Using `XPATH` for web scraping

Example 3. Product price scraping from https://nepalfoods.gov.np/

In [1]:

```python
import pandas as pd
import requests
from bs4 import BeautifulSoup
from lxml import html

#loading webpage content
webpage = requests.get("https://nepalfoods.gov.np/").content
```

In [2]:

```python
tree = html.fromstring(webpage)

list_xpath = tree.xpath("//div[@class='product-category']")
category = [x.text_content().strip() for x in list_xpath]
print(len(category))
print(category)

product = [x.text_content().strip() for x in tree.xpath("//h2")]
print(len(product))
print(product)

price = [x.text_content().strip() for x in tree.xpath("//div[@class='product-price']")]
print(len(price))
print(price)
```

```
27
['अन्य', 'अन्य', 'चामल', 'चामल', 'चामल', 'दाल', 'दाल', 'चामल', 'चामल', 'चामल', 'चामल', 'अन्य', 'अन्य', 'तेल एवं घ्यू', 'तेल एवं घ्यू',
'तेल एवं घ्यू', 'तेल एवं घ्यू', 'अन्य', 'चामल', 'चामल', 'दाल', 'चामल', 'गेडागुडी', 'चामल', 'चामल', 'अन्य', 'अन्य']
27
['उवा १ केजी', 'चियापत्ती ५०० ग्राम', 'Long Grain चामल  १० केजी', 'हुम्लाको कागुनोको चामल १ केजी', 'हुम्लाको चिनोको चामल १ केजी', 'कर्णालीको
सिमि १ केजी', 'मुसुरो दाल(सानो) १ केजी', 'अरुवा सोना मन्सुली  चामल २५   केजी', 'अरुवा मोटा चामल ३० केजी', 'हुम्लाको कागुनोको चामल १ केजी',
'हुम्लाको चिनोको चामल १ केजी', 'टाइमपास टाइचिन चिउरा १ केजी', 'गहुँ आटा 5 केजी', 'भटमासको तेल १ लिटर', 'सनफ्लावर तेल १ लिटर', 'तोरीको तेल
(शान्ती) १ लिटर', 'डी.डी.सी डेरी घ्यू १ लि', 'डी.डी.सी डेरी घ्यू १/२ लि', 'मार्सी चामल १ केजी', 'ब्राउन चामल ५ किलो ग्राम', 'कर्णालीको सिमि १ केजी',
'स्टिम जिरा मसिनो चामल 25 केजी', 'क्काँटी १ केजी', 'बासमती चामल २०  किलो ग्राम', 'काला नुनीया (काला नमक चामल ) 10 kg', 'STC ग्यास सिलिण्डर(E
xchange only STC Cylinder)', 'STC ग्यास  सिलिण्\u200dडर सहित']
27
['NPR\xa0200.00', 'NPR\xa0270.00', 'NPR\xa01780.00', 'NPR\xa0260.00', 'NPR\xa0260.00', 'NPR\xa0240.00', 'NPR\xa0165.00', 'NPR\x
a01700.00', 'NPR\xa01560.00', 'NPR\xa0260.00', 'NPR\xa0260.00', 'NPR\xa0100.00', 'NPR\xa0360.00', 'NPR\xa0215.00', 'NPR\xa0220.
00', 'NPR\xa0385.00', 'NPR\xa01160.00', 'NPR\xa0580.00', 'NPR\xa0230.00', 'NPR\xa0325.00', 'NPR\xa0240.00', 'NPR\xa02125.00',
'NPR\xa0145.00', 'NPR\xa02700.00', 'NPR\xa01500.00', 'NPR\xa01910.00', 'NPR\xa04425.00']
```

In [3]:
```python
df = pd.DataFrame({'category': category, 'product':product, 'price':price})
df = df.sort_values(by='category')
display(df.head())

df.to_csv('example3-python.csv', index=False)
```

	category	product	price
0	अन्य	उवा १ केजी	NPR 200.00
17	अन्य	डी.डी.सी डेरी घ्यू १/२ लि	NPR 580.00
25	अन्य	STC ग्यास सिलिण्डर(Exchange only STC Cylinder)	NPR 1910.00
12	अन्य	गहुँ आटा 5 केजी	NPR 360.00
11	अन्य	टाइमपास टाइचिन चिउरा १ केजी	NPR 100.00

using `Beautifulsoup` (alternate way)

In [4]:
```python
soup = BeautifulSoup(webpage, 'html.parser')
sp = soup.find_all('div', attrs={'class':'product-category'})
category = [x.text.strip() for x in sp]
```

```
print(len(category))
print(category)

product = [x.text.strip() for x in soup.find_all('h2')]
print(len(product))
print(product)

price = [x.text.strip() for x in soup.find_all('div', attrs={'class':'product-price'})]
print(len(price))
print(price)
```

```
27
['अन्य', 'अन्य', 'चामल', 'चामल', 'चामल', 'दाल', 'दाल', 'चामल', 'चामल', 'चामल', 'चामल', 'अन्य', 'अन्य', 'तेल एवं घ्यू', 'तेल एवं घ्यू',
'तेल एवं घ्यू', 'तेल एवं घ्यू', 'अन्य', 'चामल', 'चामल', 'दाल', 'चामल', 'गेडागुडी', 'चामल', 'चामल', 'अन्य', 'अन्य']
27
['उवा १ केजी', 'चियापत्ती ५०० ग्राम', 'Long Grain चामल  १० केजी', 'हुम्लाको कागुनोको चामल १ केजी', 'हुम्लाको चिनोको चामल १ केजी', 'कर्णालीको
सिमि १ केजी', 'मुसुरो दाल(सानो) १ केजी', 'अरुवा सोना मन्सुली  चामल २५   केजी', 'अरुवा मोटा चामल ३० केजी', 'हुम्लाको कागुनोको चामल १ केजी',
'हुम्लाको चिनोको चामल १ केजी', 'टाइमपास टाइचिन चिउरा १ केजी', 'गहुँ आटा 5 केजी', 'भटमासको तेल १ लिटर', 'सनफ्लावर तेल १ लिटर', 'तोरीको तेल
(शान्ती) १ लिटर', 'डी.डी.सी डेरी घ्यू १ लि', 'डी.डी.सी डेरी घ्यू १/२ लि', 'मार्सी चामल १ केजी', 'ब्राउन चामल ५ किलो ग्राम', 'कर्णालीको सिमि १ केजी',
'स्टिम जिरा मसिनो चामल 25 केजी', 'क्वाँटी १ केजी', 'बासमती चामल २०  किलो ग्राम', 'काला नुनीया (काला नमक चामल ) 10 kg', 'STC ग्यास सिलिण्डर(E
xchange only STC Cylinder)', 'STC ग्यास  सिलिण्\u200dडर सहित']
27
['NPR\xa0200.00', 'NPR\xa0270.00', 'NPR\xa01780.00', 'NPR\xa0260.00', 'NPR\xa0260.00', 'NPR\xa0240.00', 'NPR\xa0165.00', 'NPR\x
a01700.00', 'NPR\xa01560.00', 'NPR\xa0260.00', 'NPR\xa0260.00', 'NPR\xa0100.00', 'NPR\xa0360.00', 'NPR\xa0215.00', 'NPR\xa0220.
00', 'NPR\xa0385.00', 'NPR\xa01160.00', 'NPR\xa0580.00', 'NPR\xa0230.00', 'NPR\xa0325.00', 'NPR\xa0240.00', 'NPR\xa02125.00',
'NPR\xa0145.00', 'NPR\xa02700.00', 'NPR\xa01500.00', 'NPR\xa01910.00', 'NPR\xa04425.00']
```

In [5]:

```
df = pd.DataFrame({'category': category, 'product':product, 'price':price})
df = df.sort_values(by='category')
display(df.head())

df.to_csv('example3-python.csv', index=False)
```

	category	product	price
0	अन्य	उवा १ केजी	NPR 200.00
17	अन्य	डी.डी.सी डेरी घ्यू १/२ लि	NPR 580.00
25	अन्य	STC ग्यास सिलिण्डर(Exchange only STC Cylinder)	NPR 1910.00
12	अन्य	गहुँ आटा 5 केजी	NPR 360.00
11	अन्य	टाइमपास टाइचिन चिउरा १ केजी	NPR 100.00

Example 4. Extract information on Top Box Office movies from https://www.imdb.com/chart/boxoffice

In [6]:

```
#Loading webpage content
hdr = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safar
}
webpage = requests.get("https://www.imdb.com/chart/boxoffice", headers = hdr).content
```

In [7]:

```
tree = html.fromstring(webpage)

movie = [x.text_content().strip() for x in tree.xpath("//a[@class='ipc-title-link-wrapper']")]
print(len(movie))
print(movie)

weekend_gross = [x.text_content().strip() for x in tree.xpath("//span[contains(.,'Weekend Gross:')]/parent::*/span[2]")]
print(len(weekend_gross))
print(weekend_gross)

total_gross = [x.text_content().strip() for x in tree.xpath("//span[contains(.,'Total Gross:')]/parent::*/span[2]")]
print(len(total_gross))
print(total_gross)

weeks_released = [x.text_content().strip() for x in tree.xpath("//span[contains(.,'Weeks Released:')]/parent::*/span[2]")]
```

```
print(len(weeks_released))
print(weeks_released)

rating = [x.text_content().strip() for x in tree.xpath("//span[@data-testid='ratingGroup--imdb-rating']")]
print(len(rating))
print(rating)
```

```
10
['1. Inside Out 2', '2. Bad Boys: Ride or Die', '3. Kingdom of the Planet of the Apes', '4. The Garfield Movie', '5. IF', '6. T
he Watchers', '7. Furiosa: A Mad Max Saga', '8. The Fall Guy', '9. The Strangers: Chapter 1', '10. The Lord of the Rings: The F
ellowship of the Ring']
10
['$154M', '$34M', '$5.5M', '$4.8M', '$3.6M', '$3.5M', '$2.6M', '$1.6M', '$759K', '$633K']
10
['$154M', '$113M', '$158M', '$78M', '$101M', '$14M', '$63M', '$88M', '$34M', '$319M']
10
['1', '2', '6', '4', '5', '2', '4', '7', '5', '2']
10
['8.0\xa0(16K)', '7.0\xa0(19K)', '7.2\xa0(53K)', '5.8\xa0(8.4K)', '6.7\xa0(14K)', '5.8\xa0(6.9K)', '7.8\xa0(85K)', '7.0\xa0(101
K)', '4.7\xa0(11K)', '8.9\xa0(2M)']
```

In [8]:

```
df = pd.DataFrame({'movie': movie, 'weekend_gross':weekend_gross, 'total_gross':total_gross, 'weeks_released':weeks_released,
display(df.head())

df.to_csv('example4-python.csv', index=False)
```

	movie	weekend_gross	total_gross	weeks_released	rating
0	1. Inside Out 2	$154M	$154M	1	8.0 (16K)
1	2. Bad Boys: Ride or Die	$34M	$113M	2	7.0 (19K)
2	3. Kingdom of the Planet of the Apes	$5.5M	$158M	6	7.2 (53K)
3	4. The Garfield Movie	$4.8M	$78M	4	5.8 (8.4K)
4	5. IF	$3.6M	$101M	5	6.7 (14K)

Practice 2. From https://www.imdb.com/chart/moviemeter, prepare a table of most popular movies with movie name, year, length, and ratings.

In [9]:

```
#Loading webpage content
hdr = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safar
}
webpage = requests.get("https://www.imdb.com/chart/moviemeter", headers = hdr).content
```

In [10]:

```
tree = html.fromstring(webpage)

movie = [x.text_content().strip() for x in tree.xpath("//div[contains(@class,'cli-children')]/div[2]")]
print(len(movie))
print(movie)

year = [x.text_content().strip() for x in tree.xpath("//div[contains(@class,'cli-children')]/div[3]/span[1]")]
print(len(year))
print(year)

length = [x.text_content().strip() for x in tree.xpath("//div[contains(@class,'cli-children')]/div[3]/span[2]")]
print(len(length))
print(length)

grading = [x.text_content().strip() for x in tree.xpath("//div[contains(@class,'cli-children')]/div[3]/span[3]")]
print(len(grading))
print(grading)

rating = [x.text_content().strip() for x in tree.xpath("//div[contains(@class,'cli-children')]/span/div/span[1]")]
print(len(rating))
print(rating)
```

```
25
['Hit Man', 'Bad Boys: Ride or Die', 'Inside Out 2', 'Furiosa: A Mad Max Saga', 'Sous la Seine', 'The Watchers', 'The Fall Gu
y', 'Gojira -1.0', 'Civil War', 'Inside Out', 'Dune: Part Two', 'Kingdom of the Planet of the Apes', 'The Strangers: Chapter
1', 'Munjya', 'Challengers', 'Deadpool & Wolverine', 'The Bikeriders', 'Mad Max: Fury Road', 'Anyone But You', 'The First Ome
n', 'IF', 'Am I OK?', 'Kinds of Kindness', 'The Ministry of Ungentlemanly Warfare', 'Atlas']
25
['2023', '2024', '2024', '2024', '2024', '2024', '2024', '2023', '2024', '2015', '2024', '2024', '2024', '2024', '2024', '202
4', '2023', '2015', '2023', '2024', '2024', '2022', '2024', '2024', '2024']
25
['1h 55m', '1h 55m', '1h 36m', '2h 28m', '1h 44m', '1h 42m', '2h 6m', '2h 4m', '1h 49m', '1h 35m', '2h 46m', '2h 25m', '1h 31
m', '2h 20m', '2h 11m', '2h 7m', '1h 56m', '2h', '1h 43m', '1h 59m', '1h 44m', '1h 26m', '2h 44m', '2h', '1h 58m']
24
['R', 'R', 'PG', 'R', 'TV-MA', 'PG-13', 'PG-13', 'PG-13', 'R', 'PG', 'PG-13', 'PG-13', 'R', 'R', 'R', 'R', 'R', 'R', 'R', 'PG',
'R', 'R', 'R', 'PG-13']
25
['7.0\xa0(43K)', '7.0\xa0(19K)', '8.0\xa0(16K)', '7.8\xa0(85K)', '5.2\xa0(18K)', '5.8\xa0(6.9K)', '7.0\xa0(101K)', '7.8\xa0(119
K)', '7.2\xa0(114K)', '8.1\xa0(795K)', '8.6\xa0(450K)', '7.2\xa0(53K)', '4.7\xa0(11K)', '7.7\xa0(12K)', '7.3\xa0(69K)', '', '7.
4\xa0(1.7K)', '8.1\xa0(1.1M)', '6.1\xa0(93K)', '6.5\xa0(35K)', '6.7\xa0(14K)', '6.1\xa0(4K)', '6.9\xa0(2.7K)', '6.9\xa0(44K)',
'5.6\xa0(41K)']
```

In [11]:
```
df = pd.DataFrame({'movie': movie, 'year':year, 'length':length, 'rating':rating})
display(df.head())

df.to_csv('practice2.csv', index=False)
```

	movie	year	length	rating
0	Hit Man	2023	1h 55m	7.0 (43K)
1	Bad Boys: Ride or Die	2024	1h 55m	7.0 (19K)
2	Inside Out 2	2024	1h 36m	8.0 (16K)
3	Furiosa: A Mad Max Saga	2024	2h 28m	7.8 (85K)
4	Sous la Seine	2024	1h 44m	5.2 (18K)

Web scraping using `for` loop

If you look at the above example, you will notice that grading is missing for a movie. In this case, creating a balanced table is not possible. To resolve this issue, we need to use looping.

In [12]:

```
tree = html.fromstring(webpage)
elems = tree.xpath("//div[contains(@class,'cli-children')]")

movie = []
year = []
length = []
rating = []
grading = []

for e in elems:
    val = e.xpath("div[2]")
    movie.append(val[0].text_content().strip() if len(val) > 0 else '')

    val = e.xpath("div[3]/span[1]")
    year.append(val[0].text_content().strip() if len(val) > 0 else '')

    val = e.xpath("div[3]/span[2]")
    length.append(val[0].text_content().strip() if len(val) > 0 else '')

    val = e.xpath("div[3]/span[3]")
    grading.append(val[0].text_content().strip() if len(val) > 0 else '')

    val = e.xpath("span/div/span[1]")
    rating.append(val[0].text_content().strip() if len(val) > 0 else '')
```

In [13]:

```
df = pd.DataFrame({'movie': movie, 'year':year, 'length':length, 'grading':grading, 'rating':rating})
display(df.head())

df.to_csv('practice2_forloop.csv', index=False)
```

	movie	year	length	grading	rating
0	Hit Man	2023	1h 55m	R	7.0 (43K)
1	Bad Boys: Ride or Die	2024	1h 55m	R	7.0 (19K)
2	Inside Out 2	2024	1h 36m	PG	8.0 (16K)
3	Furiosa: A Mad Max Saga	2024	2h 28m	R	7.8 (85K)
4	Sous la Seine	2024	1h 44m	TV-MA	5.2 (18K)