□□ Web Scraping Project – IBM Data Analyst Capstone

This project demonstrates how to:

- Download webpages
- Scrape hyperlinks and images
- Extract tabular data

Case Study: Web Scraping Lab

Objectives

After completing this lab we will be able to:

- Download a webpage using requests module.
- Scrape all links from a webpage.
- Scrape all image URLs from a web page.
- Scrape data from html tables.

Scrape www.ibm.com

Import the required modules and functions

In [2]:

```
from bs4 import BeautifulSoup # this module helps in web scrapping.
import requests  # this module helps us to download a webpage
```

Download the contents of the webpage

In [5]:

```
url = "http://www.ibm.com"
```

In [7]:

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

Create a soup object using the class BeautifulSoup

In [10]:

```
soup = BeautifulSoup(data,"html.parser")  # create a soup object using the variable 'data'
```

Scrape all links

In [13]:

```
for link in soup.find_all('a'):  # in html anchor/link is represented by the tag <a>
    print(link.get('href'))
```

```
https://www.ibm.com/granite?lnk=hpad1us
https://developer.ibm.com/technologies/artificial-intelligence?lnk=hpad2us
https://www.ibm.com/products/watsonx-code-assistant?lnk=hpad3us
https://www.ibm.com/watsonx/developer/?lnk=hpad4us
https://www.ibm.com/thought-leadership/institute-business-value/report/ceo-generative-ai?lnk=hpab1us
https://www.ibm.com/think/reports/ai-in-action?lnk=hpab2us
https://www.ibm.com/artificial-intelligence/ai-ethics
https://www.ibm.com/account/reg/signup?formid=news-urx-52954&lnk=hpab4us
https://www.ibm.com/artificial-intelligence?lnk=hpfp1us
https://www.ibm.com/hybrid-cloud?lnk=hpfp2us
https://www.ibm.com/consulting?lnk=hpfp3us
https://www.ft.com/partnercontent/ibm/how-smaller-industry-tailored-ai-models-can-offer-greater-benefits.html
```

Scrape all images

In [16]:

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

Scrape data from html tables

In [19]:

```
#The below URL contains a html table with data about colors and color codes.
URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/HTMLColorCodes.h
```

Before proceeding to scrape a website, 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 [22]:

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

In [40]:

```
soup = BeautifulSoup(data,"html.parser")
print(soup.prettify()[:500])  # Preview first 500 characters of parsed HTML
```

```
<html>
 <body>
  <h1>
   Partial List  of HTML5 Supported Colors
  </h1>
  <table border="1" class="main-table">
   <tr>
    <td>
     Number
    </td>
    <td>
     Color
    </td>
    <td>
     Color Name
    </td>
    <td>
     Hex Code
     <br/>
     #RRGGBB
    </td>
    <td>
     Decimal Code
     <br/>
     (R,G,B)
    </td>
   </tr>
   <tr>
    <td>
     1
    </td>
    <td style="background:lightsalmon;">
    </td>
    <td>
     lightsalmon
    </td>
    <td>
     #FFA07A
    </td>
```

In [26]:

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

Get all rows from the table

In [29]:

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

```
Color Name--->Hex Code#RRGGBB
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
```

In [ ]:

In [52]:

```
import pandas as pd

# Define the extracted color data as a dictionary
color_data = {
    "Color Name": [
        "lightsalmon", "salmon", "darksalmon", "lightcoral", "coral", "tomato",
        "orangered", "gold", "orange", "darkorange", "lightyellow", "lemonchiffon",
        "papayawhip", "moccasin", "peachpuff", "palegoldenrod", "khaki", "darkkhaki",
        "yellow", "lawngreen", "chartreuse", "limegreen", "lime", "forestgreen",
        "green", "powderblue", "lightblue", "lightskyblue", "skyblue", "deepskyblue",
        "lightsteelblue", "dodgerblue"
    ],
    "Hex Code": [
        "#FFA07A", "#FA8072", "#E9967A", "#F08080", "#FF7F50", "#FF6347",
        "#FF4500", "#FFD700", "#FFA500", "#FF8C00", "#FFFFE0", "#FFFACD",
        "#FFEFD5", "#FFE4B5", "#FFDAB9", "#EEE8AA", "#F0E68C", "#BDB76B",
        "#FFFF00", "#7CFC00", "#7FFF00", "#32CD32", "#00FF00", "#228B22",
        "#008000", "#B0E0E6", "#ADD8E6", "#87CEFA", "#87CEEB", "#00BFFF",
        "#B0C4DE", "#1E90FF"
    ]
}

# Create a DataFrame
df_colors = pd.DataFrame(color_data)

# Save as CSV
df_colors.to_csv('data/scraped_colors.csv', index=False)

# Save as Excel
df_colors.to_excel('data/scraped_colors.xlsx', index=False)

print("Export completed: Data saved as CSV and Excel files inside 'data/' folder.")

# Display the table
df_colors
```

```
Export completed: Data saved as CSV and Excel files inside 'data/' folder.
```

Out[52]:

	Color Name	Hex Code
0	lightsalmon	#FFA07A
1	salmon	#FA8072
2	darksalmon	#E9967A
3	lightcoral	#F08080
4	coral	#FF7F50
5	tomato	#FF6347
6	orangered	#FF4500
7	gold	#FFD700
8	orange	#FFA500
9	darkorange	#FF8C00
10	lightyellow	#FFFFE0
11	lemonchiffon	#FFFACD
12	papayawhip	#FFEFD5
13	moccasin	#FFE4B5
14	peachpuff	#FFDAB9
15	palegoldenrod	#EEE8AA
16	khaki	#F0E68C
17	darkkhaki	#BDB76B
18	yellow	#FFFF00
19	lawngreen	#7CFC00
20	chartreuse	#7FFF00
21	limegreen	#32CD32
22	lime	#00FF00
23	forestgreen	#228B22
24	green	#008000
25	powderblue	#B0E0E6
26	lightblue	#ADD8E6
27	lightskyblue	#87CEFA
28	skyblue	#87CEEB
29	deepskyblue	#00BFFF
30	lightsteelblue	#B0C4DE
31	dodgerblue	#1E90FF

In [48]:

```
import matplotlib.pyplot as plt

# Data for bar chart
categories = ["Links Scraped", "Images Scraped", "Table Rows Scraped"]
counts = [12, 0, 32]  # 12 links, 0 images, 32 color rows

# Create colorful bar chart
plt.figure(figsize=(8, 5))
bars = plt.bar(categories, counts)

# Color bars
for i, bar in enumerate(bars):
    bar.set_color(plt.cm.Set1(i))

# Chart details
plt.title("Summary of Web Scraping Results")
plt.ylabel("Count")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

# Save the chart
bar_chart_path = "web_scraping_summary_chart.png"
plt.savefig(bar_chart_path)
plt.close()

bar_chart_path
```

Out[48]:

```
'web_scraping_summary_chart.png'
```

In [46]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Sample data for Color Name vs Hex Code
color_data = {
    "Color Name": [
        "lightsalmon", "salmon", "darksalmon", "lightcoral", "coral",
        "tomato", "orangered", "gold", "orange", "darkorange",
        "lightyellow", "lemonchiffon", "papayawhip", "moccasin", "peachpuff"
    ],
    "Hex Code": [
        "#FFA07A", "#FA8072", "#E9967A", "#F08080", "#FF7F50",
        "#FF6347", "#FF4500", "#FFD700", "#FFA500", "#FF8C00",
        "#FFFFE0", "#FFFACD", "#FFEFD5", "#FFE4B5", "#FFDAB9"
    ]
}

# Create DataFrame
df_colors = pd.DataFrame(color_data)

# Plot the table
fig, ax = plt.subplots(figsize=(10, 6))
ax.axis('off')
table = ax.table(cellText=df_colors.values,
                 colLabels=df_colors.columns,
                 loc='center',
                 cellLoc='center')

table.auto_set_font_size(False)
table.set_fontsize(10)
table.auto_set_column_width([0, 1])

# Save the table visual
color_table_path = "scraped_color_table_snapshot.png"
plt.savefig(color_table_path, bbox_inches='tight')
plt.close()

color_table_path
```

Out[46]:

```
'scraped_color_table_snapshot.png'
```

In [34]:

```
import os

# Get the absolute file path of the notebook file
file_path = os.path.abspath("Web-Scraping-Review-Lab.ipynb")
print("The notebook is located at:", file_path)
```

```
The notebook is located at: C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_5_Web-Scraping\Web-Scraping-Review-Lab.ipynb
```

In [38]:

```
import nbconvert
import nbformat
import pdfkit

# Corrected file paths (Using raw string notation or forward slashes)
input_file_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_5_Web-Scraping\Web
output_pdf_path = r"C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_5_Web-Scraping\W

# Load the Jupyter Notebook file
with open(input_file_path, 'r', encoding='utf-8') as f:
    notebook_content = nbformat.read(f, as_version=4)

# Convert the notebook to HTML
html_exporter = nbconvert.HTMLExporter()
html_exporter.exclude_input = False  # Include code cells in the output
(body, resources) = html_exporter.from_notebook_node(notebook_content)

# Convert HTML to PDF
pdfkit.from_string(body, output_pdf_path)

# Return the PDF file path
print(f"Notebook successfully converted to PDF: {output_pdf_path}")
```

```
Notebook successfully converted to PDF: C:\Users\Ede\Desktop\IBM_Capstone_Data_Analyst_2025\Module_1_Real_World_Projects\Project_5_Web-Scraping\W
eb-Scraping-Review-Lab.pdf
```

In [37]:

```
!jupyter nbconvert --to html "Web-Scraping-Review-Lab.ipynb"
```

```
[NbConvertApp] Converting notebook Web-Scraping-Review-Lab.ipynb to html
[NbConvertApp] Writing 301814 bytes to Web-Scraping-Review-Lab.html
```

Congratulations to us for having successfully completed the above lab!

Authors:

Kelechukwu Innocent Ede and Ramesh Sannareddy

Other Contributors:

- Rav Ahuja

In [ ]: