CSC461
INTRODUCTION TO DATA SCIENCE

Dr. Muhammad Sharjeel

https://muhmmadsharjeel.github.io/

02

DATA COLLECTION AND WEB SCRAPING

DATA GATHERING

- First step in data science pipeline is to get some data
- Some of the ways to get data are:
  - Download file(s) or data manually
  - Collect data using forms
  - Query from a database
  - Query an API (usually web-based)
  - Scrap from a webpage

HTTP QUERIES

- To get data from web applications, we use HTTP requests (queries)

```
import requests
response = requests.get("https://lahore.comsats.edu.pk/default.aspx")
response.status_code
response.content
response.text
response.headers
response.headers['Content-Type']
```

- Processing long URLs
  - Example: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8

```
parm = {"sa" : "t", "rct" : "j", "q" : "", .... , "uact" : "8"}
response = requests.get("http://www.google.com/url" , params = parm)
```

- HTTP GET is the most common method, but there are others, i.e., PUT, POST, and DELETE

```
response = requests.put()
response = requests.post()
response = requests.delete()
```

RESTful APIs

- REST (REpresentational State Transfer) APIs are used to get data from web services
- REST is more a design architecture, but a few key points:
  - Uses standard HTTP interface and methods (GET, PUT, POST, DELETE)
  - Stateless – the server doesn't remember what you were doing
  - Query to a REST API is almost like standard HTTP request, but always require to include parameters
  - With RESTful APIs you must use authentication, mostly using OAuth

DATA FORMATS

- The three most common formats are:
  - CSV (Comma Separated Values)
  - JSON (JavaScript Object Notation)
  - HTML/XML (Hyper Text and Extensible Markup Language)

COMMA SEPARATED VALUES

- Refers to any ‘delimited’ text file (delimiter, in this case, is the comma)
  - Example: *Semester, Course, Section, F16, 15688, 1st Jun 2016, 30 Aug 2016 4:34*
- If values themselves contain commas, you can enclose them in quotes

```
import pandas as pd
dataframe = pd.read_csv(”myFile.csv", delimiter=',', quotechar='"')
```

JAVASCRIPT OBJECT NOTATION

- JSON originated as a way of encapsulating JavaScript objects
- It supports several different data types, e.g., number, string, boolean, list (array), dictionary (key value pair)
  - Example: { "name":"John", "age":30, "car":null }

```
import json
print(json.loads(response.content))
```

```
data = {"id": "02", "name": "Naveed", "department": "HR"}
j_obj = json.dumps(data, indent = 4)
print(j_obj)
```

```
data = {"name" : "Hira", "rollno" : 56, "cgpa" : 3.6}
with open("sample.json", "w") as outfile:
    json.dump(data, outfile)
```

HYPER TEXT AND EXTENSIBLE MARKUP LANGUAGE

- HTML is syntactically like XML but horrible (e.g., open tags are not always closed)
  - More fundamentally, HTML is meant to describe appearance
- XML files contain hierarchical content delineated by tags

```
<tag attribute="value">
        <subtag>
                Some text here
        </subtag>
</tag>
```

WEB SCRAPING

- Send an HTTP request to the URL of a webpage
  - The server responds to the request by returning the HTML content of the webpage
- The HTML content is then parsed to get the data
  - Since most of the HTML data is nested, it cannot be extracted simply through string processing
  - A parser is required which can create a nested/tree structure of the HTML data
  - Several HTML parser libraries available in Python but the most advanced one is **html5lib**
- The parser would create a parse tree which can be used to navigate and search for the data
- For this task, another Python library called **BeautifulSoup** is used
  - It supports pulling data out of HTML and XML files

```
from bs4 import BeautifulSoup
import requests
response = requests.get("https://www.bbc.com/urdu")
soup = BeautifulSoup(response.content)
soup.find("section", id="schedule")
soup.find("table")
soup.findAll("a")
soup = BeautifulSoup(response.content, 'html5lib')
print(soup.prettify())
```

WEB SCRAPING

- Example of a simple web scraper

```
import requests
from bs4 import BeautifulSoup
response = requests.get("http://www.values.com/inspirational-quotes")
soup = BeautifulSoup(response.content, 'html5lib')
quotes = []
table = soup.find('div', attrs = {'id':'all_quotes'})
for row in table.findAll('div', attrs = {'class':'col-6 col-lg-3 text-center margin-30px-bottom sm-margin-30px-top'}):
    quote = {}
    quote['theme'] = row.h5.text
    quote['url'] = row.a['href']
    quote['img'] = row.img['src']
    quote['lines'] = row.img['alt'].split(" #")[0]
    quote['author'] = row.img['alt'].split(" #")[1]
    quotes.append(quote)
```

WEB SCRAPING

- Further reading
  - https://oxylabs.io/blog/python-web-scraping
  - https://www.youtube.com/watch?v=mDveiNIpqyw&t=3s
  - https://www.dataquest.io/blog/web-scraping-python-using-beautiful-soup/
  - https://www.analyticsvidhya.com/blog/2017/07/web-scraping-in-python-using-scrapy/

THANKS