lab-scraping-esp

November 27, 2024

1 Laboratorio de Web Scraping

Encontrarás en este cuaderno algunos ejercicios de web scraping para practicar tus habilidades de scraping usando `requests` y `Beautiful Soup`.

Consejos:

- Verifica el código de estado de la respuesta para cada solicitud para asegurarte de haber obtenido el contenido previsto.
- Observa el código HTML en cada solicitud para entender el tipo de información que estás obteniendo y su formato.
- Busca patrones en el texto de respuesta para extraer los datos/información solicitados en cada pregunta.
- Visita cada URL y echa un vistazo a su fuente a través de Chrome DevTools. Necesitarás identificar las etiquetas HTML, nombres de clases especiales, etc., utilizados para el contenido HTML que se espera extraer.
- Revisa los selectores CSS.

1.0.1 Recursos Útiles

- Documentación de la biblioteca Requests
- Doc de Beautiful Soup
- Lista de códigos de estado HTTP
- Conceptos básicos de HTML
- Conceptos básicos de CSS

Primero que todo, reuniendo nuestras herramientas.

[1]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import json
```

Nuevamente, recuerda limitar tu salida antes de la entrega para que tu código no se pierda en la salida.

Desafío 1 - Descargar, analizar (usando BeautifulSoup) e imprimir el contenido de la página de Desarrolladores en Tendencia de GitHub:

[2]:
```
# Esta es la URL que vas a extraer en este ejercicio
url = 'https://github.com/trending/developers'
```

[3]:
```
github_html=requests.get(url).text
soup = BeautifulSoup(github_html, "html.parser")
```

Muestra los nombres de los desarrolladores en tendencia recuperados en el paso anterior. Tu salida debe ser una lista de Python con los nombres de los desarrolladores. Cada nombre no debe contener ninguna etiqueta HTML.

Instrucciones:

1. Descubre la etiqueta HTML y los nombres de clase usados para los nombres de los desarrolladores. Puedes lograr esto usando Chrome DevTools.
2. Usa BeautifulSoup para extraer todos los elementos HTML que contienen los nombres de los desarrolladores.
3. Utiliza técnicas de manipulación de cadenas para reemplazar espacios en blanco y saltos de línea (es decir, `\n`) en el *texto* de cada elemento HTML. Usa una lista para almacenar los nombres limpios.
4. Imprime la lista de nombres.

Tu salida debería lucir como abajo (con nombres diferentes):

```
['trimstray (@trimstray)',
 'joewalnes (JoeWalnes)',
 'charlax (Charles-AxelDein)',
 'ForrestKnight (ForrestKnight)',
 'revery-ui (revery-ui)',
 'alibaba (Alibaba)',
 'Microsoft (Microsoft)',
 'github (GitHub)',
 'facebook (Facebook)',
 'boazsegev (Bo)',
 'google (Google)',
 'cloudfetch',
 'sindresorhus (SindreSorhus)',
 'tensorflow',
 'apache (TheApacheSoftwareFoundation)',
 'DevonCrawford (DevonCrawford)',
 'ARMmbed (ArmMbed)',
 'vuejs (vuejs)',
 'fastai (fast.ai)',
 'QiShaoXuan (Qi)',
 'joelparkerhenderson (JoelParkerHenderson)',
 'torvalds (LinusTorvalds)',
 'CyC2018',
 'komeiji-satori (   )',
```

```
 'script-8']
```

[4]:
```
# Tu código aquí

developer_links = soup.select("h1:nth-child(1).h3 > a:nth-child(1).Link")
developer_names=[]
for link in developer_links:
    developer_names.append(link.get_text().strip())
developer_names
```

[4]:
```
['Bagheera',
 'Folke Lemaitre',
 'Saúl Ibarra Corretgé',
 'Pontus Abrahamsson',
 'yhirose',
 'yetone',
 'Dan Gohman',
 'Tom Payne',
 'Abubakar Abid',
 'Miles Cranmer',
 'Henry Heng',
 'Victor Berchet',
 'jdx',
 'Bob Nystrom',
 'lauren',
 'Luca Forstner',
 'leogermani',
 'Mark M',
 'tangly1024',
 'Vik Paruchuri',
 'Leonid Bugaev',
 'Sascha Willems',
 'Gal Schlezinger',
 'MelkeyDev',
 'Bo-Yi Wu']
```

Desafío 2 - Mostrar los repositorios de Python en tendencia en GitHub Los pasos para resolver este problema son similares al anterior, excepto que necesitas encontrar los nombres de los repositorios en lugar de los nombres de los desarrolladores.

[5]:
```
# This is the url you will scrape in this exercise
url2 = 'https://github.com/trending/python?since=daily'

datos2 = requests.get(f"{url2}").text
soup2 = BeautifulSoup(datos2, 'html.parser')
```

[6]:
```
# Tu código aquí

repository_links = soup2.select("h2:nth-child(2).h3 > a:nth-child(1).Link")
repository_names = []
for link in repository_links:
    repository_name = link.get_text().strip()
    repository_names.append(repository_name.split("\n")[2].strip())
repository_names
```

[6]:
```
['bbot',
 'langflow',
 'freqtrade',
 'pytorch',
 'gpt4free',
 'poetry',
 'GitHub520',
 'airflow',
 'AutoRAG',
 'dspy',
 'cvat',
 'mlflow',
 'garak',
 'llama-models']
```

Desafío 3 - Mostrar todos los enlaces de imágenes de la página de Wikipedia de Walt Disney

[7]:
```
# Esta es la URL que vas a extraer en este ejercicio
url3 = 'https://en.wikipedia.org/wiki/Walt_Disney'

disney = requests.get(f"{url3}").text
soup3 = BeautifulSoup(disney, 'html.parser')
```

[8]:
```
# Tu código aquí

image_links = soup3.find_all("img")
image_urls = []
for img in image_links:
    if img.has_attr("src"):
        image_urls.append(img["src"])
image_urls
```

[8]:
```
['/static/images/icons/wikipedia.png',
 '/static/images/mobile/copyright/wikipedia-wordmark-en.svg',
 '/static/images/mobile/copyright/wikipedia-tagline-en.svg',
 '//upload.wikimedia.org/wikipedia/en/thumb/e/e7/Cscr-featured.svg/20px-Cscr-
featured.svg.png',
```

'//upload.wikimedia.org/wikipedia/en/thumb/8/8c/Extended-protection-shackle.svg/20px-Extended-protection-shackle.svg.png',
'//upload.wikimedia.org/wikipedia/commons/thumb/d/df/Walt_Disney_1946.JPG/220px-Walt_Disney_1946.JPG',
'//upload.wikimedia.org/wikipedia/commons/thumb/8/87/Walt_Disney_1942_signature.svg/150px-Walt_Disney_1942_signature.svg.png',
'//upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Walt_Disney_Birthplace_Exterior_Hermosa_Chicago_Illinois.jpg/220px-Walt_Disney_Birthplace_Exterior_Hermosa_Chicago_Illinois.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/c/c4/Walt_Disney_envelope_ca._1921.jpg/220px-Walt_Disney_envelope_ca._1921.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/9/95/Walt_Disney_with_Mickey_Mouse_drawing.jpg/209px-Walt_Disney_with_Mickey_Mouse_drawing.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/4/43/Walt_Disney_with_film_roll_and_Mickey_Mouse_on_his_right_arm%2C_year_1935.jpg/196px-Walt_Disney_with_film_roll_and_Mickey_Mouse_on_his_right_arm%2C_year_1935.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/c/cd/Walt_Disney_Snow_white_1937_trailer_screenshot_%2813%29.jpg/220px-Walt_Disney_Snow_white_1937_trailer_screenshot_%2813%29.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/1/15/Disney_drawing_goofy.jpg/170px-Disney_drawing_goofy.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/8/8c/WaltDisneyplansDisneylandDec1954.jpg/220px-WaltDisneyplansDisneylandDec1954.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Walt_Disney_and_Dr._Wernher_von_Braun_-_GPN-2000-000060.jpg/220px-Walt_Disney_and_Dr._Wernher_von_Braun_-_GPN-2000-000060.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/f/ff/Walt_disney_portrait_right.jpg/170px-Walt_disney_portrait_right.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/7/79/Walt_Disney_with_Company_at_Press_Conference.jpg/244px-Walt_Disney_with_Company_at_Press_Conference.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/1/1a/Walt_Disney_Grave.JPG/170px-Walt_Disney_Grave.JPG',
'//upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Nuvola_apps_kaboodle.svg/16px-Nuvola_apps_kaboodle.svg.png',
'//upload.wikimedia.org/wikipedia/commons/thumb/1/13/DisneySchiphol1951.jpg/220px-DisneySchiphol1951.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/6/6c/Disney1968.jpg/170px-Disney1968.jpg',
'//upload.wikimedia.org/wikipedia/commons/thumb/a/a8/Walt_Disney_Receives_Presidential_Medal_of_Freedom_1964.jpg/276px-Walt_Disney_Receives_Presidential_Medal_of_Freedom_1964.jpg',
'//upload.wikimedia.org/wikipedia/en/thumb/4/4a/Commons-logo.svg/20px-Commons-logo.svg.png',
'//upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Wikiquote-logo.svg/23px-Wikiquote-logo.svg.png',
'//upload.wikimedia.org/wikipedia/commons/thumb/4/4c/Wikisource-logo.svg/26px-Wikisource-logo.svg.png',

```
 '//upload.wikimedia.org/wikipedia/en/thumb/8/8a/OOjs_UI_icon_edit-ltr-
progressive.svg/10px-OOjs_UI_icon_edit-ltr-progressive.svg.png',
 '//upload.wikimedia.org/wikipedia/en/thumb/8/8a/OOjs_UI_icon_edit-ltr-
progressive.svg/10px-OOjs_UI_icon_edit-ltr-progressive.svg.png',
 '//upload.wikimedia.org/wikipedia/en/thumb/9/96/Symbol_category_class.svg/16px-
Symbol_category_class.svg.png',
 '//upload.wikimedia.org/wikipedia/commons/thumb/e/e3/Disneyland_Resort_logo.svg
/135px-Disneyland_Resort_logo.svg.png',
 '//upload.wikimedia.org/wikipedia/commons/thumb/d/da/Animation_disc.svg/20px-
Animation_disc.svg.png',
 '//upload.wikimedia.org/wikipedia/en/thumb/6/69/P_vip.svg/19px-P_vip.svg.png',
 '//upload.wikimedia.org/wikipedia/commons/thumb/f/fe/Mickey_Mouse_colored_%28he
ad%29.svg/20px-Mickey_Mouse_colored_%28head%29.svg.png',
 '//upload.wikimedia.org/wikipedia/en/thumb/e/e7/Video-x-generic.svg/19px-Video-
x-generic.svg.png',
 '//upload.wikimedia.org/wikipedia/commons/thumb/a/a3/Flag_of_Los_Angeles_County
%2C_California.svg/21px-Flag_of_Los_Angeles_County%2C_California.svg.png',
 '//upload.wikimedia.org/wikipedia/commons/thumb/8/8c/Blank_television_set.svg/2
1px-Blank_television_set.svg.png',
 '//upload.wikimedia.org/wikipedia/en/thumb/a/a4/Flag_of_the_United_States.svg/2
1px-Flag_of_the_United_States.svg.png',
 '//upload.wikimedia.org/wikipedia/en/thumb/8/8a/OOjs_UI_icon_edit-ltr-
progressive.svg/10px-OOjs_UI_icon_edit-ltr-progressive.svg.png',
 'https://login.wikimedia.org/wiki/Special:CentralAutoLogin/start?type=1x1',
 '/static/images/footer/wikimedia-button.svg',
 '/w/resources/assets/poweredby_mediawiki.svg']
```

Desafío 4 - Recuperar todos los enlaces a páginas en Wikipedia que se refieren a algún tipo de Python.

[9]:
```
# Esta es la URL que vas a extraer en este ejercicio
url4 = 'https://en.wikipedia.org/wiki/Python'

python = requests.get(f"{url4}").text
soup4 = BeautifulSoup(python, 'html.parser')
```

[10]:
```
# Tu código aquí

python_links = soup4.find_all("a", title=lambda title: title and "Python" in␣
 ↪title)

python_pages = []
for link in python_links:
    python_pages.append(link.get("href"))
python_pages
```

```
[10]: ['https://af.wikipedia.org/wiki/Python',
       'https://als.wikipedia.org/wiki/Python',
       'https://az.wikipedia.org/wiki/Python_(d%C9%99qiql%C9%99%C5%9Fdirm%C9%99)',
       'https://be.wikipedia.org/wiki/Python',
       'https://cs.wikipedia.org/wiki/Python_(rozcestn%C3%ADk)',
       'https://da.wikipedia.org/wiki/Python',
       'https://de.wikipedia.org/wiki/Python',
       'https://eu.wikipedia.org/wiki/Python_(argipena)',
       'https://fr.wikipedia.org/wiki/Python',
       'https://hr.wikipedia.org/wiki/Python_(razdvojba)',
       'https://id.wikipedia.org/wiki/Python',
       'https://ia.wikipedia.org/wiki/Python_(disambiguation)',
       'https://is.wikipedia.org/wiki/Python_(a%C3%B0greining)',
       'https://it.wikipedia.org/wiki/Python_(disambigua)',
       'https://la.wikipedia.org/wiki/Python_(discretiva)',
       'https://lb.wikipedia.org/wiki/Python',
      'https://hu.wikipedia.org/wiki/Python_(egy%C3%A9rtelm%C5%B1s%C3%ADt%C5%91_lap)',
       'https://nl.wikipedia.org/wiki/Python',
       'https://pt.wikipedia.org/wiki/Python_(desambigua%C3%A7%C3%A3o)',
       'https://ru.wikipedia.org/wiki/Python_(%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%
      B8%D1%8F)',
       'https://sk.wikipedia.org/wiki/Python',
       'https://sh.wikipedia.org/wiki/Python',
       'https://fi.wikipedia.org/wiki/Python',
       'https://tr.wikipedia.org/wiki/Python_(anlam_ayr%C4%B1m%C4%B1)',
       'https://vi.wikipedia.org/wiki/Python',
       'https://zh.wikipedia.org/wiki/Python_(%E6%B6%88%E6%AD%A7%E4%B9%89)',
       'https://en.wiktionary.org/wiki/Python',
       '/wiki/Pythonidae',
       '/wiki/Python_(genus)',
       '/wiki/Python_(mythology)',
       '/wiki/Python_(programming_language)',
       '/wiki/Python_of_Aenus',
       '/wiki/Python_(painter)',
       '/wiki/Python_of_Byzantium',
       '/wiki/Python_of_Catana',
       '/wiki/Python_Anghelo',
       '/wiki/Python_(Efteling)',
       '/wiki/Python_(Busch_Gardens_Tampa_Bay)',
       '/wiki/Python_(Coney_Island,_Cincinnati,_Ohio)',
       '/wiki/Python_(automobile_maker)',
       '/wiki/Python_(Ford_prototype)',
       '/wiki/Python_(missile)',
       '/wiki/Python_(nuclear_primary)',
       '/wiki/Colt_Python',
       '/wiki/Python_(codename)',
       '/wiki/Python_(film)',
```

```
 '/wiki/Monty_Python',
 '/wiki/Python_(Monty)_Pictures']
```

Desafío 5 - Títulos en el Código de los Estados Unidos desde su lanzamiento

[11]:
```
# This is the url you will scrape in this exercise
url = 'http://uscode.house.gov/download/download.shtml'

html = requests.get(url).text
soup5 = BeautifulSoup(html, 'html.parser')
```

[12]:
```
# Tu código aquí
titles_divs = soup5.select('div.usctitle[id^="us/usc/"]')

titles = []

for row in titles_divs:
    titles.append(row.text.strip())
titles
```

[12]:
```
['Title 1 - General Provisions  ',
 'Title 2 - The Congress',
 'Title 3 - The President  ',
 'Title 4 - Flag and Seal, Seat of Government, and the States  ',
 'Title 5 - Government Organization and Employees  ',
 'Title 6 - Domestic Security',
 'Title 7 - Agriculture',
 'Title 8 - Aliens and Nationality',
 'Title 9 - Arbitration  ',
 'Title 10 - Armed Forces  ',
 'Title 11 - Bankruptcy  ',
 'Title 12 - Banks and Banking',
 'Title 13 - Census  ',
 'Title 14 - Coast Guard  ',
 'Title 15 - Commerce and Trade',
 'Title 16 - Conservation',
 'Title 17 - Copyrights  ',
 'Title 18 - Crimes and Criminal Procedure  ',
 'Title 19 - Customs Duties',
 'Title 20 - Education',
 'Title 21 - Food and Drugs',
 'Title 22 - Foreign Relations and Intercourse',
 'Title 23 - Highways  ',
 'Title 24 - Hospitals and Asylums',
 'Title 25 - Indians',
 'Title 26 - Internal Revenue Code',
 'Title 27 - Intoxicating Liquors',
```

```
 'Title 28 - Judiciary and Judicial Procedure  ',
 'Title 29 - Labor',
 'Title 30 - Mineral Lands and Mining',
 'Title 31 - Money and Finance  ',
 'Title 32 - National Guard  ',
 'Title 33 - Navigation and Navigable Waters',
 'Title 34 - Crime Control and Law Enforcement',
 'Title 35 - Patents  ',
 'Title 36 - Patriotic and National Observances, Ceremonies, and Organizations
',
 'Title 37 - Pay and Allowances of the Uniformed Services  ',
 "Title 38 - Veterans' Benefits " ,
 'Title 39 - Postal Service  ',
 'Title 40 - Public Buildings, Property, and Works  ',
 'Title 41 - Public Contracts  ',
 'Title 42 - The Public Health and Welfare',
 'Title 43 - Public Lands',
 'Title 44 - Public Printing and Documents  ',
 'Title 45 - Railroads',
 'Title 46 - Shipping  ',
 'Title 47 - Telecommunications',
 'Title 48 - Territories and Insular Possessions',
 'Title 49 - Transportation  ',
 'Title 50 - War and National Defense',
 'Title 51 - National and Commercial Space Programs  ',
 'Title 53 [Reserved]',
 'Title 54 - National Park Service and Related Programs  '[
```

Desafío 6 - Una lista y un DataFrame Python con los diez nombres más buscados por el FBI

[13]:
```python
# Esta es la URL que vas a extraer en este ejercicio
# url7 = 'https://www.fbi.gov/wanted/topten'
url7 = 'https://en.wikipedia.org/wiki/FBI_Ten_Most_Wanted_Fugitives'

wanted = requests.get(f"{url7}").text
soup7 = BeautifulSoup(wanted, 'html.parser')
```

[14]:
```python
# Tu código aquí
wantedtag = soup7.find_all('h3')
wantedtable = soup7.find("tbody")
wantedrows = wantedtable.find_all("div", class_="center")
wantes = []

for i in range(0, len(wantedrows), 3):
    name = wantedrows[i].get_text()
    date = wantedrows[i + 1].get_text()
```

```
    seq_number = wantedrows[i + 2].get_text()
    wantes.append([name, date, seq_number])
wantes
```

[14]:
```
[['Alexis Flores', 'June 2, 2007', '487'],
 ['Bhadreshkumar Chetanbhai Patel', 'April 18, 2017', '514'],
 ['Alejandro Castillo', 'October 24, 2017', '516'],
 ['Arnoldo Jimenez', 'May 8, 2019', '522'],
 ['Yulan Adonay Archaga Carias', 'November 3, 2021', '526'],
 ['Ruja Ignatova', 'June 30, 2022', '527'],
 ['Omar Alexander Cardenas', 'July 20, 2022', '528'],
 ['Wilver Villegas-Palomino', 'April 14, 2023', '530'],
 ['Donald Eugene Fields II', 'May 25, 2023', '531'],
 ["Vitel'Homme Innocent", 'November 15, 2023', '532']]
```

[15]:
```
df = pd.DataFrame(wantes, columns=["Name", "Date added", "Sequence number"])
df
```

[15]:

	Name	Date added	Sequence number
0	Alexis Flores	June 2, 2007	487
1	Bhadreshkumar Chetanbhai Patel	April 18, 2017	514
2	Alejandro Castillo	October 24, 2017	516
3	Arnoldo Jimenez	May 8, 2019	522
4	Yulan Adonay Archaga Carias	November 3, 2021	526
5	Ruja Ignatova	June 30, 2022	527
6	Omar Alexander Cardenas	July 20, 2022	528
7	Wilver Villegas-Palomino	April 14, 2023	530
8	Donald Eugene Fields II	May 25, 2023	531
9	Vitel'Homme Innocent	November 15, 2023	532

Desafío 7 - Listar todos los nombres de idiomas y el número de artículos relacionados en el orden en que aparecen en wikipedia.org

[16]:
```
# Esta es la URL que vas a extraer en este ejercicio
url8 = 'https://www.wikipedia.org/'

languages = requests.get(f"{url8}").content
soup8 = BeautifulSoup(languages, 'html.parser')
```

[17]:
```
# Tu código aquí

langlist = soup8.find_all("div", {"class": f"central-featured-lang"})

language_articles = []

for lang in langlist:
    language = lang.find("strong").text.strip()
```

```
    articles = ''.join(filter(str.isdigit, lang.find("small").text.strip()))
    language_articles.append((language, articles))

language_articles
```

[17]:
```
[('English', '6902000'),
 ('    ', '2006000'),
 ('  ', '1434000'),
 ('Español', '1986000'),
 ('Deutsch', '2954000'),
 ('Français', '2643000'),
 (' ', '1448000'),
 ('Italiano', '1888000'),
 ('Português', '1136000'),
 ('   ', '    '[(
```

Desafío 8 - Una lista con los diferentes tipos de conjuntos de datos disponibles en data.gov.uk

[18]:
```
# Esta es la URL que vas a extraer en este ejercicio
url82 = 'https://data.gov.uk/'

dats = requests.get(f"{url82}")
soup8 = BeautifulSoup(dats.content, 'html.parser')
```

[19]:
```
# Tu código aquí

dataset_links = soup8.select("h3:nth-child(1) > a:nth-child(1)")
dataset_types=[]
for link in dataset_links:
    dataset_types.append(link.text.strip())
dataset_types
```

[19]:
```
['Business and economy',
 'Crime and justice',
 'Defence',
 'Education',
 'Environment',
 'Government',
 'Government spending',
 'Health',
 'Mapping',
 'Society',
 'Towns and cities',
 'Transport',
 'Digital service performance',
 'Government reference data']
```

Desafío 9 - Una lista y un DataFrame con los 10 idiomas con más hablantes nativos almacenados en un DataFrame de Pandas

[20]:
```
# Esta es la URL que vas a extraer en este ejercicio
url9 = 'https://en.wikipedia.org/wiki/
 ↪List_of_languages_by_number_of_native_speakers'

tenlang = requests.get(url9)
soup9 = BeautifulSoup(tenlang.content, "html.parser")
```

[21]:
```
# Tu código aquí

langtable = soup9.find("table", class_="wikitable")
langrows = langtable.find_all("tr")
langs = []
for row in langrows[1:11]:
    col=row.find_all("td")
    lang=col[0].text.strip()
    speakers=col[1].text.strip()
    langfamily=col[2].text.strip()
    branch=col[3].text.strip()
    langs.append([lang,speakers,langfamily,branch])

langs
```

[21]:
```
[['Mandarin Chinese', '941', 'Sino-Tibetan', 'Sinitic'],
 ['Spanish', '486', 'Indo-European', 'Romance'],
 ['English', '380', 'Indo-European', 'Germanic'],
 ['Hindi', '345', 'Indo-European', 'Indo-Aryan'],
 ['Bengali', '237', 'Indo-European', 'Indo-Aryan'],
 ['Portuguese', '236', 'Indo-European', 'Romance'],
 ['Russian', '148', 'Indo-European', 'Balto-Slavic'],
 ['Japanese', '123', 'Japonic', 'Japanese'],
 ['Yue Chinese', '86', 'Sino-Tibetan', 'Sinitic'],
 ['Vietnamese', '85', 'Austroasiatic', 'Vietic']]
```

[22]:
```
df = pd.DataFrame(langs, columns=["Language", "Native Speakers", "Language␣
 ↪Family", "Branch"])
df
```

[22]:

	Language	Native Speakers	Language Family	Branch
0	Mandarin Chinese	941	Sino-Tibetan	Sinitic
1	Spanish	486	Indo-European	Romance
2	English	380	Indo-European	Germanic
3	Hindi	345	Indo-European	Indo-Aryan
4	Bengali	237	Indo-European	Indo-Aryan
5	Portuguese	236	Indo-European	Romance
6	Russian	148	Indo-European	Balto-Slavic

```
7          Japanese              123         Japonic      Japanese
8       Yue Chinese               86    Sino-Tibetan       Sinitic
9        Vietnamese               85   Austroasiatic        Vietic
```

1.0.2 Subiendo el nivel

Desafío 10 - La información de los 20 últimos terremotos (fecha, hora, latitud, longitud y nombre de la región) como un DataFrame de pandas

[23]:
```
# Esta es la URL que vas a extraer en este ejercicio
url7 = 'http://ds.iris.edu/seismon/eventlist/index.phtml'

response = requests.get(url7)
soup10 = BeautifulSoup(response.content, "html.parser")
```

[24]:
```
# Tu código aquí

eqtable = soup10.find("table", class_="tablesorter")
eqrows = eqtable.find_all("tr")

earthquakes = []

for row in eqrows[1:21]:
    col=row.find_all("td")
    date =col[0].text.strip()
    lat =col[1].text.strip()
    lon =col[2].text.strip()
    mag=col[3].text.strip()
    depth=col[4].text.strip()
    location=col[5].text.strip()
    earthquakes.append([date,lat,lon,mag,depth,location])

earthquakes;
```

[25]:
```
df = pd.DataFrame(earthquakes, columns=["Date-Time", "Lat", "Lon",␣
 ↪"Mag","Depth","Location"])
df
```

[25]:

	Date-Time	Lat	Lon	Mag	Depth	\
0	21-NOV-2024 04:59:15	29.31	51.30	4.9	10	
1	21-NOV-2024 02:23:43	8.03	-82.88	4.8	10	
2	21-NOV-2024 00:26:47	-23.10	-69.07	4.2	96	
3	20-NOV-2024 21:19:33	-6.53	130.46	4.1	148	
4	20-NOV-2024 20:56:30	16.74	-100.42	4.3	10	
5	20-NOV-2024 18:44:16	-17.22	-174.34	5.1	200	
6	20-NOV-2024 15:09:02	46.43	136.13	5.1	407	
7	20-NOV-2024 14:55:41	18.19	145.47	4.9	282	
8	20-NOV-2024 14:43:53	79.81	1.49	5.7	14	

```
9   20-NOV-2024 10:42:57  -20.73  -176.27  5.5   228
10  20-NOV-2024 10:24:03   53.09   159.84  4.1    63
11  20-NOV-2024 09:26:28  -12.12   165.74  5.3    10
12  20-NOV-2024 08:09:50   38.03   141.86  4.9    63
13  20-NOV-2024 08:09:00   -7.28   120.17  4.4   453
14  20-NOV-2024 06:40:15   41.05   140.98  4.9    10
15  20-NOV-2024 04:03:44  -57.44    -7.75  5.0    10
16  20-NOV-2024 01:40:54   44.58   150.55  4.7    10
17  19-NOV-2024 20:50:44   -8.86   122.05  4.6   106
18  19-NOV-2024 18:36:11   34.08   138.38  5.0   249
19  19-NOV-2024 18:06:19  -37.90   -73.37  4.0    40

                          Location
0                     SOUTHERN IRAN
1    PANAMA-COSTA RICA BORDER REGION
2                    NORTHERN CHILE
3                         BANDA SEA
4     NEAR COAST OF GUERRERO, MEXICO
5                      TONGA ISLANDS
6                  PRIMOR'YE, RUSSIA
7                    MARIANA ISLANDS
8                      GREENLAND SEA
9                FIJI ISLANDS REGION
10      NEAR EAST COAST OF KAMCHATKA
11                SANTA CRUZ ISLANDS
12  NEAR EAST COAST OF HONSHU, JAPAN
13                       FLORES SEA
14            HOKKAIDO, JAPAN REGION
15    EAST OF SOUTH SANDWICH ISLANDS
16             EAST OF KURIL ISLANDS
17          FLORES REGION, INDONESIA
18    NEAR S. COAST OF HONSHU, JAPAN
19       NEAR COAST OF CENTRAL CHILE
```

Desafío 11 - Datos del Top de peliculas (nombre de la película, lanzamiento inicial, lugar de origen, nombre del director y puntaje) como un DataFrame de pandas

[26]:
```
# Esta es la URL que vas a extraer en este ejercicio
url11 = 'https://www.filmaffinity.com/es/userlist.php?
 ↪user_id=882839&list_id=1006'

pelis = requests.get(url11)
soup11 = BeautifulSoup(pelis.content, "html.parser")
```

[27]:
```
# Tu código aquí

movieslist = soup11.find("ul", class_="fa-list-group mx-md-2")
```

```
moviesrows = movieslist.find_all("li")

movies=[]

for row in moviesrows:
    title = row.find("a", class_="d-none d-md-inline-block").get_text().strip()
    year = row.find("span", class_="mc-year ms-1").text.strip()
    country = row.find("img", class_="nflag")["alt"].strip()
    director = row.find_all("span", class_="nb")[0].text.strip()
    rating = row.find("div", class_="avg mx-0").text.strip()
    movies.append([title,year,country,director,rating])

movies;
```

[28]:
```
df = pd.DataFrame(movies, columns=["Title", "Year", "Country",␣
 ↪"Director","Rating"])
df
```

[28]:

	Title	Year	Country
0	12 hombres sin piedad	1957	Estados Unidos
1	El padrino	1972	Estados Unidos
2	El padrino. Parte II	1974	Estados Unidos
3	La guerra de las galaxias. Episodio V: El impe…	1980	Estados Unidos
4	La guerra de las galaxias. Episodio IV: Una nu…	1977	Estados Unidos
5	La guerra de las galaxias. Episodio VI: El ret…	1983	Estados Unidos
6	Indiana Jones y la última cruzada	1989	Estados Unidos
7	En busca del arca perdida	1981	Estados Unidos
8	Casablanca	1942	Estados Unidos
9	Seven	1995	Estados Unidos
10	Forrest Gump	1994	Estados Unidos
11	Salvar al soldado Ryan	1998	Estados Unidos
12	El viaje de Chihiro	2001	Japón
13	El castillo ambulante	2004	Japón
14	La princesa Mononoke	1997	Japón
15	Érase una vez en América	1984	Estados Unidos
16	Gran Torino	2008	Estados Unidos
17	Toy Story	1995	Estados Unidos
18	Cadena perpetua	1994	Estados Unidos
19	Pulp Fiction	1994	Estados Unidos
20	Ratatouille	2007	Estados Unidos
21	El señor de los anillos: Las dos torres	2002	Nueva Zelanda
22	El señor de los anillos: El retorno del rey	2003	Nueva Zelanda
23	El señor de los anillos: La comunidad del anillo	2001	Nueva Zelanda
24	Piratas del Caribe: La maldición de la Perla N…	2003	Estados Unidos
25	El bueno, el feo y el malo	1966	Italia
26	Origen	2010	Estados Unidos
27	Matrix	1999	Estados Unidos

28	Regreso al futuro	1985	Estados Unidos
29	El show de Truman	1998	Estados Unidos
30	El pianista	2002	Reino Unido
31	La lista de Schindler	1993	Estados Unidos
32	La bella y la bestia	1991	Estados Unidos
33	Shutter Island	2010	Estados Unidos
34	Star Trek	2009	Estados Unidos
35	Kill Bill. Volumen 1	2003	Estados Unidos
36	Reservoir Dogs	1992	Estados Unidos
37	Django desencadenado	2012	Estados Unidos
38	Malditos bastardos	2009	Estados Unidos
39	El club de la lucha	1999	Estados Unidos

	Director	Rating
0	Sidney Lumet	8,7
1	Francis Ford Coppola	9,0
2	Francis Ford Coppola	8,9
3	Irvin Kershner	8,1
4	George Lucas	7,9
5	Richard Marquand	7,9
6	Steven Spielberg	7,8
7	Steven Spielberg	7,8
8	Michael Curtiz	8,4
9	David Fincher	8,3
10	Robert Zemeckis	8,2
11	Steven Spielberg	7,8
12	Hayao Miyazaki	8,1
13	Hayao Miyazaki	7,8
14	Hayao Miyazaki	8,0
15	Sergio Leone	8,3
16	Clint Eastwood	8,2
17	John Lasseter	7,7
18	Frank Darabont	8,6
19	Quentin Tarantino	8,6
20	Brad Bird	7,3
21	Peter Jackson	8,0
22	Peter Jackson	8,2
23	Peter Jackson	8,0
24	Gore Verbinski	7,2
25	Sergio Leone	8,2
26	Christopher Nolan	8,0
27	Lilly Wachowski,	7,9
28	Robert Zemeckis	7,5
29	Peter Weir	7,7
30	Roman Polanski	8,2
31	Steven Spielberg	8,6
32	Gary Trousdale,	7,3

```
33       Martin Scorsese    7,6
34           J.J. Abrams    6,8
35     Quentin Tarantino    7,6
36     Quentin Tarantino    8,1
37     Quentin Tarantino    7,9
38     Quentin Tarantino    7,8
39         David Fincher    8,1
```

Desafío 12 - Nombre de la película, año, lugar, director, raiting, cast, raiting usuarios y rating count de las 10 películas aleatorias como un DataFrame de pandas.

[29]:
```
# Esta es la URL que vas a extraer en este ejercicio
url11 = 'https://www.filmaffinity.com/es/userlist.php?
 ↪user_id=882839&list_id=1006'

pelis = requests.get(url11)
soup11 = BeautifulSoup(pelis.content, "html.parser")
```

[30]:
```
# Tu código aquí

import random

movieslist = soup11.find("ul", class_="fa-list-group mx-md-2")
moviesrows = movieslist.find_all("li")
moviesrows = random.sample(moviesrows, 10)

movies = []

for row in moviesrows:
    title = row.find("a", class_="d-none d-md-inline-block").get_text().strip()
    year = row.find("span", class_="mc-year ms-1").text.strip()
    country = row.find("img", class_="nflag")["alt"].strip()
    director = row.find_all("span", class_="nb")[0].text.strip()
    rating = row.find("div", class_="avg mx-0").text.strip()
    cast = [actor.text.strip() for actor in row.find_all("span", class_="nb")[1:
 ↪]]
    ratmovieusrs = row.find("div", class_="fa-user-rat-box not-me ms-auto").
 ↪text.strip()
    ratcount = row.find("div", class_="count text-nowrap ms-2").text.strip()
    movies.append([title, year, country, director, rating, cast, ratmovieusrs,␣
 ↪ratcount])

movies;
```

[31]:
```
df = pd.DataFrame(movies, columns=["Title", "Year", "Country",␣
 ↪"Director","Rating", "Cast", "RatMovieUsrs", "RatCount"])
df
```

[31]:

	Title	Year	Country	Director	Rating	Cast	RatMovieUsrs	RatCount
0	La princesa Mononoke	1997	Japón	Hayao Miyazaki	8,0	[]	8	71.240
1	12 hombres sin piedad	1957	Estados Unidos	Sidney Lumet	8,7	[Henry Fonda,, Lee J. Cobb,, Jack Warden,, E.G…	9	72.500
2	Gran Torino	2008	Estados Unidos	Clint Eastwood	8,2	[Clint Eastwood,, Christopher Carley,, Bee Van…	8	149.039
3	El show de Truman	1998	Estados Unidos	Peter Weir	7,7	[Jim Carrey,, Laura Linney,, Noah Emmerich,, E…	6	127.100
4	Matrix	1999	Estados Unidos	Lilly Wachowski,	7,9	[Lana Wachowski,, Hermanas Wachowski, Keanu Re…	7	197.723
5	Salvar al soldado Ryan	1998	Estados Unidos	Steven Spielberg	7,8	[Tom Hanks,, Tom Sizemore,, Edward Burns,, Mat…	6	142.852
6	Cadena perpetua	1994	Estados Unidos	Frank Darabont	8,6	[Tim Robbins,, Morgan Freeman,, Bob Gunton,, J…	9	171.112
7	Shutter Island	2010	Estados Unidos	Martin Scorsese	7,6	[Leonardo DiCaprio,, Mark Ruffalo,, Ben Kingsl…	7	127.728
8	El castillo ambulante	2004	Japón	Hayao Miyazaki	7,8	[]	8	61.454
9	Toy Story	1995	Estados Unidos	John Lasseter	7,7	[Tom Hanks,, Tim Allen,, Don Rickles]	7	160.686

Desafío 13 - Encontrar el reporte meteorológico en vivo (temperatura, velocidad del viento, descripción y clima) de una ciudad dada.

[32]:
```
# Esta es la URL que vas a extraer en este ejercicio
# https://openweathermap.org/current

# Usa la API oficial con tu clave API real y con tu ciudad
```

[33]:
```
# Tu código aquí

def obtener_clima(api_key, ciudad):
    url_base = "http://api.openweathermap.org/data/2.5/weather?"
    nombre_ciudad = ciudad
    url_completa = f"{url_base}appid={api_key}&q={nombre_ciudad}"
    respuesta = requests.get(url_completa)
    return respuesta.json()

# Reemplaza 'YOUR_API_KEY' con tu clave API real y 'CITY_NAME' con tu ciudad
api_key = '9da0f0082cf35d67520a646b8f2ff3a7'
nombre_ciudad = 'Barcelona'
datos_clima = obtener_clima(api_key, nombre_ciudad)

print(f"Clima en {nombre_ciudad}:")
```

```
print(datos_clima)
```

```
Clima en Barcelona:
{'coord': {'lon': 2.159, 'lat': 41.3888}, 'weather': [{'id': 802, 'main':
'Clouds', 'description': 'scattered clouds', 'icon': '03d'}], 'base':
'stations', 'main': {'temp': 286.86, 'feels_like': 285.98, 'temp_min': 285.73,
'temp_max': 287.41, 'pressure': 1011, 'humidity': 65, 'sea_level': 1011,
'grnd_level': 1003}, 'visibility': 10000, 'wind': {'speed': 0.45, 'deg': 311,
'gust': 2.24}, 'clouds': {'all': 48}, 'dt': 1732172236, 'sys': {'type': 2, 'id':
18549, 'country': 'ES', 'sunrise': 1732171643, 'sunset': 1732206456},
'timezone': 3600, 'id': 3128760, 'name': 'Barcelona', 'cod': 200}
```

Desafío 14 - Nombre del libro, precio y disponibilidad de stock como un DataFrame de pandas.

[34]:
```
# Esta es la URL que vas a extraer en este ejercicio.
# Es una librería ficticia creada para ser extraída.
url14 = 'http://books.toscrape.com'
```

[35]:
```
# Tu código aquí

books = []

for i in range(1, 51):
    url = f'http://books.toscrape.com/catalogue/page-{i}.html'
    response = requests.get(url)
    soup = BeautifulSoup(response.content, "html.parser")

    libros = soup.select('h3:nth-child(3) > a:nth-child(1)')
    precios = soup.select('div:nth-child(4) > p:nth-child(1)')
    stocks= soup.select('div:nth-child(4) > p:nth-child(2)')

    for libro, precio, stock in zip(libros, precios, stocks):
        books.append([libro['title'], precio.text.strip(), stock.text.strip()])

books;
```

[36]:
```
df = pd.DataFrame(books, columns=["Boock", "Price", "Stock"])
df
```

[36]:

	Boock	Price	Stock
0	A Light in the Attic	£51.77	In stock
1	Tipping the Velvet	£53.74	In stock
2	Soumission	£50.10	In stock
3	Sharp Objects	£47.82	In stock
4	Sapiens: A Brief History of Humankind	£54.23	In stock
..	…	…	…
995	Alice in Wonderland (Alice's Adventures in Won…	£55.53	In stock

```
996   Ajin: Demi-Human, Volume 1 (Ajin: Demi-Human #1)  £57.06  In stock
997  A Spy's Devotion (The Regency Spies of London #1)  £16.97  In stock
998               1st to Die (Women's Murder Club #1)  £53.98  In stock
999                1,000 Places to See Before You Die  £26.08  In stock

[1000 rows x 3 columns]
```