Hands-on Activity 7.2 Webscraping using BeautifulSoup and Requests

Name: Bautista, Jhon Hendricks
Section: CPE22S3

Performed on: 04/09/2025
Submitted on: 04/11/2025
Submitted to: Engr. Roman M. Richard

Data Gathering

Sources of Data

A vast amount of historical data can be found in files such as:

- MS Word documents
- Emails
- Spreadsheets
- MS PowerPoints
- PDFs
- HTML
- and plaintext files

Public and Private Archives

CSV, JSON, and XML files use plaintext, a common format, and are compatible with a wide range of applications The Web can be mined for data using a web scraping application The IoT uses sensors create data Sensors in smartphones, cars, airplanes, street lamps, and home appliances capture raw data

Open Data and Private Data

1. Open Data

The Open Knowledge Foundation describes Open Data as “any content, information or data that people are free to use, reuse, and redistribute without any legal, technological, or social restriction."

2. . Private Data

Data related to an expectation of privacy and regulated by a particular country/government

Structured and Unstructured Data

1. Structured Data Data entered and maintained in fixed fields within a file or record Easily entered, classified, queried, and analyzed Relational databases or spreadsheets

2. Unstructured Data Lacks organization

   Raw data Photo contents, audio, video, web pages, blogs, books, journals, white papers, PowerPoint presentations, articles, email, wikis, word processing documents, and text in general

Example of gathering image data using webcam

In [272…

```
import cv2
from google.colab.patches import cv2_imshow

key = cv2.waitKey(1)
webcam = cv2.VideoCapture(0)

while True:
    try:
        check, frame = webcam.read()
        print(check)  # prints true as long as the webcam is running
        print(frame)  # prints matrix values of each frame
        cv2.imshow("Capturing", frame)

        key = cv2.waitKey(1)

        if key == ord('s'):
            cv2.imwrite(filename='saved_img.jpg', img=frame)
            webcam.release()
            img_new = cv2.imread('saved_img.jpg', cv2.IMREAD_GRAYSCALE)
            cv2.imshow("Captured Image", img_new)
            cv2.waitKey(1650)
            cv2.destroyAllWindows()
            print("Processing image...")

            img_ = cv2.imread('saved_img.jpg', cv2.IMREAD_ANYCOLOR)
            print("Converting RGB image to grayscale...")
            gray = cv2.cvtColor(img_, cv2.COLOR_BGR2GRAY)
            print("Converted RGB image to grayscale...")

            print("Resizing image to 28x28 scale...")
            img_ = cv2.resize(gray, (28, 28))
            print("Resized...")

            img_resized = cv2.imwrite(filename='saved_img-final.jpg', img=img_)
            print("Image saved!")
```

```
            break
        elif key == ord('q'):
            print("Turning off camera.")
            webcam.release()
            print("Camera off.")
            print("Program ended.")
            cv2.destroyAllWindows()
            break

    except KeyboardInterrupt:
        print("Turning off camera.")
        webcam.release()
        print("Camera off.")
        print("Program ended.")
        cv2.destroyAllWindows()
        break
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[272], line 1
----> 1 import cv2
      2 from google.colab.patches import cv2_imshow
      4 key = cv2.waitKey(1)

ModuleNotFoundError: No module named 'cv2'
```

Example of gathering voice data using microphone

In [82]:
```
!pip3 install sounddevice
```

```
Collecting sounddevice
  Downloading sounddevice-0.5.1-py3-none-win_amd64.whl.metadata (1.4 kB)
Requirement already satisfied: CFFI>=1.0 in c:\users\win10\anaconda3\lib\site-packag
es (from sounddevice) (1.17.1)
Requirement already satisfied: pycparser in c:\users\win10\anaconda3\lib\site-packag
es (from CFFI>=1.0->sounddevice) (2.21)
Downloading sounddevice-0.5.1-py3-none-win_amd64.whl (363 kB)
Installing collected packages: sounddevice
Successfully installed sounddevice-0.5.1
```

In [84]:
```
!pip3 install wavio
```

```
Collecting wavio
  Downloading wavio-0.0.9-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: numpy>=1.19.0 in c:\users\win10\anaconda3\lib\site-pa
ckages (from wavio) (1.26.4)
Downloading wavio-0.0.9-py3-none-any.whl (9.5 kB)
Installing collected packages: wavio
Successfully installed wavio-0.0.9
```

In [86]:
```
!pip3 install scipy
```

```
Requirement already satisfied: scipy in c:\users\win10\anaconda3\lib\site-packages
(1.13.1)
Requirement already satisfied: numpy<2.3,>=1.22.4 in c:\users\win10\anaconda3\lib\si
te-packages (from scipy) (1.26.4)
```

In [110…
```
! apt-get install libportaudio2
```

```
ERROR: Could not find a version that satisfies the requirement libportaudio2 (from v
ersions: none)
ERROR: No matching distribution found for libportaudio2
```

In [104…
```
!pip install PyAudio
```

```
Collecting PyAudio
  Downloading PyAudio-0.2.14-cp312-cp312-win_amd64.whl.metadata (2.7 kB)
Downloading PyAudio-0.2.14-cp312-cp312-win_amd64.whl (164 kB)
Installing collected packages: PyAudio
Successfully installed PyAudio-0.2.14
```

In [115…
```
# import required libraries
import sounddevice as sd
from scipy.io.wavfile import write
import wavio as wv

# Sampling frequency
freq = 44100

# Recording duration
duration = 5

# Start recorder with the given values
# of duration and sample frequency
recording = sd.rec(int(duration * freq),
                   samplerate=freq, channels=2)

# Record audio for the given number of seconds
sd.wait()

# This will convert the NumPy array to an audio
# file with the given sampling frequency
write("recording0.wav", freq, recording)

# Convert the NumPy array to audio file
wv.write("recording1.wav", recording, freq, sampwidth=2)
```

Web Scraping

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. The web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying in which specific data is gathered and

copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis.

Image Scraping using BeautifulSoup and Request

In [33]:
```
!pip install bs4
```

```
Requirement already satisfied: bs4 in c:\users\win10\anaconda3\lib\site-packages (0.
0.2)
Requirement already satisfied: beautifulsoup4 in c:\users\win10\anaconda3\lib\site-p
ackages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\win10\anaconda3\lib\site-pa
ckages (from beautifulsoup4->bs4) (2.5)
```

In [94]:
```
!pip install requests
```

```
Requirement already satisfied: requests in c:\users\win10\anaconda3\lib\site-package
s (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\win10\anaconda3
\lib\site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\win10\anaconda3\lib\site-pac
kages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\win10\anaconda3\lib\si
te-packages (from requests) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\win10\anaconda3\lib\si
te-packages (from requests) (2025.1.31)
```

In [95]:
```
import requests
from bs4 import BeautifulSoup

def getdata(url):
    r = requests.get(url)
    return r.text

htmldata = getdata("https://www.google.com/")
soup = BeautifulSoup(htmldata, 'html.parser')
for item in soup.find_all('img'):
    print(item['src'])
```

```
/images/branding/googlelogo/1x/googlelogo_white_background_color_272x92dp.png
```

In [98]:
```
!pip install selenium
```

```
Collecting selenium
  Downloading selenium-4.31.0-py3-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: urllib3<3,>=1.26 in c:\users\win10\anaconda3\lib\site
-packages (from urllib3[socks]<3,>=1.26->selenium) (2.2.3)
Collecting trio~=0.17 (from selenium)
  Downloading trio-0.29.0-py3-none-any.whl.metadata (8.5 kB)
Collecting trio-websocket~=0.9 (from selenium)
  Downloading trio_websocket-0.12.2-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\win10\anaconda3\lib\si
te-packages (from selenium) (2025.1.31)
Requirement already satisfied: typing_extensions~=4.9 in c:\users\win10\anaconda3\li
b\site-packages (from selenium) (4.11.0)
Requirement already satisfied: websocket-client~=1.8 in c:\users\win10\anaconda3\lib
\site-packages (from selenium) (1.8.0)
Collecting attrs>=23.2.0 (from trio~=0.17->selenium)
  Downloading attrs-25.3.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: sortedcontainers in c:\users\win10\anaconda3\lib\site
-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: idna in c:\users\win10\anaconda3\lib\site-packages (f
rom trio~=0.17->selenium) (3.7)
Collecting outcome (from trio~=0.17->selenium)
  Downloading outcome-1.3.0.post0-py2.py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\win10\anaconda3\lib\site-p
ackages (from trio~=0.17->selenium) (1.3.0)
Requirement already satisfied: cffi>=1.14 in c:\users\win10\anaconda3\lib\site-packa
ges (from trio~=0.17->selenium) (1.17.1)
Collecting wsproto>=0.14 (from trio-websocket~=0.9->selenium)
  Downloading wsproto-1.2.0-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in c:\users\win10\anacond
a3\lib\site-packages (from urllib3[socks]<3,>=1.26->selenium) (1.7.1)
Requirement already satisfied: pycparser in c:\users\win10\anaconda3\lib\site-packag
es (from cffi>=1.14->trio~=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\win10\anaconda3\lib\site-pa
ckages (from wsproto>=0.14->trio-websocket~=0.9->selenium) (0.14.0)
Downloading selenium-4.31.0-py3-none-any.whl (9.4 MB)
   ---------------------------------------- 0.0/9.4 MB ? eta -:--:--
   ---------------------------------------- 9.4/9.4 MB 83.4 MB/s eta 0:00:00
Downloading trio-0.29.0-py3-none-any.whl (492 kB)
Downloading trio_websocket-0.12.2-py3-none-any.whl (21 kB)
Downloading attrs-25.3.0-py3-none-any.whl (63 kB)
Downloading outcome-1.3.0.post0-py2.py3-none-any.whl (10 kB)
Downloading wsproto-1.2.0-py3-none-any.whl (24 kB)
Installing collected packages: wsproto, attrs, outcome, trio, trio-websocket, seleni
um
  Attempting uninstall: attrs
    Found existing installation: attrs 23.1.0
    Uninstalling attrs-23.1.0:
      Successfully uninstalled attrs-23.1.0
Successfully installed attrs-25.3.0 outcome-1.3.0.post0 selenium-4.31.0 trio-0.29.0
trio-websocket-0.12.2 wsproto-1.2.0
```

Image Scraping using Selenium

In [100…

```
!pip install selenium
!apt-get update # to update ubuntu to correctly run apt install
```

```
!apt install chromium-chromedriver
!cp /usr/lib/chromium-browser/chromedriver /usr/bin
import sys
sys.path.insert(0,'/usr/lib/chromium-browser/chromedriver')
from selenium import webdriver
import time
import requests
import shutil
import os
import getpass
import urllib.request
import io
import time
from PIL import Image
user = getpass.getuser()
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome('chromedriver',chrome_options=chrome_options)
search_url = "https://www.google.com/search?q={q}&tbm=isch&tbs=sur%3Afc&hl=en&ved=
driver.get(search_url.format(q='Car'))
def scroll_to_end(driver):
   driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
   time.sleep(5)#sleep_between_interactions

def getImageUrls(name,totalImgs,driver):

   search_url = "https://www.google.com/search?q={q}&tbm=isch&tbs=sur%3Afc&hl=en&v
   driver.get(search_url.format(q=name))
   img_urls = set()
   img_count = 0
   results_start = 0

   while(img_count<totalImgs): #Extract actual images now

       scroll_to_end(driver)

       thumbnail_results = driver.find_elements_by_xpath("//img[contains(@class,'Q
       totalResults=len(thumbnail_results)
       print(f"Found: {totalResults} search results. Extracting links from{results

       for img in thumbnail_results[results_start:totalResults]:

           img.click()
           time.sleep(2)
           actual_images = driver.find_elements_by_css_selector('img.n3VNCb')
           for actual_image in actual_images:
               if actual_image.get_attribute('src') and 'https' in actual_image.ge
                   img_urls.add(actual_image.get_attribute('src'))

           img_count=len(img_urls)

           if img_count >= totalImgs:
               print(f"Found: {img_count} image links")
               break
```

```
            else:
                print("Found:", img_count, "looking for more image links ...")
                load_more_button = driver.find_element_by_css_selector(".mye4qd")
                driver.execute_script("document.querySelector('.mye4qd').click();")
                results_start = len(thumbnail_results)
    return img_urls
 def downloadImages(folder_path,file_name,url):
    try:

        image_content = requests.get(url).content
    except Exception as e:
        print(f"ERROR - COULD NOT DOWNLOAD {url} - {e}")
    try:
        image_file = io.BytesIO(image_content)
        image = Image.open(image_file).convert('RGB')

        file_path = os.path.join(folder_path, file_name)

        with open(file_path, 'wb') as f:
            image.save(f, "JPEG", quality=85)
        print(f"SAVED - {url} - AT: {file_path}")
    except Exception as e:
        print(f"ERROR - COULD NOT SAVE {url} - {e}")


def saveInDestFolder(searchNames,destDir,totalImgs,driver):
    for name in list(searchNames):
        path=os.path.join(destDir,name)
        if not os.path.isdir(path):
            os.mkdir(path)
        print('Current Path',path)
        totalLinks=getImageUrls(name,totalImgs,driver)
        print('totalLinks',totalLinks)
    if totalLinks is None:
            print('images not found for :',name)

    else:
        for i, link in enumerate(totalLinks):
            file_name = f"{i:150}.jpg"
            downloadImages(path,file_name,link)

searchNames=['cat']
destDir=f'/content/drive/My Drive/Colab Notebooks/Dataset/'
totalImgs=5
saveInDestFolder(searchNames,destDir,totalImgs,driver)
```

```
Requirement already satisfied: selenium in c:\users\win10\anaconda3\lib\site-package
s (4.31.0)
Requirement already satisfied: urllib3<3,>=1.26 in c:\users\win10\anaconda3\lib\site
-packages (from urllib3[socks]<3,>=1.26->selenium) (2.2.3)
Requirement already satisfied: trio~=0.17 in c:\users\win10\anaconda3\lib\site-packa
ges (from selenium) (0.29.0)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\win10\anaconda3\lib\s
ite-packages (from selenium) (0.12.2)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\win10\anaconda3\lib\si
te-packages (from selenium) (2025.1.31)
Requirement already satisfied: typing_extensions~=4.9 in c:\users\win10\anaconda3\li
b\site-packages (from selenium) (4.11.0)
Requirement already satisfied: websocket-client~=1.8 in c:\users\win10\anaconda3\lib
\site-packages (from selenium) (1.8.0)
Requirement already satisfied: attrs>=23.2.0 in c:\users\win10\anaconda3\lib\site-pa
ckages (from trio~=0.17->selenium) (25.3.0)
Requirement already satisfied: sortedcontainers in c:\users\win10\anaconda3\lib\site
-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: idna in c:\users\win10\anaconda3\lib\site-packages (f
rom trio~=0.17->selenium) (3.7)
Requirement already satisfied: outcome in c:\users\win10\anaconda3\lib\site-packages
(from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\win10\anaconda3\lib\site-p
ackages (from trio~=0.17->selenium) (1.3.0)
Requirement already satisfied: cffi>=1.14 in c:\users\win10\anaconda3\lib\site-packa
ges (from trio~=0.17->selenium) (1.17.1)
Requirement already satisfied: wsproto>=0.14 in c:\users\win10\anaconda3\lib\site-pa
ckages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in c:\users\win10\anacond
a3\lib\site-packages (from urllib3[socks]<3,>=1.26->selenium) (1.7.1)
Requirement already satisfied: pycparser in c:\users\win10\anaconda3\lib\site-packag
es (from cffi>=1.14->trio~=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\win10\anaconda3\lib\site-pa
ckages (from wsproto>=0.14->trio-websocket~=0.9->selenium) (0.14.0)
```

```
'apt-get' is not recognized as an internal or external command,
operable program or batch file.
'apt' is not recognized as an internal or external command,
operable program or batch file.
'cp' is not recognized as an internal or external command,
operable program or batch file.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[100], line 22
     20 chrome_options.add_argument('--no-sandbox')
     21 chrome_options.add_argument('--disable-dev-shm-usage')
---> 22 driver = webdriver.Chrome('chromedriver',chrome_options=chrome_options)
     23 search_url = "https://www.google.com/search?q={q}&tbm=isch&tbs=sur%3Afc&hl=e
n&ved=0CAIQpwVqFwoTCKCa1c6s4-oCFQAAAAAdAAAAABAC&biw=1251&bih=568"
     24 driver.get(search_url.format(q='Car'))

TypeError: WebDriver.__init__() got an unexpected keyword argument 'chrome_options'
```

Web Scraping of Movies Information using BeautifulSoup

We want to analyze the distributions of IMDB and Metacritic movie ratings to see if we find anything interesting. To do this, we'll first scrape data for over 2000 movies.

In [132…
```
from requests import get
url = 'https://www.imdb.com/search/title?release_date=2017&sort=num_votes,desc&page
response = get(url)
print(response.text[:500])
```

```
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
</body>
</html>
```

Using BeautifulSoup to parse the HTML content To parse our HTML document and extract the 50 div containers, we'll use a Python module called BeautifulSoup, the most common web scraping module for Python. In the following code cell we will:

- Import the BeautifulSoup class creator from the package bs4.
- Parse response.text by creating a BeautifulSoup object, and assign this object to html_soup. The 'html.parser' argument indicates that we want to do the parsing using Python's built-in HTML parser.

In [122…
```
from bs4 import BeautifulSoup
html_soup = BeautifulSoup(response.text, 'html.parser')
headers = {'Accept-Language': 'en-US,en;q=0.8'}
type(html_soup)
```

Out[122… `bs4.BeautifulSoup`

Before extracting the 50 div containers, we need to figure out what distinguishes them from other div elements on that page. Often, the distinctive mark resides in the class attribute. If you inspect the HTML lines of the containers of interest, you'll notice that the class attribute has two values: lister-item and mode-advanced. This combination is unique to these div containers. We can see that's true by doing a quick search (Ctrl + F). We have 50 such containers, so we expect to see only 50 matches:

Now let's use the find_all() method to extract all the div containers that have a class attribute of lister-item mode-advanced:

In [126…
```
movie_containers = html_soup.find_all('div', class_ = 'lister-item mode-advanced')
print(type(movie_containers))
print(len(movie_containers))
```

```
<class 'bs4.element.ResultSet'>
0
```

find_all() returned a ResultSet object which is a list containing all the 50 divs we are interested in.

Now we'll select only the first container, and extract, by turn, each item of interest:

- The name of the movie.
- The year of release.
- The IMDB rating.
- The Metascore.
- The number of votes

Extracting the data for a single movie We can access the first container, which contains information about a single movie, by using list notation on movie_containers

In [128…
```
first_movie = movie_containers[0]
first_movie
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[128], line 1
----> 1 first_movie = movie_containers[0]
      2 first_movie

IndexError: list index out of range
```

In [130…
```
first_movie.div
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[130], line 1
----> 1 first_movie.div

NameError: name 'first_movie' is not defined
```

In [134…
```
first_movie.a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[134], line 1
----> 1 first_movie.a

NameError: name 'first_movie' is not defined
```

In [ ]:
```
first_movie.h3
```

In [ ]:
```
first_movie.h3.a
```

In [ ]:
```
first_name = first_movie.h3.a.text
first_name
```

In [ ]:
```
first_year = first_movie.h3.find('span', class_ = 'lister-item-year text-muted unbo
first_year
```

In [ ]:
```
first_year = first_year.text
 first_year
```

The IMDB rating

In [ ]:
```
first_movie.strong
```

In [49]:
```
first_imdb = float(first_movie.strong.text)
first_imdb
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[49], line 1
----> 1 first_imdb = float(first_movie.strong.text)
      2 first_imdb

NameError: name 'first_movie' is not defined
```

The Metascore

In [52]:
```
first_mscore = first_movie.find('span', class_ = 'metascore favorable')
first_mscore = int(first_mscore.text)
print(first_mscore)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[52], line 1
----> 1 first_mscore = first_movie.find('span', class_ = 'metascore favorable')
      2 first_mscore = int(first_mscore.text)
      3 print(first_mscore)

NameError: name 'first_movie' is not defined
```

The number of votes

In [ ]:
```
first_votes = first_movie.find('span', attrs = {'name':'nv'})
first_votes
```

In [ ]:
```
first_votes['data-value']
```

In [ ]:
```
first_votes = int(first_votes['data-value'])
```

In [142…
```
# Lists to store the scraped data in
names = []
years = []
imdb_ratings = []
metascores = []
votes = []

# Extract data from individual movie container
```

```
for container in movie_containers:

    # If the movie has a Metascore, then extract:
    if container.find('div', class_='ratings-metascore') is not None:
        # The name
        name = container.h3.a.text
        names.append(name)

        # The year
        year = container.h3.find('span', class_='lister-item-year').text
        years.append(year)

        # The IMDB rating
        imdb = float(container.strong.text)
        imdb_ratings.append(imdb)

        # The Metascore
        m_score = container.find('span', class_='metascore').text.strip()
        metascores.append(int(m_score))

        # The number of votes
        vote = container.find('span', attrs={'name': 'nv'})['data-value']
        votes.append(int(vote))
```

In [144…

```
import pandas as pd
test_df = pd.DataFrame({'movie': names,
'year': years,
'imdb': imdb_ratings,
'metascore': metascores,
'votes': votes
})
print(test_df.info())
test_df
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 0 entries
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   movie      0 non-null      float64
 1   year       0 non-null      float64
 2   imdb       0 non-null      float64
 3   metascore  0 non-null      float64
 4   votes      0 non-null      float64
dtypes: float64(5)
memory usage: 132.0 bytes
None
```

Out[144…

movie	year	imdb	metascore	votes

The script for multiple pages

In [ ]:

```
!pip install IPython
```

In [148…

```
from time import time
from time import sleep
from random import randint
from IPython.core.display import clear_output
pages = [ '1','2','3','4','5']
years_url = [ '2017', '2018', '2019', '2020']
# Redeclaring the lists to store data in
names = []
years = []
imdb_ratings = []
metascores = []


votes = []
# Preparing the monitoring of the loop
start_time = time()
requests = 0
# For every year in the interval 2000-2017
for year_url in years_url:
   # For every page in the interval 1-4
   for page in pages:
       # Make a get request
       response = get('https://www.imdb.com/search/title?release_date=' + year_url
       '&sort=num_votes,desc&page=' + page, headers = headers)
       # Pause the loop
       sleep(randint(8,15))
       # Monitor the requests
       requests += 1
       elapsed_time = time() - start_time
       print('Request:{}; Frequency: {} requests/s'.format(requests, requests/elap
       clear_output(wait = True)
       # Throw a warning for non-200 status codes
       if response.status_code != 200:
           warn('Request: {}; Status code: {}'.format(requests, response.status_co
       # Break the loop if the number of requests is greater than expected
       if requests > 72:
           warn('Number of requests was greater than expected.')
           break
       # Parse the content of the request with BeautifulSoup
       page_html = BeautifulSoup(response.text, 'html.parser')
       # Select all the 50 movie containers from a single page
       mv_containers = page_html.find_all('div', class_ = 'lister-item mode-advanc
       # For every movie of these 50
       for container in mv_containers:
           # If the movie has a Metascore, then:
           if container.find('div', class_ = 'ratings-metascore') is not None:

               # Scrape the name
               name = container.h3.a.text
               names.append(name)
               # Scrape the year
               year = container.h3.find('span', class_ = 'lister-item-year').text
               years.append(year)
               # Scrape the IMDB rating
               imdb = float(container.strong.text)
```

```
                imdb_ratings.append(imdb)
                # Scrape the Metascore
                m_score = container.find('span', class_ = 'metascore').text
                metascores.append(int(m_score))
                # Scrape the number of votes
                vote = container.find('span', attrs = {'name':'nv'})['data-value']
                votes.append(int(vote))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[148], line 34
     32 # Throw a warning for non-200 status codes
     33 if response.status_code != 200:
---> 34     warn('Request: {}; Status code: {}'.format(requests, response.status_cod
e))
     35 # Break the loop if the number of requests is greater than expected
     36 if requests > 72:

NameError: name 'warn' is not defined
```

In [153…
```
movie_ratings = pd.DataFrame({'movie': names,
 'year': years,
 'imdb': imdb_ratings,
 'metascore': metascores,
 'votes': votes
 })
print(movie_ratings.info())
movie_ratings.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 0 entries
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   movie      0 non-null      float64
 1   year       0 non-null      float64
 2   imdb       0 non-null      float64
 3   metascore  0 non-null      float64
 4   votes      0 non-null      float64
dtypes: float64(5)
memory usage: 132.0 bytes
None
```

Out[153…

	movie	year	imdb	metascore	votes

In [155…
```
movie_ratings.tail(10)
```

Out[155…

	movie	year	imdb	metascore	votes

In [159…
```
movie_ratings.to_csv('movie_ratings.csv')
```

Data Preparation

- Collected data may not be compatible or formatted correctly
- Data must be prepared before it can be added to a data set
- Extract, Transform and Load (ETL)

process for collecting data from a variety of sources, transforming the data, and then loading the data into a database

Data preprocessing

Data Processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data preprocessing.

Most of the real-world data is messy, some of these types of data are: 1. Missing data: Missing data can be found when it is not continuously created or due to technical issues in the application (IOT system). 2. Noisy Data This type of data is also called outliners, this can occur due to human errors (human manually gathering the data) or some technical problem of the device at the time of collection of data. 3. Inconsistent data: This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

These are some of the basic pre processing techniques that can be used to convert raw data. 1. Conversion of data: As we know that Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features. 2. Ignoring the missing values: Whenever we encounter missing data in the data set then we can remove the row or column of data depending on our need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset. 3. Filling the missing values: Whenever we encounter missing data in the data set then we can fill the missing data manually, most commonly the mean, median or highest frequency value is used.

1 . Machine learning: If we have some missing data then we can predict what data shall be present at the empty position by using the existing data. 5. Outliers detection: There are some error data that might be present in our data set that deviates drastically from other observations in a data set. [Example: human weight = 800 Kg; due to mistyping of extra 0]

Example of Data Preparation of movie_rating.csv

In [161…
```
movie_ratings['year'].unique()
```

Out[161…
```
array([], dtype=float64)
```

In [163…
```
movie_ratings.dtypes
```

Out[163…
```
movie        float64
year         float64
imdb         float64
metascore    float64
votes        float64
dtype: object
```

In [165…
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace('(I)','')))
```

In [167…
```
movie_ratings['year'].unique()
```

Out[167…
```
array([], dtype=float64)
```

In [169…
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace('(II)','')))
```

In [171…
```
 movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace('(III)','')))
```

In [173…
```
movie_ratings['year'].unique()
```

Out[173…
```
array([], dtype=float64)
```

In [175…
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace('(','')))
```

In [177…
```
movie_ratings['year'].unique()
```

Out[177…
```
array([], dtype=float64)
```

In [179…
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace(')','')))
```

In [181…
```
movie_ratings['year'].unique()
```

Out[181…
```
array([], dtype=float64)
```

In [183…
```
movie_ratings['year'] = movie_ratings['year'].astype(int)
```

In [185…
```
movie_ratings['year'].unique()
```

Out[185…
```
array([], dtype=int32)
```

In [187…
```
movie_ratings.dtypes
```

Out[187…
```
movie        float64
year           int32
imdb         float64
metascore    float64
votes        float64
dtype: object
```

In [189…
```
movie_ratings.head(10)
```

Out[189…

	movie	year	imdb	metascore	votes

In [191…
```
movie_ratings.tail(10)
```

Out[191…

	movie	year	imdb	metascore	votes

In [193…
```
movie_ratings
```

Out[193…

	movie	year	imdb	metascore	votes

In [ ]:

Submission Requirements

In [10]:
```
#Webscrapping of my choice

url = 'https://www.popvortex.com/music/charts/top-100-songs.php'
```

```
  Cell In[10], line 1
    Webscrapping of my choice
                  ^
SyntaxError: invalid syntax
```

In [45]:
```
from bs4 import BeautifulSoup # importing libraries
import requests

source = requests.get('https://www.popvortex.com/music/charts/top-100-songs.php') #

soup = BeautifulSoup(source.text, 'lxml') # # use the module and a parser

'''I  will first ttry to get the first instance of the data I need'''


chart = soup.find('div', class_='chart-wrapper') # the parent div

box = soup.find('div', class_='feed-item music-chart chart-feed-grid flex-row')
song_name = box.find('p', class_= 'title-artist').cite.text  # getting html tag ins

print(song_name) # checking
print()

artist_name = box.find('p', class_= 'title-artist').em.text # getting the artist na

print(artist_name) # checking
print()

genre = box.find('div', class_= 'chart-content').ul.li.a.text # getting the genre

print(genre)
print()
```

```
release_date = box.select('ul > li') # getting the release date
# this has the select method because this data is in a nested html tag

print(release_date[1].text) # checking
print()
```

```
Ordinary

Alex Warren

Pop

Release Date: September 27, 2024
```

In [55]:
```
# since there was a none instance I will implement the try except function

from bs4 import BeautifulSoup
import requests
import csv


source = requests.get('https://www.popvortex.com/music/charts/top-100-songs.php') #
soup = BeautifulSoup(source.text, 'html.parser') # use the module and a parser

# open the CSV file to write the data
with open('topsongs.csv', 'w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(['Rank', 'Song Name', 'Artist', 'Genre', 'Release Date'])  # fi

    # parent div
    chart_items = soup.find_all('div', class_='chart-wrapper')
    boxes = soup.find_all('div', class_='feed-item music-chart chart-feed-grid flex

    rank = 1  # rank counter
    for box in boxes:
        song_info = box.find('p', class_='title-artist')
        song_name = song_info.cite.text # getting song name
        artist_name = song_info.em.text # artist name

        try:
            genre = box.find('div', class_='chart-content').ul.li.a.text.strip()
        except AttributeError:
            genre = 'None'

        release_info = box.select('ul > li')
        release_date = release_info[1].text # release date

        # print for checking
        print(f"{rank}. {song_name} by {artist_name} | Genre: {genre} | Released: {

        # write to CSV
        writer.writerow([rank, song_name, artist_name, genre, release_date])

        rank += 1
```

1. Ordinary by Alex Warren | Genre: Pop | Released: Release Date: September 27, 2024
2. I'm The Problem by Morgan Wallen | Genre: Country | Released: Release Date: January 31, 2025
3. Anxiety by Doechii | Genre: Hip-Hop / Rap | Released: Release Date: March 5, 2025
4. NOKIA by Drake | Genre: R&B / Soul | Released: Release Date: February 14, 2025
5. Pink Pony Club by Chappell Roan | Genre: Pop | Released: Release Date: April 3, 2020
6. Just In Case by Morgan Wallen | Genre: Country | Released: Release Date: March 21, 2025
7. I'm A Little Crazy by Morgan Wallen | Genre: Country | Released: Release Date: March 21, 2025
8. Love Somebody by Morgan Wallen | Genre: Country | Released: Release Date: October 18, 2024
9. Die With A Smile by Lady Gaga & Bruno Mars | Genre: Pop | Released: Release Date: August 16, 2024
10. Gratitude by Brandon Lake | Genre: Christian & Gospel | Released: Release Date: August 28, 2020
11. Beautiful Things by Benson Boone | Genre: Pop | Released: Release Date: January 18, 2024
12. APT. by ROSÉ & Bruno Mars | Genre: Pop | Released: Release Date: October 17, 2024
13. Smile by Morgan Wallen | Genre: Country | Released: Release Date: December 31, 2024
14. Lose Control by Teddy Swims | Genre: Pop | Released: Release Date: June 23, 2023
15. A Bar Song (Tipsy) by Shaboozey | Genre: Country | Released: Release Date: April 12, 2024
16. Messy by Lola Young | Genre: Alternative | Released: Release Date: May 29, 2024
17. Lies Lies Lies by Morgan Wallen | Genre: Country | Released: Release Date: July 5, 2024
18. Hard Fought Hallelujah by Brandon Lake & Jelly Roll | Genre: Christian & Gospel | Released: Release Date: February 7, 2025
19. Azizam by Ed Sheeran | Genre: Pop | Released: Release Date: March 19, 2025
20. YOUR WAY'S BETTER by Forrest Frank | Genre: Christian & Gospel | Released: Release Date: October 24, 2024
21. Bad Dreams by Teddy Swims | Genre: Pop | Released: Release Date: September 13, 2024
22. Not Like Us by Kendrick Lamar | Genre: Hip-Hop / Rap | Released: Release Date: May 4, 2024
23. luther by Kendrick Lamar & SZA | Genre: Hip-Hop / Rap | Released: Release Date: November 21, 2024
24. The Door by Teddy Swims | Genre: Pop | Released: Release Date: April 17, 2024
25. Abracadabra by Lady Gaga | Genre: Pop | Released: Release Date: February 3, 2025
26. Worst Way by Riley Green | Genre: Country | Released: Release Date: March 29, 2024
27. BIRDS OF A FEATHER by Billie Eilish | Genre: Alternative | Released: Release Date: May 17, 2024
28. Good News by Shaboozey | Genre: Country | Released: Release Date: November 15, 2024
29. Volveré by Rubby Pérez | Genre: None | Released: Release Date: January 1, 1989
30. I Am Not Okay by Jelly Roll | Genre: Country | Released: Release Date: June 12, 2024
31. A Lot More Free by Max McNown | Genre: Country | Released: Release Date: February 21, 2024
32. I Never Lie by Zach Top | Genre: Traditional Country | Released: Release Date: April 5, 2024
33. YIPPEE-KI-YAY. by Kesha | Genre: Pop | Released: Release Date: March 27, 2025

34. Stargazing by Myles Smith | Genre: None | Released: Release Date: May 10, 2024
35. Boots on the Ground by 803Fresh | Genre: Blues | Released: Release Date: Decembe r 20, 2024
36. Better Me For You (Brown Eyes) by Max McNown | Genre: Country | Released: Releas e Date: November 14, 2024
37. Emergence by Sleep Token | Genre: Heavy Metal | Released: Release Date: March 1 3, 2025
38. Liar by Jelly Roll | Genre: Country | Released: Release Date: August 2, 2024
39. I Had Some Help (feat. Morgan Wallen) by Post Malone | Genre: Country | Release d: Release Date: May 10, 2024
40. The Search by NF | Genre: Hip-Hop / Rap | Released: Release Date: May 30, 2019
41. The Sound of Silence by Disturbed | Genre: Hard Rock | Released: Release Date: A ugust 21, 2015
42. Ain't No Love In Oklahoma by Luke Combs | Genre: Country | Released: Release Dat e: May 18, 2024
43. Texas by Blake Shelton | Genre: Country | Released: Release Date: November 15, 2 024
44. Steve's Lava Chicken by Jack Black | Genre: Soundtrack | Released: Release Date: March 28, 2025
45. Espresso by Sabrina Carpenter | Genre: Pop | Released: Release Date: April 11, 2 024
46. The Truth by Megan Woods | Genre: Christian & Gospel | Released: Release Date: M ay 31, 2024
47. Buscando Tus Besos by Rubby Pérez | Genre: None | Released: Release Date: Januar y 1, 1987
48. Happen To Me by Russell Dickerson | Genre: Country | Released: Release Date: Feb ruary 21, 2025
49. I'm Gonna Love You by Cody Johnson & Carrie Underwood | Genre: Country | Release d: Release Date: September 27, 2024
50. twilight zone by Ariana Grande | Genre: Pop | Released: Release Date: March 28, 2025
51. Praise (feat. Brandon Lake, Chris Brown & Chandler Moore) by Elevation Worship | Genre: Christian & Gospel | Released: Release Date: May 19, 2023
52. tv off (feat. Lefty Gunplay) by Kendrick Lamar | Genre: Hip-Hop / Rap | Release d: Release Date: November 22, 2024
53. GOOD DAY by Forrest Frank | Genre: Christian & Gospel | Released: Release Date: January 19, 2024
54. MUTT by Leon Thomas | Genre: R&B / Soul | Released: Release Date: August 8, 2024
55. My Bubble Gum (Dirty) by Rasheeda | Genre: Hip-Hop / Rap | Released: Release Dat e: July 26, 2005
56. Wondering Why by The Red Clay Strays | Genre: Country | Released: Release Date: March 9, 2022
57. Always Remember Us This Way by Lady Gaga | Genre: Soundtrack | Released: Release Date: October 5, 2018
58. That’s So True by Gracie Abrams | Genre: Pop | Released: Release Date: October 1 8, 2024
59. Sports car by Tate McRae | Genre: Pop | Released: Release Date: January 24, 2025
60. you look like you love me by Ella Langley & Riley Green | Genre: Country | Relea sed: Release Date: June 21, 2024
61. Hi Ren by Ren | Genre: None | Released: Release Date: December 15, 2022
62. Afterlife (From the Netflix Series "Devil May Cry") by Evanescence | Genre: Soun dtrack | Released: Release Date: March 28, 2025
63. Somethin' 'Bout A Woman (feat. Teddy Swims) by Thomas Rhett | Genre: Pop | Relea sed: Release Date: November 15, 2024
64. Too Sweet by Hozier | Genre: Alternative | Released: Release Date: March 22, 202 4

65. Starburster by Fontaines D.C. | Genre: Alternative | Released: Release Date: April 17, 2024
66. Three Six Five by Shinedown | Genre: Rock | Released: Release Date: January 24, 2025
67. Sorry I'm Here For Someone Else by Benson Boone | Genre: Pop | Released: Release Date: February 27, 2025
68. The Sound of Silence (CYRIL Remix) by Disturbed | Genre: Dance | Released: Release Date: February 13, 2024
69. Take Me Home, Country Roads (Original Version) by John Denver | Genre: Pop | Released: Release Date: April 6, 1971
70. Tennessee Whiskey by Chris Stapleton | Genre: Country | Released: Release Date: April 23, 2015
71. Lil Boo Thang by Paul Russell | Genre: Pop | Released: Release Date: August 18, 2023
72. That's Who I Praise by Brandon Lake | Genre: Christian & Gospel | Released: Release Date: July 29, 2024
73. FISHIN' IN THE DARK by Niko Moon | Genre: Country | Released: Release Date: January 19, 2024
74. Thunderstruck by AC/DC | Genre: Hard Rock | Released: Release Date: September 10, 1990
75. How Does It Feel To Be Forgotten by Selena Gomez & benny blanco | Genre: Pop | Released: Release Date: March 21, 2025
76. How Do I Say Goodbye by Dean Lewis | Genre: None | Released: Release Date: September 1, 2022
77. Weight Of Your World by Chris Stapleton | Genre: Country | Released: Release Date: November 10, 2023
78. Last Of My Kind (feat. Paul Cauthen) by Shaboozey | Genre: Country | Released: Release Date: May 31, 2024
79. squabble up by Kendrick Lamar | Genre: Hip-Hop / Rap | Released: Release Date: November 22, 2024
80. Don't Stop Believin' (2024 Remaster) by Journey | Genre: Rock | Released: Release Date: June 3, 1981
81. We Hug Now by Sydney Rose | Genre: Pop | Released: Release Date: February 13, 2025
82. Y No Voy a Llorar by Rubby Pérez | Genre: None | Released: Release Date: March 1, 2001
83. Hometown Home by LOCASH | Genre: Country | Released: Release Date: April 19, 2024
84. Good Luck, Babe! by Chappell Roan | Genre: Pop | Released: Release Date: April 5, 2024
85. Desperate by Jamie MacDonald | Genre: Christian & Gospel | Released: Release Date: January 3, 2025
86. Someone You Loved by Lewis Capaldi | Genre: Alternative | Released: Release Date: November 8, 2018
87. HOT TO GO! by Chappell Roan | Genre: Pop | Released: Release Date: August 11, 2023
88. Goodness of God (Live) by CeCe Winans | Genre: Christian & Gospel | Released: Release Date: March 12, 2021
89. YIPPEE-KI-YAY. (feat. T-Pain) by Kesha | Genre: Pop | Released: Release Date: March 27, 2025
90. One Last Breath by Creed | Genre: Hard Rock | Released: Release Date: November 20, 2001
91. weren't for the wind by Ella Langley | Genre: Country | Released: Release Date: October 4, 2024
92. Nothing's Gonna Stop Us Now by Starship | Genre: Rock | Released: Release Date: January 30, 1987

```
93. The Giver by Chappell Roan | Genre: Country | Released: Release Date: March 13, 2025
```

In [154…
```python
from bs4 import BeautifulSoup # importing libraries
import requests
import csv # for creating the file


source = requests.get('https://www.popvortex.com/music/charts/top-100-songs.php') #
soup = BeautifulSoup(source.text, 'lxml') #

chart = soup.find('div', class_='chart-wrapper')
boxes = soup.find_all('div', class_='feed-item music-chart chart-feed-grid flex-row
print(f"Found {len(boxes)} songs.")  # check how many songs are found
```

```
Found 93 songs.
```

In [256…
```python
# preprocessing of the dataset

'''Now that I have created the the set I can perform data cleaning so that it would
when doing analysis'''

import pandas as pd

songs = pd.read_csv('topsongs.csv')

songs.head() # checking of initial values
```

Out[256…

	Rank	Song Name	Artist	Genre	Release Date
0	1	Ordinary	Alex Warren	Pop	Release Date: September 27, 2024
1	2	I'm The Problem	Morgan Wallen	Country	Release Date: January 31, 2025
2	3	Anxiety	Doechii	Hip-Hop / Rap	Release Date: March 5, 2025
3	4	NOKIA	Drake	R&B / Soul	Release Date: February 14, 2025
4	5	Pink Pony Club	Chappell Roan	Pop	Release Date: April 3, 2020

In [258…
```python
songs.dtypes # checking of data types
```

Out[258…
```
Rank            int64
Song Name      object
Artist         object
Genre          object
Release Date   object
dtype: object
```

In [260…
```python
songs.shape # checking the number of entries
```

Out[260…
```
(93, 5)
```

In [262…
```python
songs.isnull().sum() # checking if their is null values
```

Out[262…
```
Rank            0
Song Name       0
Artist          0
Genre           6
Release Date    0
dtype: int64
```

In [264…
```
songs.columns
```

Out[264…
```
Index(['Rank', 'Song Name', 'Artist', 'Genre', 'Release Date'], dtype='object')
```

after checking the dataset I decided to change the released date column since there is a conflict in the data extracted

In [266…
```
# checking of error
songs['Release Date']
```

Out[266…
```
0     Release Date: September 27, 2024
1       Release Date: January 31, 2025
2          Release Date: March 5, 2025
3      Release Date: February 14, 2025
4          Release Date: April 3, 2020
                    ...
88        Release Date: March 27, 2025
89     Release Date: November 20, 2001
90       Release Date: October 4, 2024
91      Release Date: January 30, 1987
92        Release Date: March 13, 2025
Name: Release Date, Length: 93, dtype: object
```

In [270…
```
# there is a release date string
# now i will remove that and keep the date only
songs['Release Date'] = songs['Release Date'].astype(str)

# split by ':' and take the part after it (index 1) to clean up the text
songs['Release Date'] = songs['Release Date'].str.split(':').str[1].str.
# convert the cleaned 'Release Date' column to datetime
songs['Release Date'] = pd.to_datetime(songs['Release Date'], format='%B %d, %Y')

# shecking
songs['Release Date']

# save the modified DataFrame back to CSV
songs.to_csv('top_100_songs.csv', index=False, encoding='utf-8')
```

```
0    2024-09-27
1    2025-01-31
2    2025-03-05
3    2025-02-14
4    2020-04-03
        ...
88   2025-03-27
89   2001-11-20
90   2024-10-04
91   1987-01-30
92   2025-03-13
Name: Release Date, Length: 93, dtype: datetime64[ns]
```

In [98]:
```
songs.dtypes # checking of modification
```

Out[98]:
```
Rank                     int64
Song Name               object
Artist                  object
Genre                   object
Release Date    datetime64[ns]
dtype: object
```

In [104…
```
duplicates = songs.duplicated() # checking for duplicates
duplicates.sum()
```

Out[104…
```
0
```

In [106…
```
songs.head()
```

Out[106…

	Rank	Song Name	Artist	Genre	Release Date
0	1	Ordinary	Alex Warren	Pop	2024-09-27
1	2	I'm The Problem	Morgan Wallen	Country	2025-01-31
2	3	Anxiety	Doechii	Hip-Hop / Rap	2025-03-05
3	4	NOKIA	Drake	R&B / Soul	2025-02-14
4	5	Pink Pony Club	Chappell Roan	Pop	2020-04-03

In [ ]:
```
```

Now that the dataset is modified I can load some analysis about the dataset

In [115…
```
# checking the best genre by count

topGenre = songs['Genre'].value_counts() # getting the frequency of each genre
topGenre.nlargest(1) # gets the top1
```

Out[115…
```
Genre
Country    27
Name: count, dtype: int64
```

In [117…
```
# which artist has many songs in the ranking
topArtist = songs['Artist'].value_counts()
topArtist.nlargest(1) # gets top1
```

Out[117…
```
Artist
Morgan Wallen    6
Name: count, dtype: int64
```

In [131…
```
# getting the month with the most released top songs
songs1 = songs.copy()
songs1['Month'] = songs1['Release Date'].dt.month_name()

# count the number of songs released each month
month_counts = songs1['Month'].value_counts()
```

```
# find the month with the most releases
month_counts.nlargest()
```

Out[131…

```
Month
March       17
January     10
April       10
May         10
November    10
Name: count, dtype: int64
```

In [139…

```
# getting all the pop genre songs
songs[songs['Genre'] == 'Pop']
```

Out[139…

	Rank	Song Name	Artist	Genre	Release Date
0	1	Ordinary	Alex Warren	Pop	2024-09-27
4	5	Pink Pony Club	Chappell Roan	Pop	2020-04-03
8	9	Die With A Smile	Lady Gaga & Bruno Mars	Pop	2024-08-16
10	11	Beautiful Things	Benson Boone	Pop	2024-01-18
11	12	APT.	ROSÉ & Bruno Mars	Pop	2024-10-17
13	14	Lose Control	Teddy Swims	Pop	2023-06-23
18	19	Azizam	Ed Sheeran	Pop	2025-03-19
20	21	Bad Dreams	Teddy Swims	Pop	2024-09-13
23	24	The Door	Teddy Swims	Pop	2024-04-17
24	25	Abracadabra	Lady Gaga	Pop	2025-02-03
32	33	YIPPEE-KI-YAY.	Kesha	Pop	2025-03-27
44	45	Espresso	Sabrina Carpenter	Pop	2024-04-11
49	50	twilight zone	Ariana Grande	Pop	2025-03-28
57	58	That's So True	Gracie Abrams	Pop	2024-10-18
58	59	Sports car	Tate McRae	Pop	2025-01-24
62	63	Somethin' 'Bout A Woman (feat. Teddy Swims)	Thomas Rhett	Pop	2024-11-15
66	67	Sorry I'm Here For Someone Else	Benson Boone	Pop	2025-02-27
68	69	Take Me Home, Country Roads (Original Version)	John Denver	Pop	1971-04-06
70	71	Lil Boo Thang	Paul Russell	Pop	2023-08-18
74	75	How Does It Feel To Be Forgotten	Selena Gomez & benny blanco	Pop	2025-03-21
80	81	We Hug Now	Sydney Rose	Pop	2025-02-13
83	84	Good Luck, Babe!	Chappell Roan	Pop	2024-04-05
86	87	HOT TO GO!	Chappell Roan	Pop	2023-08-11
88	89	YIPPEE-KI-YAY. (feat. T-Pain)	Kesha	Pop	2025-03-27

In [ ]:

In [149…

```
# getting songs ranked from 15 to 30
songs = songs.set_index('Rank')
```

In [147…
```
songs[14:30]
```

Out[147…

Rank	Song Name	Artist	Genre	Release Date
15	A Bar Song (Tipsy)	Shaboozey	Country	2024-04-12
16	Messy	Lola Young	Alternative	2024-05-29
17	Lies Lies Lies	Morgan Wallen	Country	2024-07-05
18	Hard Fought Hallelujah	Brandon Lake & Jelly Roll	Christian & Gospel	2025-02-07
19	Azizam	Ed Sheeran	Pop	2025-03-19
20	YOUR WAY'S BETTER	Forrest Frank	Christian & Gospel	2024-10-24
21	Bad Dreams	Teddy Swims	Pop	2024-09-13
22	Not Like Us	Kendrick Lamar	Hip-Hop / Rap	2024-05-04
23	luther	Kendrick Lamar & SZA	Hip-Hop / Rap	2024-11-21
24	The Door	Teddy Swims	Pop	2024-04-17
25	Abracadabra	Lady Gaga	Pop	2025-02-03
26	Worst Way	Riley Green	Country	2024-03-29
27	BIRDS OF A FEATHER	Billie Eilish	Alternative	2024-05-17
28	Good News	Shaboozey	Country	2024-11-15
29	Volveré	Rubby Pérez	NaN	1989-01-01
30	I Am Not Okay	Jelly Roll	Country	2024-06-12

In [151…
```
# checking the songs containing the word "The"
songs_with_the = songs[songs['Song Name'].str.contains('The', case=False, na=False)]
songs_with_the
```

Out[151...

Rank	Song Name	Artist	Genre	Release Date
2	I'm The Problem	Morgan Wallen	Country	2025-01-31
23	luther	Kendrick Lamar & SZA	Hip-Hop / Rap	2024-11-21
24	The Door	Teddy Swims	Pop	2024-04-17
27	BIRDS OF A FEATHER	Billie Eilish	Alternative	2024-05-17
35	Boots on the Ground	803Fresh	Blues	2024-12-20
40	The Search	NF	Hip-Hop / Rap	2019-05-30
41	The Sound of Silence	Disturbed	Hard Rock	2015-08-21
46	The Truth	Megan Woods	Christian & Gospel	2024-05-31
62	Afterlife (From the Netflix Series "Devil May ...	Evanescence	Soundtrack	2025-03-28
68	The Sound of Silence (CYRIL Remix)	Disturbed	Dance	2024-02-13
73	FISHIN' IN THE DARK	Niko Moon	Country	2024-01-19
78	Last Of My Kind (feat. Paul Cauthen)	Shaboozey	Country	2024-05-31
91	weren't for the wind	Ella Langley	Country	2024-10-04
93	The Giver	Chappell Roan	Country	2025-03-13

In [ ]:

CONCLUSION

I learned the different ways to do web scrapping in python. I learned the different libraries needed for scapping particular data which is beautifulsoup for basic web scrapping and selenium for image scrapping. I learned that to do web scrapping you must be aware about the html elements in the website you need. For me it is important to do double checking on the html elements you access since it may have a conflict when trying to run your scrapper. In my web scrapping code, I did a lot of trial and error because of the way i handled the scrapping. At first i thought it was working since i got the first instance but when I try to run through all of the website, I only got the first. I think it is important to be mindful on what html element we access. I also learned that sometimes the website itself my have a conflict in loading the data to your scraper just like mine which has the lazy attribute and it may even had dynamic content since I saw 100 songs in the website but when i loaded it in my scraper

it only got 93. Moving forward, I need to learn more on how to handle the different situations where the website structure affect my web scrapper program.

In [ ]: