CHALLENGE 1

CHALLENGE 1

**Download, parse (using BeautifulSoup), and print the content from the Trending Developers page from GitHub:**

In [ ]:



*# This is the url you will scrape in this exercise*

url **=** 'https://github.com/trending/developers'

import requests

from bs4 import BeautifulSoup

url = 'https://github.com/trending/developers'

# Descargar la página

html = requests.get(url).tex

# Verificar si la solicitud fue exitosa (código de estado 200)

if html.status\_code == 200:

# Analizar el contenido HTML con BeautifulSoup

soup = BeautifulSoup(html, 'html.parser') #response=es la variable que almacena la solicitud, text= formato texto, html.parser= Indica que se debe utilizar el analizador HTML predeterminado incluido en BeautifulSoup para analizar el contenido HTML.

# Encontrar y extraer el contenido deseado

content = soup.find('div', class\_='Box-row')

# Imprimir el contenido

print(content)

else:

print('Error al obtener la página:', response.status\_code)

#### Display the names of the trending developers retrieved in the previous step.

Your output should be a Python list of developer names. Each name should not contain any html tag.

**Instructions:**

**url = 'https://github.com/trending/developers'**

**html = requests.get(url)**

**if response.status\_code == 200:**

**soup = BeautifulSoup(.text, 'html.parser')**

**# Encuentra todos los elementos que contienen los nombres de los desarrolladores**

**developer\_elements = soup.find\_all('h1', class\_='h3 lh-condensed')**

**# Extrae y limpia los nombres de los desarrolladores**

**developer\_names = [element.text.strip() for element in developer\_elements]**

**# Imprime la lista de nombres**

**print(developer\_names)**

**else:**

**print('Error al obtener la página:', html.status\_code)**

#### Display the trending Python repositories in GitHub

The steps to solve this problem is similar to the previous one except that you need to find out the repository names instead of developer names.

url = 'https://github.com/trending/python?since=daily'

html = requests.get(url)

if html.status\_code == 200:

soup = BeautifulSoup(html.text, 'html.parser')

# Encuentra todos los elementos que contienen los nombres de los repositorios

repository\_elements = soup.find\_all('h1', class\_='h3 lh-condensed')

# Extrae y limpia los nombres de los repositorios

repository\_names = [element.text.strip() for element in repository\_elements]

# Imprime la lista de nombres

print(repository\_names)

#### Display all the image links from Walt Disney wikipedia page

#### url = 'https://en.wikipedia.org/wiki/Walt\_Disney'

#### html = requests.get(url)

#### if html.status\_code == 200:

#### soup = BeautifulSoup(html.text, 'html.parser')

#### # Encuentra todos los elementos <img=imagen> en la página

#### image\_elements = soup.find\_all('img')

#### # Extrae la URL de cada imagen

#### image\_links = [element['src'] for element in image\_elements]

#### # Imprime las URLs de las imágenes

#### for link in image\_links:

#### print(link)

#### else:

#### print('Error al obtener la página:', html.status\_code)

#### Retrieve an arbitary Wikipedia page of "Python" and create a list of links on that page

url = 'https://es.wikipedia.org/wiki/Python'

html = requests.get(url)

if html.status\_code == 200:

soup = BeautifulSoup(html.text, 'html.parser')

#Cada elemento <a> contiene un atributo href que especifica la URL de destino del enlace. Al extraer este atributo de cada elemento, puedes obtener la URL del enlace y trabajar con ella, como en el ejemplo anterior donde se creó una lista de enlaces.

enlace\_elements = soup.find\_all('a')

# Extrae el atributo href de cada <a> para obtener la URL del enlace

enlaces = [enlace['href'] for enlace in enlace\_elements]

# Filtra las URL para los enlaces internos de Wikipedia

enlaces\_wikipedia = [enlace for enlace in enlaces if enlace.startswith('/wiki/')]

# Imprime los enlaces

for enlace in enlaces\_wikipedia:

print(enlace)

else:

print('Error al obtener la página:', html.status\_code)

#### Number of Titles that have changed in the United States Code since its last release point

import requests

from bs4 import BeautifulSoup

url = 'http://uscode.house.gov/download/download.shtml'

html = requests.get(url)

if html.status\_code == 200:

soup = BeautifulSoup(html.text, 'html.parser')

# Encuentra el elemento que contiene la información sobre los cambios en el título.

# div es una etiqueta HTML utilizada para crear contenedores o secciones, y en este código se utiliza como una forma de encontrar elementos específicos dentro del HTML utilizando BeautifulSoup.

#Es importante destacar que el atributo id debe ser único en una página HTML, lo que significa que solo debe haber un único elemento con ese valor de id

div\_cambios = soup.find('div', id='usctitlechanged')

# Extrae el texto del elemento

texto\_cambios = div\_cambios.get\_text()

# Analiza el texto para determinar la cantidad de títulos que han cambiado

titulos\_cambiados = len(texto\_cambios.split(','))

# Imprime la cantidad de títulos cambiados

print("Número de títulos cambiados:", titulos\_cambiados)

else:

print('Error al obtener la página:', html.status\_code)

#### A Python list with the top ten FBI's Most Wanted names

url = 'https://www.fbi.gov/wanted/topten'

html = requests.get(url)

if html.status\_code == 200:

soup = BeautifulSoup(html.text, 'html.parser')

# Encuentra el contenedor que contiene los nombres de los más buscados

# utilizando soup.find('ul', class\_='wanted-person-list')

contenedor\_nombres = soup.find('ul', class\_='wanted-person-list')

# Encuentra todos los elementos de la lista dentro del contenedor utilizando contenedor\_nombres.find\_all('li')

elementos\_lista = contenedor\_nombres.find\_all('li')

# Extrae los nombres de los elementos de la lista utilizando el método get\_text() y los guarda en una lista llamada nombres\_mas\_buscados, que luego se imprime en la consola.

nombres\_mas\_buscados = [elemento.get\_text().strip() for elemento in elementos\_lista]

# Imprime la lista de nombres

for nombre in nombres\_mas\_buscados:

print(nombre)

else:

print('Error al obtener la página:', html.status\_code)

#### 20 latest earthquakes info (date, time, latitude, longitude and region name) by the EMSC as a pandas dataframe

url = 'https://www.emsc-csem.org/Earthquake/'

html = requests.get(url)

if html.status\_code == 200:

soup = BeautifulSoup(html.text, 'html.parser')

# Encuentra la tabla que contiene la información de los terremotos

tabla\_terremotos = soup.find('table')

# Encuentra todas las filas en la tabla

filas = tabla\_terremotos.find\_all('tr')

# Extrae los datos de cada fila y almacénalos en una lista de diccionarios

datos\_terremotos = []

for fila in filas:

columna = fila.find\_all('td')

fecha = columna[3].get\_text()

hora = columna[4].get\_text()

latitud = columna[5].get\_text()

longitud = columna[6].get\_text()

region = columna[11].get\_text()

datos\_terremotos.append({

'Fecha': fecha,

'Hora': hora,

'Latitud': latitud,

'Longitud': longitud,

'Región': region

})

# Crea un DataFrame a partir de la lista de diccionarios

df\_terremotos = pd.DataFrame(datos\_terremotos)

# Imprime el DataFrame

print(df\_terremotos)

else:

print('Error al obtener la página:', html.status\_code)

#### Display the date, days, title, city, country of next 25 hackathon events as a Pandas dataframe table

url = 'https://hackevents.co/hackathons'

html = requests.get(url)

if html.status\_code == 200:

soup = BeautifulSoup(html.text, 'html.parser')

# Encuentra la tabla que contiene la información de los eventos

tabla\_eventos = soup.find('table')

# Encuentra todas las filas en la tabla

filas = tabla\_eventos.find\_all('tr')

# Extrae los datos de cada fila y almacénalos en una lista de diccionarios

datos\_eventos = []

for fila in filas:

columna = fila.find\_all('td')

if len(columna) == 5:

fecha = columna[0].get\_text().strip()

dias = columna[1].get\_text().strip()

titulo = columna[2].get\_text().strip()

ciudad = columna[3].get\_text().strip()

pais = columna[4].get\_text().strip()

datos\_eventos.append({

'Fecha': fecha,

'Días': dias,

'Título': titulo,

'Ciudad': ciudad,

'País': pais

})

# Crea un DataFrame a partir de la lista de diccionarios

df\_eventos = pd.DataFrame(datos\_eventos)

# Imprime el DataFrame

print(df\_eventos.head(25))

else:

print('Error al obtener la página:', html.status\_code)

#### Count number of tweets by a given Twitter account.

You will need to include a ***try/except block*** for account names not found.  
***Hint:*** the program should count the number of tweets for any provided account

import tweepy

#Ten en cuenta que necesitarás tener una cuenta de desarrollador de Twitter y las credenciales de la API para acceder a la

#API de Twitter mediante tweepy. Además, ten en cuenta que hay un límite de solicitudes a la API de Twitter, así que ten

#cuidado al hacer muchas consultas seguidas.

# Agrega tus credenciales de la API de Twitter

#eemplazar TU\_CONSUMER\_KEY, TU\_CONSUMER\_SECRET, TU\_ACCESS\_TOKEN, TU\_ACCESS\_TOKEN\_SECRET y NOMBRE\_DE\_LA\_CUENTA con tus propias

#credenciales de la API de Twitter y el nombre de la cuenta de Twitter que deseas contar los tweets.

consumer\_key = 'TU\_CONSUMER\_KEY'

consumer\_secret = 'TU\_CONSUMER\_SECRET'

access\_token = 'TU\_ACCESS\_TOKEN'

access\_token\_secret = 'TU\_ACCESS\_TOKEN\_SECRET'

# Configura las credenciales de autenticación

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

# Crea una instancia de la API

api = tweepy.API(auth)

# Nombre de la cuenta de Twitter que deseas contar los tweets

nombre\_cuenta = 'NOMBRE\_DE\_LA\_CUENTA'

try:

# Obtiene el objeto del usuario

usuario = api.get\_user(screen\_name=nombre\_cuenta)

# Obtiene el número de tweets del usuario

num\_tweets = usuario.statuses\_count

print(f"El número de tweets de @{nombre\_cuenta} es: {num\_tweets}")

except tweepy.error.TweepError:

print(f"La cuenta @{nombre\_cuenta} no fue encontrada.")

#### Number of followers of a given twitter account

You will need to include a ***try/except block*** in case account/s name not found.  
***Hint:*** the program should count the followers for any provided account

import tweepy

# Agrega tus credenciales de la API de Twitter

consumer\_key = 'TU\_CONSUMER\_KEY'

consumer\_secret = 'TU\_CONSUMER\_SECRET'

access\_token = 'TU\_ACCESS\_TOKEN'

access\_token\_secret = 'TU\_ACCESS\_TOKEN\_SECRET'

# Configura las credenciales de autenticación

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

# Crea una instancia de la API

api = tweepy.API(auth)

# Nombre de la cuenta de Twitter de la que deseas contar los seguidores

nombre\_cuenta = 'NOMBRE\_DE\_LA\_CUENTA'

try:

# Obtiene el objeto del usuario

usuario = api.get\_user(screen\_name=nombre\_cuenta)

# Obtiene el número de seguidores del usuario

num\_seguidores = usuario.followers\_count

print(f"El número de seguidores de @{nombre\_cuenta} es: {num\_seguidores}")

except tweepy.error.TweepError:

print(f"La cuenta @{nombre\_cuenta} no fue encontrada.")

#### List all language names and number of related articles in the order they appear in wikipedia.org

import requests

from bs4 import BeautifulSoup

# URL de la página de inicio de Wikipedia

url = 'https://www.wikipedia.org/'

# Realiza una solicitud GET a la URL

response = requests.get(url)

if response.status\_code == 200:

# Crea un objeto BeautifulSoup para el contenido HTML

soup = BeautifulSoup(response.text, 'html.parser')

# encuentra todas las etiquetas <span> con la clase CSS "central-featured-lang" y las almacena en la variable lang\_spans.

lang\_spans = soup.find\_all('span', class\_='central-featured-lang')

# Itera sobre las etiquetas span y extrae el nombre del idioma y el número de artículos

for lang\_span in lang\_spans:

lang\_name = lang\_span.get\_text(strip=True)

#lang\_span.get\_text(strip=True) extrae el texto dentro de la etiqueta <span>, eliminando cualquier espacio en blanco

#redundante gracias al argumento strip=True. Esto nos da el nombre del idioma.

num\_articles = lang\_span.find('bdi').get\_text(strip=True)

#lang\_span.find('bdi') busca la etiqueta <bdi> dentro de la etiqueta <span>. La etiqueta <bdi> es donde se encuentra

#el número de artículos relacionados.

print(f"Idioma: {lang\_name}, Número de artículos: {num\_articles}")

else:

print('Error al obtener la página:', response.status\_code)

#### A list with the different kind of datasets available in data.gov.uk

url = 'https://data.gov.uk/'

# Realiza una solicitud GET a la URL

response = requests.get(url)

if response.status\_code == 200:

# Crea un objeto BeautifulSoup para el contenido HTML

soup = BeautifulSoup(response.text, 'html.parser')

#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR#SUSTITUIR

# Encuentra todos los elementos <a> que contienen enlaces en la página (links webs)

enlaces = soup.find\_all('a')

# Recorre los enlaces y muestra su texto y URL

for enlace in enlaces:

texto = enlace.text

url = enlace['href']

print(f"Texto: {texto}, URL: {url}")

else:

print('Error al obtener la página:', response.status\_code)

#### Top 10 languages by number of native speakers stored in a Pandas Dataframe[¶](http://localhost:8888/notebooks/Desktop/Ironhack/4.2-lab_web_scraping/your-code/main.ipynb" \l "Top-10-languages-by-number-of-native-speakers-stored-in-a-Pandas-Dataframe)

url = 'https://en.wikipedia.org/wiki/List\_of\_languages\_by\_number\_of\_native\_speakers'

response = requests.get(url)

if response.status\_code == 200:

# Crea un objeto BeautifulSoup para el contenido HTML

soup = BeautifulSoup(response.text, 'html.parser')

# Encuentra la tabla con la lista de idiomas

tabla = soup.find('table', class\_='wikitable')

# Crea una lista vacía para almacenar los datos de los idiomas

datos\_idiomas = []

# Itera sobre las filas de la tabla, omitiendo la primera fila de encabezados

for fila in tabla.find\_all('tr')[1:]:

columnas = fila.find\_all('td')

nombre\_idioma = columnas[1].text.strip()

hablantes\_nativos = columnas[2].text.strip()

datos\_idiomas.append({'Idioma': nombre\_idioma, 'Hablantes Nativos': hablantes\_nativos})

# Crea el dataframe con los datos

df\_idiomas = pd.DataFrame(datos\_idiomas)

# Ordena por la columna 'Hablantes Nativos' de forma descendente

df\_idiomas = df\_idiomas.sort\_values(by='Hablantes Nativos', ascending=False)

# Toma los 10 primeros idiomas

df\_top\_10\_idiomas = df\_idiomas.head(10)

# Imprime el dataframe resultante

print(df\_top\_10\_idiomas)

else:

print('Error al obtener la página:', response.status\_code)

#### IMDB's Top 250 data (movie name, Initial release, director name and stars) as a pandas dataframe

url = 'https://www.imdb.com/chart/top'

html = requests.get(url)

if html.status\_code == 200:

# Crea un objeto BeautifulSoup para el contenido HTML

soup = BeautifulSoup(html.text, 'html.parser')

# Encuentra la tabla que contiene la información de las películas

tabla = soup.find('table', class\_='chart')

# Crea listas vacías para almacenar los datos de las películas

nombres = []

anios = []

directores = []

actores = []

# Itera sobre las filas de la tabla, omitiendo la primera fila de encabezados

for fila in tabla.find\_all('tr')[1:]:

columnas = fila.find\_all('td')

nombre = columnas[1].find('a').text

anio = columnas[1].find('span', class\_='secondaryInfo').text.strip('()')

director = columnas[1].find('a', attrs={'title': 'Director'}).text

actores\_principales = [a.text for a in columnas[1].find\_all('a', attrs={'title': 'See full cast & crew'})]

nombres.append(nombre)

anios.append(anio)

directores.append(director)

actores.append(actores\_principales)

# Crea el dataframe con los datos

df\_peliculas = pd.DataFrame({'Nombre': nombres, 'Año de Lanzamiento': anios,

'Director': directores, 'Actores principales': actores})

# Imprime el dataframe resultante

print(df\_peliculas)

else:

print('Error al obtener la página:', response.status\_code)

#### Movie name, year and a brief summary of the top 10 random movies (IMDB) as a pandas dataframe.

url = 'http://www.imdb.com/chart/top'

response = requests.get(url)

if response.status\_code == 200:

# Crea un objeto BeautifulSoup para el contenido HTML

soup = BeautifulSoup(response.text, 'html.parser')

# Encuentra la lista de películas de la tabla

#En el caso de IMDb, la lista de películas se encuentra dentro del elemento <tbody> con la clase CSS lister-list.

#Por lo tanto, al utilizar BeautifulSoup para raspar la página, seleccionamos el elemento <tbody> con esa clase específica

#para obtener la lista de películas.

lista\_peliculas = soup.find('tbody', class\_='lister-list')

# Obtiene las filas de la lista de películas

filas = lista\_peliculas.find\_all('tr')

# Selecciona 10 películas al azar

peliculas\_aleatorias = random.sample(filas, 10)

# Crea listas vacías para almacenar los datos

nombres = []

year = []

resumenes = []

# Itera sobre las películas seleccionadas

for pelicula in peliculas\_aleatorias:

# Encuentra el nombre de la película

nombre = pelicula.find('td', class\_='titleColumn').find('a').text

nombres.append(nombre)

# Encuentra el año de lanzamiento de la película

year = pelicula.find('span', class\_='secondaryInfo').text.strip('()')

year.append(year)

# Encuentra el resumen de la película

resumen = pelicula.find('td', class\_='titleColumn').find('span', class\_='runtime').next\_sibling.strip()

resumenes.append(resumen)

# Crea el dataframe con los datos

df\_peliculas = pd.DataFrame({'Nombre': nombres, 'Año de Lanzamiento': year, 'Resumen': resumenes})

# Imprime el dataframe resultante

print(df\_peliculas)

else:

print('Error al obtener la página:', response.status\_code)

#### Book name,price and stock availability as a pandas dataframe.[¶](http://localhost:8888/notebooks/Desktop/Ironhack/4.2-lab_web_scraping/your-code/main.ipynb#Book-name,price-and-stock-availability-as-a-pandas-dataframe.)

url = 'http://books.toscrape.com/'

# Realiza una solicitud GET a la URL

html = requests.get(url)

if response.status\_code == 200:

# Crea un objeto BeautifulSoup para el contenido HTML

soup = BeautifulSoup(html.content, 'html.parser')

# Encuentra todos los contenedores de los libros en la página

libros = soup.find\_all('article', class\_='product\_pod')

# Crea listas vacías para almacenar los datos

nombres = []

precios = []

disponibilidades = []

# Itera sobre los contenedores de los libros

for libro in libros:

# Encuentra el nombre del libro

nombre = libro.find('h3').find('a')['title']

nombres.append(nombre)

# Encuentra el precio del libro

precio = libro.find('p', class\_='price\_color').text.strip('£')

precios.append(precio)

# Encuentra la disponibilidad de stock del libro

disponibilidad = libro.find('p', class\_='instock availability').text.strip()

disponibilidades.append(disponibilidad)

# Crea el dataframe con los datos

df\_libros = pd.DataFrame({'Nombre del Libro': nombres, 'Precio': precios, 'Disponibilidad': disponibilidades})

# Imprime el dataframe resultante

print(df\_libros)

else:

print('Error al obtener la página:', response.status\_code)