

Data and Artificial Intelligence

Cyber Shujaa Program

Week 1 Assignment Web Scraping and Data Handling in Python

NAME: PAULINE KUNGU

PROJECT: WEB SCRAPING PROJECT

DATE: May 15, 2025

INTRODUCTION

Data is important for machine learning algorithms to learn patterns, make predictions and improve their performance. Sufficient and high quality data is needed. Web scraping is the automated process of extracting data from websites.

What am going to cover:

- Practical Python coding on Jupiter Notebooks hosted on Google Colab
- Use requests and BeautifulSoup to extract data from a web page.
- Parse and clean the extracted data.
- Store structured data into a Pandas DataFrame.
- Export the final dataset to a .csv file.

Step 1: Importing the libraries to use for web scraping

The first step is to import the necessary libraries am going to use. BeautifulSoup provides HTML parsing capabilities, the request library handles HTTP request and pandas is used for data manipulation.

The screenshot shows a Jupyter Notebook interface in a web browser. The notebook is titled 'webscrapingproject.ipynb'. The code is as follows:

```
[ ] #importing libraries for scraping
from bs4 import BeautifulSoup
import requests
import pandas as pd

[ ] #here i have specified the url am going to scrape and used the request library to send a get request to the url
url = "https://www.scrapethissite.com/pages/forms"
page = requests.get(url)

[ ] #check if the request was successful
page

<Response [200]>

using beautifulsoup a python library to parse the html page we got using the request library

[ ] soup = BeautifulSoup(page.text, 'html')
print(soup)

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<title>Hockey Teams: Forms, Searching and Pagination | Scrape This Site | A public sandbox for learning web scraping</title>
<link href="/static/images/scraper-icon.png" rel="icon" type="image/png"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
```

Step 2: Fetching the page

I specified the target URL and sent a GET request to retrieve the HTML content of the web page. The 200 is a response status code that confirms that the request was successful.

The screenshot shows a Jupyter Notebook interface in a web browser. The notebook is titled 'webscrapingproject.ipynb'. The code is as follows:

```
[ ] #importing libraries for scraping
from bs4 import BeautifulSoup
import requests
import pandas as pd

[ ] #here i have specified the url am going to scrape and used the request library to send a get request to the url
url = "https://www.scrapethissite.com/pages/forms"
page = requests.get(url)

[ ] #check if the request was successful
page

<Response [200]>

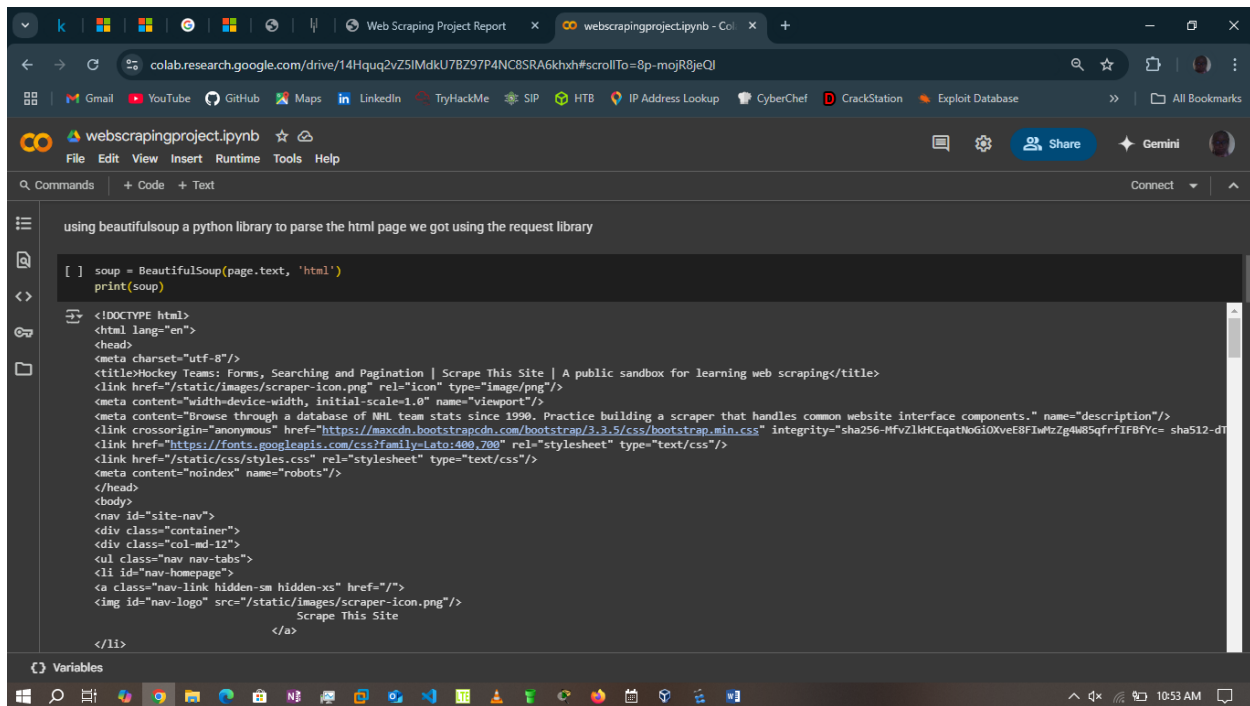
using beautifulsoup a python library to parse the html page we got using the request library

[ ] soup = BeautifulSoup(page.text, 'html')
print(soup)

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<title>Hockey Teams: Forms, Searching and Pagination | Scrape This Site | A public sandbox for learning web scraping</title>
<link href="/static/images/scraper-icon.png" rel="icon" type="image/png"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
```

Step 3: parsing the HTML content

After retrieving the web page, I used BeautifulSoup to parse the HTML content. Then using the find() function to locate the hockey table.



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes a file explorer on the left, a command bar with '+ Code' and '+ Text' buttons, and a 'Connect' button on the right. The main area displays a code cell with the following Python code:

```
using beautifulsoup a python library to parse the html page we got using the request library

[ ] soup = BeautifulSoup(page.text, 'html')
    print(soup)
```

Below the code cell, the rendered HTML output is visible, showing the document structure from the scraped page, including DOCTYPE, head, meta, title, and body elements.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<title>Hockey Teams: Forms, Searching and Pagination | Scrape This Site | A public sandbox for learning web scraping</title>
<link href="/static/images/scraper-icon.png" rel="icon" type="image/png"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
<meta content="Browse through a database of NHL team stats since 1990. Practice building a scraper that handles common website interface components." name="description"/>
<link crossorigin="anonymous" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" integrity="sha256-MfvzlkCEgatMoGi0XveE8FIwZg4W8SqfrFI8fYc= sha512-dt" />
<link href="https://fonts.googleapis.com/css?family=Lato:400,700" rel="stylesheet" type="text/css"/>
<link href="/static/css/styles.css" rel="stylesheet" type="text/css"/>
<meta content="noindex" name="robots"/>
</head>
<body>
<nav id="site-nav">
<div class="container">
<div class="col-md-12">
<ul class="nav nav-tabs">
<li id="nav-homepage">
<a class="nav-link hidden-sm hidden-xs" href="/">

Scrape This Site
</a>
</li>
```

Step 4: Extracting the header column

I extracted the column header from the table to create the structure for the pandas dataframe.

The screenshot shows a Google Colab notebook titled 'webscrapingproject.ipynb'. The first code cell contains the following Python code:

```
[ ] #use the find function to locate the hockey table
hockey_table = soup.find('table', class_='table')
print(hockey_table)
```

Below the code cell, the output is displayed as a table with the following columns: Team Name, Year, Wins, Losses, OT Losses, Win %, Goals For (GF), Goals Against (GA), and a plus/minus sign (+/-). The table is currently empty.

The second code cell contains the following Python code:

```
[ ] #the find_all() does the same as the find() but returns all that contain not just the first data.
headers = hockey_table.find_all('th')
# using the strip() function to remove the tags
column_names = [header.text.strip() for header in headers]
print("Column Titles:", column_names)
```

The output of this cell is: Column Titles: ['Team Name', 'Year', 'Wins', 'Losses', 'OT Losses', 'Win %', 'Goals For (GF)', 'Goals Against (GA)', '+ / -']

The third code cell contains the following Python code:

```
[ ] #create a dataframe using pandas library
df = pd.DataFrame(columns=column_names)
df
```

The output of this cell is an empty DataFrame with the same column names as the table above.

Step 5: Creating a dataframe and populating it

I then created the dataframe and extracted the column name in it. Then I also populated it data from the hockey table rows.

The screenshot shows the same Google Colab notebook. The fourth code cell contains the following Python code:

```
[ ] #am getting all the rows which are in the tag "tr"
rows = hockey_table.find_all('tr')[1:]
rows
```

The output of this cell is a list of HTML table rows (tr tags).

The fifth code cell contains the following Python code:

```
[ ] # Loop through each row and extract the text from each cell
for row in rows:
    cells = row.find_all('td')
    #remove the tags
    data = [cell.text.strip() for cell in cells]
    #add the text to the df using .loc which enable assessing and modification of data in dataframe
    #len()function show the length/number of objects ie rows of the dataframe
    df.loc[len(df)] = data
```

The output of this cell is the updated DataFrame with the first 5 rows of data populated.

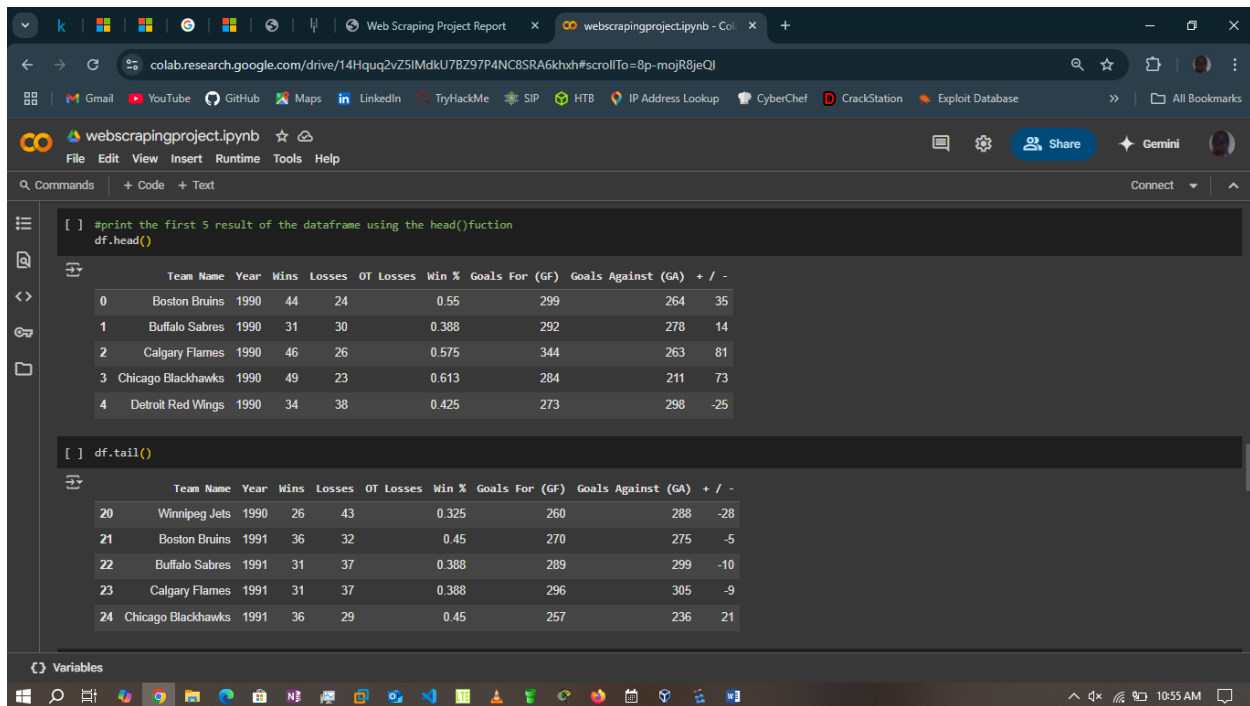
The sixth code cell contains the following Python code:

```
[ ] #print the first 5 result of the dataframe using the head()function
df.head()
```

The output of this cell is the first 5 rows of the DataFrame.

Step 6: Exploring the data

Here I was doing basic exploration of the data in the dataframe to understand the data in it. I used some common functions like head(), tail(), info(), unique() and checked for missing values.



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell uses `df.head()` to display the first five rows of a dataframe. The second cell uses `df.tail()` to display the last five rows. The dataframe contains hockey team statistics for the years 1990 and 1991.

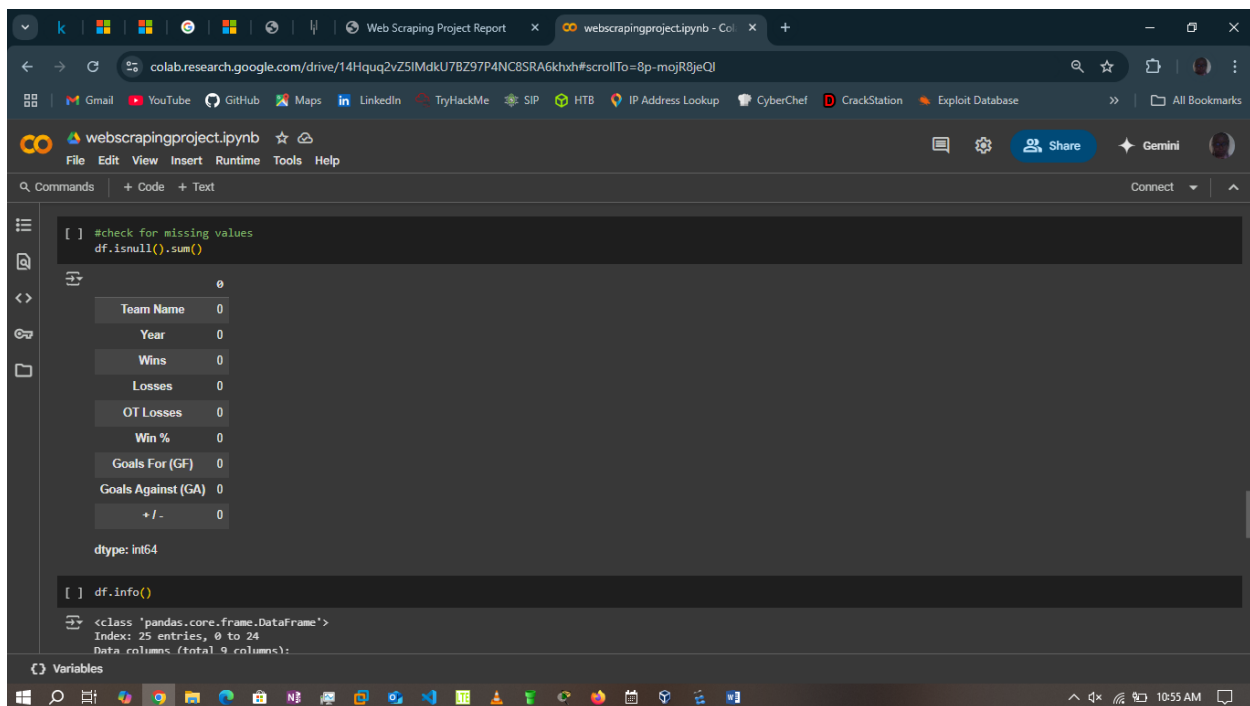
```
[ ] #print the first 5 result of the dataframe using the head()function
df.head()
```

	Team Name	Year	Wins	Losses	OT Losses	Win %	Goals For (GF)	Goals Against (GA)	+ / -
0	Boston Bruins	1990	44	24		0.55	299	264	35
1	Buffalo Sabres	1990	31	30		0.388	292	278	14
2	Calgary Flames	1990	46	26		0.575	344	263	81
3	Chicago Blackhawks	1990	49	23		0.613	284	211	73
4	Detroit Red Wings	1990	34	38		0.425	273	298	-25

```
[ ] df.tail()
```

	Team Name	Year	Wins	Losses	OT Losses	Win %	Goals For (GF)	Goals Against (GA)	+ / -
20	Winnipeg Jets	1990	26	43		0.325	260	288	-28
21	Boston Bruins	1991	36	32		0.45	270	275	-5
22	Buffalo Sabres	1991	31	37		0.388	289	299	-10
23	Calgary Flames	1991	31	37		0.388	296	305	-9
24	Chicago Blackhawks	1991	36	29		0.45	257	236	21

Variables



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell checks for missing values using `df.isnull().sum()`. The second cell displays the dataframe's information using `df.info()`.

```
[ ] #check for missing values
df.isnull().sum()
```

Team Name	0
Year	0
Wins	0
Losses	0
OT Losses	0
Win %	0
Goals For (GF)	0
Goals Against (GA)	0
+ / -	0

dtype: int64

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 25 entries, 0 to 24
Data columns (total 9 columns):
```

Variables

The screenshot shows a Google Colab notebook with the following content:

```
[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 25 entries, 0 to 24
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   Team Name       25 non-null    object
 1   Year            25 non-null    object
 2   Wins            25 non-null    object
 3   Losses          25 non-null    object
 4   OT Losses       25 non-null    object
 5   Win %           25 non-null    object
 6   Goals For (GF)  25 non-null    object
 7   Goals Against (GA) 25 non-null    object
 8   + / -           25 non-null    object
dtypes: object(9)
memory usage: 2.5+ KB

the table has data from 2 yrs 1990 and 1991

[ ] # Show all unique years
print(df['Year'].unique())

['1990' '1991']

there are 21 teams
```

The interface includes a top bar with navigation icons, a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), and a sidebar with icons for file management and a 'Variables' panel at the bottom.

Step 7: Exporting the csv

Finally I exported the dataframe to the CSV file then downloaded it from google colab. It opened as excel file as shown below:

The screenshot shows the same Google Colab notebook with the following additional content:

```
[ ] # Show all unique years
print(df['Year'].unique())

['1990' '1991']

there are 21 teams

[ ] # Show all unique teams
print(df['Team Name'].unique())

['Boston Bruins' 'Buffalo Sabres' 'Calgary Flames' 'Chicago Blackhawks'
 'Detroit Red Wings' 'Edmonton Oilers' 'Hartford Whalers'
 'Los Angeles Kings' 'Minnesota North Stars' 'Montreal Canadiens'
 'New Jersey Devils' 'New York Islanders' 'New York Rangers'
 'Philadelphia Flyers' 'Pittsburgh Penguins' 'Quebec Nordiques'
 'St. Louis Blues' 'Toronto Maple Leafs' 'Vancouver Canucks'
 'Washington Capitals' 'Winnipeg Jets']

lets save the dataframe into csv file

[ ] df.to_csv('Hockey_Stats.csv')
print("Data exported successfully to 'Hockey_Stats.csv'")

Data exported successfully to 'Hockey_Stats.csv'
```

The 'Variables' panel at the bottom is now empty.

Hockey Stats - Excel (Product Activation Failed)

File Home Insert Page Layout Formulas Data Review View Tell me what you want to do... Share

A1 Team Name

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Team Name	Year	Wins	Losses	OT Losses	Win %	Goals For	Goals Against	+/-												
2	Boston Bruins	1990	44	24		0.55	299	264	35												
3	Buffalo Sabres	1990	31	30		0.388	292	278	14												
4	Calgary Flames	1990	46	26		0.575	344	263	81												
5	Chicago Blackhawks	1990	49	23		0.613	284	211	73												
6	Detroit Red Wings	1990	34	38		0.425	273	298	-25												
7	Edmonton Oilers	1990	37	37		0.463	272	272	0												
8	Hartford Whalers	1990	31	38		0.388	238	276	-38												
9	Los Angeles Kings	1990	46	24		0.575	340	254	86												
10	Minnesota North Stars	1990	27	39		0.338	256	266	-10												
11	Montreal Canadiens	1990	39	30		0.487	273	249	24												
12	New Jersey Devils	1990	32	33		0.4	272	264	8												
13	New York Islanders	1990	25	45		0.312	223	290	-67												
14	New York Rangers	1990	36	31		0.45	297	265	32												
15	Philadelphia Flyers	1990	33	37		0.412	252	267	-15												
16	Pittsburgh Penguins	1990	41	33		0.512	342	305	37												
17	Quebec Nordiques	1990	16	50		0.2	236	354	-118												
18	St. Louis Blues	1990	47	22		0.588	310	250	60												
19	Toronto Maple Leafs	1990	23	46		0.287	241	318	-77												
20	Vancouver Canucks	1990	28	43		0.35	243	315	-72												
21	Washington Capitals	1990	37	36		0.463	258	258	0												
22	Winnipeg Jets	1990	26	43		0.325	260	288	-28												
23	Boston Bruins	1991	36	32		0.45	270	275	-5												
24	Buffalo Sabres	1991	31	37		0.388	289	299	-10												
25	Calgary Flames	1991	31	37		0.388	296	305	-9												
26	Chicago Blackhawks	1991	36	29		0.45	257	236	21												
27																					
28																					

Hockey Stats

Ready

CONCLUSION

This web scraping project successfully demonstrated the process of extracting tabular data from a website and transforming it into a structured format for analysis. I was able to use python libraries like beautifulsoup, request and pandas to explore and extract data from the website HTML document. The data gathered can be used to train ML model that can be deployed for use.

NOTEBOOK

Link:

<https://colab.research.google.com/drive/14Hquq2vZ5IMdkU7BZ97P4NC8SRA6khxh?usp=sharing>