Example of gathering image data using webcam

In [3]:
```
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 framecd
    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)
      img_new = 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.destrolAllWindows()
    break
  [40 38 38]
  [39 37 37]]

 [[57 58 56]
  [58 59 57]
  [60 61 59]
  ...
  [42 40 40]
  [41 39 39]
  [40 38 38]]

 [[58 59 57]
  [58 59 57]
  [60 61 59]
  ...
  [39 39 39]
  [39 39 39]
  [38 38 38]]

 ...
```

Example of gathering voice data using microphone

In [5]:
```
!pip3 install sounddevice
```

```
Requirement already satisfied: sounddevice in c:\users\jay-ann alorro\anaconda3\lib
\site-packages (0.4.6)
Requirement already satisfied: CFFI>=1.0 in c:\users\jay-ann alorro\anaconda3\lib\s
ite-packages (from sounddevice) (1.15.1)
Requirement already satisfied: pycparser in c:\users\jay-ann alorro\anaconda3\lib\s
ite-packages (from CFFI>=1.0->sounddevice) (2.21)
```

In [6]:
```
!pip3 install wavio
```

```
Collecting wavio
  Obtaining dependency information for wavio from https://files.pythonhosted.org/pa
ckages/bf/02/40d03e99a3d2d8d1e9392f44376f470120427ffb12483579dc7e0365f712/wavio-0.
0.8-py3-none-any.whl.metadata (https://files.pythonhosted.org/packages/bf/02/40d03e
99a3d2d8d1e9392f44376f470120427ffb12483579dc7e0365f712/wavio-0.0.8-py3-none-any.wh
l.metadata)
  Downloading wavio-0.0.8-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: numpy>=1.19.0 in c:\users\jay-ann alorro\anaconda3\l
ib\site-packages (from wavio) (1.24.3)
Downloading wavio-0.0.8-py3-none-any.whl (9.4 kB)
Installing collected packages: wavio
Successfully installed wavio-0.0.8
```

In [8]:
```
!pip3 install scipy
```

```
Requirement already satisfied: scipy in c:\users\jay-ann alorro\anaconda3\lib\site-
packages (1.11.1)
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in c:\users\jay-ann alorro\ana
conda3\lib\site-packages (from scipy) (1.24.3)
```

In [13]:
```
!apt-get install libportaudio2
```

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

In [15]:
```
# 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

Image Scraping using BeautifulSoup and Request

In [16]:
```
!pip install bs4
```

```
Collecting bs4
  Obtaining dependency information for bs4 from https://files.pythonhosted.org/pack
ages/51/bb/bf7aab772a159614954d84aa832c129624ba6c32faa559dfb200a534e50b/bs4-0.0.2-p
y2.py3-none-any.whl.metadata (https://files.pythonhosted.org/packages/51/bb/bf7aab7
72a159614954d84aa832c129624ba6c32faa559dfb200a534e50b/bs4-0.0.2-py2.py3-none-any.wh
l.metadata)
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: beautifulsoup4 in c:\users\jay-ann alorro\anaconda3
\lib\site-packages (from bs4) (4.12.2)
Requirement already satisfied: soupsieve>1.2 in c:\users\jay-ann alorro\anaconda3\l
ib\site-packages (from beautifulsoup4->bs4) (2.4)
Downloading bs4-0.0.2-py2.py3-none-any.whl (1.2 kB)
Installing collected packages: bs4
Successfully installed bs4-0.0.2
```

In [17]:
```
pip install requests
```

```
Requirement already satisfied: requests in c:\users\jay-ann alorro\anaconda3\lib\si
te-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\jay-ann alorro
\anaconda3\lib\site-packages (from requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\jay-ann alorro\anaconda3\li
b\site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\jay-ann alorro\anacon
da3\lib\site-packages (from requests) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\jay-ann alorro\anacon
da3\lib\site-packages (from requests) (2023.7.22)
Note: you may need to restart the kernel to use updated packages.
```

In [18]:
```python
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 [19]:
```
pip install selenium
```

```
   ------------------------- -------------- 6.4/10.0 MB 5.7 MB/s eta 0:00:01
   -------------------------- ------------- 6.6/10.0 MB 5.8 MB/s eta 0:00:01
   -------------------------- ------------- 6.7/10.0 MB 5.6 MB/s eta 0:00:01
   --------------------------- ------------ 7.0/10.0 MB 5.6 MB/s eta 0:00:01
   ---------------------------- ----------- 7.0/10.0 MB 5.5 MB/s eta 0:00:01
   ----------------------------- ---------- 7.3/10.0 MB 5.5 MB/s eta 0:00:01
   ------------------------------ --------- 7.5/10.0 MB 5.5 MB/s eta 0:00:01
   ------------------------------- -------- 7.8/10.0 MB 5.6 MB/s eta 0:00:01
   -------------------------------- ------- 8.1/10.0 MB 5.6 MB/s eta 0:00:01
   --------------------------------- ------ 8.4/10.0 MB 5.6 MB/s eta 0:00:01
   ---------------------------------- ----- 8.6/10.0 MB 5.7 MB/s eta 0:00:01
   ------------------------------------ --- 9.0/10.0 MB 5.7 MB/s eta 0:00:01
   ------------------------------------ --- 9.1/10.0 MB 5.7 MB/s eta 0:00:01
   ------------------------------------ --- 9.2/10.0 MB 5.6 MB/s eta 0:00:01
   ------------------------------------- -- 9.5/10.0 MB 5.6 MB/s eta 0:00:01
   -------------------------------------- - 9.7/10.0 MB 5.6 MB/s eta 0:00:01
   ---------------------------------------  10.0/10.0 MB 5.7 MB/s eta 0:00:01
   ---------------------------------------  10.0/10.0 MB 5.7 MB/s eta 0:00:01
   ---------------------------------------  10.0/10.0 MB 5.7 MB/s eta 0:00:01
   ---------------------------------------- 10.0/10.0 MB 5.3 MB/s eta 0:00:00
```

Image Scraping using Selenium

```
In [36]: !pip install selenium
         # !apt-get update # 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(options=chrome_options)

         search_url = "https://www.google.com/search?q={q}&tbm=isch&tbs=sur%3Afc&hl=en&ved=0C
         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&ve
             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('xpath', "//img[contains(@class,'Q4
                 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.get
                             img_urls.add(actual_image.get_attributes('src'))

                     img_count = len(img_urls)

                     if img_count >= totalImgs:
                         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 = r'C:\Users\Jay-ann Alorro\Downloads\data sci\Dataset'
totalImgs=5

saveInDestFolder(searchNames, destDir, totalImgs, driver)
```

```
--> 103 saveInDestFolder(searchNames,destDir,totalImgs,driver)

Cell In[36], line 87, in saveInDestFolder(searchNames, destDir, totalImgs, drive
r)
     85         os.mkdir(path)
     86     print('Current Path',path)
---> 87     totalLinks=getImageUrls(name,totalImgs,driver)
     88     print('totalLinks',totalLinks)
     90 if totalLinks is None:

Cell In[36], line 41, in getImageUrls(name, totalImgs, driver)
     38 results_start = 0
     40 while(img_count < totalImgs): # Extract actual images now
---> 41     scroll_to_end(driver)
     43     thumbnail_results = driver.find_elements('xpath', "//img[contains(@c
lass,'Q4LuWd')]")
     44     totalResults = len(thumbnail_results)

Cell In[36], line 31, in scroll_to_end(driver)
```

Web Scraping of Movies Information using BeautifulSoup

In [67]:
```python
from requests import get
url = 'https://www.imdb.com/search/title/?release_date=2017-01-01,2017-12-31&sort=nu
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKi
response = requests.get(url, headers=headers)
print(response.text[:500])
```

```
<!DOCTYPE html><html lang="en-US" xmlns:og="http://opengraphprotocol.org/schema/" x
mlns:fb="http://www.facebook.com/2008/fbml"><head><meta charSet="utf-8"/><meta name
="viewport" content="width=device-width"/><script>if(typeof uet === 'function'){ ue
t('bb', 'LoadTitle', {wb: 1}); }</script><script>window.addEventListener('load', (e
vent) => {
        if (typeof window.csa !== 'undefined' && typeof window.csa === 'function')
{
            var csaLatencyPlugin = window.csa('Content', {
```

In [68]:
```python
from bs4 import BeautifulSoup
html_soup = BeautifulSoup(response.text, 'html.parser')
headers ={'Accept-Language': 'en-US,en;q=0.8'}
type(html_soup)
```

Out[68]:
```
bs4.BeautifulSoup
```

In [71]:
```python
movie_containers = html_soup.find_all('li', class_ = 'ipc-metadata-list-summary-item
print(type(movie_containers))
print(len(movie_containers))
```

```
<class 'bs4.element.ResultSet'>
50
```

In [72]:
```
first_movie = movie_containers[0]
first_movie
```

```
Out[72]: <li class="ipc-metadata-list-summary-item"><div class="ipc-metadata-list-summary-it
         em__c"><div class="ipc-metadata-list-summary-item__tc"><span aria-disabled="false"
         class="ipc-metadata-list-summary-item__t"></span><div class="sc-ab6fa25a-3 bVYfLY d
         li-parent"><div class="sc-ab6fa25a-2 gOsifL"><div class="sc-e5a25b0f-0 jQjDIb dli-p
         oster-container"><div class="ipc-poster ipc-poster--base ipc-poster--dynamic-width
         ipc-sub-grid-item ipc-sub-grid-item--span-2" role="group"><div aria-label="add to w
         atchlist" class="ipc-watchlist-ribbon ipc-focusable ipc-watchlist-ribbon--s ipc-wat
         chlist-ribbon--base ipc-watchlist-ribbon--loading ipc-watchlist-ribbon--onImage ipc
         -poster__watchlist-ribbon" role="button" tabindex="0"><svg class="ipc-watchlist-rib
         bon__bg" height="34px" role="presentation" viewbox="0 0 24 34" width="24px" xmlns
         ="http://www.w3.org/2000/svg"><polygon class="ipc-watchlist-ribbon__bg-ribbon" fill
         ="#000000" points="24 0 0 0 0 32 12.2436611 26.2926049 24 31.7728343"></polygon><po
         lygon class="ipc-watchlist-ribbon__bg-hover" points="24 0 0 0 0 32 12.2436611 26.29
         26049 24 31.7728343"></polygon><polygon class="ipc-watchlist-ribbon__bg-shadow" poi
         nts="24 31.7728343 24 33.7728343 12.2436611 28.2926049 0 34 0 32 12.2436611 26.2926
         049"></polygon></svg><div class="ipc-watchlist-ribbon__icon" role="presentation"><s
         vg class="ipc-loader ipc-loader--circle ipc-watchlist-ribbon__loader" data-testid
         ="watchlist-ribbon-loader" height="48px" role="presentation" version="1.1" viewbox
         ="0 0 48 48" width="48px" xmlns="http://www.w3.org/2000/svg"><g class="ipc-loader__
         container" fill="currentColor"><circle class="ipc-loader__circle ipc-loader__circle
         --one" cx="24" cy="9" r="4"></circle><circle class="ipc-loader__circle ipc-loader__
         circle--two" cx="35" cy="14" r="4"></circle><circle class="ipc-loader__circle ipc-l
         oader__circle--three" cx="39" cy="24" r="4"></circle><circle class="ipc-loader__cir
         cle ipc-loader__circle--four" cx="35" cy="34" r="4"></circle><circle class="ipc-loa
         der__circle ipc-loader__circle--five" cx="24" cy="39" r="4"></circle><circle class
         ="ipc-loader__circle ipc-loader__circle--six" cx="13" cy="34" r="4"></circle><circl
         e class="ipc-loader__circle ipc-loader__circle--seven" cx="9" cy="24" r="4"></circl
         e><circle class="ipc-loader__circle ipc-loader__circle--eight" cx="13" cy="14" r
         ="4"></circle></g></svg></div></div><div class="ipc-media ipc-media--poster-27x40 i
         pc-image-media-ratio--poster-27x40 ipc-media--base ipc-media--poster-m ipc-poster__
         poster-image ipc-media__img" style="width:100%"><img alt="Hugh Jackman in Logan (20
         17)" class="ipc-image" loading="lazy" sizes="50vw, (min-width: 480px) 34vw, (min-wi
         dth: 600px) 26vw, (min-width: 1024px) 16vw, (min-width: 1280px) 16vw" src="https://
         m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkE
         yXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX140_CR0,1,140,207_.jpg" srcset="https://m.medi
         a-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqc
         GdeQXVyNjc1NTYyMjg@._V1_QL75_UX140_CR0,1,140,207_.jpg 140w, https://m.media-amazon.
         com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNj
         c1NTYyMjg@._V1_QL75_UX210_CR0,2,210,311_.jpg (https://m.media-amazon.com/images/M/M
         V5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1
         _QL75_UX210_CR0,2,210,311_.jpg) 210w, https://m.media-amazon.com/images/M/MV5BYzc5M
         TU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX
         280_CR0,3,280,414_.jpg (https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi0
         0NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX280_CR0,3,280,
         414_.jpg) 280w" width="140"/></div><a aria-label="View title page for Logan" class
         ="ipc-lockup-overlay ipc-focusable" href="/title/tt3315342/?ref_=sr_i_1"><div class
         ="ipc-lockup-overlay__screen"></div></a></div></div><div class="sc-b0691f29-0 jbYPf
         h"><div class="ipc-title ipc-title--base ipc-title--title ipc-title-link-no-icon ip
         c-title--on-textPrimary sc-b0691f29-9 klOwFB dli-title"><a class="ipc-title-link-wr
         apper" href="/title/tt3315342/?ref_=sr_t_1" tabindex="0"><h3 class="ipc-title__tex
         t">1. Logan</h3></a></div><div class="sc-b0691f29-7 hrgukm dli-title-metadata"><spa
         n class="sc-b0691f29-8 ilsLEX dli-title-metadata-item">2017</span><span class="sc-b
         0691f29-8 ilsLEX dli-title-metadata-item">2h 17m</span><span class="sc-b0691f29-8 i
         lsLEX dli-title-metadata-item">R-16</span></div><span class="sc-b0691f29-1 grHDBY">
         <div class="sc-e2dbc1a3-0 ajrIH sc-b0691f29-2 bhhtyj dli-ratings-container" data-te
         stid="ratingGroup--container"><span aria-label="IMDb rating: 8.1" class="ipc-rating
         -star ipc-rating-star--base ipc-rating-star--imdb ratingGroup--imdb-rating" data-te
         stid="ratingGroup--imdb-rating"><svg class="ipc-icon ipc-icon--star-inline" fill="c
         urrentColor" height="24" role="presentation" viewbox="0 0 24 24" width="24" xmlns
```

```
="http://www.w3.org/2000/svg"><path d="M12 20.1l5.82 3.682c1.066.675 2.37-.322 2.09
-1.584l-1.543-6.926 5.146-4.667c.94-.85.435-2.465-.799-2.567l-6.773-.602L13.29.89a
1.38 1.38 0 0 0-2.581 0l-2.65 6.53-6.774.602C.052 8.126-.453 9.74.486 10.59l5.147
4.666-1.542 6.926c-.28 1.262 1.023 2.26 2.09 1.585L12 20.099z"></path></svg>8.1<spa
n class="ipc-rating-star--voteCount"> (<!-- -->827K<!-- -->)</span></span><button a
ria-label="Rate Logan" class="ipc-rate-button sc-e2dbc1a3-1 jboOQc ratingGroup--use
r-rating ipc-rate-button--unrated ipc-rate-button--base" data-testid="rate-button">
<span class="ipc-rating-star ipc-rating-star--base ipc-rating-star--rate"><svg clas
s="ipc-icon ipc-icon--star-border-inline" fill="currentColor" height="24" role="pre
sentation" viewbox="0 0 24 24" width="24" xmlns="http://www.w3.org/2000/svg"><path
d="M22.724 8.217l-6.786-.587-2.65-6.22c-.477-1.133-2.103-1.133-2.58 0l-2.65 6.234-
6.772.573c-1.234.098-1.739 1.636-.8 2.446l5.146 4.446-1.542 6.598c-.28 1.202 1.023
2.153 2.09 1.511l5.818-3.495 5.819 3.509c1.065.643 2.37-.308 2.089-1.51l-1.542-6.612
5.145-4.446c.94-.81.45-2.348-.785-2.446zm-10.726 8.89l-5.272 3.174 1.402-5.983-4.65
5-4.026 6.141-.531 2.384-5.634 2.398 5.648 6.14.531-4.654 4.026 1.402 5.983-5.286-
3.187z"></path></svg><span class="ipc-rating-star--rate">Rate</span></span></button
></div><span class="sc-b0691f29-11 TmkKM"><span class="sc-b0901df4-0 bcQdDJ metacri
tic-score-box" style="background-color:#54A72A">77</span><span class="metacritic-sc
ore-label">Metascore</span></span></span></div><div class="sc-ab6fa25a-4 ggHbBR dli
-post-element"><button aria-disabled="false" aria-label="See more information about
Logan" class="ipc-icon-button dli-info-icon ipc-icon-button--base ipc-icon-button--
onAccent2" role="button" tabindex="0" title="See more information about Logan"><svg
class="ipc-icon ipc-icon--info" fill="currentColor" height="24" role="presentation"
viewbox="0 0 24 24" width="24" xmlns="http://www.w3.org/2000/svg"><path d="M0 0h24v
24H0V0z" fill="none"></path><path d="M11 7h2v2h-2zm0 4h2v6h-2zm1-9C6.48 2 2 6.48 2
12s4.48 10 10 10 10-4.48 10-10S17.52 2 12 2zm0 18c-4.41 0-8-3.59-8-8s3.59-8 8-8 8
3.59 8 8-3.59 8-8 8z"></path></svg></button></div></div><div class="sc-ab6fa25a-1 b
BwFsP"><div class="ipc-html-content ipc-html-content--base sc-ab6fa25a-0 bhexuD dli
-plot-container" role="presentation"><div class="ipc-html-content-inner-div">In a f
uture where mutants are nearly extinct, an elderly and weary Logan leads a quiet li
fe. But when Laura, a mutant child pursued by scientists, comes to him for help, he
must get her to safety.</div></div></div></div></div></div></li>
```

In [76]:
```
first_movie.li
```

In [77]:
```
first_movie.a
```

Out[77]:
```
<a aria-label="View title page for Logan" class="ipc-lockup-overlay ipc-focusable"
href="/title/tt3315342/?ref_=sr_i_1"><div class="ipc-lockup-overlay__screen"></div>
</a>
```

In [78]:
```
# movie title
first_movie.h3
```

Out[78]:
```
<h3 class="ipc-title__text">1. Logan</h3>
```

In [82]:
```
first_movie.h3.a
```

In [87]:
```
first_name = first_movie.h3.text[3:]
first_name
```

Out[87]:
```
'Logan'
```

In [90]:
```
# the year of the movie release
first_year = first_movie.div.find('span', class_ = 'sc-b0691f29-8 ilsLEX dli-title-m
first_year
```

Out[90]:
```
<span class="sc-b0691f29-8 ilsLEX dli-title-metadata-item">2017</span>
```

In [91]:
```
first_year = first_year.text
first_year
```

Out[91]:
```
'2017'
```

In [108]:
```
# imdb rating
first_movie.find('span', class_ = 'ipc-rating-star ipc-rating-star--base ipc-rating-
```

Out[108]:
```
<span aria-label="IMDb rating: 8.1" class="ipc-rating-star ipc-rating-star--base ip
c-rating-star--imdb ratingGroup--imdb-rating" data-testid="ratingGroup--imdb-ratin
g"><svg class="ipc-icon ipc-icon--star-inline" fill="currentColor" height="24" role
="presentation" viewbox="0 0 24 24" width="24" xmlns="http://www.w3.org/2000/svg"><
path d="M12 20.1l5.82 3.682c1.066.675 2.37-.322 2.09-1.584l-1.543-6.926 5.146-4.667
c.94-.85.435-2.465-.799-2.567l-6.773-.602L13.29.89a1.38 1.38 0 0 0-2.581 0l-2.65 6.
53-6.774.602C.052 8.126-.453 9.74.486 10.59l5.147 4.666-1.542 6.926c-.28 1.262 1.02
3 2.26 2.09 1.585L12 20.099z"></path></svg>8.1<span class="ipc-rating-star--voteCou
nt"> (<!-- -->827K<!-- -->)</span></span>
```

In [128]:
```
import re
first_imdb = first_movie.find('span', class_ = 'ipc-rating-star ipc-rating-star--bas
first_imdb.find(string=re.compile('.'))
```

Out[128]:
```
'8.1'
```

In [282]:
```
# metascore
first_mscore = first_movie.find('span', class_ = 'sc-b0901df4-0 bcQdDJ metacritic-sc
first_mscore = int(first_mscore.text)
print(first_mscore)
```

```
77
```

In [131]:
```
# the number of votes
first_votes = first_movie.find('span', class_ = 'ipc-rating-star--voteCount')
first_votes
```

Out[131]:
```
<span class="ipc-rating-star--voteCount"> (<!-- -->827K<!-- -->)</span>
```

In [136]:
```
first_votes.text[2:-1]
```

Out[136]:
```
'827K'
```

```
In [231]: # List 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 metascore, then extract:
    if container.find('h3', class_ = 'ipc-title__text') is not None:
        # the name
        name = container.h3.text[3:]
        names.append(name)

        # the year
        year = container.find('span', class_ = 'sc-b0691f29-8 ilsLEX dli-title-metad
        years.append(year)

        # the imdb rating
        imdb = first_movie.find('span', class_ = 'ipc-rating-star ipc-rating-star--b
        imdb_ratings.append(imdb.text[:4])

        # the metascore
        m_score = container.find('span', class_ = 'sc-b0901df4-0 bcQdDJ metacritic-s
        metascores.append(m_score)

        # the number of votes
        vote = container.find('span', class_ = 'ipc-rating-star--voteCount')
        votes.append(vote.text[2:-1])
```

In [180]:
```
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: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   movie      50 non-null     object
 1   year       50 non-null     object
 2   imdb       50 non-null     object
 3   metascore  41 non-null     object
 4   votes      50 non-null     object
dtypes: object(5)
memory usage: 2.1+ KB
None
```

Out[180]:

	movie	year	imdb	metascore	votes
0	Logan	2017	8.1	[77]	827K
1	Thor: Ragnarok	2017	8.1	[74]	813K
2	Guardians of the Galaxy Vol. 2	2017	8.1	[67]	756K
3	Dunkirk	2017	8.1	[94]	736K
4	Spider-Man: Homecoming	2017	8.1	[73]	716K
5	Wonder Woman	2017	8.1	[76]	698K
6	Get Out	2017	8.1	[85]	691K
7	Star Wars: Episode VIII - The Last Jedi	2017	8.1	[84]	670K
8	Blade Runner 2049	2017	8.1	[81]	658K
9	Baby Driver	2017	8.1	[86]	605K
10	It	2017	8.1	[69]	603K
11	Coco	2017	8.1	[81]	586K
12	Three Billboards Outside Ebbing, Missouri	2017	8.1	[88]	553K
13	Money Heist	2017–2021	8.1	None	529K
14	John Wick: Chapter 2	2017	8.1	[75]	509K
15	Justice League	2017	8.1	[45]	477K
16	The Shape of Water	2017	8.1	[87]	446K
17	Dark	2017–2020	8.1	None	440K
18	Jumanji: Welcome to the Jungle	2017	8.1	[58]	435K
19	Kingsman: The Golden Circle	2017	8.1	[44]	361K
20	Kong: Skull Island	2017	8.1	[62]	345K
21	Ozark	2017–2022	8.1	None	344K
22	Pirates of the Caribbean: Salazar's Revenge	2017	8.1	[39]	344K
23	Beauty and the Beast	2017	8.1	[65]	333K
24	Mindhunter	2017–2019	8.1	None	333K
25	Lady Bird	2017	8.1	[93]	326K
26	13 Reasons Why	2017–2020	8.1	None	314K
27	Call Me by Your Name	2017	8.1	[94]	313K
28	The Greatest Showman	2017	8.1	[48]	310K
29	Alien: Covenant	2017	8.1	[65]	302K
30	Murder on the Orient Express	2017	8.1	[52]	295K
31	War for the Planet of the Apes	2017	8.1	[82]	280K
32	Wind River	2017	8.1	[73]	279K
33	The Punisher	2017–2019	8.1	None	263K
34	The Handmaid's Tale	2017–	8.1	None	257K
35	Fast & Furious 8	2017	8.1	[56]	253K
36	Life	2017	8.1	[54]	252K

	movie	year	imdb	metascore	votes
37	Mother!	2017	8.1	[76]	249K
38	The Hitman's Bodyguard	2017	8.1	[47]	246K
39	I, Tonya	2017	8.1	[77]	242K
40	King Arthur: Legend of the Sword	2017	8.1	[41]	232K
41	Ghost in the Shell	2017	8.1	[52]	227K
42	Big Little Lies	2017–	8.1	None	223K
43	Darkest Hour	2017	8.1	[75]	220K
44	The End of the F***ing World	2017–2019	8.1	None	218K
45	American Made	2017	8.1	[65]	207K
46	Atomic Blonde	2017	8.1	[63]	206K
47	The Mummy	2017	8.1	[34]	206K
48	Baywatch	2017	8.1	[37]	201K
49	Bright	2017	8.1	[29]	201K

```
In [283]: # script for multiple pages

          from time import time
          from time import sleep
          from random import randint
          from IPython.display import clear_output

          years_url = ['2014', '2015', '2016', '2017', '2018','2019', '2020', '2021', '2022',

          names = []
          years = []
          imdb_ratings = []
          metascores = []
          votes = []

          start_time = time()
          requests = 0
          vote_count_str = '5.5K'

          agent = {"User-Agent": 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleW
          for year_url in years_url:
              url = f"https://www.imdb.com/search/title/?release_date={year_url}-01-01,{year_ur
              print(url)
              response = get(url, headers=agent)
              print(response.url)
              sleep(randint(8,15))

              requests += 1
              elapsed_time = time() - start_time
              print('Request:{}; Frequency: {} requests/s'.format(requests, requests/elapsed_ti
              clear_output(wait = True)

              if response.status_code != 200:
                  print('Request: {}; Status code: {}'.format(requests, response.status_code))

              if requests > 72:
                  print('Number of requests was greater than expected.')
                  break

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

              mv_containers = page_html.find_all('li', class_ = 'ipc-metadata-list-summary-item

              for container in mv_containers:
                  if container.find('span', class_="sc-b0691f29-11 TmkKM") is not None:
                      # movie name
                      name = container.h3.text[3:]
                      names.append(name)

                      # year released
                      year = container.find('span', class_ = 'sc-b0691f29-8 ilsLEX dli-title-me
                      years.append(year)

                      # imdb rating
                      imdb = float(container.find('span', class_='ipc-rating-star ipc-rating-st
                      imdb_ratings.append(imdb)

                      # metascore
                      m_score = int(container.find('span', class_ = 'sc-b0901df4-0 bcQdDJ metac
                      metascores.append(m_score)

                      # vote count
                      vote = container.find('span', class_="ipc-rating-star--voteCount").find(
                      votes.append(vote)

              del response
```

```
https://www.imdb.com/search/title/?release_date=2023-01-01,2023-12-31&sort=num_vote
s,desc (https://www.imdb.com/search/title/?release_date=2023-01-01,2023-12-31&sort=
num_votes,desc)
https://www.imdb.com/search/title/?release_date=2023-01-01,2023-12-31&sort=num_vote
s,desc (https://www.imdb.com/search/title/?release_date=2023-01-01,2023-12-31&sort=
num_votes,desc)
Request:10; Frequency: 0.0647508098266438 requests/s
```

In [284]:
```
movie_ratings = pd.DataFrame({'movie': names,
                              'year': years,
                              'imdb': imdb_ratings,
                              'metascores': metascores,
                              'votes': votes
                              })
print(movie_ratings.info())
movie_ratings.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 402 entries, 0 to 401
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   movie       402 non-null    object
 1   year        402 non-null    object
 2   imdb        402 non-null    float64
 3   metascores  402 non-null    int64
 4   votes       402 non-null    object
dtypes: float64(1), int64(1), object(3)
memory usage: 15.8+ KB
None
```

Out[284]:

	movie	year	imdb	metascores	votes
0	Interstellar	2014	8.7	74	2.1M
1	Guardians of the Galaxy	2014	8.0	76	1.3M
2	Gone Girl	2014	8.1	79	1.1M
3	Whiplash	2014	8.5	89	981K
4	Captain America: The Winter Soldier	2014	7.7	70	896K
5	The Grand Budapest Hotel	2014	8.1	88	883K
6	The Imitation Game	2014	8.0	71	822K
7	X-Men: Days of Future Past	2014	7.9	75	743K
8	John Wick	2014	7.4	68	736K
9	Edge of Tomorrow	2014	7.9	71	733K

```
In [285]: movie_ratings.tail(10)
```

Out[285]:

	movie	year	imdb	metascores	votes
392	La sociedad de la nieve	2023	7.8	72	122K
393	The Marvels	2023	5.6	50	119K
394	Scream VI	2023	6.5	61	118K
395	Fast X	2023	5.8	56	117K
396	Knock at the Cabin	2023	6.1	63	114K
397	Sound of Freedom	2023	7.7	36	111K
398	Asteroid City	2023	6.5	75	110K
399	A Haunting in Venice	2023	6.5	63	109K
400	The Hunger Games: The Ballad of Songbirds & S...	2023	6.8	54	108K
401	The Equalizer 3	2023	6.8	58	107K

```
In [298]: movie_ratings.to_csv(r'C:\Users\Jay-ann Alorro\Downloads\data sci\movie_ratings.csv'
```

The IMDB website has been updated since and a lot of changes have been made. For this reason, I have manipulated some of the syntax given in order to acquire the movie information needed. Additional dates are also added to have more data.

Data Preparation

Example of Data Preparation of movie_rating.csv

Some of these doesn't apply to the dataset because the format of the date have been already updated. I have deleted some and replaced it with some data preparation I think is needed

```
In [287]: movie_ratings['year'].unique()

Out[287]: array(['2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021',
                 '2022', '2023'], dtype=object)
```

```
In [288]: movie_ratings.dtypes

Out[288]: movie          object
          year           object
          imdb          float64
          metascores      int64
          votes          object
          dtype: object
```

In [289]:
```python
movie_ratings['year'] = movie_ratings['year'].astype(int)
```

In [290]:
```python
movie_ratings['year'].unique()
```

Out[290]:
```
array([2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023])
```

In [291]:
```python
movie_ratings.dtypes
```

Out[291]:
```
movie          object
year            int32
imdb          float64
metascores      int64
votes          object
dtype: object
```

In [292]:
```python
movie_ratings.head(10)
```

Out[292]:

	movie	year	imdb	metascores	votes
0	Interstellar	2014	8.7	74	2.1M
1	Guardians of the Galaxy	2014	8.0	76	1.3M
2	Gone Girl	2014	8.1	79	1.1M
3	Whiplash	2014	8.5	89	981K
4	Captain America: The Winter Soldier	2014	7.7	70	896K
5	The Grand Budapest Hotel	2014	8.1	88	883K
6	The Imitation Game	2014	8.0	71	822K
7	X-Men: Days of Future Past	2014	7.9	75	743K
8	John Wick	2014	7.4	68	736K
9	Edge of Tomorrow	2014	7.9	71	733K

In [293]:
```python
movie_ratings.tail(10)
```

Out[293]:

	movie	year	imdb	metascores	votes
392	La sociedad de la nieve	2023	7.8	72	122K
393	The Marvels	2023	5.6	50	119K
394	Scream VI	2023	6.5	61	118K
395	Fast X	2023	5.8	56	117K
396	Knock at the Cabin	2023	6.1	63	114K
397	Sound of Freedom	2023	7.7	36	111K
398	Asteroid City	2023	6.5	75	110K
399	A Haunting in Venice	2023	6.5	63	109K
400	The Hunger Games: The Ballad of Songbirds & S...	2023	6.8	54	108K
401	The Equalizer 3	2023	6.8	58	107K

In [294]:
```
movie_ratings.fillna(0, inplace=True)
```

In [295]:
```
movie_ratings.head(10)
```

Out[295]:

	movie	year	imdb	metascores	votes
0	Interstellar	2014	8.7	74	2.1M
1	Guardians of the Galaxy	2014	8.0	76	1.3M
2	Gone Girl	2014	8.1	79	1.1M
3	Whiplash	2014	8.5	89	981K
4	Captain America: The Winter Soldier	2014	7.7	70	896K
5	The Grand Budapest Hotel	2014	8.1	88	883K
6	The Imitation Game	2014	8.0	71	822K
7	X-Men: Days of Future Past	2014	7.9	75	743K
8	John Wick	2014	7.4	68	736K
9	Edge of Tomorrow	2014	7.9	71	733K

In [296]:
```
movie_ratings.tail(10)
```

Out[296]:

	movie	year	imdb	metascores	votes
392	La sociedad de la nieve	2023	7.8	72	122K
393	The Marvels	2023	5.6	50	119K
394	Scream VI	2023	6.5	61	118K
395	Fast X	2023	5.8	56	117K
396	Knock at the Cabin	2023	6.1	63	114K
397	Sound of Freedom	2023	7.7	36	111K
398	Asteroid City	2023	6.5	75	110K
399	A Haunting in Venice	2023	6.5	63	109K
400	The Hunger Games: The Ballad of Songbirds & S...	2023	6.8	54	108K
401	The Equalizer 3	2023	6.8	58	107K

In [297]:
```
movie_ratings
```

Out[297]:

	movie	year	imdb	metascores	votes
0	Interstellar	2014	8.7	74	2.1M
1	Guardians of the Galaxy	2014	8.0	76	1.3M
2	Gone Girl	2014	8.1	79	1.1M
3	Whiplash	2014	8.5	89	981K
4	Captain America: The Winter Soldier	2014	7.7	70	896K
...	...	...	...	...	...
397	Sound of Freedom	2023	7.7	36	111K
398	Asteroid City	2023	6.5	75	110K
399	A Haunting in Venice	2023	6.5	63	109K
400	The Hunger Games: The Ballad of Songbirds & S...	2023	6.8	54	108K
401	The Equalizer 3	2023	6.8	58	107K

402 rows × 5 columns

In [299]:
```
movie_ratings.to_csv(r'C:\Users\Jay-ann Alorro\Downloads\data sci\movie_ratings.csv'
```

In [ ]:
```
```