uasin_Gishu_
County" title="Uasin Gishu County">Uasin Gishu</a></span></td>
 <td><sup class="reference" id="cite_ref-Kenya_Census_2019_3-3"><a href="#c
ite_note-Kenya_Census_2019-3">[3]</a></sup>
 </td></tr>]
```

In [244]:
```
# Extracting data from each table row and cleaning up individual row data.

# Initialize an empty list to store individual row data
individual_row_data = []

# Iterate over each table row
for row in table_rows:
    # Find all 'td' elements within the row to extract the data
    row_data = row.findAll('td')

    # Use list comprehension to extract the text content of each data cell a
    cleaned_row_data = [data.text.strip() for data in row_data]

    # Append the cleaned row data to the list
    individual_row_data.append(cleaned_row_data)

# Print first 10 individual rows of  data
individual_row_data[:10]
```

Out[244]:
```
[['1.', 'Nairobi', 'Capital', '4,397,073', 'Nairobi', '[2]'],
 ['2.', 'Mombasa', 'City', '1,208,333', 'Mombasa', '[3]'],
 ['3.', 'Nakuru', 'City', '570,674', 'Nakuru', '[3]'],
 ['4.',
  'Ruiru',
  'Municipality (Ruiru + Githurai)',
  '490,120',
  'Kiambu',
  '[3]'],
 ['5.', 'Eldoret', 'Municipality', '475,716', 'Uasin Gishu', '[3]'],
 ['6.', 'Kisumu', 'City', '397,957', 'Kisumu', '[3]'],
 ['7.', 'Kikuyu', 'Municipality', '323,881', 'Kiambu', '[3]'],
 ['8.',
  'Ngong',
  'Municipality (Ngong + O/Rongai)',
  '279,757',
  'Kajiado',
  '[3]'],
 ['9.',
  'Mavoko',
  'Municipality (Mlolongo + Athi River + Syokimau + Katani + Githunguri)',
  '258,498',
  'Machakos',
  '[4]'],
 ['10.', 'Thika', 'Municipality', '251,407', "Kiambu/Murang'a", '[3]']]
```

In [245]:
```
for row in individual_row_data:
    length = len(df)
    df.loc[length] = row
df.head(10)
```

Out[245]:

	№	City/ Town	Status	Population	County	Ref
0	1.	Nairobi	Capital	4,397,073	Nairobi	[2]
1	2.	Mombasa	City	1,208,333	Mombasa	[3]
2	3.	Nakuru	City	570,674	Nakuru	[3]
3	4.	Ruiru	Municipality (Ruiru + Githurai)	490,120	Kiambu	[3]
4	5.	Eldoret	Municipality	475,716	Uasin Gishu	[3]
5	6.	Kisumu	City	397,957	Kisumu	[3]
6	7.	Kikuyu	Municipality	323,881	Kiambu	[3]
7	8.	Ngong	Municipality (Ngong + O/Rongai)	279,757	Kajiado	[3]
8	9.	Mavoko	Municipality (Mlolongo + Athi River + Syokimau...	258,498	Machakos	[4]
9	10.	Thika	Municipality	251,407	Kiambu/Murang'a	[3]

In [246]:
```
# Setting the index of the DataFrame using the first cleaned table header.

# Use pandas set_index to set the index of the DataFrame using the first cle
df = df.set_index(cleaned_table_headers[0])

# Print the final DataFrame with the index set
df.head(10)
```

Out[246]:

№	City/ Town	Status	Population	County	Ref
1.	Nairobi	Capital	4,397,073	Nairobi	[2]
2.	Mombasa	City	1,208,333	Mombasa	[3]
3.	Nakuru	City	570,674	Nakuru	[3]
4.	Ruiru	Municipality (Ruiru + Githurai)	490,120	Kiambu	[3]
5.	Eldoret	Municipality	475,716	Uasin Gishu	[3]
6.	Kisumu	City	397,957	Kisumu	[3]
7.	Kikuyu	Municipality	323,881	Kiambu	[3]
8.	Ngong	Municipality (Ngong + O/Rongai)	279,757	Kajiado	[3]
9.	Mavoko	Municipality (Mlolongo + Athi River + Syokimau...	258,498	Machakos	[4]
10.	Thika	Municipality	251,407	Kiambu/Murang'a	[3]

In [247]:
```
# Adding a new column for timestamp with the current date.

# Importing the datetime module
import datetime

# Use datetime.date.today() to obtain the current date and assign it to the
df["date"] = datetime.date.today()

# Print the DataFrame with the new timestamp column
df.head(10)
```

Out[247]:

№	City/ Town	Status	Population	County	Ref	date
1.	Nairobi	Capital	4,397,073	Nairobi	[2]	2023-11-26
2.	Mombasa	City	1,208,333	Mombasa	[3]	2023-11-26
3.	Nakuru	City	570,674	Nakuru	[3]	2023-11-26
4.	Ruiru	Municipality (Ruiru + Githurai)	490,120	Kiambu	[3]	2023-11-26
5.	Eldoret	Municipality	475,716	Uasin Gishu	[3]	2023-11-26
6.	Kisumu	City	397,957	Kisumu	[3]	2023-11-26
7.	Kikuyu	Municipality	323,881	Kiambu	[3]	2023-11-26
8.	Ngong	Municipality (Ngong + O/Rongai)	279,757	Kajiado	[3]	2023-11-26
9.	Mavoko	Municipality (Mlolongo + Athi River + Syokimau...	258,498	Machakos	[4]	2023-11-26
10.	Thika	Municipality	251,407	Kiambu/Murang'a	[3]	2023-11-26

In [248]:
```
# Exporting the DataFrame to a CSV file named 'pop.census.csv'.

# Use pandas to_csv method to export the DataFrame to a CSV file
df.to_csv('pop.census.csv')

# Print a message indicating the successful export
print("Data has been successfully exported to 'pop.census.csv'")
```

```
Data has been successfully exported to 'pop.census.csv'
```