Hands On Activity 7.2 (CALINGO)

Name: CALINGO, CHRISTIAN LEI

Section: CPE22S3

Course: Computational Thinking with Python

Course Code: CPE311

Data Gathering Using Webcam

```
import cv2
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)
        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.destroyAllWindows()
      break
```

```
 True
 [[[109 114 141]
   [112 117 144]
   [114 120 143]
   ...
   [111 120 146]
   [112 120 149]
   [113 121 150]]

  [[110 115 142]
   [113 118 145]
   [115 121 144]
   ...
   [111 120 146]
   [112 120 149]
   [113 121 150]]

  [[109 120 140]
```

```
 [113 124 144]
 [113 125 141]
 ...
 [115 122 143]
 [116 122 145]
 [116 122 145]]

...

[[ 81  93 117]
 [ 84  96 120]
 [ 87  96 121]
 ...
 [108 116 133]
 [107 115 132]
 [106 114 131]]

[[ 86  95 120]
 [ 87  96 121]
 [ 88  96 121]
 ...
 [111 118 133]
 [107 116 131]
 [105 114 129]]

[[ 81  90 115]
 [ 81  90 115]
 [ 81  89 114]
 ...
 [114 121 136]
 [110 119 134]
 [108 117 132]]]
True
[[[107 122 137]
  [108 123 138]
  [109 123 141]
  ...
  [115 125 147]
  [112 122 144]
  [110 120 142]]
```

```
!pip3 install sounddevice
!pip3 install wavio
!pip3 install scipy
!apt-get install libportaudio2
```

```
Requirement already satisfied: sounddevice in c:\users\user\anaconda3\lib\site-packages (0.4.6)
Requirement already satisfied: CFFI>=1.0 in c:\users\user\anaconda3\lib\site-packages (from sounddevice) (1.15.1)
Requirement already satisfied: pycparser in c:\users\user\anaconda3\lib\site-packages (from CFFI>=1.0->sounddevice) (2.21)
Requirement already satisfied: wavio in c:\users\user\anaconda3\lib\site-packages (0.0.8)
Requirement already satisfied: numpy>=1.19.0 in c:\users\user\anaconda3\lib\site-packages (from wavio) (1.24.4)
Requirement already satisfied: scipy in c:\users\user\anaconda3\lib\site-packages (1.9.1)
Requirement already satisfied: numpy<1.25.0,>=1.18.5 in c:\users\user\anaconda3\lib\site-packages (from scipy) (1.24.4)
'apt-get' is not recognized as an internal or external command,
operable program or batch file.
```

```
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)

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.

```
!pip install bs4
!pip install requests
```

```
Requirement already satisfied: bs4 in c:\users\user\anaconda3\lib\site-packages (0.0.2)
Requirement already satisfied: beautifulsoup4 in c:\users\user\anaconda3\lib\site-packages (from bs4) (4.11.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\user\anaconda3\lib\site-packages (from beautifulsoup4->bs4) (2.3.1)
Requirement already satisfied: requests in c:\users\user\anaconda3\lib\site-packages (2.28.1)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\user\anaconda3\lib\site-packages (from requests) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\user\anaconda3\lib\site-packages (from requests) (2023.7.22)
Requirement already satisfied: idna<4,>=2.5 in c:\users\user\anaconda3\lib\site-packages (from requests) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\user\anaconda3\lib\site-packages (from requests) (1.26.11)
```

```
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
```

Image Scraping Using Selenium

```
pip install selenium
```

```
Requirement already satisfied: selenium in c:\users\user\anaconda3\lib\site-packages (4.18.1)
Requirement already satisfied: urllib3[socks]<3,>=1.26 in c:\users\user\anaconda3\lib\site-packages (from selenium) (1.26.11)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\user\anaconda3\lib\site-packages (from selenium) (2023.7.22)
Requirement already satisfied: trio~=0.17 in c:\users\user\anaconda3\lib\site-packages (from selenium) (0.25.0)
```

```
Requirement already satisfied: typing_extensions>=4.9.0 in c:\users\user\anaconda3\lib\site-packages (from selenium) (4.10.0)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\user\anaconda3\lib\site-packages (from selenium) (0.11.1)
Requirement already satisfied: exceptiongroup in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.2.0)
Requirement already satisfied: idna in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (3.3)
Requirement already satisfied: cffi>=1.14 in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.15.1)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.1)
Requirement already satisfied: sortedcontainers in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: attrs>=23.2.0 in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (23.2.0)
Requirement already satisfied: outcome in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied: wsproto>=0.14 in c:\users\user\anaconda3\lib\site-packages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied: PySocks!=1.5.7,<2.0,>=1.5.6 in c:\users\user\anaconda3\lib\site-packages (from urllib3[socks]<3,>=1.26->s
Requirement already satisfied: pycparser in c:\users\user\anaconda3\lib\site-packages (from cffi>=1.14->trio~=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\user\anaconda3\lib\site-packages (from wsproto>=0.14->trio-websocket~=0.9->sele
Note: you may need to restart the kernel to use updated packages.
```

```
!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=0CAIQpwVqFwoTCKCa1c6s4-oCFQAAAAAdAAAAABAC&biw=1251&bih=568
driver.get(search_url.format(q='Car'))
def scroll_to_end(driver):
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    ime.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&ved=0CAIQpwVqFwoTCKCa1c6s4-oCFQAAAAAdAAAAABAC&biw=1251&bih
    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,'Q4LuWd')]")
        totalResults=len(thumbnail_results)
        print(f"Found: {totalResults} search results. Extracting links from{results_start}:{totalResults}")

        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_attribute('src'):
                    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\user\anaconda3\lib\site-packages (4.18.1)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\user\anaconda3\lib\site-packages (from selenium) (2023.7.22)
Requirement already satisfied: urllib3[socks]<3,>=1.26 in c:\users\user\anaconda3\lib\site-packages (from selenium) (1.26.11)
Requirement already satisfied: trio~=0.17 in c:\users\user\anaconda3\lib\site-packages (from selenium) (0.25.0)
Requirement already satisfied: typing_extensions>=4.9.0 in c:\users\user\anaconda3\lib\site-packages (from selenium) (4.10.0)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\user\anaconda3\lib\site-packages (from selenium) (0.11.1)
Requirement already satisfied: outcome in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied: cffi>=1.14 in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.15.1)
Requirement already satisfied: attrs>=23.2.0 in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (23.2.0)
Requirement already satisfied: exceptiongroup in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.2.0)
Requirement already satisfied: sortedcontainers in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.1)
Requirement already satisfied: idna in c:\users\user\anaconda3\lib\site-packages (from trio~=0.17->selenium) (3.3)
Requirement already satisfied: wsproto>=0.14 in c:\users\user\anaconda3\lib\site-packages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied: PySocks!=1.5.7,<2.0,>=1.5.6 in c:\users\user\anaconda3\lib\site-packages (from urllib3[socks]<3,>=1.26->s
Requirement already satisfied: pycparser in c:\users\user\anaconda3\lib\site-packages (from cffi>=1.14->trio~=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\user\anaconda3\lib\site-packages (from wsproto>=0.14->trio-websocket~=0.9->sele
'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)
~\AppData\Local\Temp\ipykernel_21716\42671164.py in <module>
     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=en&ved=0CAIQpwVqFwoTCKCa1c6s4-
oCFQAAAAAdAAAAABAC&biw=1251&bih=568"
     24 driver.get(search_url.format(q='Car'))

TypeError:   init  () got an unexpected keyword argument 'chrome_options'
```

Web Scraping of Movies Information using BeautifulSoup

```
from requests import get
url = 'https://www.imdb.com/search/title?release_date=2017&sort=num_votes,desc&page=1'
agent = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36"}
response = get(url,headers = agent)
```

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

```
bs4.BeautifulSoup
```

```
movie_containers = html_soup.find_all('div', class_ = 'ipc-metadata-list-summary-item__c')
print(type(movie_containers))
print(len(movie_containers))
```

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

Selecting the following:

• The name of the movie

• The year of release.

• The IMDB rating.

• The Metascore.

• The number of votes

```
first_movie = movie_containers[0]
```

```
first_movie.h3
```

```
<h3 class="ipc-title__text">1. Logan</h3>
```

```
first_movie.a
```

```
<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>
```

Name of the first Movie

```
first_name = first_movie.find('h3', class_= "ipc-title__text").text[3:]
first_name
```

```
'Logan'
```

Year Release of the first Movie

```
first_year = first_movie.find('span', class_="sc-b0691f29-8 ilsLEX dli-title-metadata-item").text
first_year
```

```
'2017'
```

Ratings of the first movie

```
first_rate = first_movie.find('span',class_="ipc-rating-star ipc-rating-star--base ipc-rating-star--imdb ratingGroup--imdb-rating").text[:3]
first_rate
```

```
'8.1'
```

Metascore of the first movie

```
first_score = first_movie.find('span', class_='sc-b0901df4-0 bcQdDJ metacritic-score-box').text
first_score
```

```
'77'
```

Votes of the first movie

```
first_votes = first_movie.find('span', class_='ipc-rating-star--voteCount').text[1:]
first_votes
```

```
'(827K)'
```

Trying the Second Movie

Accessing the movie through the movie_containers

```
second_movie = movie_containers[1]
```

Name of the Second Movie

```
second_name = second_movie.find('h3', class_= "ipc-title__text").text[3:]
second_name
```

```
'Thor: Ragnarok'
```

Year Release of the Second Movie

```
second_year = second_movie.find('span', class_= "sc-b0691f29-8 ilsLEX dli-title-metadata-item").text
second_year
```

```
'2017'
```

IMDB Rating of the Second Movie

```
second_rate = second_movie.find('span',class_="ipc-rating-star ipc-rating-star--base ipc-rating-star--imdb ratingGroup--imdb-rating").text[:
second_rate
```

```
'7.9'
```

Metascore of the Second Movie

```
second_score = second_movie.find('span', class_='sc-b0901df4-0 bcQdDJ metacritic-score-box').text
second_score
```

```
'74'
```

Votes of the Second Movie

```
second_votes = second_movie.find('span', class_='ipc-rating-star--voteCount').text[1:]
second_votes
```

```
'(813K)'
```

Script

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

for container in movie_containers:
    if container.find('span', class_='sc-b0901df4-0 bcQdDJ metacritic-score-box') is not None:

        name = container.find('h3', class_= "ipc-title__text").text[3:]
        names.append(name)

        year = container.find('span', class_= "sc-b0691f29-8 ilsLEX dli-title-metadata-item").text
        years.append(year)

        imdb_rating = float(container.find('span',class_="ipc-rating-star ipc-rating-star--base ipc-rating-star--imdb ratingGroup--imdb-rati
        imdb_ratings.append(imdb_rating)

        metascore = int(container.find('span', class_='sc-b0901df4-0 bcQdDJ metacritic-score-box').text)
        metascores.append(metascore)

        vote = container.find('span', class_='ipc-rating-star--voteCount').text[1:]
        votes.append(vote)

print(names)
print(years)
print(imdb_ratings)
print(metascores)
print(votes)
```

```
['Logan', 'Thor: Ragnarok', 'Guardians of the Galaxy Vol. 2', 'Dunkirk', 'Spider-Man: Homecoming', 'Wonder Woman', 'Get Out', 'Star Wars
['2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017', '2017',
[8.1, 7.9, 7.6, 7.8, 7.4, 7.3, 7.8, 6.9, 8.0, 7.5, 7.3, 8.4, 8.1, 7.4, 6.1, 7.3, 6.9, 6.7, 6.7, 6.5, 7.1, 7.4, 7.8, 7.5, 6.4, 6.5, 7.4,
[77, 74, 67, 94, 73, 76, 85, 84, 81, 86, 69, 81, 88, 75, 45, 87, 58, 44, 62, 39, 65, 93, 94, 48, 65, 52, 82, 73, 56, 54, 76, 47, 77, 41,
['(827K)', '(813K)', '(756K)', '(736K)', '(716K)', '(698K)', '(691K)', '(670K)', '(658K)', '(605K)', '(603K)', '(586K)', '(553K)', '(509
```

```
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: 41 entries, 0 to 40
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   movie      41 non-null     object
 1   year       41 non-null     object
 2   imdb       41 non-null     float64
 3   metascore  41 non-null     int64
 4   votes      41 non-null     object
dtypes: float64(1), int64(1), object(3)
memory usage: 1.7+ KB
None
```

	movie	year	imdb	metascore	votes
0	Logan	2017	8.1	77	(827K)
1	Thor: Ragnarok	2017	7.9	74	(813K)
2	Guardians of the Galaxy Vol. 2	2017	7.6	67	(756K)
3	Dunkirk	2017	7.8	94	(736K)
4	Spider-Man: Homecoming	2017	7.4	73	(716K)
5	Wonder Woman	2017	7.3	76	(698K)
6	Get Out	2017	7.8	85	(691K)
7	Star Wars: Episode VIII - The Last Jedi	2017	6.9	84	(670K)
8	Blade Runner 2049	2017	8.0	81	(658K)
9	Baby Driver	2017	7.5	86	(605K)
10	It	2017	7.3	69	(603K)
11	Coco	2017	8.4	81	(586K)
12	Three Billboards Outside Ebbing, Missouri	2017	8.1	88	(553K)
13	John Wick: Chapter 2	2017	7.4	75	(509K)
14	Justice League	2017	6.1	45	(477K)
15	The Shape of Water	2017	7.3	87	(446K)
16	Jumanji: Welcome to the Jungle	2017	6.9	58	(436K)
17	Kingsman: The Golden Circle	2017	6.7	44	(361K)
18	Kong: Skull Island	2017	6.7	62	(345K)
19	Pirates of the Caribbean: Salazar's Revenge	2017	6.5	39	(344K)
20	Beauty and the Beast	2017	7.1	65	(333K)
21	Lady Bird	2017	7.4	93	(326K)
22	Call Me by Your Name	2017	7.8	94	(313K)
23	The Greatest Showman	2017	7.5	48	(310K)
24	Alien: Covenant	2017	6.4	65	(302K)
25	Murder on the Orient Express	2017	6.5	52	(295K)
26	War for the Planet of the Apes	2017	7.4	82	(280K)
27	Wind River	2017	7.7	73	(279K)
28	Fast & Furious 8	2017	6.6	56	(253K)
29	Life	2017	6.6	54	(252K)
30	Mother!	2017	6.6	76	(249K)
31	The Hitman's Bodyguard	2017	6.9	47	(246K)
32	I, Tonya	2017	7.5	77	(242K)
33	King Arthur: Legend of the Sword	2017	6.7	41	(232K)
34	Ghost in the Shell	2017	6.3	52	(227K)
35	Darkest Hour	2017	7.4	75	(220K)
36	American Made	2017	7.1	65	(207K)
37	Atomic Blonde	2017	6.7	63	(206K)
38	The Mummy	2017	5.4	34	(206K)

38	The Mummy	2017	5.4	34 (206K)
39	Baywatch	2017	5.5	37 (201K)
40	Bright	2017	6.3	29 (201K)

```
from time import time
from time import sleep
from requests import get
from random import randint

from IPython.core.display import clear_output
pages = ['1','2','3','4','5']
years_url = [ '2015','2016','2017', '2018', '2019', '2023']

# 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:

    # Make a get request
    url = f'https://www.imdb.com/search/title?release_date={year_url}-01-01,{year_url}-12-31&sort=num_votes,desc&page=1'
    agent = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36'
    response = get(url,headers = agent)
    #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/elapsed_time))
    clear_output(wait = True)

    # Throw a warning for non-200 status codes
    if response.status_code != 200:
        print('Request: {}; Status code: {}'.format(requests, response.status_code))

    # Break the loop if the number of requests is greater than expected
    if requests > 72:
        print('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_ = 'sc-ab6fa25a-3 bVYfLY dli-parent')

    # For every movie of these 50
    for container in mv_containers:
        # If the movie has a Metascore, then:
        if container.find('span', class_ = 'sc-b0901df4-0 bcQdDJ metacritic-score-box') is not None:
            # Scrape the name
            name = container.find('h3',class_='ipc-title__text').text[3:]
            names.append(name)

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

            # Scrape the IMDB rating
            imdb = container.find('span', class_ = 'ipc-rating-star ipc-rating-star--base ipc-rating-star--imdb ratingGroup--imdb-rating').
            imdb_ratings.append(float(imdb))

            # Scrape the Metascore
            m_score = container.find('span', class_ = 'sc-b0901df4-0 bcQdDJ metacritic-score-box').text
            metascores.append(int(m_score))

            # Scrape the number of votes
```

```
            vote = container.find('span', class_ = 'ipc-rating-star--voteCount').text[2:-1]
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: 250 entries, 0 to 249
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   movie      250 non-null    object 
 1   year       250 non-null    object 
 2   imdb       250 non-null    float64
 3   metascore  250 non-null    int64  
 4   votes      250 non-null    object 
dtypes: float64(1), int64(1), object(3)
memory usage: 9.9+ KB
None
```

	movie	year	imdb	metascore	votes
0	Mad Max: Fury Road	2015	8.1	90	1.1M
1	Star Wars: Episode VII - The Force Awakens	2015	7.8	80	971K
2	The Martian	2015	8.0	80	919K
3	Avengers: Age of Ultron	2015	7.3	66	918K
4	The Revenant	2015	8.0	76	870K
5	Inside Out	2015	8.1	94	781K
6	Ant-Man	2015	7.2	64	719K