

Handout Data Collection II

With Handout_Data Collection I, we learned to use Python to collect a movie data from a website. Because the procedures and the repeated code patterns are simple, it is relatively easy to collect data. However, the code patterns in websites do not always repeat even for the same information. Such different code patterns for the same information often cause some errors in data collection. In this section, we are going to learn about how to handle such errors and how to export the data you collect to the working directory.

1. Create a new Python file as you did in Data Collection I. Please check the site below for your reference.

https://chunshengj.github.io/579-class/data_analysis/Install Anaconda Jupyter Package.html

2. Import packages, set working directory, and scrape the movie information

```
import os
import requests
import bs4
import pandas as pd

os.chdir("/Users/sheng/Jupyter/AESHM_579") # for Mac
# os.chdir("C:\\\\Users\\\\sheng\\\\Jupyter\\\\AESHM_579") # for Windows

df='https://www.boxofficemojo.com/year/2023/?ref_=bo_y1_table_1'
page = requests.get(df)

bs_obj = bs4.BeautifulSoup(page.content, "html.parser")
movie = bs_obj.findAll("tr")
```

3. You are asked to collect two more components (number of theaters and release date). Here are the additional command lines for the data collection.

- 1) Create empty containers which will accumulate values for these two columns

```
movie_title = []
movie_gross = []
movie_theaters = []
movie_date = []
```

- 2) Make the commands to collect the number of theaters and release date in the loop

- a. Use the repeated elements for these components (Right mouse click on the webpage => Inspect Element)
- b. Use the “text” function

```
<tr>
  <td class="a-text-right mojo-header-column mojo-truncate mojo-field-type-rank mojo-sort-column" style="width: 49px; height: 31px; min-width: 49px; min-height: 31px;">1</td>
  ><td class="a-text-left mojo-field-type-release mojo-cell-wide" style="width: 352px; height: 31px; min-width: 352px; min-height: 31px;">...</td>
  <td class="a-text-left mojo-field-type-genre hidden" style="width: 0px; height: 0px; min-width: 0px; min-height: 0px;">...</td>
  <td class="a-text-right mojo-field-type-money hidden" style="width: 0px; height: 0px; min-width: 0px; min-height: 0px;">...</td>
  <td class="a-text-right mojo-field-type-duration hidden" style="width: 0px; height: 0px; min-width: 0px; min-height: 0px;">...</td>
  <td class="a-text-right mojo-field-type-money mojo-estimatable" style="width: 99px; height: 31px; min-width: 99px; min-height: 31px;">$570,554,370</td>
  <td class="a-text-right mojo-field-type-positive_integer" style="width: 73px; height: 31px; min-width: 73px; min-height: 31px;">4,371</td> == $0
  <td class="a-text-right mojo-field-type-money mojo-estimatable" style="width: 99px; height: 31px; min-width: 99px; min-height: 31px;">$570,554,370</td>
  <td class="a-text-left mojo-field-type-date a-nowrap" style="width: 100px; height: 31px; min-width: 100px; min-height: 31px;">Apr 5</td>
  ><td class="a-text-left mojo-field-type-money" style="width: 203px; height: 31px; min-width: 203px; min-height: 31px;">...</td>
  <td class="a-text-right mojo-field-type-boolean hidden" style="width: 0px; height: 0px; min-width: 0px; min-height: 0px;">false</td>

for item in movie[1:201]:
    title = item.find("a", {"class": "a-link-normal"}).text
    gross = item.find("td", {"class": "a-text-right mojo-field-type-money mojo-estimatable"}).text
    theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text
    date = item.find("td", {"class": "a-text-left mojo-field-type-date a-nowrap"}).text
```

- 3) Store the collected values in the containers

- a. Use the “append” function

```
movie_title.append(title)
movie_gross.append(gross)
movie_theaters.append(theaters)
movie_date.append(date)
```

- b. Add the collected values to a dataframe

```
iter_df = pd.DataFrame({"Title": movie_title, "Gross": movie_gross, "Theaters": movie_theaters, "Date": movie_date})
print(iter_df)
```

	Title	Gross	Theaters	Date
0	The Super Mario Bros. Movie	\$570,554,370	4,371	Apr 5
1	Guardians of the Galaxy Vol. 3	\$336,521,512	4,450	May 5
2	Avatar: The Way of Water	\$283,067,859	4,340	Dec 16
3	Spider-Man: Across the Spider-Verse	\$232,339,759	4,332	Jun 2
4	The Little Mermaid	\$232,317,534	4,320	May 26
..
213	Let It Be Morning	\$42,719	26	Feb 3
214	Close to Vermeer	\$41,039	23	May 26
215	Cinema Sabaya	\$39,668	4	Feb 10
216	Full Time	\$38,658	8	Feb 3
217	MindReader	\$37,965	15	Oct 24

4. Fix errors when we extract movie information different from what we want

- 1) Let's try to add the Distributor information to the table.

- a. Create a column for Distributor in the table
 - b. Create a list for the values of Distributor

```

movie_title = []
movie_gross = []
movie_theaters = []
movie_date = []
movie_distributor = []

```

- c. Check the element for Distributor
- Right mouse click on Distributor

Rank	Release	Gross	Theaters	Total Gross	Release Date	Distributor
1	The Super Mario Bros. Movie	\$570,554,370	4,371	\$570,554,370	Apr 5	Universal Pictures
2	Guardians of the Galaxy Vol. 3	\$336,521,512	4,450	\$336,521,512	May 5	Walt Disney Studios Motion Pictures
3	Avatar: The Way of Water	\$283,067,859	4,340	\$684,075,767	Dec 16	20th Century Studios
4	Spider-Man: Across the Spider-Verse	\$232,339,759	4,332	\$232,339,759	Jun 2	Columbia Pictures
5	The Little Mermaid	\$232,317,534	4,320	\$232,317,534	May 26	Walt Disney Studios Motion Pictures

```

<td class="a-text-right mojo-field-type-money mojo-estimatable" style="width: 99px; height: 31px; min-width: 99px; min-height: 31px;">$570,554,370</td>
<td class="a-text-right mojo-field-type-positive_integer" style="width: 73px; height: 31px; min-width: 73px; min-height: 31px;">4,371</td>
<td class="a-text-right mojo-field-type-money mojo-estimatable" style="width: 99px; height: 31px; min-width: 99px; min-height: 31px;">$570,554,370</td>
<td class="a-text-left mojo-field-type-date a-nowrap" style="width: 100px; height: 31px; min-width: 100px; min-height: 31px;">Apr 5</td>
<td class="a-text-left mojo-field-type-studio" style="width: 203px; height: 31px; min-width: 203px; min-height: 31px;">
    <a class="a-link-normal" target="_blank" rel="noopener" href="https://pro.imdb.com/company/co0005073/boxoffice/?view=releases&ref_=mojo_yld_table_1&rf=mojo_yld_table_1"> == $0
        "Universal Pictures"
        ><img alt="link icon" class="mojo-new-window-svg" href="https://pro.imdb.com/company/co0005073/boxoffice/?view=releases&ref_=mojo_yld_table_1&rf=mojo_yld_table_1" style="vertical-align: middle;"/>
    </a>
</td>

```

- d. Use the element to write a command to collect the information of Distributor

```

for item in movie[1:201]:
    title = item.find("a", {"class": "a-link-normal"}).text
    gross = item.find("td", {"class": "a-text-right mojo-field-type-money mojo-estimatable"}).text
    theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text
    date = item.find("td", {"class": "a-text-left mojo-field-type-date a-nowrap"}).text
    studio = item.find("a", {"class": "a-link-normal"}).text

```

- e. Add the collected values to the table

```

iter_df = pd.DataFrame({
    "Title": movie_title,
    "Gross": movie_gross,
    "Theaters": movie_theaters,
    "Date": movie_date,
    "Distributor": movie_distributor
})
print(iter_df)

```

- f. Entire commands

```

import os
import requests
import bs4
import pandas as pd

os.chdir("/Users/sheng/Jupyter/AESHM_579") # for Mac
# os.chdir("C:\\\\Users\\\\sheng\\\\Jupyter\\\\AESHM_579") # for Windows

df='https://www.boxofficemojo.com/year/2023/?ref_=bo_yl_table_1'
page = requests.get(df)

bs_obj = bs4.BeautifulSoup(page.content, "html.parser")
movie = bs_obj.findAll("tr")

movie_title = []
movie_gross = []
movie_theaters = []
movie_date = []
movie_distributor = []

for item in movie[1:201]:
    title = item.find("a", {"class": "a-link-normal"}).text
    gross = item.find("td", {"class": "a-text-right mojo-field-type-money mojo-estimatable"}).text
    theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text
    date = item.find("td", {"class": "a-text-left mojo-field-type-date a-nowrap"}).text
    studio = item.find("a", {"class": "a-link-normal"}).text

    movie_title.append(title)
    movie_gross.append(gross)
    movie_theaters.append(theaters)
    movie_date.append(date)
    movie_distributor.append(studio)

iter_df = pd.DataFrame({"Title": movie_title, "Gross": movie_gross, "Theaters": movie_theaters,
                       "Date": movie_date, "Distributor": movie_distributor})
print(iter_df)

```

g. Returns of the commands

- i. Our results are wrong because the Distributor column has movie titles instead of distributor names. So, we need check if our command for Distributor is correct. We can run each command to check the issue.

	Title	Gross	Theaters	Date
0	The Super Mario Bros. Movie	\$570,554,370	4,371	Apr 5
1	Guardians of the Galaxy Vol. 3	\$336,521,512	4,450	May 5
2	Avatar: The Way of Water	\$283,067,859	4,340	Dec 16
3	Spider-Man: Across the Spider-Verse	\$232,339,759	4,332	Jun 2
4	The Little Mermaid	\$232,317,534	4,320	May 26
..
195	Let It Be Morning	\$42,719	26	Feb 3
196	Close to Vermeer	\$41,039	23	May 26
197	Cinema Sabaya	\$39,668	4	Feb 10
198	Full Time	\$38,658	8	Feb 3
199	MindReader	\$37,965	15	Oct 24

	Distributor
0	The Super Mario Bros. Movie
1	Guardians of the Galaxy Vol. 3
2	Avatar: The Way of Water
3	Spider-Man: Across the Spider-Verse
4	The Little Mermaid
..	...
195	Let It Be Morning
196	Close to Vermeer
197	Cinema Sabaya
198	Full Time
199	MindReader

- h. Run the initial commands before the loop

```
import os
import requests
import bs4
import pandas as pd

os.chdir("/Users/sheng/Jupyter/AESHM_579") # for Mac
# os.chdir("C:\\\\Users\\\\sheng\\\\Jupyter\\\\AESHM_579") # for Windows

df='https://www.boxofficemojo.com/year/2023/?ref_=bo_y1_table_1'
page = requests.get(df)

bs_obj = bs4.BeautifulSoup(page.content, "html.parser")
movie = bs_obj.findAll("tr")
```

- i. Check if our command can collect the distributor information from the individual movie content

- Do not use the loop for this
- Just check the command for an individual movie

```
import os
import requests
import bs4
import pandas as pd

os.chdir("/Users/sheng/Jupyter/AESHM_579") # for Mac
# os.chdir("C:\\\\Users\\\\sheng\\\\Jupyter\\\\AESHM_579") # for Windows

df='https://www.boxofficemojo.com/year/2023/?ref_=bo_y1_table_1'
page = requests.get(df)

bs_obj = bs4.BeautifulSoup(page.content, "html.parser")
movie = bs_obj.findAll("tr")

movie[1].find("a", {"class": "a-link-normal"}).text
'The Super Mario Bros. Movie'
```

- The command returns the movie title. It means that the movie title has been stored because the same element is used for Title and Distributor, and Title is written first. To fix this problem, we should know how many components are stored with this element and where the distributor information is placed among the same elements.
- Use “findAll” function to collect all the content using the “a” element with the “class” attribute and the “a-link-normal” attribute value

```
movie[1].findAll("a", {"class": "a-link-normal"})
```

[The Super Mario Bros. Movie](/release/r11930593025/?ref_=bo_yld_table_1),
[Universal Pictures](https://pro.imdb.com/company/co0005073/boxoffice/?view=releases&ref_=mojo_yld_table_1&rf=mojo_yld_table_1)<svg class="mojo-new-window-svg" viewBox="0 0 32 32" xmlns="http://www.w3.org/2000/svg">
<path d="M24,15.5725113,3V23.5A3.5A3.50424,3.50424,0,0,1,23.5,27H8.5A3.50424,3.50424,0,0,1,5,23.5V8.5A3.50424,3.50424,0,0,1,8.5,5h4.9275513,3H8.5a.50641.50641,0,0,0-.5v15a.50641.50641,0,0,0,.5h15a.50641.50641,0,0,0,.5-.5ZM19.81952,8.56372,12.8844,17.75a.49989.49989,0,0,0,.04547.654791.66534.66528a.49983.49983,0,0,0,.65479.0455319.18628-6.93518,2.12579,2.12585a.5.5,0,0,0,.84741-.2752611.48273-9.35108a.50006.50006,0,0,0-.57214-.57214L17.969,5.59058a.5.5,0,0,0-.27526.84741z">/</path></svg>

First content

Second content

- i. There are two observations collected. The first one includes the title information while the second one has the distributor information.
- k. To collect the distributor information from the second observation, we need to specify where the data we want is located in.
 - i. You should remember that the Python data starts from 0
 - ii. The first observation can be retrieved with the following command:

```
movie[1].findAll("a", {"class": "a-link-normal"})[0]
```

```
movie[1].findAll("a", {"class": "a-link-normal"})[0]
```

[The Super Mario Bros. Movie](/release/r11930593025/?ref_=bo_yld_table_1)

```
movie[1].findAll("a", {"class": "a-link-normal"})[1]
```

[Universal Pictures](https://pro.imdb.com/company/co0005073/boxoffice/?view=releases&ref_=mojo_yld_table_1&rf=mojo_yld_table_1)<svg class="mojo-new-window-svg" viewBox="0 0 32 32" xmlns="http://www.w3.org/2000/svg">
<path d="M24,15.5725113,3V23.5A3.5A3.50424,3.50424,0,0,1,23.5,27H8.5A3.50424,3.50424,0,0,1,5,23.5V8.5A3.50424,3.50424,0,0,1,8.5,5h4.9275513,3H8.5a.50641.50641,0,0,0-.5v15a.50641.50641,0,0,0,.5h15a.50641.50641,0,0,0,.5-.5ZM19.81952,8.56372,12.8844,17.75a.49989.49989,0,0,0,.04547.654791.66534.66528a.49983.49983,0,0,0,.65479.0455319.18628-6.93518,2.12579,2.12585a.5.5,0,0,0,.84741-.2752611.48273-9.35108a.50006.50006,0,0,0-.57214-.57214L17.969,5.59058a.5.5,0,0,0-.27526.84741z">/</path></svg>

- iii. The second observation can be retrieved with the following command:
- iv. Then, collect the content in text within the second observation

```
movie[1].findAll("a", {"class": "a-link-normal"})[1].text
```

'Universal Pictures\n\n'

- l. The return shows the collected distributor information includes unnecessary strings "\n\n" So we need to separate the distributor information from these unnecessary strings.

- i. Use "split" function

```
movie[1].findAll("a", {"class": "a-link-normal"})[1].text
```

'Universal Pictures\n\n'

```
movie[1].findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")
```

['Universal Pictures', '']

ii. Ask ChatGPT how “split” function works

The screenshot shows a ChatGPT interface. The user asks, "What is 'split' function in Python?" ChatGPT responds: "In Python, the 'split()' function is a built-in method that allows you to split a string into a list of substrings based on a specified delimiter. The split operation divides the original string at each occurrence of the delimiter and returns the substrings as elements in a list." Below this, ChatGPT provides the basic syntax: "Here's the basic syntax of the split() function:". An example code snippet is shown in a code editor window:

```
python
sentence = "Hello, how are you today?"
words = sentence.split() # Using default delimiter (whitespace)
print(words)
# Output: ['Hello,', 'how', 'are', 'you', 'today?']
```

iii. Collected data is divided into two parts: 1) Universal Pictures and blank ''

m. Then, extract the first part

```
movie[1].findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")  
'Universal Pictures'
```

- i. We can extract what we want (distributor information only)
- ii. Use this updated command for Distributor in the loop

5. Fix errors when we have error messages caused in the loop

- 1) Even though you update the command for Distributor in the loop, the error message may pop up (see the below).

```
for item in movie[1:201]:  
    title = item.find("a", {"class": "a-link-normal"}).text  
    gross = item.find("td", {"class": "a-text-right mojo-field-type-money mojo-estimatable"}).text  
    theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text  
    date = item.find("td", {"class": "a-text-left mojo-field-type-date a-nowrap"}).text  
    studio = item.findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")[0]  
  
    movie_title.append(title)  
    movie_gross.append(gross)  
    movie_theaters.append(theaters)  
    movie_date.append(date)  
    movie_distributor.append(studio)  
  
iter_df = pd.DataFrame({"Title": movie_title, "Gross": movie_gross, "Theaters": movie_theaters,  
                      "Date": movie_date, "Distributor": movie_distributor})  
print(iter_df)
```

```
-----  
IndexError                                                 Traceback (most recent call last)  
Cell In[29], line 26  
24     theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text  
25     date = item.find("td", {"class": "a-text-left mojo-field-type-date a-nowrap"}).text  
---> 26     studio = item.findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")[0]  
28     movie_title.append(title)  
29     movie_gross.append(gross)  
  
IndexError: list index out of range
```

The error message says that we have a problem in line 26.

- 2) We know that the command for Distributor works for the first movie. But it may not work in the loop. It means that there is an unusual case that this command cannot apply to. So, we should find the movies that caused this error message.

- a. Print the title when we successfully complete data collection for a movie

```
for item in movie[1:201]:  
    title = item.find("a", {"class": "a-link-normal"}).text  
    gross = item.find("td", {"class": "a-text-right mojo-field-type-money mojo-estimatable"}).text  
    theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text  
    date = item.find("td", {"class": "a-text-left mojo-field-type-date a-nowrap"}).text  
    studio = item.findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")[0]  
  
    print(title)  
  
    movie_title.append(title)  
    movie_gross.append(gross)  
    movie_theaters.append(theaters)  
    movie_date.append(date)  
    movie_distributor.append(studio)
```

```

The Super Mario Bros. Movie
Guardians of the Galaxy Vol. 3
Avatar: The Way of Water
Spider-Man: Across the Spider-Verse
The Little Mermaid
Ant-Man and the Wasp: Quantumania
John Wick: Chapter 4
Creed III
Fast X
Puss in Boots: The Last Wish
Scream VI
M3GAN
Dungeons & Dragons: Honor Among Thieves
Evil Dead Rise
Transformers: Rise of the Beasts
Cocaine Bear
A Man Called Otto
Shazam! Fury of the Gods

```

```

-----
IndexError                                     Traceback (most recent call last)
Cell In[30], line 26
  24 theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text
  25 date = item.find("td", {"class": "a-text-left mojo-field-type-date anowrap"}).text
--> 26 studio = item.findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")[0]
  28 print(title)
  30 movie_title.append(title)

IndexError: list index out of range

```

- b. In this case, the return above shows the command for Distributor was working for many movies before the run stopped.
- c. In my case, after we collected the distributor information of the movie “Shazam! Fury of the Gods,” the error arose.
- d. Then, we need to check the next movie after “Shazam! Fury of the Gods” and understand the data structure of that movie.

16 Cocaine Bear	\$64,388,510	3,571	\$64,388,510	Feb 24	Universal Pictures
17 A Man Called Otto	\$64,228,358	3,957	\$64,267,657	Dec 30	Columbia Pictures
18 Shazam! Fury of the Gods	\$57,638,006	4,071	\$57,638,006	Mar 17	Warner Bros.
19 Air	\$52,460,106	3,507	\$52,460,106	Apr 5	-
20 Jesus Revolution	\$52,102,987	2,575	\$52,102,987	Feb 24	Lionsgate

- i. We can find that there is no distributor information of the movie “Air”
- ii. Inspect its element. We can see that there is no “a” element under the “td” element that has distributor information.

```

▼<tr>
  <td class="a-text-right mojo-header-column mojo-truncate mojo-field-type-rank mojo-sort-column" style="width: 49px; height: 31px; min-width: 49px; min-height: 31px;">>19</td>
  ><td class="a-text-left mojo-field-type-release mojo-cell-wide" style="width: 352px; height: 31px; min-width: 352px; min-height: 31px;"><span></span></td>
  <td class="a-text-right mojo-field-type-money hidden" style="width: 0px; height: 0px; min-width: 0px; min-height: 0px;"></td>
  <td class="a-text-right mojo-field-type-duration hidden" style="width: 0px; height: 0px; min-width: 0px; min-height: 0px;"></td>
  <td class="a-text-right mojo-field-type-money mojo-estimatable" style="width: 99px; height: 31px; min-width: 99px; min-height: 31px;">$52,460,106</td>
  <td class="a-text-right mojo-field-type-positive_integer" style="width: 73px; height: 31px; min-width: 73px; min-height: 31px;">3,507</td>
  <td class="a-text-right mojo-field-type-money mojo-estimatable" style="width: 99px; height: 31px; min-width: 99px; min-height: 31px;">$52,460,106</td>
  <td class="a-text-left mojo-field-type-date anowrap" style="width: 100px; height: 31px; min-width: 100px; min-height: 31px;">Apr 5</td>
  <td class="a-text-left mojo-field-type-studio" style="width: 203px; height: 31px; min-width: 203px; min-height: 31px;">=&#039;0
  <td class="a-text-right mojo-field-type-boolean hidden" style="width: 0px; height: 0px; min-width: 0px; min-height: 0px;"></td>

```

- In this situation, we should detect the error type and manually put a blank as the value of Distributor
- Use the “try” and “except” function
- Ask ChatGPT how “try” and “except” work in Python

How does "try" and "except" work in Python?

In Python, the "try" and "except" statements are used together to handle exceptions or errors that may occur during the execution of a program. They provide a way to gracefully handle errors and prevent the program from crashing.

The basic syntax of the "try-except" block is as follows:

```
python
try:
    # Code that might raise an exception
    # ...
except ExceptionType:
    # Code to handle the exception
    # ...
```

- Need to know the error type: IndexError
- The argument says that try to run the command: studio = item.findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")[0] and that if IndexError is detected, print “no studio information” and the value of the studio variable is none or a blank.

```
for item in movie[1:201]:
    title = item.find("a", {"class": "a-link-normal"}).text
    gross = item.find("td", {"class": "a-text-right mojo-field-type-money mojo-estimatable"}).text
    theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text
    date = item.find("td", {"class": "a-text-left mojo-field-type-date a-nowrap"}).text

    try:
        studio = item.findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")[0]
    except IndexError as e:
        # IndexError: list index out of range
        print("no studio information")
        studio = None

    movie_title.append(title)
    movie_gross.append(gross)
    movie_theaters.append(theaters)
    movie_date.append(date)
    movie_distributor.append(studio)

iter_df = pd.DataFrame({"Title": movie_title, "Gross": movie_gross, "Theaters": movie_theaters,
                       "Date": movie_date, "Distributor": movie_distributor})
print(iter_df)
```

	Title	Gross	Theaters	Date
0	The Super Mario Bros. Movie	\$570,554,370	4,371	Apr 5
1	Guardians of the Galaxy Vol. 3	\$336,521,512	4,450	May 5
2	Avatar: The Way of Water	\$283,067,859	4,340	Dec 16
3	Spider-Man: Across the Spider-Verse	\$232,339,759	4,332	Jun 2
4	The Little Mermaid	\$232,317,534	4,320	May 26
..
195	Let It Be Morning	\$42,719	26	Feb 3
196	Close to Vermeer	\$41,039	23	May 26
197	Cinema Sabaya	\$39,668	4	Feb 10
198	Full Time	\$38,658	8	Feb 3
199	MindReader	\$37,965	15	Oct 24
	Distributor			
0	Universal Pictures			
1	Walt Disney Studios Motion Pictures			
2	20th Century Studios			
3	Columbia Pictures			
4	Walt Disney Studios Motion Pictures			
..	...			
195	Cohen Media Group			
196	Kino Lorber			
197	Kino Lorber			
198	Music Box Films			
199	Five & Two Pictures			

- iii. There are several messages of “no studio information,” which indicates that the movies do not have distributor information

- 4) Exclude movies without distributor information from the table
- You may want to collect movies that have no missing values because you are not going to use those movies in your analysis.
 - To remove those movies with missing values, we can also use the “try” and “except” function
 - “else” statement is newly added
 - Ask ChatGPT what “try”, “except”, and “else” are in Python

What are “try”, “except”, and “else” in Python?

In Python, “try”, “except”, and “else” are control flow statements used together to handle exceptions and define the flow of code execution in the presence of exceptions. They are often used in combination to create more sophisticated error handling logic.

Here's a brief explanation of each statement:

- “try”: The “try” statement is used to enclose a block of code that might potentially raise an exception. The code inside the “try” block is executed, and if an exception occurs, the control flow jumps to the corresponding “except” block.
- “except”: The “except” statement is used to specify how to handle specific exceptions that may occur within the “try” block. It allows you to define code that is executed when a particular exception is raised. You can have multiple “except” blocks to handle different types of exceptions. If an exception matches the specified type, the code within the corresponding “except” block is executed.
- “else”: The “else” statement is optional and follows all the “except” blocks. It is used to define a block of code that is executed if no exception occurs in the “try” block. The code in the “else” block is executed only if the “try” block completes without any exceptions being raised. It is commonly used to specify the code that should run when the desired operations within the “try” block succeed.

- This try command says that try to collect the movie information only if there is a value for Distributor but that if there is an error (IndexError specifically), print “no studio information” and then end the run without any follow-up actions.

```

for item in movie[1:201]:
    title = item.find("a", {"class": "a-link-normal"}).text
    gross = item.find("td", {"class": "a-text-right mojo-field-type-money mojo-estimatable"}).text
    theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text
    date = item.find("td", {"class": "a-text-left mojo-field-type-date a-nowrap"}).text

    try:
        studio = item.findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")[0]
    except IndexError as e:          # IndexError: list index out of range
        print("no studio information")
    #   studio = None
    else:
        movie_title.append(title)
        movie_gross.append(gross)
        movie_theaters.append(theaters)
        movie_date.append(date)
        movie_distributor.append(studio)

iter_df = pd.DataFrame({"Title": movie_title, "Gross": movie_gross, "Theaters": movie_theaters,
                       "Date": movie_date, "Distributor": movie_distributor})
print(iter_df)

```

```

no studio information

          Title      Gross Theaters   Date
0       The Super Mario Bros. Movie $570,554,370    4,371 Apr 5
1     Guardians of the Galaxy Vol. 3 $336,521,512    4,450 May 5
2           Avatar: The Way of Water $283,067,859    4,340 Dec 16
3 Spider-Man: Across the Spider-Verse $232,339,759    4,332 Jun 2
4           The Little Mermaid    $232,317,534    4,320 May 26
..
..          ...
176        Let It Be Morning      $42,719       26 Feb 3
177        Close to Vermeer      $41,039       23 May 26
178        Cinema Sabaya        $39,668        4 Feb 10
179        Full Time            $38,658        8 Feb 3
180        MindReader           $37,965       15 Oct 24

          Distributor
0           Universal Pictures
1 Walt Disney Studios Motion Pictures
2           20th Century Studios
3           Columbia Pictures
4 Walt Disney Studios Motion Pictures
..
..          ...
176        Cohen Media Group
177        Kino Lorber
178        Kino Lorber
179        Music Box Films
180        Five & Two Pictures

```

iv. As a result, we have only 181 movies in the table.

6. Export the collected data to the working directory

- 1) We already set up the working directory.

 - a. It means when we import and export the datasets, we are going to use the specific folder in our computer
 - b. If you need to change the working directory, you should set up the new working directory with the following code.

```
os.chdir("/Users/sheng/Jupyter/AESHM_579") # for Mac  
# os.chdir("C:\\Users\\sheng\\Jupyter\\AESHM_579") # for Windows
```

- 2) Export the dataset in CSV
 - a. Dataframe.to_csv("csv file name")
 - i. iter_df.to_csv("Movies_Year.csv")=> In your case, "Movies_2021.csv"

```
iter df.to_csv('Movies 2023.csv')
```

- b. After running these commands, go to the working directory
 - i. Can find the movie file

The screenshot shows a Mac desktop environment. On the left, a sidebar lists 'Desktop', 'Documents', 'AirDrop', and 'Recents'. To the right, a file browser window is open with the title 'Movies_2023.csv'. The file list contains three items: 'Data_collection_1.ipynb', 'Data_collection_2.ipynb', and 'Movies_2023.csv', with the latter being the selected item.

	A	B	C	D	E	F	G	H
1	Title		Gross	Theaters	Date	Distributor		
2	0 The Super Mario Bros. Movie	\$570,554,370	4,371	5-Apr	Universal Pictures			
3	1 Guardians of the Galaxy Vol. 3	\$336,521,512	4,450	5-May	Walt Disney Studios Motion Pictures			
4	2 Avatar: The Way of Water	\$283,067,859	4,340	16-Dec	20th Century Studios			
5	3 Spider-Man: Across the Spider-Verse	\$232,339,759	4,332	2-Jun	Columbia Pictures			
6	4 The Little Mermaid	\$232,317,534	4,320	26-May	Walt Disney Studios Motion Pictures			
7	5 Ant-Man and the Wasp: Quantumania	\$214,503,921	4,345	17-Feb	Walt Disney Studios Motion Pictures			
8	6 John Wick: Chapter 4	\$187,091,533	3,855	24-Mar	Lionsgate			
9	7 Creed III	\$156,248,615	4,007	3-Mar	United Artists Releasing			
10	8 Fast X	\$138,760,790	4,088	19-May	Universal Pictures			
11	9 Puss in Boots: The Last Wish	\$129,870,190	4,121	21-Dec	Universal Pictures			
12	10 Scream VI	\$108,161,389	3,676	10-Mar	Paramount Pictures			
13	11 M3GAN	\$95,043,350	3,628	6-Jan	Universal Pictures			
14	12 Dungeons & Dragons: Honor Among Thieves	\$93,277,026	3,856	31-Mar	Paramount Pictures			
15	13 Evil Dead Rise	\$67,206,770	3,417	21-Apr	Warner Bros.			
16	14 Transformers: Rise of the Beasts	\$66,207,458	3,678	9-Jun	Paramount Pictures			
17	15 Cocaine Bear	\$64,388,510	3,571	24-Feb	Universal Pictures			
18	16 A Man Called Otto	\$64,228,358	3,957	30-Dec	Columbia Pictures			
19	17 Shazam! Fury of the Gods	\$57,638,006	4,071	17-Mar	Warner Bros.			
20	18 Jesus Revolution	\$52,102,987	2,575	24-Feb	Lionsgate			
21	19 80 for Brady	\$39,330,588	3,939	3-Feb	Paramount Pictures			

7. Concatenate csv files that contain movie information

Thus far, we have learned to collect movie information in a year and export the data. Now, we can apply this data collection process to other years.

1) Let's collect movies in the year of 2022.

a. Change the URL in the variable “df”

i. Copy the webpage for the movies in the previous year

Year	Total Gross	%± LY	Releases	Average	#1 Release
2023	\$3,885,988,821	-	284	\$13,683,059	The Super Mario Bros. Movie
2022	\$7,369,357,270	+64.4%	496	\$14,857,575	Top Gun: Maverick
2021	\$4,482,808,453	+112.1%	440	\$10,188,201	Spider-Man: No Way Home
2020	\$2,113,846,800	-81.4%	456	\$4,635,628	Bad Boys for Life

Domestic Box Office For 2022

The screenshot shows a web-based application for tracking domestic box office data. At the top, there are dropdown menus for '2022' and 'Calendar grosses'. Below this is a table with the following columns: Rank, Release, Gross, Theaters, Total Gross, Release Date, and Distributor.

Rank	Release	Gross	Theaters	Total Gross	Release Date	Distributor
1	Top Gun: Maverick	\$718,318,561	4,751	\$718,732,821	May 27	Paramount Pictures
2	Black Panther: Wakanda Forever	\$436,499,646	4,396	\$453,829,060	Nov 11	Walt Disney Studios Motion Pictures
3	Doctor Strange in the Multiverse of Madness	\$411,331,607	4,534	\$411,331,607	May 6	Walt Disney Studios Motion Pictures
4	Avatar: The Way of Water	\$401,007,908	4,340	\$684,075,767	Dec 16	20th Century Studios



← → C [boxofficemojo.com/year/2022/?ref_=bo_yl_table_2](https://www.boxofficemojo.com/year/2022/?ref_=bo_yl_table_2)



```
# df='https://www.boxofficemojo.com/year/2023/?ref_=bo_yl_table_1'
df='https://www.boxofficemojo.com/year/2022/?ref_=bo_yl_table_2'
```

- ii. After changing the URL, deactivate the URL for the movies in the current year
- b. All the collection procedures are the same, but we need to change the csv file name that will contain the movie information in the previous year (e.g. 2022)
 - i. Ignore the previous command as you did for the URL variable

```
import os
import requests
import bs4
import pandas as pd

os.chdir("/Users/sheng/Jupyter/AESHM_579") # for Mac
# os.chdir("C:\\\\Users\\\\sheng\\\\Jupyter\\\\AESHM_579") # for Windows

# df='https://www.boxofficemojo.com/year/2023/?ref_=bo_yl_table_1'
df='https://www.boxofficemojo.com/year/2022/?ref_=bo_yl_table_2'

page = requests.get(df)

bs_obj = bs4.BeautifulSoup(page.content, "html.parser")
movie = bs_obj.findAll("tr")

movie_title = []
movie_gross = []
movie_theaters = []
movie_date = []
movie_distributor = []

for item in movie[1:201]:
    title = item.find("a", {"class": "a-link-normal"}).text
    gross = item.find("td", {"class": "a-text-right mojo-field-type-money mojo-estimatable"}).text
    theaters = item.find("td", {"class": "a-text-right mojo-field-type-positive_integer"}).text
    date = item.find("td", {"class": "a-text-left mojo-field-type-date anowrap"}).text

    try:
        studio = item.findAll("a", {"class": "a-link-normal"})[1].text.split("\n\n")[0]
    except IndexError as e: # IndexError: list index out of range
        print("no studio information")
    #
    else:
        movie_title.append(title)
        movie_gross.append(gross)
        movie_theaters.append(theaters)
        movie_date.append(date)
        movie_distributor.append(studio)

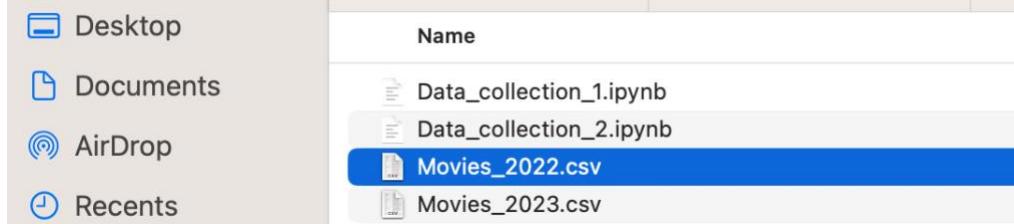
iter_df = pd.DataFrame({"Title": movie_title, "Gross": movie_gross, "Theaters": movie_theaters,
                       "Date": movie_date, "Distributor": movie_distributor})
print(iter_df)

iter_df.to_csv('Movies_2022.csv')
```

c. Return

		Title	Gross	Theaters
0		Top Gun: Maverick	\$718,318,561	4,751
1		Black Panther: Wakanda Forever	\$436,499,646	4,396
2	Doctor Strange in the Multiverse of Madness		\$411,331,607	4,534
3		Avatar: The Way of Water	\$401,007,908	4,340
4		Jurassic World: Dominion	\$376,851,080	4,697
..	
187		Brian and Charles	\$430,635	279
188		King Richard	\$408,056	3,302
189		The Inspection	\$385,627	144
190		Alice	\$372,529	169
191		Emergency Declaration	\$372,123	42
	Date	Distributor		
0	May 27	Paramount Pictures		
1	Nov 11	Walt Disney Studios Motion Pictures		
2	May 6	Walt Disney Studios Motion Pictures		
3	Dec 16	20th Century Studios		
4	Jun 10	Universal Pictures		
..		
187	Jun 17	Focus Features		
188	Nov 19	Warner Bros.		
189	Nov 18		A24	
190	Mar 18	Roadside Attractions		
191	Aug 12	Well Go USA Entertainment		

d. The 2022 movie file in the working directory



- 2) Combine the collected data files (concatenation)
 - a. Create a new Python file for concatenation
 - b. Import the necessary packages
 - i. Packages: os and pandas
 - ii. If you want to use a different working directory, use the command “os.chdir”

```
import os
import pandas as pd

os.chdir("/Users/sheng/Jupyter/AESHM_579") # for Mac
# os.chdir("C:\\\\Users\\\\sheng\\\\Jupyter\\\\AESHM_579") # for Windows
```

- c. Create an empty container for the combined values

- i. Use a variable “result” in a pandas data frame format

```
result = pd.DataFrame()
```

- d. To combine the collected movie information, we should import the movie file into the Python system and then concatenate them.
- e. To import the movie data, we can use the loop
 - i. The file names are “Movies_2022.csv” and “Movies_2023.csv”
 - 1. The years in the file name and the following codes can change depending on the time period you are focusing on.
 - ii. So, the year in the file name should change in the loop.

```
result = pd.DataFrame()  
  
for year in range(2022, 2024):
```

- iii. The indicator for values in “range” is named as “year” for clarity.
- iv. Set the variable “readfile” to change depending on “year” in the loop.
- v. In other words, the first component in the loop is 2022 and the variable “readfile” becomes “Movies_2022.csv.” The next component is 2023 and the “readfile” is “Movies_2023.csv”

```
for year in range(2022, 2024):  
    readfile = "Movies_" + str(year) + ".csv"  
    df = pd.read_csv(readfile)  
    result = pd.concat([result, df])
```

- f. In the loop, we read the csv files to put them in the pandas data frame
- g. Then, combine the specific information of movies in each file in the container “result” with the concatenation function
- h. Get code explanation from ChatGPT

CJ

Explain the code below:

```
result = pd.DataFrame()

for year in range(2022, 2024):
    readfile = "Movies_" + str(year) + ".csv"
    df = pd.read_csv(readfile)
    result = pd.concat([result, df])
```



This code appears to be written in Python and makes use of the pandas library for data manipulation. It is designed to read data from multiple CSV files, specifically movie data, and concatenate them into a single DataFrame called "result." Let's break down the code step by step:

1. `'result = pd.DataFrame()'`: This line initializes an empty DataFrame called "result" using pandas. This DataFrame will eventually store the combined data from multiple CSV files.
2. `'for year in range(2022, 2024):'`: This line sets up a for loop that iterates over the years 2022 and 2023. The loop will read data from CSV files for these two years.
3. `'readfile = "Movies_" + str(year) + ".csv"'`: Inside the loop, this line constructs the name of the CSV file to be read for the current year. It does so by combining the string "Movies_" with the current year (converted to a string) and ".csv". For example, for the first iteration, it will create the string "Movies_2022.csv".
4. `'df = pd.read_csv(readfile)'`: This line uses pandas' `'read_csv'` function to read the data from the CSV file specified by the "readfile" variable. The resulting data is stored in a DataFrame called "df." This DataFrame will contain the movie data for the current year.
5. `'result = pd.concat([result, df])'`: After reading the data for the current year, this line concatenates the "df" DataFrame with the "result" DataFrame. The `'pd.concat'` function is used to vertically stack the rows from "df" below the rows of "result." This way, the data for each year is added to the "result" DataFrame one after another.

- i. Check the returns with “print”
- j. Export the data in a csv file.

k. Entire commands

```
import os
import pandas as pd

os.chdir("/Users/sheng/Jupyter/AESHM_579") # for Mac
# os.chdir("C:\\\\Users\\\\sheng\\\\Jupyter\\\\AESHM_579") # for Windows

result = pd.DataFrame()

for year in range(2022, 2024):
    readfile = "Movies_" + str(year) + ".csv"
    df = pd.read_csv(readfile)
    result = pd.concat([result, df])

print(result)

result.to_csv("Movies_2012-2023.csv")
```

	Unnamed: 0	Title	Gross \
0	0	Top Gun: Maverick	\$718,318,561
1	1	Black Panther: Wakanda Forever	\$436,499,646
2	2	Doctor Strange in the Multiverse of Madness	\$411,331,607
3	3	Avatar: The Way of Water	\$401,007,908
4	4	Jurassic World: Dominion	\$376,851,080
..
176	176	Let It Be Morning	\$42,719
177	177	Close to Vermeer	\$41,039
178	178	Cinema Sabaya	\$39,668
179	179	Full Time	\$38,658
180	180	MindReader	\$37,965
Theaters	Date	Distributor	
0	4,751 May 27	Paramount Pictures	
1	4,396 Nov 11	Walt Disney Studios Motion Pictures	
2	4,534 May 6	Walt Disney Studios Motion Pictures	
3	4,340 Dec 16	20th Century Studios	
4	4,697 Jun 10	Universal Pictures	
..	
176	26 Feb 3	Cohen Media Group	
177	23 May 26	Kino Lorber	
178	4 Feb 10	Kino Lorber	
179	8 Feb 3	Music Box Films	
180	15 Oct 24	Five & Two Pictures	

[373 rows x 6 columns]

l. Final return in your working directory



	Unnamed: 0	Title	Gross	Theaters	Date	Distributor
0	0	Top Gun: Maverick	\$718,318,561.00	4,751	May 27	Paramount Pictures
1	1	Black Panther: Wakanda Forever	\$436,499,646.00	4,396	Nov 11	Walt Disney Studios Motion Pictures
2	2	Doctor Strange in the Multiverse of Madness	\$411,331,607.00	4,534	May 6	Walt Disney Studios Motion Pictures
3	3	Avatar: The Way of Water	\$401,007,908.00	4,340	Dec 16	20th Century Studios
4	4	Jurassic World: Dominion	\$376,851,080.00	4,697	Jun 10	Universal Pictures
5	5	Minions: The Rise of Gru	\$369,695,210.00	4,427	Jul 1	Universal Pictures
6	6	The Batman	\$369,345,583.00	4,417	Mar 4	Warner Bros.
7	7	Thor: Love and Thunder	\$343,256,830.00	4,375	Jul 8	Walt Disney Studios Motion Pictures
8	8	Spider-Man: No Way Home	\$231,808,708.00	4,336	Dec 17	Sony Pictures Entertainment (SPE)
9	9	Sonic the Hedgehog 2	\$190,872,904.00	4,258	Apr 8	Paramount Pictures
10	10	Black Adam	\$168,054,237.00	4,402	Oct 21	Warner Bros.
11	11	Elvis	\$151,040,048.00	3,932	Jun 24	Warner Bros.
12	12	Uncharted	\$148,649,929.00	4,275	Feb 18	Sony Pictures Entertainment (SPE)
13	13	Nope	\$123,277,080.00	3,807	Jul 22	Universal Pictures
14	14	Lightyear	\$118,307,188.00	4,255	Jun 17	Walt Disney Studios Motion Pictures

2022



2023



163	163	Smoking Causes Coughing	\$64,484.00	63	Mar 31	Magnolia Pictures
164	164	My Happy Ending	\$59,329.00	126	Feb 24	Roadside Attractions
165	165	Godland	\$59,124.00	7	Feb 3	Janus Films
166	166	Nam June Paik: Moon Is the Oldest TV	\$59,023.00	5	Mar 24	Greenwich Entertainment
167	167	The Trial	\$58,659.00	5	Dec 9	Rialto Pictures
168	168	32 Sounds	\$56,480.00	6	Apr 28	Abramorama
169	169	Fire of Love	\$56,332.00	191	Jul 6	Neon
170	170	Tori and Lokita	\$55,283.00	18	Mar 24	Janus Films
171	171	The End of Sex	\$54,527.00	185	Apr 28	Blue Fox Entertainment
172	172	Cat Daddies	\$49,910.00	5	Oct 14	Sky Island Films
173	173	Decision to Leave	\$49,185.00	169	Oct 14	MUBI
174	174	Nuclear Now	\$48,064.00	350	Apr 28	Abramorama
175	175	R.M.N.	\$46,360.00	41	Apr 28	IFC Films
176	176	Let It Be Morning	\$42,719.00	26	Feb 3	Cohen Media Group
177	177	Close to Vermeer	\$41,039.00	23	May 26	Kino Lorber
178	178	Cinema Sabaya	\$39,668.00	4	Feb 10	Kino Lorber
179	179	Full Time	\$38,658.00	8	Feb 3	Music Box Films
180	180	MindReader	\$37,965.00	15	Oct 24	Five & Two Pictures