SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Aug 2024

Data Scraping Seminar 6

ICT233 Data Programming

Veronica Hu huhe001@suss.edu.sg

RECAP

S5 Key Learning Objectives

1) Determine data types in Pandas and convert them between various forms
2) Handle and manipulate string and text data in Pandas
3) Understand how to apply builtin or selfdefined functions on Pandas vector
4) Appreciate the use of techniques to group and combine Pandas data for calculation and analytical purposes

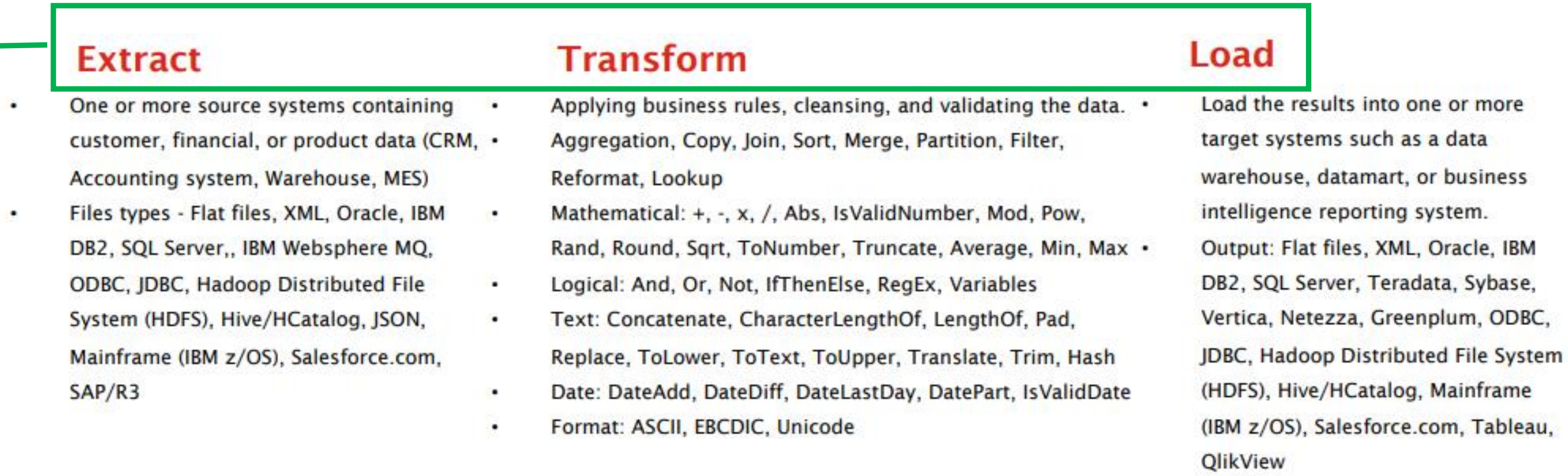
SEMINAR OVERVIEW

Data Scraping – LEARNING OBJECTIVES

1) Design methods to extract and parse information from the internet
2) Retrieve and consume data from Web APIs
3) Understand and appreciate simple and complex methods of retrieval and apply the right methods given the use cases
4) Construct a Python program to retrieve, analyze, and visualize data

ETL Process

Seminar 6



Extract

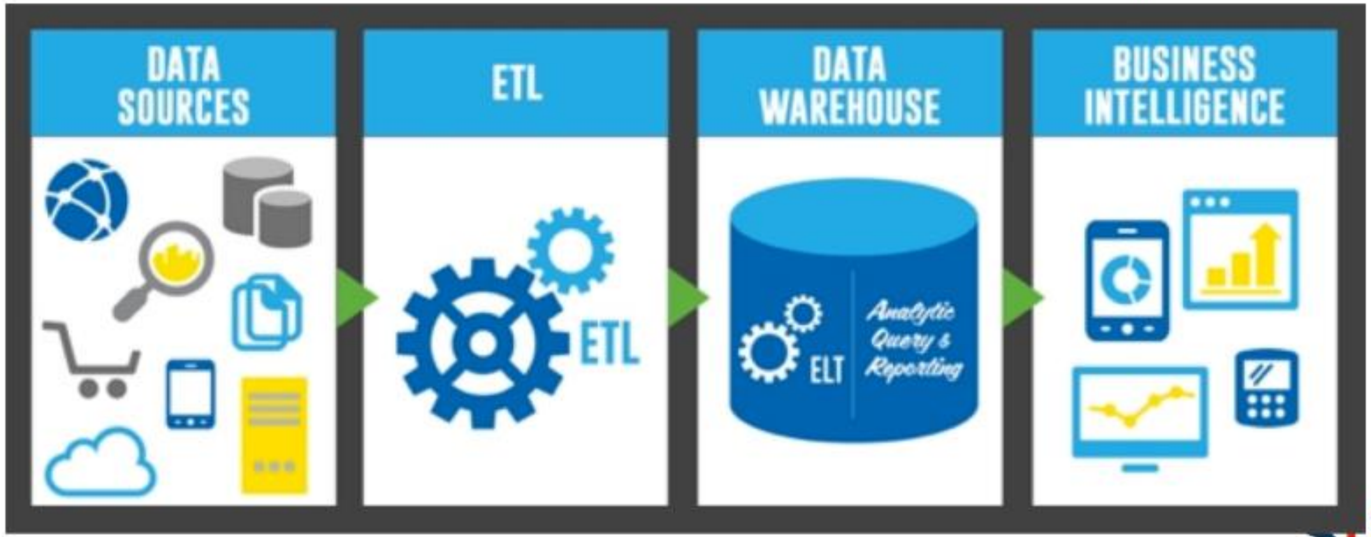
- One or more source systems containing customer, financial, or product data (CRM, Accounting system, Warehouse, MES)
- Files types - Flat files, XML, Oracle, IBM DB2, SQL Server,, IBM Websphere MQ, ODBC, JDBC, Hadoop Distributed File System (HDFS), Hive/HCatalog, JSON, Mainframe (IBM z/OS), Salesforce.com, SAP/R3

Transform

- Applying business rules, cleansing, and validating the data.
- Aggregation, Copy, Join, Sort, Merge, Partition, Filter, Reformat, Lookup
- Mathematical: +, -, x, /, Abs, IsValidNumber, Mod, Pow, Rand, Round, Sqrt, ToNumber, Truncate, Average, Min, Max
- Logical: And, Or, Not, IfThenElse, RegEx, Variables
- Text: Concatenate, CharacterLengthOf, LengthOf, Pad, Replace, ToLower, ToText, ToUpper, Translate, Trim, Hash
- Date: DateAdd, DateDiff, DateLastDay, DatePart, IsValidDate
- Format: ASCII, EBCDIC, Unicode

Load

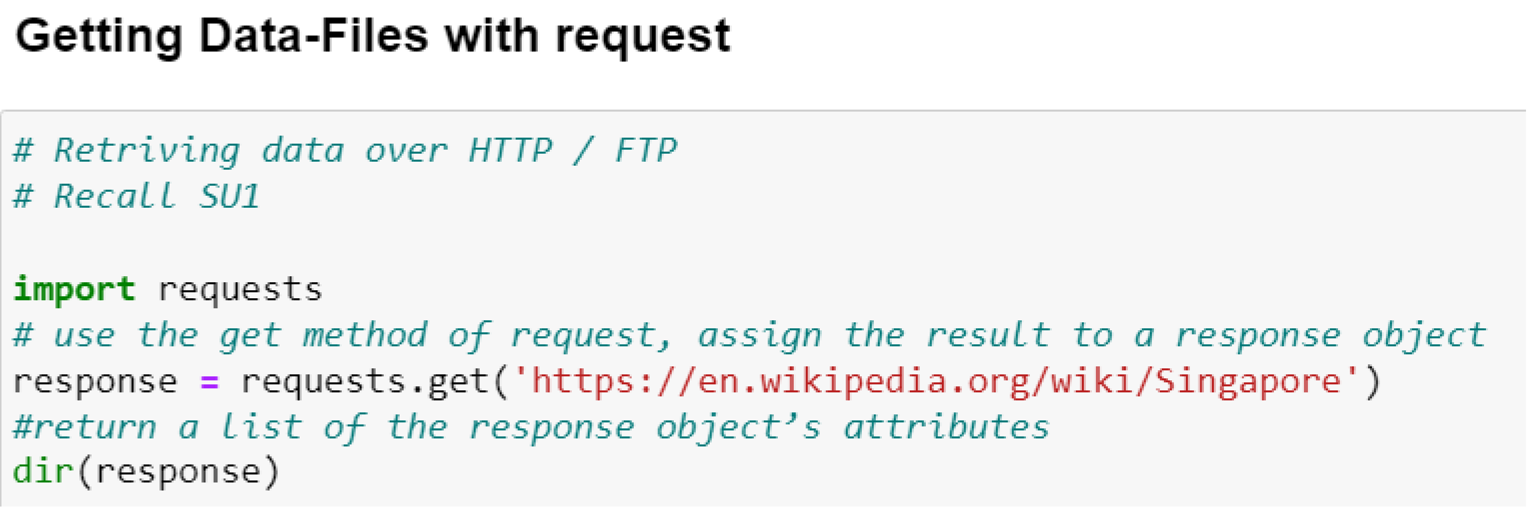
- Load the results into one or more target systems such as a data warehouse, datamart, or business intelligence reporting system.
- Output: Flat files, XML, Oracle, IBM DB2, SQL Server, Teradata, Sybase, Vertica, Netezza, Greenplum, ODBC, JDBC, Hadoop Distributed File System (HDFS), Hive/HCatalog, Mainframe (IBM z/OS), Salesforce.com, Tableau, QlikView



Chapter 1: Getting Web Data using API

1.1 Introduction

- Ways to get data off the web
  - Get a raw data file over HTTP or FTP
  - Use a dedicated API provided by web services to get the data
  - Scrap the data by getting web pages by HTTP/S, and parsing the data locally for content.



Getting Data-Files with request

```
# Retriving data over HTTP / FTP
# Recall SU1

import requests
# use the get method of request, assign the result to a response object
response = requests.get('https://en.wikipedia.org/wiki/Singapore')
#return a list of the response object’s attributes
dir(response)
```

Chapter 1: Getting Web Data using API

1.2 Using Python to Consume Data from a WebAPI

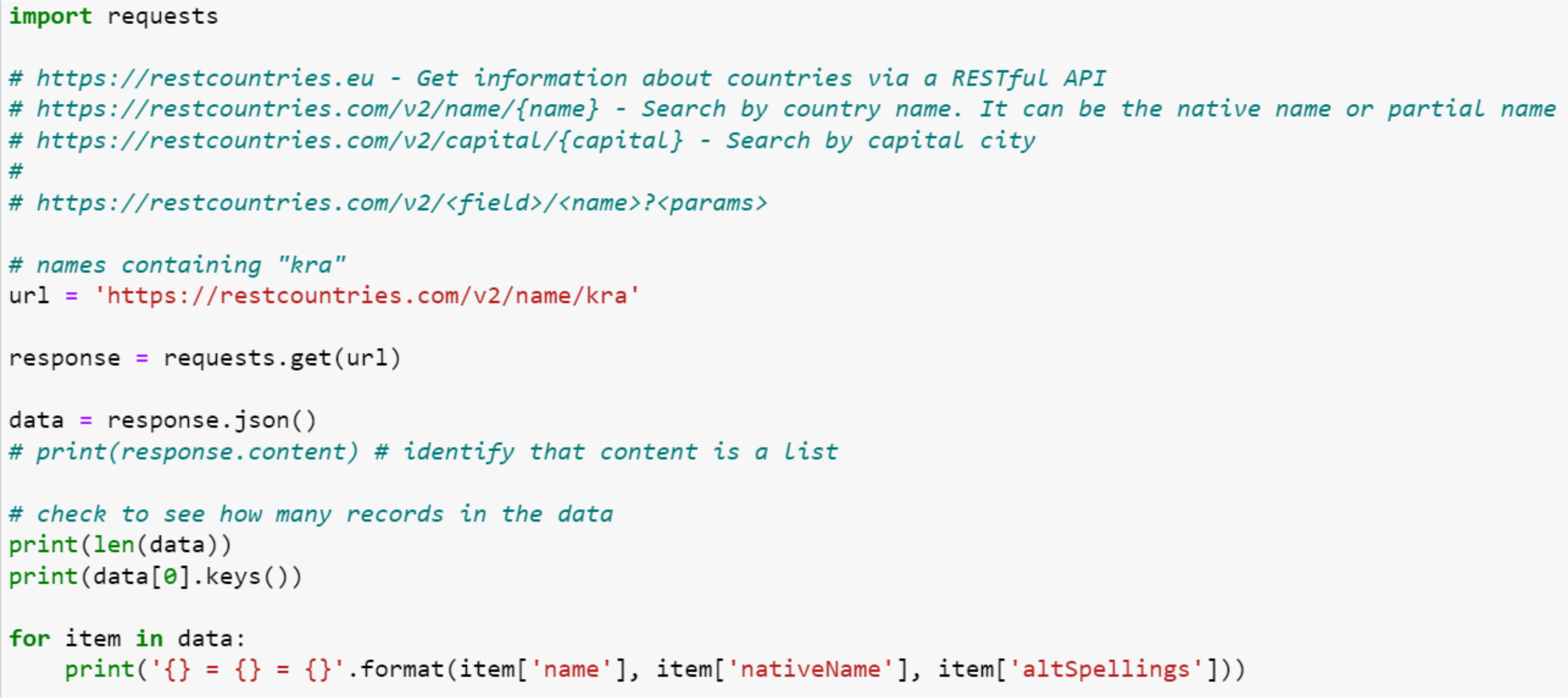
Ways to consume such APIs:

- **REST**
  - Representational State Transfer
  - Using a combination of HTTP verbs (GET, POST, etc.) and Uniform Resource Identifiers (URIs)
  - e.g. /items/id to access, create and update data.
- SOAP
  - Simple Object Access Protocol
  - Using Complex XML (Header, Body) and HTTP/SMTP/ other protocols
- XML - RPC
  - A remote procedure call protocol (RPC)
  - Using simple XML encoding (Method calls & Params) and HTTP transport.

Chapter 1: Getting Web Data using API

1.2 Using Python to Consume Data from a WebAPI

- Read the documentation (if available) on the API's site
  - Example 1 Getting data from RESTcountries



```
import requests

# https://restcountries.eu - Get information about countries via a RESTful API
# https://restcountries.com/v2/name/{name} - Search by country name. It can be the native name or partial name
# https://restcountries.com/v2/capital/{capital} - Search by capital city
#
# https://restcountries.com/v2/<field>/<name>?<params>

# names containing "kra"
url = 'https://restcountries.com/v2/name/kra'

response = requests.get(url)

data = response.json()
# print(response.content) # identify that content is a list

# check to see how many records in the data
print(len(data))
print(data[0].keys())

for item in data:
    print('{} = {} = {}'.format(item['name'], item['nativeName'], item['altSpellings']))
```

Chapter 1: Getting Web Data using API

1.2 Using Python to Consume Data from a WebAPI

- Example 2 & 3 - Getting data from Twitter using tweepy APIs
  - Free version – no longer available
  - To subscribe to scrape social media data from Twitter, e.g.
    - tweets from home timeline, list of followers
    - live tweets by keywords

Chapter 2: Web Scraping

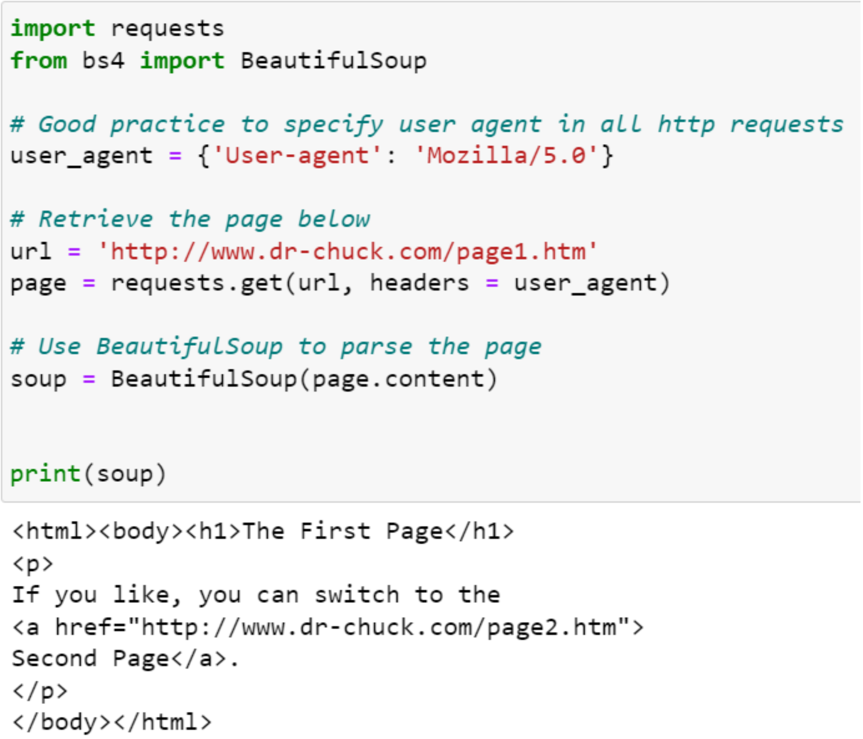
2.1 Introduction

- WWW as a big data repository
- Using selection patterns to get what we need
- Unstructured form such as repeatable HTML structures tables, div/css classes, li
  - ‘clean’ / structured form or proper common formats such as JSON, XML, CSV
  - think about "cut & paste" versus using automation

Chapter 2: Web Scraping

2.2 BeautifulSoup

- Python library for pulling data out of HTML and XML files
  - https://www.crummy.com/software/BeautifulSoup/bs4/doc/
  - pip install beautifulsoup



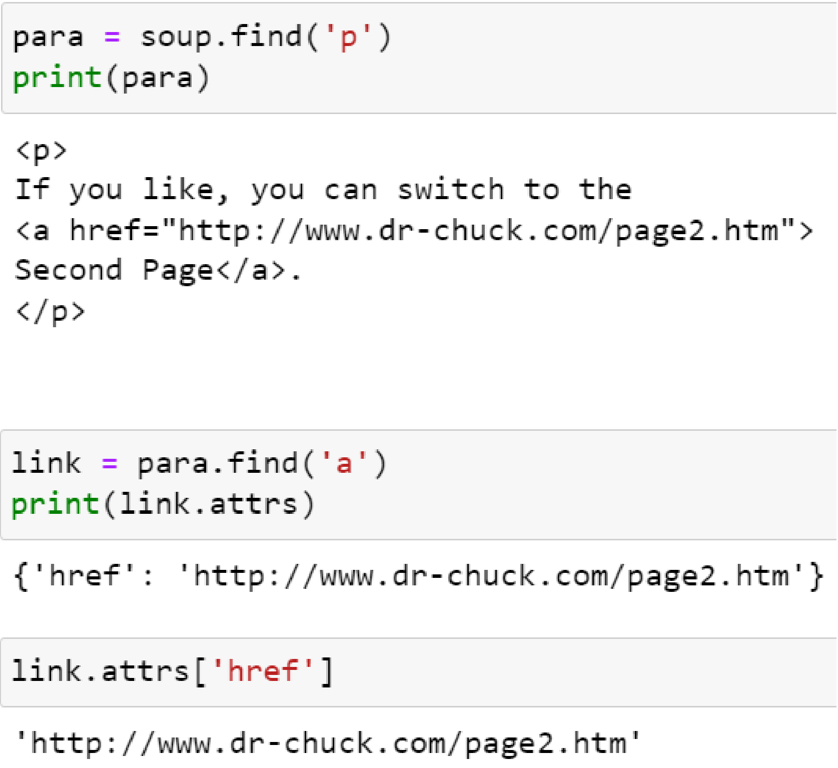
```python
import requests
from bs4 import BeautifulSoup

# Good practice to specify user agent in all http requests
user_agent = {'User-agent': 'Mozilla/5.0'}

# Retrieve the page below
url = 'http://www.dr-chuck.com/page1.htm'
page = requests.get(url, headers = user_agent)

# Use BeautifulSoup to parse the page
soup = BeautifulSoup(page.content)


print(soup)
```

```
<html><body><h1>The First Page</h1>
<p>
If you like, you can switch to the
<a href="http://www.dr-chuck.com/page2.htm">
Second Page</a>.
</p>
</body></html>
```

Chapter 2: Web Scraping

2.2 BeautifulSoup

- find() method: to get one anchor
- find_all() method to return a list of all anchors found



```
para = soup.find('p')
print(para)
```

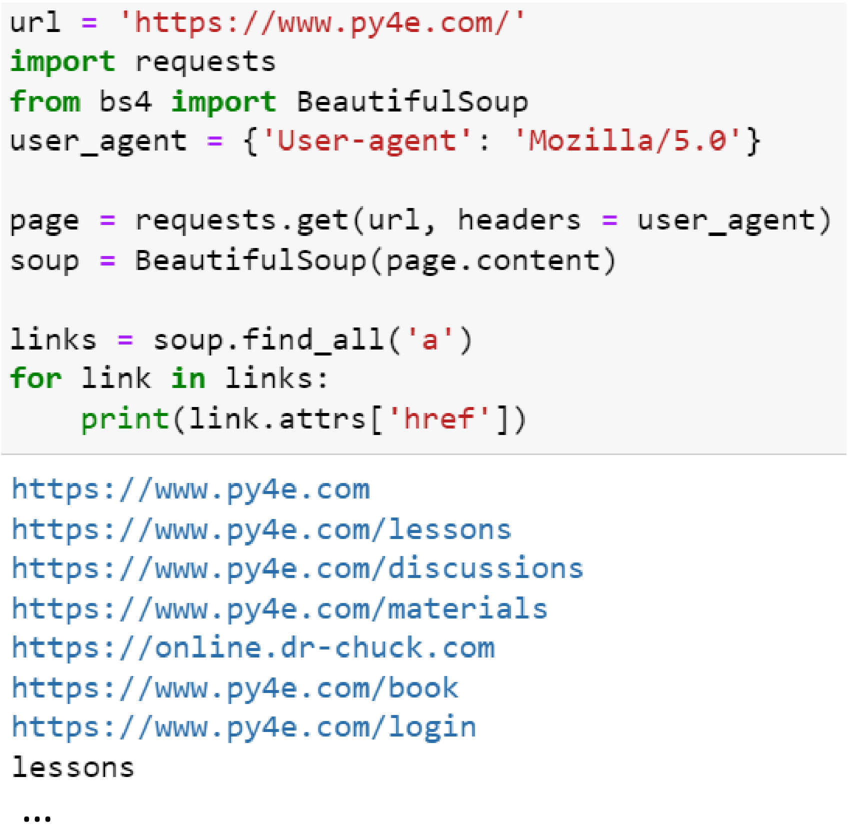
```
<p>
If you like, you can switch to the
<a href="http://www.dr-chuck.com/page2.htm">
Second Page</a>.
</p>
```

```
link = para.find('a')
print(link.attrs)
```

```
{'href': 'http://www.dr-chuck.com/page2.htm'}
```

```
link.attrs['href']
```

```
'http://www.dr-chuck.com/page2.htm'
```



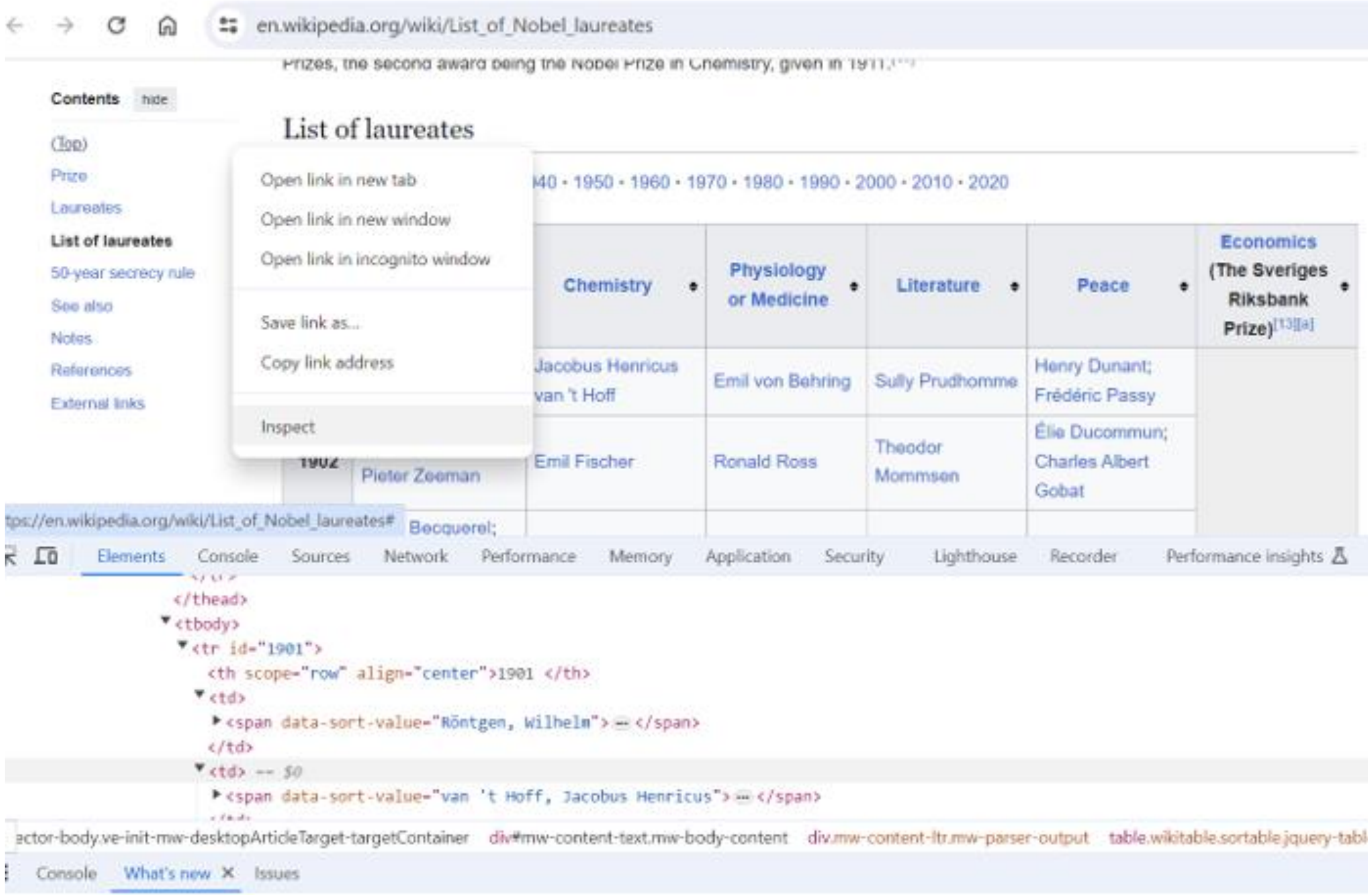
```
url = 'https://www.py4e.com/'
import requests
from bs4 import BeautifulSoup
user_agent = {'User-agent': 'Mozilla/5.0'}

page = requests.get(url, headers = user_agent)
soup = BeautifulSoup(page.content)

links = soup.find_all('a')
for link in links:
    print(link.attrs['href'])
```

```
https://www.py4e.com
https://www.py4e.com/lessons
https://www.py4e.com/discussions
https://www.py4e.com/materials
https://online.dr-chuck.com
https://www.py4e.com/book
https://www.py4e.com/login
lessons
 ...
```

Chapter 2: Web Scraping

2.2 BeautifulSoup Web Scraping - Complicated Example

- Scrape table from Wikipedia Nobel page https://en.wikipedia.org/wiki/List_of_Nobel_laureates
- Approach
  - "Inspect" the page using your browser development tools
  - Craft your Selection Patterns



Prizes, the second award being the Nobel Prize in Chemistry, given in 1911.

List of laureates

40 · 1950 · 1960 · 1970 · 1980 · 1990 · 2000 · 2010 · 2020

		Chemistry	Physiology or Medicine	Literature	Peace	Economics (The Sveriges Riksbank Prize)[13][a]
		Jacobus Henricus van 't Hoff	Emil von Behring	Sully Prudhomme	Henry Dunant; Frédéric Passy	
1902	Pieter Zeeman	Emil Fischer	Ronald Ross	Theodor Mommsen	Élie Ducommun; Charles Albert Gobat	
	Becquerel;					

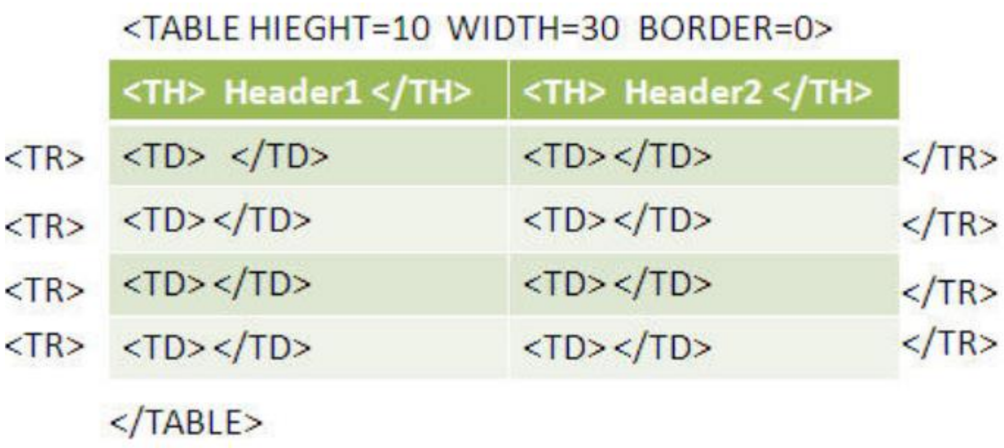
```
</thead>
▼<tbody>
 ▼<tr id="1901">
    <th scope="row" align="center">1901 </th>
   ▼<td>
     ▶<span data-sort-value="Röntgen, Wilhelm">…</span>
    </td>
   ▼<td> == $0
     ▶<span data-sort-value="van 't Hoff, Jacobus Henricus">…</span>
```

ector-body.ve-init-mw-desktopArticleTarget-targetContainer div#mw-content-text.mw-body-content div.mw-content-ltr.mw-parser-output table.wikitable.sortable.jquery-tabl

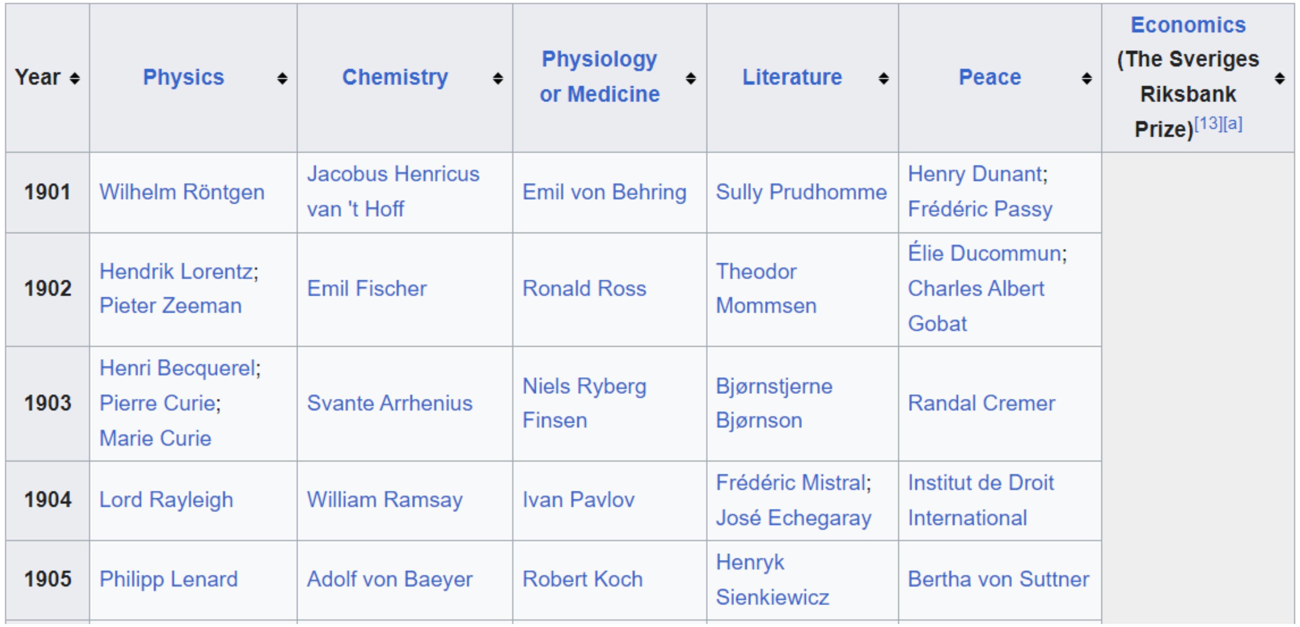
Chapter 2: Web Scraping

2.2 BeautifulSoup Web Scraping - Complicated Example

- Crafting Selection Patterns
  - html tag structure of Table



```
     <TABLE HIEGHT=10  WIDTH=30  BORDER=0>
     <TH>  Header1 </TH>   <TH>  Header2 </TH>
<TR> <TD>   </TD>          <TD></TD>           </TR>
<TR> <TD></TD>             <TD></TD>           </TR>
<TR> <TD></TD>             <TD></TD>           </TR>
<TR> <TD></TD>             <TD></TD>           </TR>
     </TABLE>
```



Year ⬍	Physics ⬍	Chemistry ⬍	Physiology or Medicine ⬍	Literature ⬍	Peace ⬍	Economics (The Sveriges Riksbank Prize)[13][a] ⬍
1901	Wilhelm Röntgen	Jacobus Henricus van 't Hoff	Emil von Behring	Sully Prudhomme	Henry Dunant; Frédéric Passy	
1902	Hendrik Lorentz; Pieter Zeeman	Emil Fischer	Ronald Ross	Theodor Mommsen	Élie Ducommun; Charles Albert Gobat	

1903	Henri Becquerel;
	Pierre Curie;
	Marie Curie
		Svante Arrhenius	Niels Ryberg Finsen	Bjørnstjerne Bjørnson	Randal Cremer	

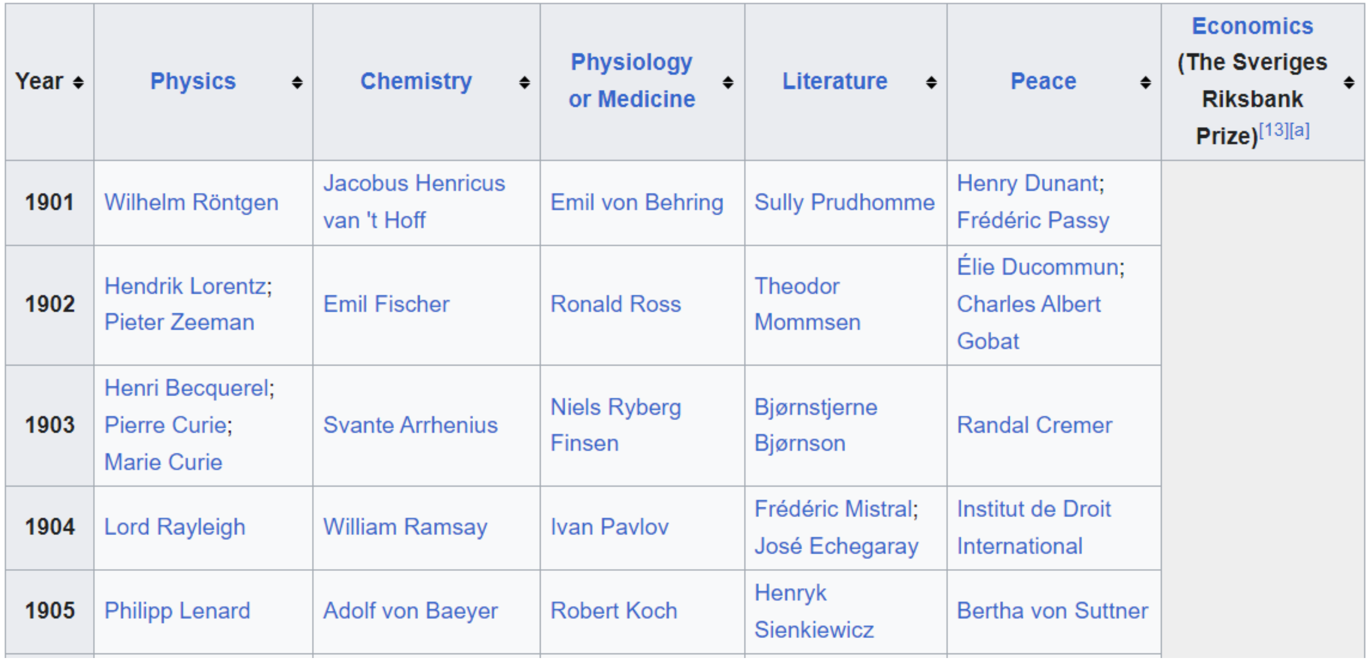
1904	Lord Rayleigh	William Ramsay	Ivan Pavlov	Frédéric Mistral; José Echegaray	Institut de Droit International	
1905	Philipp Lenard	Adolf von Baeyer	Robert Koch	Henryk Sienkiewicz	Bertha von Suttner	

...

Chapter 2: Web Scraping

2.2 BeautifulSoup Web Scraping - Complicated Example

- Output in json



Year	Physics	Chemistry	Physiology or Medicine	Literature	Peace	Economics (The Sveriges Riksbank Prize)[13][a]
1901	Wilhelm Röntgen	Jacobus Henricus van 't Hoff	Emil von Behring	Sully Prudhomme	Henry Dunant; Frédéric Passy	
1902	Hendrik Lorentz; Pieter Zeeman	Emil Fischer	Ronald Ross	Theodor Mommsen	Élie Ducommun; Charles Albert Gobat	

1903	Henri Becquerel;
	Pierre Curie;
	Marie Curie
		Svante Arrhenius	Niels Ryberg Finsen	Bjørnstjerne Bjørnson	Randal Cremer	

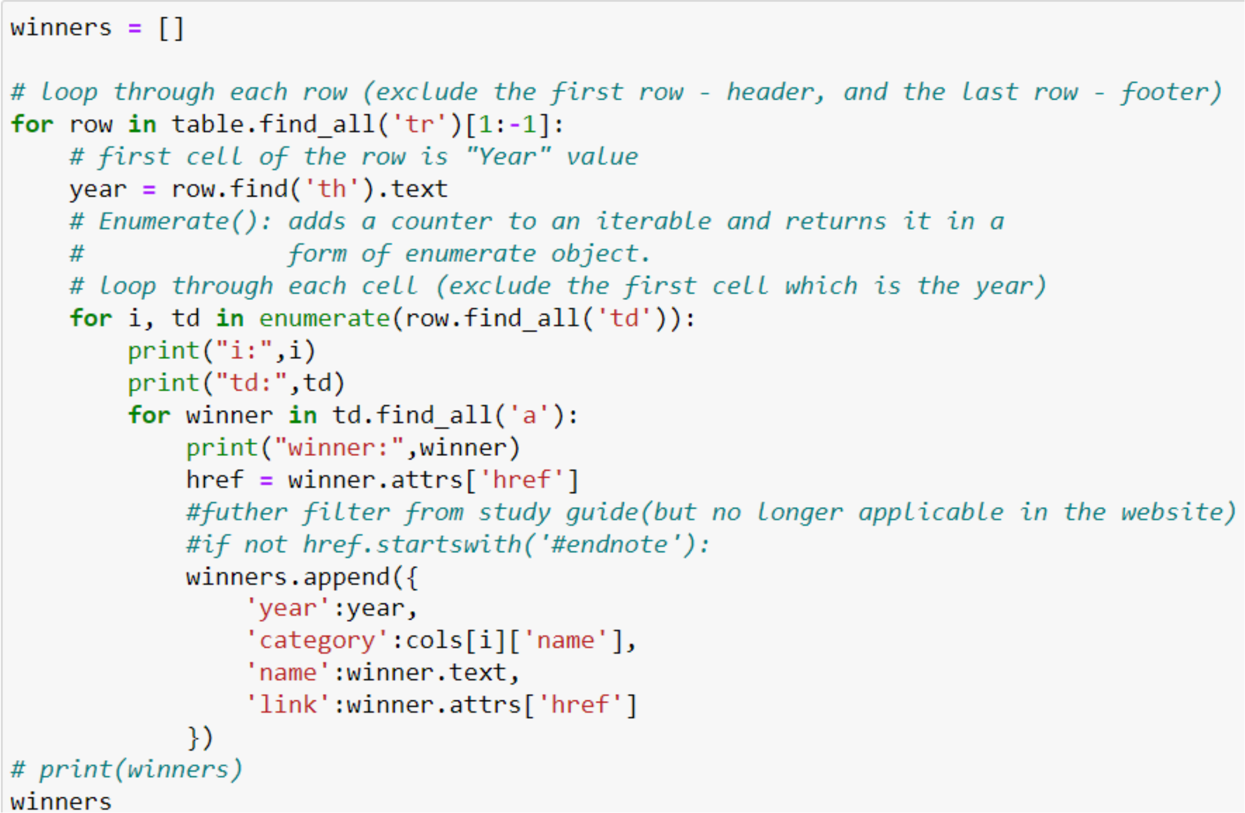
1904	Lord Rayleigh	William Ramsay	Ivan Pavlov	Frédéric Mistral; José Echegaray	Institut de Droit International	
1905	Philipp Lenard	Adolf von Baeyer	Robert Koch	Henryk Sienkiewicz	Bertha von Suttner	

```
[{'year': '1901\n',
  'category': 'Physics',
  'name': 'Wilhelm Röntgen',
  'link': '/wiki/Wilhelm_R%C3%B6ntgen'},
 {'year': '1901\n',
  'category': 'Chemistry',
  'name': "Jacobus Henricus van 't Hoff",
  'link': '/wiki/Jacobus_Henricus_van_%27t_Hoff'},
 {'year': '1901\n',
  'category': 'Physiologyor Medicine',
  'name': 'Emil von Behring',
  'link': '/wiki/Emil_von_Behring'},
 {'year': '1901\n',
  'category': 'Literature',
  'name': 'Sully Prudhomme',
  'link': '/wiki/Sully_Prudhomme'},
 {'year': '1901\n',
  'category': 'Peace',
  'name': 'Henry Dunant',
  'link': '/wiki/Henry_Dunant'} …
```

Chapter 2: Web Scraping

2.2 BeautifulSoup Web Scraping - Complicated Example

- Crafting Selection Patterns



```python
winners = []

# loop through each row (exclude the first row - header, and the last row - footer)
for row in table.find_all('tr')[1:-1]:
    # first cell of the row is "Year" value
    year = row.find('th').text
    # Enumerate(): adds a counter to an iterable and returns it in a
    #              form of enumerate object.
    # loop through each cell (exclude the first cell which is the year)
    for i, td in enumerate(row.find_all('td')):
        print("i:",i)
        print("td:",td)
        for winner in td.find_all('a'):
            print("winner:",winner)
            href = winner.attrs['href']
            #futher filter from study guide(but no longer applicable in the website)
            #if not href.startswith('#endnote'):
            winners.append({
                'year':year,
                'category':cols[i]['name'],
                'name':winner.text,
                'link':winner.attrs['href']
            })
# print(winners)
winners
```

```
[{'year': '1901\n',
  'category': 'Physics',
  'name': 'Wilhelm Röntgen',
  'link': '/wiki/Wilhelm_R%C3%B6ntgen'},
 {'year': '1901\n',
  'category': 'Chemistry',
  'name': "Jacobus Henricus van 't Hoff",
  'link': '/wiki/Jacobus_Henricus_van_%27t_Hoff'},
 {'year': '1901\n',
  'category': 'Physiologyor Medicine',
  'name': 'Emil von Behring',
  'link': '/wiki/Emil_von_Behring'},
 {'year': '1901\n',
  'category': 'Literature',
  'name': 'Sully Prudhomme',
  'link': '/wiki/Sully_Prudhomme'},
 {'year': '1901\n',
  'category': 'Peace',
  'name': 'Henry Dunant',
  'link': '/wiki/Henry_Dunant'} …
```

Chapter 3: Heavy Weight Web Scraping

3.1 First Hand with Scrapy

- Large-scale data scrapes
- Provide built-in caching (with expiration times), asynchronous threading, User-Agent randomization.. etc
  - User-Agent: a string that is sent along to any website you visit. This is a sort of "fingerprint" your browser leaves behind.
  - Some of the website could detect & block the identities which crawl the data frequently
- Installation
  - pip install scrapy

Chapter 3: Heavy Weight Web Scraping

3.1 First Hand with Scrapy

- Large-scale data scrapes
- Provide built-in caching (with expiration times), asynchronous threading, User-Agent randomization.. etc
  - User-Agent: a string that is sent along to any website you visit. This is a sort of "fingerprint" your browser leaves behind.
  - Some of the website could detect & block the identities which crawl the data frequently
- Installation
  - pip install scrapy

Chapter 3: Heavy Weight Web Scraping

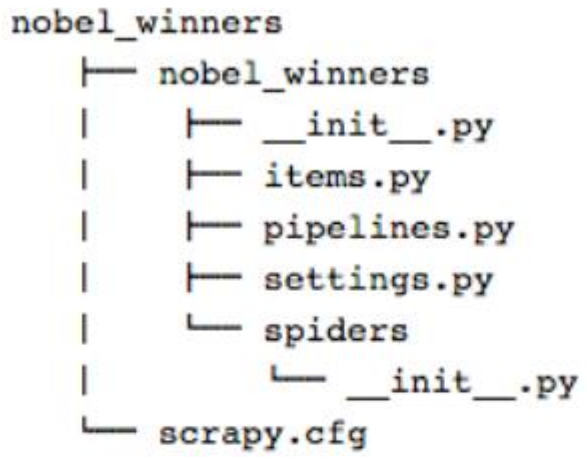
3.1 First Hand with Scrapy

- Generating a new project

scrapy startproject nobel_winners

cd nobel_winners

scrapy genspider example example.com



```
nobel_winners
    ├── nobel_winners
    |       ├── __init__.py
    |       ├── items.py
    |       ├── pipelines.py
    |       ├── settings.py
    |       └── spiders
    |               └── __init__.py
    └── scrapy.cfg
```

Chapter 3: Heavy Weight Web Scraping

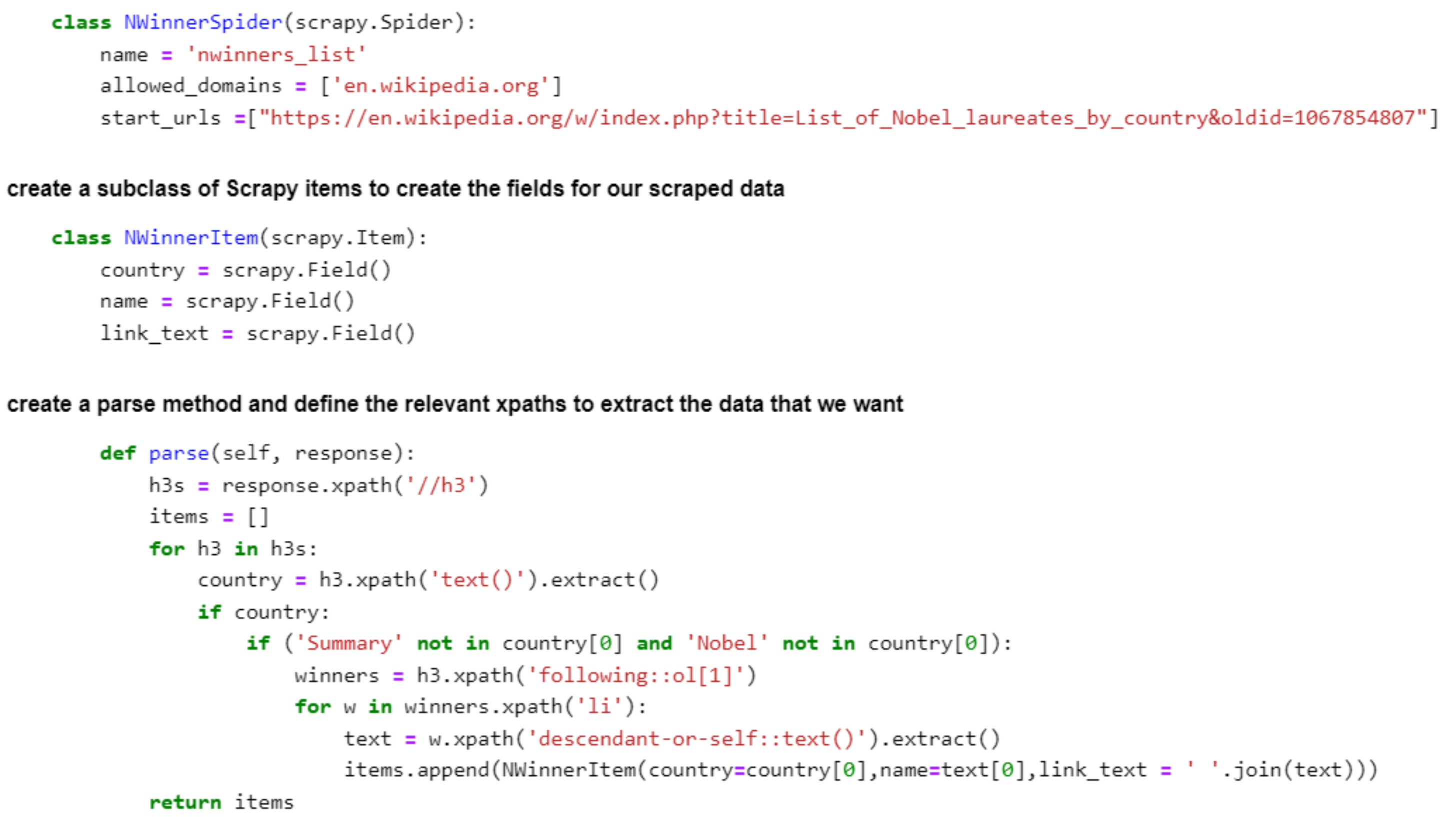
3.1 First Hand with Scrapy

- Establish the Scraping Target
  - To find the nationalities for each of the winners
    - https://en.wikipedia.org/w/index.php?title=List_of_Nobel_laureates_by_country&oldid=1067854807
  - Targeting HTML with Xpaths
  - Test out the xpath in scrapy shell

Chapter 3: Heavy Weight Web Scraping

3.2 Web Scraping using Scrapy Spider

- scraper spider is essentially a Python module
  - eg. nwinner_list_spider.py



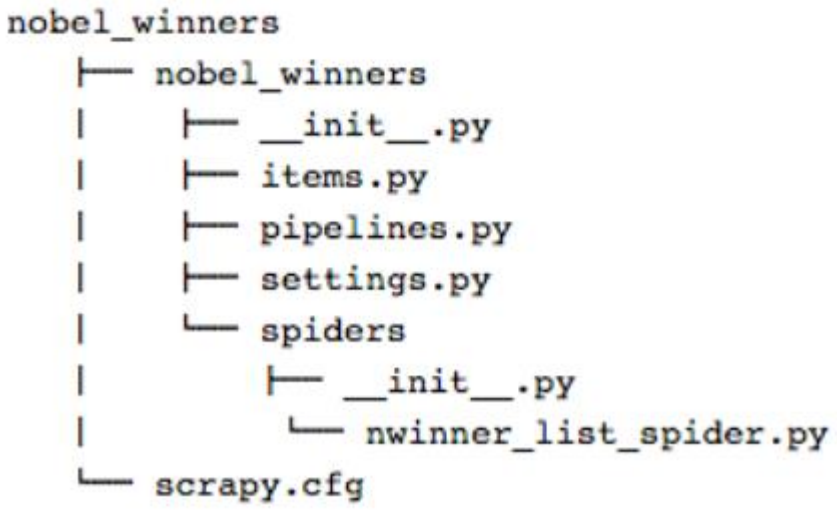
```
class NWinnerSpider(scrapy.Spider):
    name = 'nwinners_list'
    allowed_domains = ['en.wikipedia.org']
    start_urls =["https://en.wikipedia.org/w/index.php?title=List_of_Nobel_laureates_by_country&oldid=1067854807"]
```

create a subclass of Scrapy items to create the fields for our scraped data

```
class NWinnerItem(scrapy.Item):
    country = scrapy.Field()
    name = scrapy.Field()
    link_text = scrapy.Field()
```

create a parse method and define the relevant xpaths to extract the data that we want

```
    def parse(self, response):
        h3s = response.xpath('//h3')
        items = []
        for h3 in h3s:
            country = h3.xpath('text()').extract()
            if country:
                if ('Summary' not in country[0] and 'Nobel' not in country[0]):
                    winners = h3.xpath('following::ol[1]')
                    for w in winners.xpath('li'):
                        text = w.xpath('descendant-or-self::text()').extract()
                        items.append(NWinnerItem(country=country[0],name=text[0],link_text = ' '.join(text)))
        return items
```



```
nobel_winners
 ├── nobel_winners
 |     ├── __init__.py
 |     ├── items.py
 |     ├── pipelines.py
 |     ├── settings.py
 |     └── spiders
 |           ├── __init__.py
 |            └── nwinner_list_spider.py
 └── scrapy.cfg
```

Chapter 3: Heavy Weight Web Scraping

3.2 Web Scraping using Scrapy Spider

- Run the spider

  scrapy list

  scrapy crawl **nwinners_list** -o nwinners.json

- Generates output file nwinners.json

```
[
{"country": "Argentina", "name": "C\u00e9sar Milstein", "link_text": "C\u00e9sar Milstein *, Physiology or Medicine, 1984"},
{"country": "Argentina", "name": "Adolfo P\u00e9rez Esquivel", "link_text": "Adolfo P\u00e9rez Esquivel , Peace, 1980"},
{"country": "Argentina", "name": "Luis Federico Leloir", "link_text": "Luis Federico Leloir , Chemistry, 1970"},
{"country": "Argentina", "name": "Bernardo Houssay", "link_text": "Bernardo Houssay , Physiology or Medicine, 1947"},
{"country": "Argentina", "name": "Carlos Saavedra Lamas", "link_text": "Carlos Saavedra Lamas , Peace, 1936"},
{"country": "Australia", "name": "Brian Schmidt", "link_text": "Brian Schmidt ,  born in the United States , Physics, 2011"},
…
```

Chapter 3: Heavy Weight Web Scraping

3.2 Web Scraping using Scrapy Spider

- Saving data into database for further analysis
    - NoSQL (pymongo)
    - SQL (ORM – sqlalchemy)

SUMMARY

Data Scraping – LEARNING OBJECTIVES

1) Design methods to extract and parse information from the internet
2) Retrieve and consume data from Web APIs
3) Understand and appreciate simple and complex methods of retrieval and apply the right methods given the use cases
4) Construct a Python program to retrieve, analyze, and visualize data

THANK YOU