Web scrapping for extracting School Details

Web scraping is a technique used to extract information from websites. Imagine visiting a website and copying the information you need into a document; web scraping automates this process. Accessing the HTML of the webpage and extracting useful information/data from it. This technique is called web scrapping.

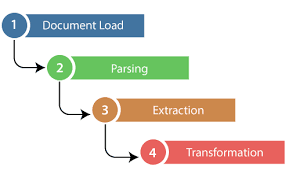
**Main 4 Steps Involved in Web Scrapping:**

**Requests:** Firstly we need to send the HTTP request to the URL of the webpage we need to access. The server responds to the request by returning the Raw HTML content of the webpage. For extraction we need to use HTTP library for python i.e., Requests.This request library has inbuilt functionalities for managing both the request and response.

**Beautifulsoup Library:**Beautifulsoup is a library of python which is useful in navigating, searching, and modifying a parse tree. This automatically converts incoming documents to unicode and outgoing documents to UTF-8.

**Pandas:** Later after extracting the data we need to store the data into structured format by using python library called pandas.

**Structured Data:** For the final step we need to store the structured data into Excel , csv, json etc.



**Extracting the School URLs of the School:**

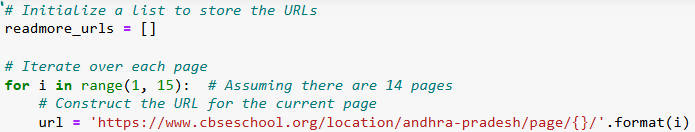
**Step 1 : Choose the Website and Webpage URL:**

To extract the school details from the website , I have selected the website : cbse school, which consists of all the school details in chronological order .

**Website Url :** https://www.cbseschool.org/location/

Firstly, I have choose to extract the school details urls from one state at a time. The url is used to extract school details urls from different pages. The url goes as follows

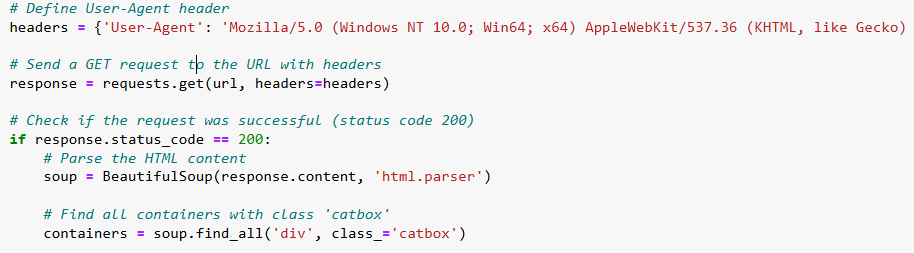
**url = 'https://www.cbseschool.org/location/state-name/page/{}/'.format(i)** here we can change the state name accordingly and pages are iterated using "i".



**Step 2: Inspect the Website :**

To extract the data we need to know which elements are used in html code. The main two elements are tags and class. Right click on the website to select "Inspect". This will open the HTML code. Use the inspector tool to see the name of all the elements that are used in the code.

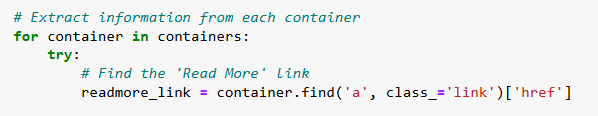
At the end of each container we have readmore (yellow highlight) which is redirected to another webpage which has all the required details of the individual schools.

https://static.zohocdn.com/writer3/images/spacer.ed280a0ea3cc38f3cbbc747acfbef47d.gif 

The HTML elements with the class "catbox" and tag "div" have the readmore links url data.

**Step 3: Looping through each containers:**

To extract multiple containers containing school details, we need to iterate through each containers having the tag "a" and class "link". The URL from the href attribute is extracted and appended to readmore\_url list.

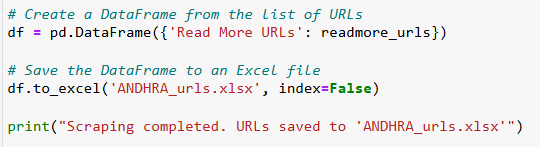


**Step 4: Looping through each pages:**

* A for loop is written for the iteration process through **pages 1 to n pages**. After inspecting we have **tag "div" and class "catbox"** contains desired information.
* After getting into current webpage , a dictionary Headers defines a User-Agent which avoids the error and security of the page by mimicking the webpage. We need to check the response code , which need to 200 for successfully entering into the webpage.
* Later the library BeautifulSoup ("response.content","html.parser") parses the downloaded HTML content. "**soup.find\_all**" will retrieve the all elements with the corresponding class.
* Later it iterates through each containers in the current page and extracted data is stored.
* I have used try and exception method which handles potential errors during the link extraction. This try and exception method helps in continuing the process without interruption.

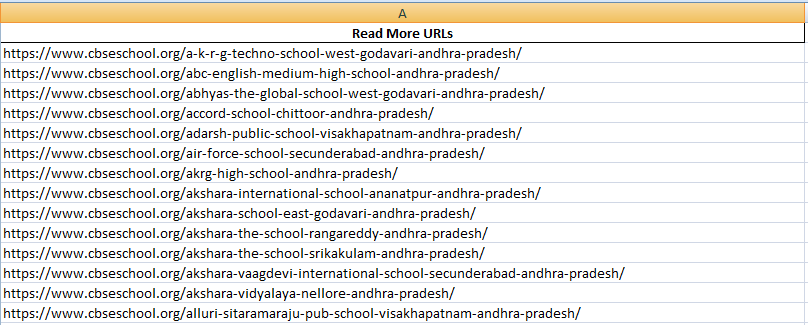
**Step 5: Data Oraganization with Pandas:**

A Dataframe is created from readmore\_url list with a single column named "Read More URLs" and this dataframe is saved as an excel file name with etension ".xlsx".



I have selected this process for extracting urls of each schools in each state separately.

**Results:**



Next step,

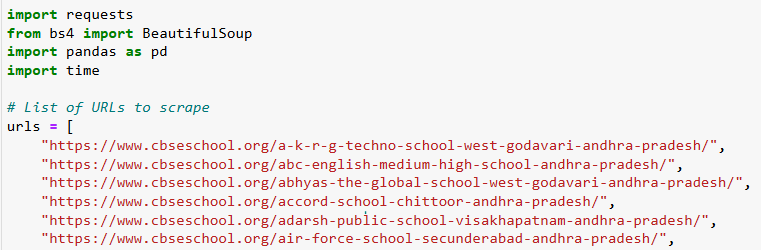
**Extracting School Details by Using URLs of each state:**

We need to extract school details from the URLs extracted earlier. The process of doing that is by sending urls one by one and with help of request and beautiful soup libraries  we can send requests and response. The data is extracted from each URLs after successful entering to the webpage. Following is the code explanation :

**Iterating through URLs:**

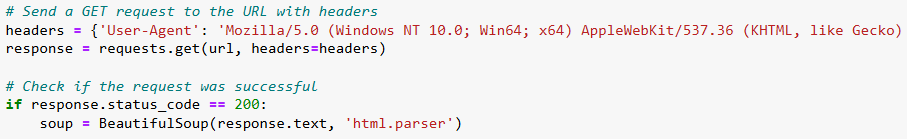
Firstly we need to do is iterating the urls , for which we can write a for loop. The code iterates through each URL in the urls list given.

Inside the loop we need to send request to the webpage for extraction and checking the status code whether we are getting 200 or any other number.



**Request and Status code:**

* THis code sends a GET request to the current URL with the defined headers containing User-Agent string.
* The response status code is returned as 200 then the status is success otherwise error meassage will print indicating the failed URL and status code.

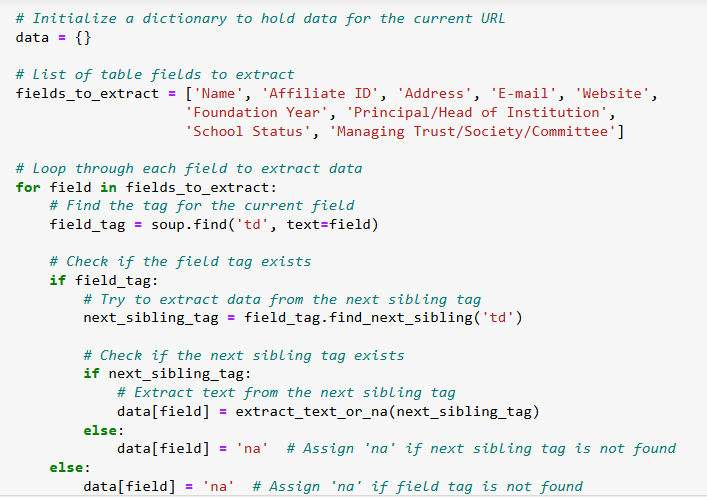


**Parse HTML:**

BeautifulSoup is used for parsing the downloaded HTML content into a beautifulsoup object for easier navigation. Next we need to define the fields to extract according to our requirement. Here I wanted Name, Affiliated ID, Address, Email id, Website link, principal Name, committee/Trust Name.

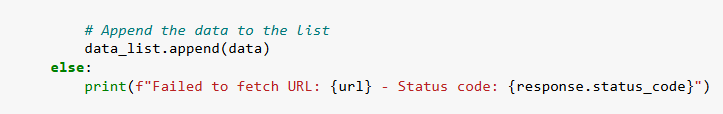
**Looping through Fields:**

* this loop iterates each field in the "fields to extract " list , After inspecting in HTML code of webpage, we need to extract data which is stored in table. The table has the tag as td and class text=field.
* soup.find searches for the table cell containing text of current field name. If  it is found it tries to find the next field and to extract data we use extract\_text\_or\_na function is used and send the data to "data" dictionary and if it is not found "na" is assisgned in missing data.



**Append Data:**

* THe complete "data" dictionary containing the extracted information for the current url is appended to the "data\_list".



**Delay TIme:**

We need to use time.sleep(1) statement as the urls crowds and overwhelms the process . So to avoid, we use this statement which delays the requests for one second.



**Error Handling:**

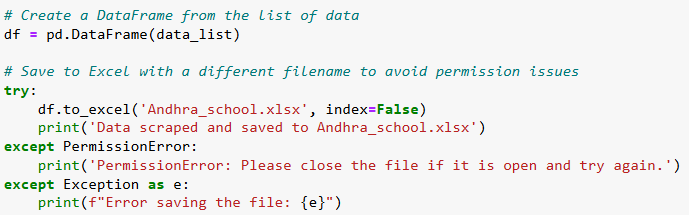
To maintain the continous process in the extraction, I have used error handling for capturing exceptions that might occur during the process and print a error message indicating the URL.

**Creating the Data frame using Pandas:**

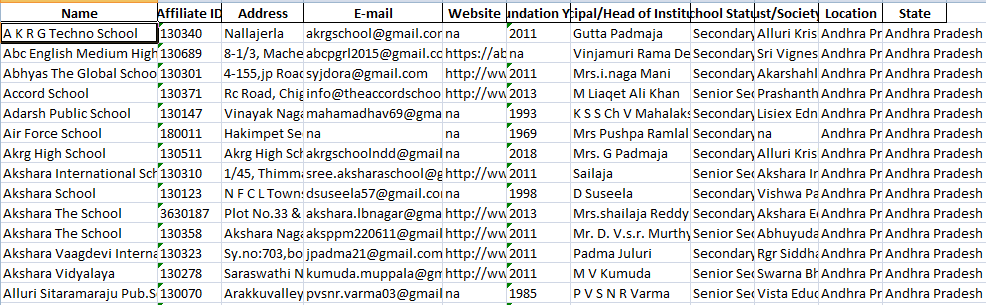
We need to create a structured data from the Data\_list dictionary and make it as dataframe.

**Storing the Data:**

After the structured data is created we need to store the data into excel, csv, json etc to utilise.



**Result:**



**This code is used for extracting data from different states by changing the url given.**

**Below is the code for Url extraction:**

**# Initialize a list to store the URLs**

**readmore\_urls = []**

**# Iterate over each page**

**for i in range(1, 15): # Assuming there are 14 pages**

**# Construct the URL for the current page**

**url = 'https://www.cbseschool.org/location/andhra-pradesh/page/{}/'.format(i)**

**# Define User-Agent header**

**headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}**

**# Send a GET request to the URL with headers**

**response = requests.get(url, headers=headers)**

**# Check if the request was successful (status code 200)**

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

**# Parse the HTML content**

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

**# Find all containers with class 'catbox'**

**containers = soup.find\_all('div', class\_='catbox')**

**# Extract information from each container**

**for container in containers:**

**try:**

**# Find the 'Read More' link**

**readmore\_link = container.find('a', class\_='link')['href']**

**# Append the URL to the list**

**readmore\_urls.append(readmore\_link)**

**except Exception as e:**

**print("Error:", e)**

**# Print progress message**

**print("Scraped page", i)**

**# Create a DataFrame from the list of URLs**

**df = pd.DataFrame({'Read More URLs': readmore\_urls})**

**# Save the DataFrame to an Excel file**

**df.to\_excel('ANDHRA\_urls.xlsx', index=False)**

**print("Scraping completed. URLs saved to 'ANDHRA\_urls.xlsx'")**

**Below is the code for Details of School from URLs extracted:**

import requests

from bs4 import BeautifulSoup

import pandas as pd

import time

# List of URLs to scrape

urls = [

"https://www.cbseschool.org/a-k-r-g-techno-school-west-godavari-andhra-pradesh/",

"https://www.cbseschool.org/abc-english-medium-high-school-andhra-pradesh/",

"https://www.cbseschool.org/abhyas-the-global-school-west-godavari-andhra-pradesh/",

"https://www.cbseschool.org/accord-school-chittoor-andhra-pradesh/",

write rest of url here

]

# Function to extract text or return 'na'

def extract\_text\_or\_na(tag):

return tag.get\_text(strip=True) if tag else 'na'

# List to hold data for each URL

data\_list = []

# Loop through each URL

for url in urls:

try:

# Send a GET request to the URL with headers

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}

response = requests.get(url, headers=headers)

# Check if the request was successful

if response.status\_code == 200:

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

# Initialize a dictionary to hold data for the current URL

data = {}

# List of table fields to extract

fields\_to\_extract = ['Name', 'Affiliate ID', 'Address', 'E-mail', 'Website',

'Foundation Year', 'Principal/Head of Institution',

'School Status', 'Managing Trust/Society/Committee']

# Loop through each field to extract data

for field in fields\_to\_extract:

# Find the tag for the current field

field\_tag = soup.find('td', text=field)

# Check if the field tag exists

if field\_tag:

# Try to extract data from the next sibling tag

next\_sibling\_tag = field\_tag.find\_next\_sibling('td')

# Check if the next sibling tag exists

if next\_sibling\_tag:

# Extract text from the next sibling tag

data[field] = extract\_text\_or\_na(next\_sibling\_tag)

else:

data[field] = 'na' # Assign 'na' if next sibling tag is not found

else:

data[field] = 'na' # Assign 'na' if field tag is not found

# Extract location

location\_tag = soup.find('a', rel='tag')

data['Location'] = extract\_text\_or\_na(location\_tag) if location\_tag else 'na'

# Add State

data['State'] = 'Andhra Pradesh' # Hardcoded as per the given data

# Append the data to the list

data\_list.append(data)

else:

print(f"Failed to fetch URL: {url} - Status code: {response.status\_code}")

# Introduce a delay between requests

time.sleep(1)

except Exception as e:

print(f"Error processing URL: {url} - {e}")

# Create a DataFrame from the list of data

df = pd.DataFrame(data\_list)

# Save to Excel with a different filename to avoid permission issues

try:

df.to\_excel('Andhra\_school.xlsx', index=False)

print('Data scraped and saved to Andhra\_school.xlsx')

except PermissionError:

print('PermissionError: Please close the file if it is open and try again.')

except Exception as e:

print(f"Error saving the file: {e}")