Basic Web Scraping from HTML

- Web Scraping is an automatic method used to gather data from websites.
- It can be used as a method for one of the first steps of **ML Lifecycle**: `Data Collection`
- Most HTML data from websites are in unstructured format which is further processed and converted into structured format using web scraping.
- There are multiple methods to scrape data, such as using API's or even using custom written code.
- Python has several open-source libraries for web scraping such as BeautifulSoup, Scrapy, Selenium and so on. In this demo, we are going to use the `BeautifulSoup` library.

Beautiful Soup creates a parse tree which can be further used to extract data from a website's HTML.

In [1]:
```
import bs4
from bs4 import BeautifulSoup
import csv
import requests
import time
import pandas as pd
import urllib
import re
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
with open('home.html','r') as html_file:
    content=html_file.read()
    soup = BeautifulSoup(content,'html.parser')

print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8"/>
  <meta content="width=device-width, initial-scale=1" name="viewport"/>
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstra
p.min.css" rel="stylesheet"/>
  <title>
   My Courses
  </title>
 </head>
 <body>
  <h1>
   Hello, Start Learning
  </h1>
  <div class="card" id="card-python-for-beginners">
   <div class="class-header">
    Python
   </div>
   <div class="card-body">
    <h5 class="card-title">
     Python for beginners
    </h5>
    <p class="card-text">
     If you are new to Python, this is the course you should buy!
    </p>
    <a class="btn btn-primary" href="#">
     Start for 20$
    </a>
   </div>
  </div>
  <div class="card" id="card-python-web-development">
   <div class="class-header">
    Python
   </div>
   <div class="card-body">
    <h5 class="card-title">
     Python Web Development
    </h5>
    <p class="card-text">
     If you feel confident enough with Python, then you can enroll in this co
urse to learn how to create your own website!
    </p>
    <a class="btn btn-primary" href="#">
     Start for 50$
    </a>
   </div>
  </div>
  <div class="card" id="card-python-machine-learning">
   <div class="class-header">
    Python
   </div>
   <div class="card-body">
    <h5 class="card-title">
     Python Machine Learning
    </h5>
    <p class="card-text">
```

```
     Become a Python Machine Learning Master!
    </p>
    <a class="btn btn-primary" href="#">
     Start for 100$
    </a>
   </div>
  </div>
 </body>
</html>
```

Extracting Content from basic tags

In [3]:
```
tags = soup.find_all('h5')
print(tags)

for course in tags:
    print(course.text)
```

```
[<h5 class="card-title">Python for beginners</h5>, <h5 class="card-title">Pyt
hon Web Development</h5>, <h5 class="card-title">Python Machine Learning</h5
>]
Python for beginners
Python Web Development
Python Machine Learning
```

Extracting specific content from div tags and class attribute

In [4]:
```
tags = soup.find_all('div',class_='card')
for course in tags:
    course_name = course.h5.text
    course_price = course.a.text.split()[-1]
    #print(course_price)
    print("{} costs {}".format(course_name,course_price))
```

```
Python for beginners costs 20$
Python Web Development costs 50$
Python Machine Learning costs 100$
```

We can use the above pieces of code to extract relevant course information from sites like Udemy which constantly updates its course library.

Web Scraping using Requests library

- `Requests` library is used to request information from specific URLs and store it for further processing.
- It sends an *HTTP* request to a website and stores the response object within a variable.
- The `get()` method is used to perform this operation.
- `req.content` extracts the HTML code of the website and the `parser` converts the code to a Python object.

- The `find_all()` method is used to search and retrieve the specified tag and the data contained inside as HTML code.
- We can use the `read_html()` method from Pandas to extract the data from the HTML code and convert it into a Data Frame.

In [5]:
```
# IPL Dataset scrape
import pandas as pd
import requests
from bs4 import BeautifulSoup
req = requests.get("http://selfish-branch.surge.sh/",verify=False)
soup = BeautifulSoup(req.content, 'lxml')
print(soup.prettify())
```

```
<html>
 <head>
  <title>
   IPL Table
  </title>
  <link href="bootstrap.min.css" rel="stylesheet" type="text/css"/>
 </head>
 <body>
  <div class="container">
   <div>
    <h2>
     Scrape the given rows into a Dataframe
    </h2>
    <br/>
    <h4>
     Note: if there is no data it means it's a NULL value.
    </h4>
   </div>
  </div>
```

In [6]:
```
table = soup.find_all('table')[0]
df_ipl = pd.read_html(str(table),index_col=None)[0]
```

In [7]:
```
table
```

Out[7]:
```
<table class="table table-bordered table-striped table-hover table-responsi
ve">
<thead class="thead-dark">
<tr>
<th>id</th>
<th>season</th>
<th>city</th>
<th>date</th>
<th>team1</th>
<th>team2</th>
<th>toss_winner</th>
<th>toss_decision</th>
<th>result</th>
<th>dl_applied</th>
<th>winner</th>
<th>win_by_runs</th>
<th>win_by_wickets</th>
<th>player_of_match</th>
<th>venue</th>
```

In [8]:
```
df_ipl.head()
```

Out[8]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_
0	1.0	2008.0	Bangalore	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2.0	2008.0	Chandigarh	2008-04-19	Chennai Super Kings	Kings XI Punjab	Chennai Super Kings	bat	normal	
2	3.0	2008.0	Delhi	2008-04-19	Rajasthan Royals	Delhi Daredevils	Rajasthan Royals	bat	normal	
3	4.0	2008.0	Mumbai	2008-04-20	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	normal	
4	5.0	2008.0	Kolkata	2008-04-20	Deccan Chargers	Kolkata Knight Riders	Deccan Chargers	bat	normal	

In [ ]: