Published in The Pythoneers · Follow

Abhay Parashar · Follow
Feb 10 · 8 min read ★

Photo by Nguyen Dang Hoang Nhu on Unsplash

Web Scraping Using Selenium Python

Detailed Tutorial With One Project

Web Scraping is an essential skill for all data scientists and automation engineers around the world. It is the process of scraping data from the web. The data can be in the form of text, links, tables, or images. In general, you can only scrape static data from the web. Here Come the role of selenium.

Selenium is a python library that can scrape dynamic web easily. It is used also used for web automation & testing. Scraping data from the web is a small part of its library. Let's See Some of the features of this library.

Features:-

- Easy to read & code,
- Open Source,
- Fast In Execution,
- Can run tests across different browsers,
- Automates Browser Easily,
- Beginners Friendly

Installation: `pip install selenium`

Basic Code —

```
from selenium import webdriver
driver = webdriver.Chrome('chromedriver')
driver.get('facebook.com')
```

. . .

Key Components of Selenium —

1. Web Driver

Web Driver is an automated tool that helps test web apps across different browsers. It provides the capability to navigate to different elements of the web page like user input, buttons, JavaScript execution, and more. Selenium supports multiple web browsers and offers web drivers for each. but for me, Chrome works best with selenium.

Visit the selenium documentation, select your browser and download the stable version of the web driver. Once downloaded move your web driver to a location where you can access it easily like

`C:\users\webdriver\chromedriver.exe` or for simplicity just move it to the folder you are working on.

Now That your web driver is with you let's scrape the web by locating different elements.

2. Element Locators

Selenium web driver offers a variety of locater functions to locate elements on the web page. Let's see each one with an example —

▶ Locating Elements Using ID

You should use this when you know the ID of the attribute on the web page. It will return the first element with the matching id attribute.

```
driver.find_element_by_id('login')
```

▶ Locating Elements Using Class Name

You should use this when you know the class of the attributes on the web page.

It will return the first element with a matching class name. If there is no matching class then it will raise a NoSuchElementException.

```
driver.find_element_by_class_name('tags')
```

▶ Locating Elements Using Name

It will return the first matching name attribute. It is useful for scraping names, titles, input fields, etc.

```
driver.find_element_by_name('username')
```

▶ Locating Elements Using Xpath (☆)

XPath is a language for locating nodes in XML documents. As HTML is an implementation of XML, selenium users have the power to access the attributes with Xpath also. It is the best choice to use when we don't have a

suitable name, class, or id for our attribute. Xpath has the capability to scrape multiple data at the same time.

You can write the XPath of an element in absolute terms or in relative to a DOM object.

The Syntax for XPath is —

```
Xpath = //tagname[@Attribute='Value']

//        ➞ Select Current Node
tagname   ➞ Tagname like input, div,td,tr
@         ➞ Selects attribute
Attribute➞ Attribute name (class,id,name,etc)
value     ➞ value of the attribute


Xpath = //div[@class='tag']
## It will return all the divs with a class tag in a list.

Xpath = //div[@class='tag']/a[@class='social']
## It will return all the links with class social inside the div that
has a class of tag. Written XPath HTML Looks Like
HTML
<div class='tag'>
    <a class='social' href='Facebook.com'> Facebook</a>
</div>
```

▶ Locating Elements Using Link Text

It is used when you know the text used in the anchor tag. it will locate an attribute link with its link text.

```
HTML
 <a href="testimonials.html">Testimonials</a>

LOCATING ELEMENT USING LINK TEXT
driver.find_element_by_link_text('Testimonials')
```

▶ Locating Elements Using Partial Link Text

This method will locate an link with its partial text means a part of the text.

```
HTML
 <a href="testimonials.html">Testimonials</a>

LOCATING ELEMENT USING PARTIAL LINK TEXT
driver.find_element_by_partial_link_text('Testi')
```

▶ Locating Elements by Tag Name

You can use this strategy to retrieve the first occurrence of a tag inside an element.

```
HTML
  <h1>Elon Musk</h1>
  <p>CEO & Co Founder, Tesla</p>
  <p>Elon Musk Was Born in 28 June 1971.</p>


LOCATING ELEMENT BY TAG NAME
driver.find_element_by_tag_name('p')


----
CEO & Co Founder, Tesla
```

▶ Locating Elements by CSS Selectors

This method is for all the beautiful soup lovers. It allows you to select an element with the help of CSS selectors. where you can put the tag name and then for class `.` and for id `#` to select an element.

```
HTML
<p class='content'>Tesla's Share Seeing a Hike After Long Time.</p>
```

```
LOCATING ELEMENTS BY CSS SELECTORS
driver.find_element_by_css_selector('p.content')
```

Selenium is not limited to just locating elements, it provides the capability to intract with them like giving inputs, etc.

3. Web Interaction & Navigation

After getting an element the next step is to interact with it. whether you wanted to scrape the text, put some text inside an input field, and just scroll randomly on the web page. Selenium web driver can do it all. You can import keys from the selenium web driver that allows you the control different keys.

```
from selenium.webdriver.common.keys import Keys


element.send_keys("some text") ## pass the text to the element
element.send_keys("some text", Keys.ARROW_DOWN)
element.send_keys(Keys.ENTER)          ##Press ENTER
```

You can fill forms using this, let's log in to Facebook with the help of selenium.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome(executable_path='chromedriver.exe')
driver.get("https://www.facebook.com/")


FB_ID = 'YOUR_FB_ID'
FB_PSD = 'YOUR_FB_PSD'


email = driver.find_element_by_id("email") ## locating email input
email.send_keys(id)                        ## Sending Username as input


Password = driver.find_element_by_id("pass")
Password.send_keys(password)


## Clicking Login Button
button = driver.find_element_by_name("login").click()


or


button = driver.find_element_by_name("login")
button.send_keys(Keys.ENTER)
```

You can use the driver's inbuilt function to control the forward and backward moment in the browser.

```
driver.forward()   ## To Move Forward
driver.back()      ## To Move Backward
```

4. Waits

Every Modern Day Website uses ajax to load the contents of the web. Thus, different parts of the website load at different times making it difficult to scrape data from them. Waits can resolve this issue by providing a slack of interval between two continuous actions.

There are two types of waits you can add —

1. **Explicit Wait**: It is a piece of code that waits for a certain condition to happen before processing code execution.

```
from selenium.webdriver.support.ui import WebDriverWait


try:
    element = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.ID, "myDynamicElement"))
    )
finally:
driver.quit()
```

2. **Implicit Wait**: It is a piece of code that stops the execution of the program for a certain amount of time. tells WebDriver to poll the DOM for a certain amount of time when trying to find any element (or elements) not immediately available.

```
driver.implicitly_wait(10) # 10 second wait
```

5. Action Chains

Actions Chain is a feature of selenium that allows you to combine multiple low-level interactions in a chain. For example, mouse clicks, key combinations, different key presses, etc.

```
from selenium.webdriver.common.action_chains import ActionChains


actions= ActionChains(driver)
actions.send_keys(Keys.TAB)              ## Key Press
actions.send_keys(Keys.ENTER)            ## Key Press
actions.send_keys(username)              ## Variable input
actions.send_keys(Keys.Enter)            ## Key Press
actions.send_keys(Keys.Tab)              ## Key Press
actions.perform()      ## To Perform all the actions one by one
```

Now, that you have a basic understanding let's move to our projects.

{ Important Code Snippets }

1. SCROLLING TILL BOTTOM OF THE PAGE

```
driver.execute_script() ## Allows Inserting JS Code In Python Script


EX:- driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
## Above Code Will Scrool Till The Bottom of Web Page


page_height = driver.execute_script("return
document.body.scrollHeight")
for value in range(0,page_height):
 driver.execute_script(f"window.scrollTo(0, {value});")
## Above Code Will Smooth Scrolling Till The End
```

2. RUNNING SELENIUM CODE IN BACKGROUND

```
from selenium.webdriver.chrome.options import Options
```

```
chrome_options = Options()
chrome_options.add_argument('--headless')


PATH = 'WEBDRIVE_PATH'
driver = webdriver.Chrome(PATH,options=chrome_options)
```

3. CLOSE EVERYTING

```
drive.quit()
```

. . .

Project: Scraping Top Grossing Movies Data

Task: Our task is to scrape the top 200 movies with their lifetime collection and release data and save it in a CSV file. The website we will be using is **box office mojo** they have a chart of the top 200 highest-grossing movies.

Step 1: Importing Libraries and Modules

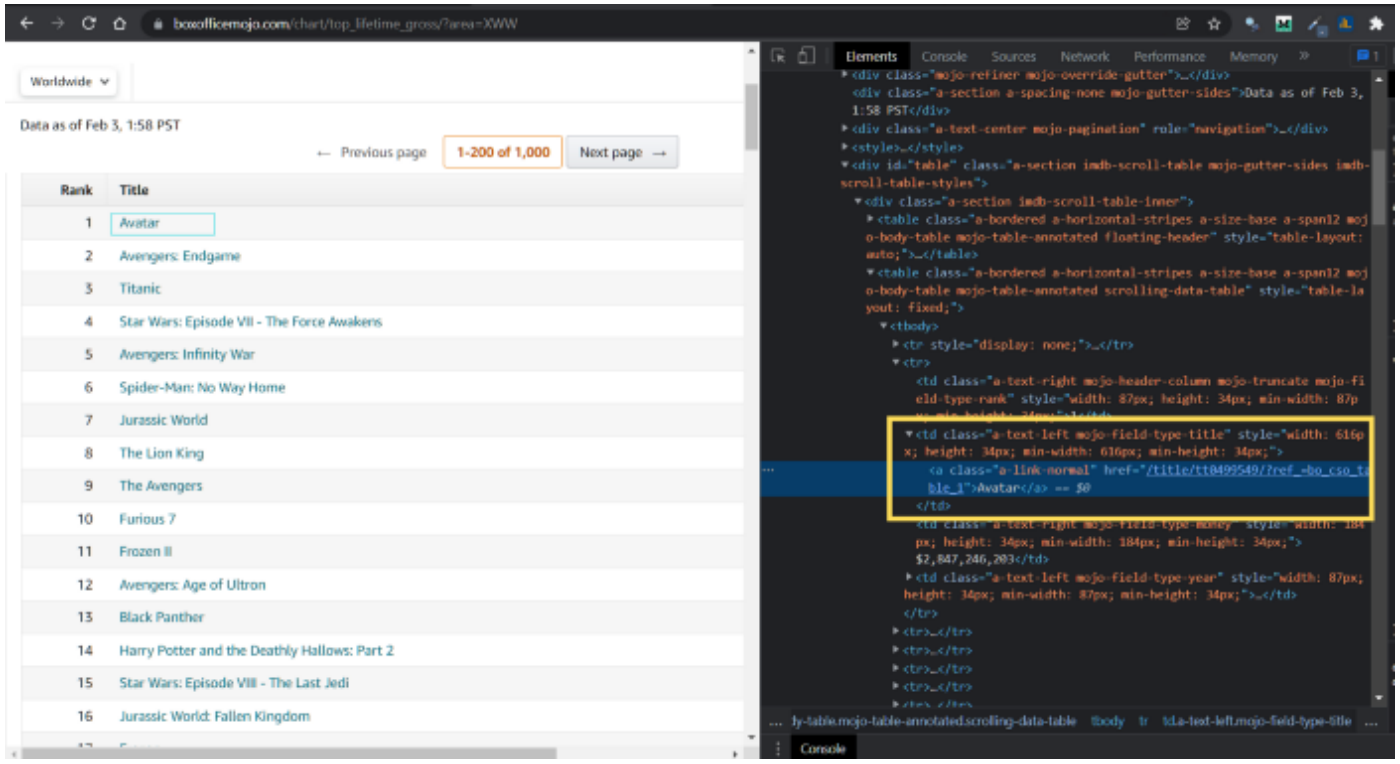
The End goal of this project is to store all the records in a CSV file, so we are using the panda's library to do the work for us.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import pandas as pd
```

Step 2: Accessing Driver & Website

I am using chrome as my browser so I have downloaded chrome driver. if you are using something else then change the code accordingly.

```
driver = webdriver.Chrome('chromedriver.exe')
driver.get('https://www.boxofficemojo.com/chart/top_lifetime_gross/?area=XWW')
```



Web Page Image With Code Inspector — Image By Author

Step 3: Scraping Movie Names

In the above image, you will see in the right-side code inspector is opened, in which a yellow box highlights a piece of code. This code contains the movie name **Avtar**. The HTML structure for it —

```
<td class="a-text-left mojo-field-type-title">
    <a class='a-link-normal'> Avtar </a>
</td>
```

To access the name of the movie we are going to write Xpath.

```
//td[@class="a-text-left mojo-field-type-title"]/a[@class="a-link-
normal"]
```

With the help of the above Xpath, all the movie's names can be scraped.

```
movies_names = driver.find_elements_by_xpath('//td[@class="a-text-left
mojo-field-type-title"]/a[@class="a-link-normal"]')


movie_name_list = []
for movie in range(len(movies_names)):
    movie_name_list.append(movies_names[movie].text)
print(movie_name_list)


''' It will print all the movie names inside a list '''
```

Step 3: Scraping Movie Release Dates.

The Same Process We Follow For Scraping movie Release Date. Xpath for scraping movie dates will be —

```
//td[@class="a-text-left mojo-field-type-year"]/a[@class="a-link-
normal
```

Let's use it to scrape movie release dates.

```
release_year = driver.find_elements_by_xpath('//td[@class="a-text-left
mojo-field-type-year"]/a[@class="a-link-normal"]')


release_year_list = []
for year in range(len(release_year)):
    release_year_list.append(release_year[year].text)
print(release_year_list)


''' It will print all the movies release dates inside a list '''
```

Step 4: Scraping Movie Crossings.

The process will be the same but this time Xpath will be smaller. We can scrape movie grossing directly from the `<td>` above them. The Xpath for Scraping Movie Grossings will be —

```
//td[@class="a-text-right mojo-field-type-money
```

Let's use it to scrape movie grossing.

```
lifetime_gross = driver.find_elements_by_xpath('//td[@class="a-text-
right mojo-field-type-money"]')


lifetime_gross_list = []
for i in range(len(lifetime_gross)):
    lifetime_gross_list.append(lifetime_gross[i].text)
print(lifetime_gross_list)


''' It will print all the movies Grossings inside a list '''
```

Step 5: Storing Data In a CSV

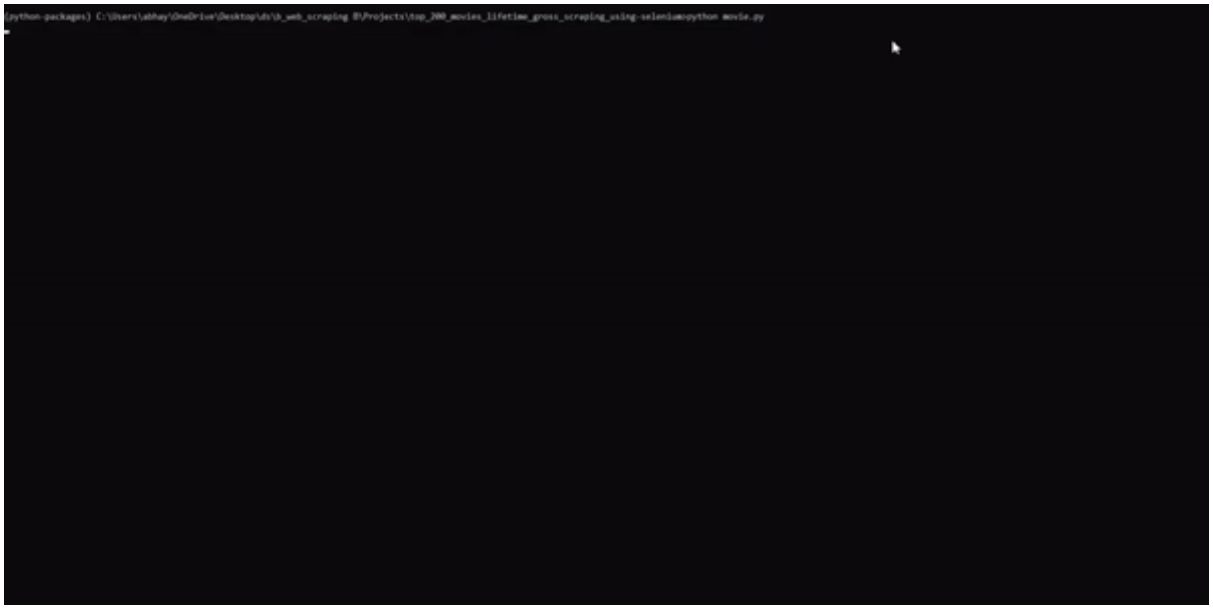
We got data in three separate lists. Now we can easily combine them using the zip function. After combining with the help of pandas Dataframe we can create a data frame and then we can drop it on a CSV file.

```
data =list( zip(movie_name_list, release_year_list,
lifetime_gross_list))

df = pd.DataFrame(data,columns=['Movie Name', 'Release Date','Lifetime
Earnings'])

df.to_csv('top_200_movies_with_lifetime_gross.csv',index=False)
```

Combining It All Together



Code Output Gif By Author

We have successfully scraped the data of the top 200 movies' lifetime grossing. Now, your task is to reduce the size of the code by combining all the three loops in one, etc.

Master The Art of Writing Xpath For Web Scraping

A Gentle Introduction To The RegEX of Web Scraping

levelup.gitconnected.com

Master Web Scraping Completely From Zero To Hero 🕸

Using Beautiful Soup and Requests Library with One Project

medium.com

. . .

Conclusion

Selenium is a great combination of scraping and automation libraries. you can use it for both. It can easily scrape Java Script-loaded content so it become one of the top-notch libraries of web scraping. Although I use it mostly when I need to automate some tasks on a web browser with a little bit of scraping. I hope

you like this sweet little introduction to selenium. Create more projects, tag me at the end and I will definitely give my review and comment on it.

. . .

Thanks For Reading Till Here, If You Like My Content and Want To Support Me The Best Way is —

1. Follow Me On *Medium*.
2. Connect With Me On *LinkedIn*.
3. Become a Medium Member Using *My Referral Link*. a small part of your membership fee will go to me.
4. Subscribe To *My Email List* To Never Miss An Article From Me.

A Newsletter That Contains Top Stories From The Week, Coming On Your Door Every Saturday Take a look.

Your email

Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.

More from The Pythoneers

Follow

We Share Innovative Stories Related to Python Programming, Machine learning, Data Science, Computer Vision, Automation, Web Scraping, Software Development, and more related to AI.

Amit Chauhan · Feb 6 ★

Make WebApp With Streamlit With Python

Machine learning and data science web framework — In this article, the streamlit web framework is used to make the web app of fruit detection and counting from the image. The streamlit is a very useful web...

Data Science 4 min read

Amir Ali Hashemi · Feb 6 ★

Fetch YouTube Subtitles With Python

Get subtitle text and download it in PDF format — Personally, a lot of times I came across situations in which I needed to reference some sentences of a Youtube video but I had no chance to do so without...

Programming 3 min read

Abhay Parashar · Feb 4 ★

12 Things Every Python Programmer Should Know

Amazing List of Python Key Concepts — Python has one of the biggest communities of developers around the world in comparison to other programming languages. There is no secret of python that is hidden fro...

Python 8 min read

Abhay Parashar · Feb 1 ★

7 Websites To Visit When You Are Stuck At Code

Coding and Decoding is a part of developers life — Developers Spent Most of Their Time Writing Code, Generating Errors and Then Decoding Them. Looks Easy But The Fact is Sometimes how hard we try the errors...

Code 4 min read

Паша Дубовик · Feb 1

Visualize your multiprocessing calculations in python with parallelbar module

Perhaps every data scientist sooner or later faces the need to use multiprocessor data processing. In this article, we will not analyze in det...

Multiprocessing 6 min read

Read more from The Pythoneers

More from Medium

Hacking for fun

Discussing smartphones' privacy and how to...

Self Joins in Rails

I recently used self joins in a rails project whe...

SQL Primer — Basic Operations

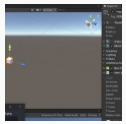
I've been doing a lot of coding on the front-e...

Things to learn to become Web Developer

Get your hands dirty into the coolest technol...

blog_post = {name: 'new entry'}

A little about myself:



Journal 05 — C# Programming, Basics of Variables

Ubuntu/Linux Server — Creating a New User

Here is a guide that teaches you how to creat...

Hack Java Streams : Print Progress using Lombok