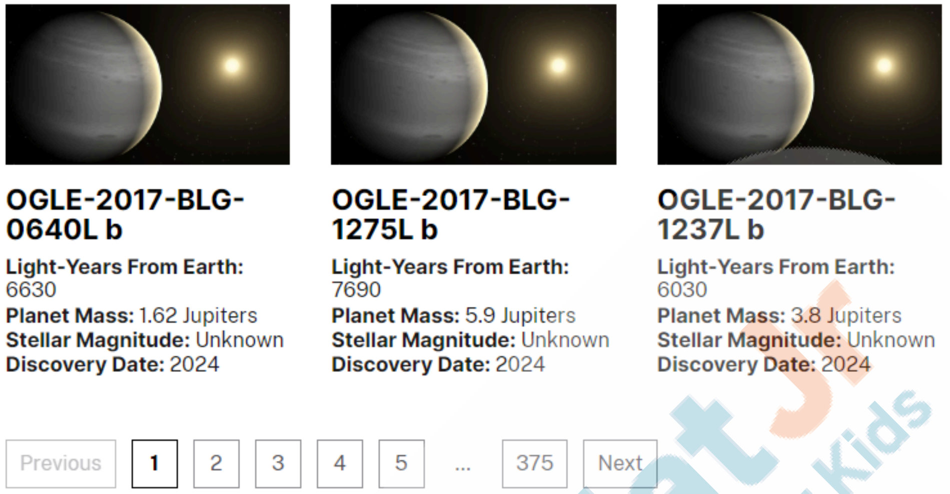
Topic	WEB SCRAPING - 1	
Class Description	**Students will scrape the data from NASA's website to analyze and filter the same for future classes.**	
Class	**PRO C127**	
Class time	**45 mins**	
Goal	● Introduction to Web Scraping ● Use Selenium to perform browser automation to open browser and get web page source ● Use selenium to click ● Use BeautifulSoup4 to extract webpage content	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher-Led Activity 1** **Student-Led Activity 1** **Wrap-Up**	**10 mins** **10 mins** **20 mins** **05 mins**
Credit & Permissions:	**Exoplanet Exploration by NASA** **Beautiful Soup by Crummy (webspace of Leonard Richardson)** **Selenium under Apache license 2.0**	
WARM-UP SESSION - 10 mins		

Teacher Starts Slideshow **Slide 1 to 4** Refer to speaker notes and follow the instructions on each slide.	
Teacher Action	**Student Action**
Hey <student's name>. How are you? It's great to see you! Can you tell me what we learned in the previous class? ***Note**: Encourage the student to give answers and be more involved in the discussion.* **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities. ● Quizzes.	**ESR**: Hi, thanks! We integrated the chatbot
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide 5 to 13**	
Activity Details **Following are the session deliverables:** ● Appreciate the student. ● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
Teacher Action	**Student Action**
Have you seen any movies related to stars and galaxies etc?	**ESR:** Varied

Note: *Encourage the student to give answers and connect the answer with today's topic.* So, from this information, we know that stars and galaxies are far away from us. We need some data to calculate the different characteristics of these planets and stars such as their distance, size, composition and weight etc. NASA has provided us with the data on exoplanets on its website called **EXOPLANET EXPLORATION**. ***Exoplanets*** *are those planets that are present beyond our solar system.* Let's visit the website and check the data first. ***Note:*** *Open Teacher Activity 1 to show the website to the student. Scroll down to find the data in tabular format.* ***Note:*** *NASA's exoplanet catalog web page keeps updating as per the new planet discoveries. At the time of writing this document, the web page had* ***375 Pages*** *with* ***15 Planets per page*** *(except the last page) showing a total of* ***5613 planets data****.*	**ESR:** Yes



OGLE-2017-BLG-0640L b

Light-Years From Earth: 6630
Planet Mass: 1.62 Jupiters
Stellar Magnitude: Unknown
Discovery Date: 2024

OGLE-2017-BLG-1275L b

Light-Years From Earth: 7690
Planet Mass: 5.9 Jupiters
Stellar Magnitude: Unknown
Discovery Date: 2024

OGLE-2017-BLG-1237L b

Light-Years From Earth: 6030
Planet Mass: 3.8 Jupiters
Stellar Magnitude: Unknown
Discovery Date: 2024

Here you can see 5,631 exoplanets are being discovered and data are given in a grid format. ***Note:*** *The total number of exoplanets may change due to the timely updation of data on the website. Open the link and mention the number accordingly.*	**ESR:** Varied
Till now we have always provided you with the datasets. But what if we want to get the data from the website and use it for some purpose? ***Note:*** *Let the student think about what can be done and proceed with the explanation.*	
In this case, we need to write a program that can read the data from the website. The process of accessing data from a website in our program is known as **'Web Scraping'**. In today's class, we will learn about Web Scraping, where we will write a program that can fetch all the useful data from NASA's website for us. Are you excited?	**ESR:** Yes

Let’s dive into the code.

Teacher Ends Slideshow

TEACHER-LED ACTIVITY - 10 mins

Teacher Initiates Screen Share

ACTIVITY

- **Introduction to Beautiful Soup and selenium for Web Scraping**
- **Use of inspect tool for finding**

Teacher Action	Student Action
Let’s create a new directory. Give a name to your Python file as **scraper.py.** We’ll be using four Python libraries: ● BeautifulSoup4 ● Selenium ● Time ● Pandas **Extract HTML Page Content:** **bs4 (BeautifulSoup version 4)** is a Python module, which is famously used for **parsing or separating text as HTML** and then performing actions on it, such as finding specific HTML tags with a particular class/id or listing out all the **<div>** tags inside the page, etc. Teacher Activity 2: Beautiful Soup 4 **Open browser and get HTML page code:**	

Selenium is a **browser automation** Python module, that means, it can help us to open a browser, click on web pages, fill in some forms on the web page and perform other browser operations automatically.

Teacher Activity 3: Selenium

Since we have to scrape data from 491 pages, clicking on the button to go to the next page of the data would come in handy.

Note: *The number of pages may change depending upon the updated data.*

To start scraping data first we need to open the browser using Python Script. The **Selenium** module can be used to open the web page in a browser automatically using Python Scripts.

For this we'll need a **webdriver** from **Selenium**.

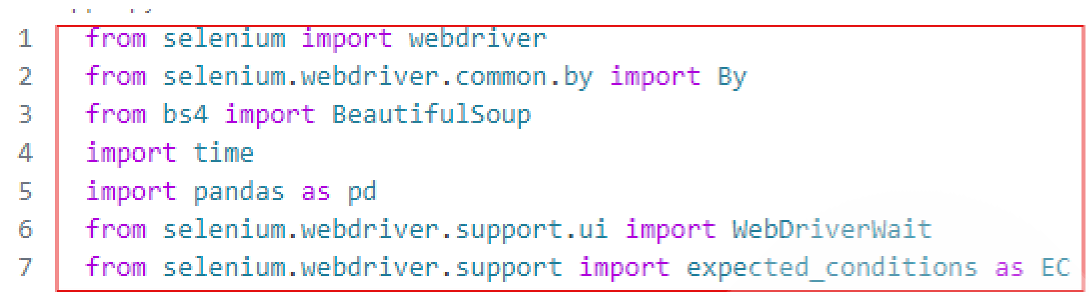
Also, we'll be using **Selenium** for clicking a button. For this, we need to import **By** from **selenium.webdriver.common by**.

Note: *Installation of webdriver is covered in the later section of this class.*

Let's import these modules.

Next import the **time** library to make our code sleep for some time so that the web page could load properly before we start scraping.

We are importing the **pandas** library so that we can export the data that we scrape into a CSV file.



```
from selenium import webdriver
from selenium.webdriver.common.by import By
from bs4 import BeautifulSoup
import time
import pandas as pd
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

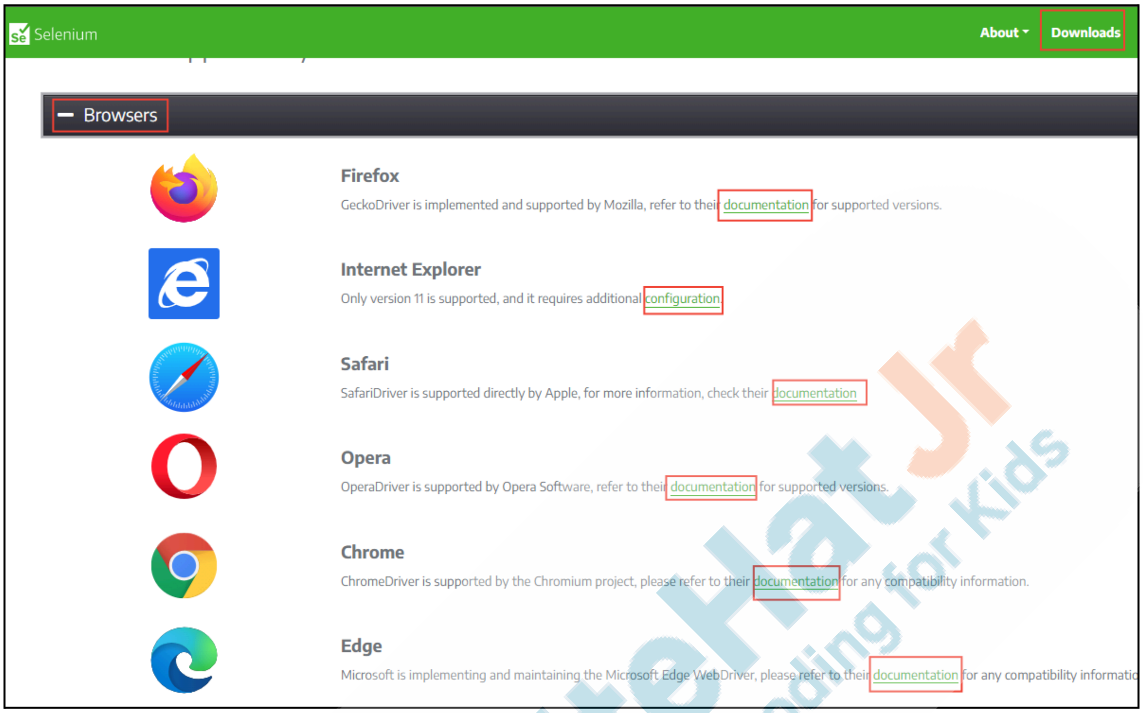
Now, we have to define:

1. **Link of the website we want to open:**
   Provide the link of the website In the variable **START_URL.**

2. **Driver for the browser:**
   Download a webdriver to open the browser using selenium. Depending upon your choice of web browser we can get the drivers at the below link:

Teacher Activity 4: Webdriver for Selenium

Note: *Refer Teacher Activity 5: Browser installation Version Check. Depending upon the system (32-bit or 64-bit) and browser version, download the Webdriver.*



Firefox
GeckoDriver is implemented and supported by Mozilla, refer to their documentation for supported versions.

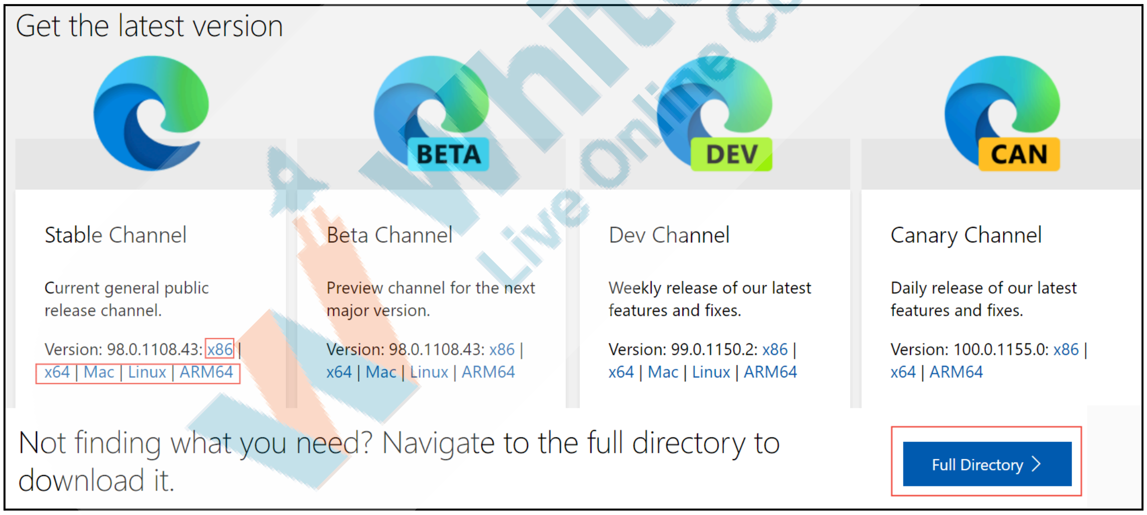
Internet Explorer
Only version 11 is supported, and it requires additional configuration

Safari
SafariDriver is supported directly by Apple, for more information, check their documentation

Opera
OperaDriver is supported by Opera Software, refer to their documentation for supported versions.

Chrome
ChromeDriver is supported by the Chromium project, please refer to their documentation for any compatibility information.

Edge
Microsoft is implementing and maintaining the Microsoft Edge WebDriver, please refer to their documentation for any compatibility informatic


Get the latest version

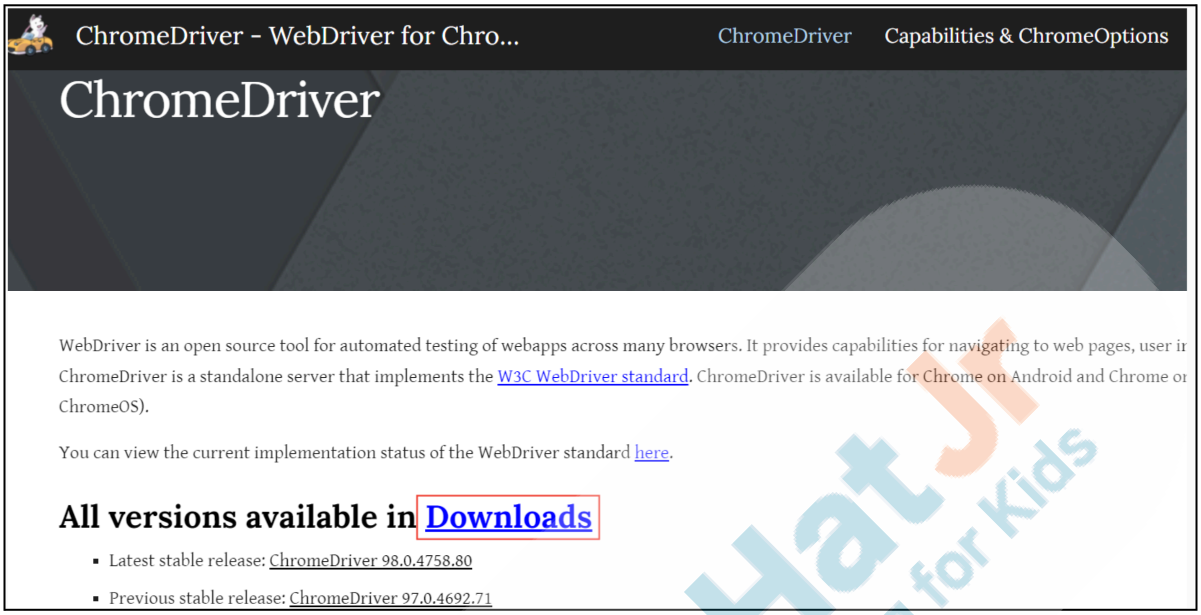
Stable Channel
Current general public release channel.
Version: 98.0.1108.43: x86 | x64 | Mac | Linux | ARM64

Beta Channel
Preview channel for the next major version.
Version: 98.0.1108.43: x86 | x64 | Mac | Linux | ARM64

Dev Channel
Weekly release of our latest features and fixes.
Version: 99.0.1150.2: x86 | x64 | Mac | Linux | ARM64

Canary Channel
Daily release of our latest features and fixes.
Version: 100.0.1155.0: x86 | x64 | ARM64

Not finding what you need? Navigate to the full directory to download it.



ChromeDriver - WebDriver for Chro... ChromeDriver Capabilities & ChromeOptions

ChromeDriver

WebDriver is an open source tool for automated testing of webapps across many browsers. It provides capabilities for navigating to web pages, user ir

ChromeDriver is a standalone server that implements the W3C WebDriver standard. ChromeDriver is available for Chrome on Android and Chrome or ChromeOS).

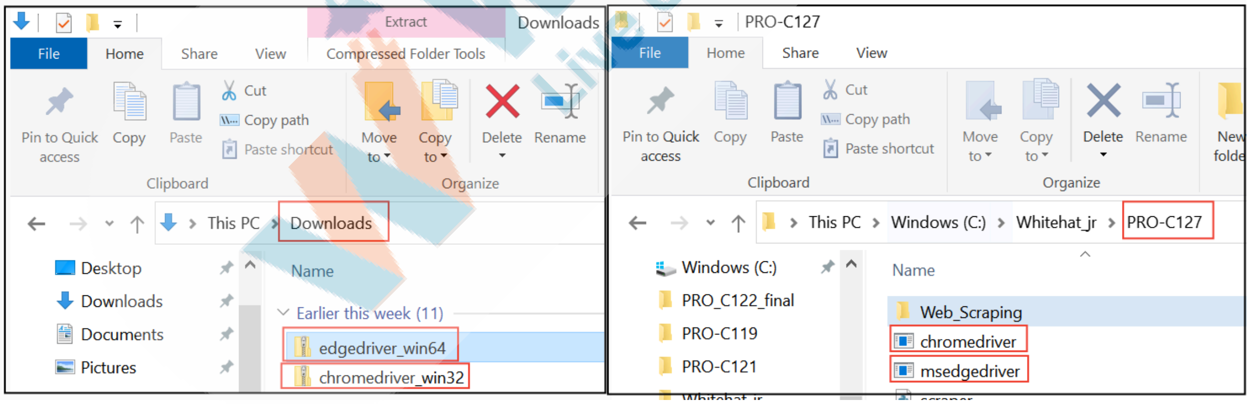
You can view the current implementation status of the WebDriver standard here.

All versions available in Downloads

- Latest stable release: ChromeDriver 98.0.4758.80
- Previous stable release: ChromeDriver 97.0.4692.71

3. Download the driver.

   - It will be in zip format so extract the file.
   - Save it in the same directory where you have the Python file.



4. Define a **browser** variable and assign the webdriver to it.

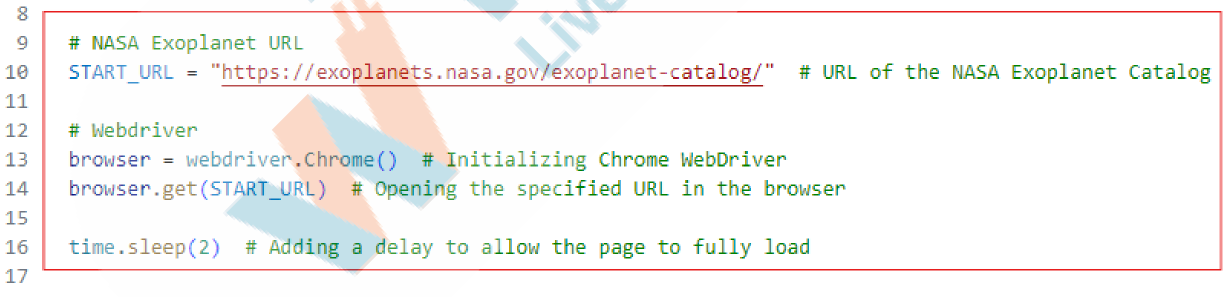
   Syntax:

browser = webdriver.name_of_browser(<path of webdriver.exe file>)

- **name_of_browser** can be **Chrome** (Google Chrome) or **Edge** (Microsoft Edge) etc.
- **path of webdriver.exe file:** Provide the path of the **webdriver.exe** file downloaded in your system. Leave it empty if the file is in same folder as the scrapper.py

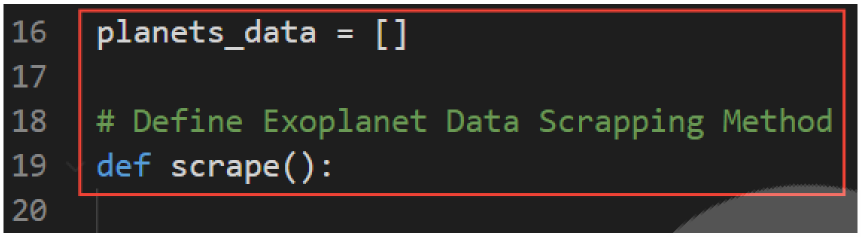
***Note:** While providing the path of the web driver replace the backward slash with forward slash.*

5. **Open the link using the browser:**
   Use **browser.get(<URL>)** method to open the link. Pass the link in this method.



```

# NASA Exoplanet URL
START_URL = "https://exoplanets.nasa.gov/exoplanet-catalog/"  # URL of the NASA Exoplanet Catalog

# Webdriver
browser = webdriver.Chrome()  # Initializing Chrome WebDriver
browser.get(START_URL)  # Opening the specified URL in the browser

time.sleep(2)  # Adding a delay to allow the page to fully load

```

1. Create a list **planet_data,** we'll save all the details of the planet.
2. We will create a function called **scrape()**, to scrape()



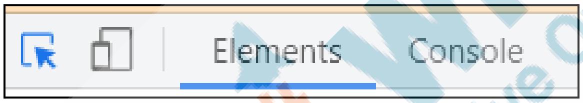
```
planets_data = []

# Define Exoplanet Data Scrapping Method
def scrape():
```

Now, let's just try to scrape the first page only.

Inspect a webpage:

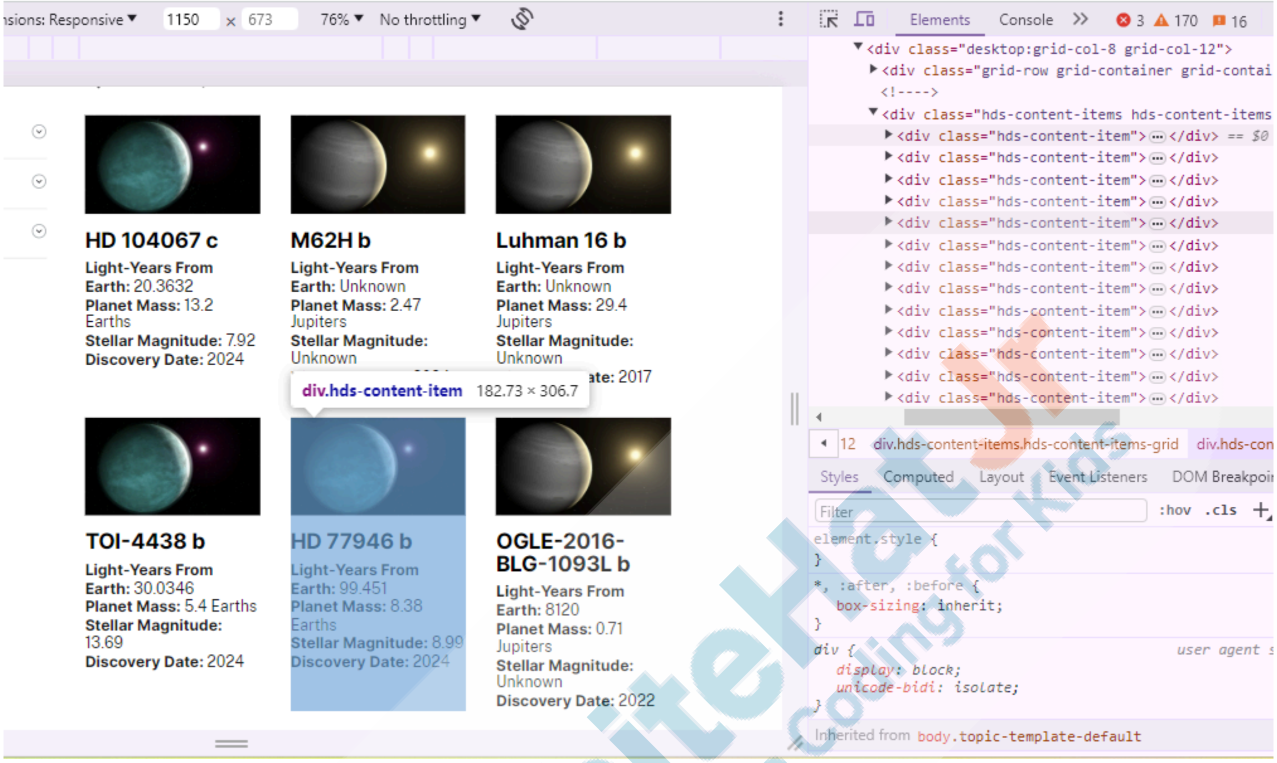
Before we do that, let's inspect the page:

1. Open EXOPLANET EXPLORATION.
2. Press **Ctrl + Shift + i** or Right-click the **webpage** and click on **inspect** option to open inspect window
3. Click on the "**Elements**" and diagonally pointing arrow on the left most corner of the inspect window menu.



4. Hover over the elements to inspect the HTML tags

We can see that each planet info is inside a **<div>** with the **class** as **'hds-content-item'**.



HD 104067 c
Light-Years From Earth: 20.3632
Planet Mass: 13.2 Earths
Stellar Magnitude: 7.92
Discovery Date: 2024

M62H b
Light-Years From Earth: Unknown
Planet Mass: 2.47 Jupiters
Stellar Magnitude: Unknown

Luhman 16 b
Light-Years From Earth: Unknown
Planet Mass: 29.4 Jupiters
Stellar Magnitude: Unknown
…ate: 2017

div.hds-content-item 182.73 × 306.7

TOI-4438 b
Light-Years From Earth: 30.0346
Planet Mass: 5.4 Earths
Stellar Magnitude: 13.69
Discovery Date: 2024

HD 77946 b
Light-Years From Earth: 99.451
Planet Mass: 8.38 Earths
Stellar Magnitude: 8.99
Discovery Date: 2024

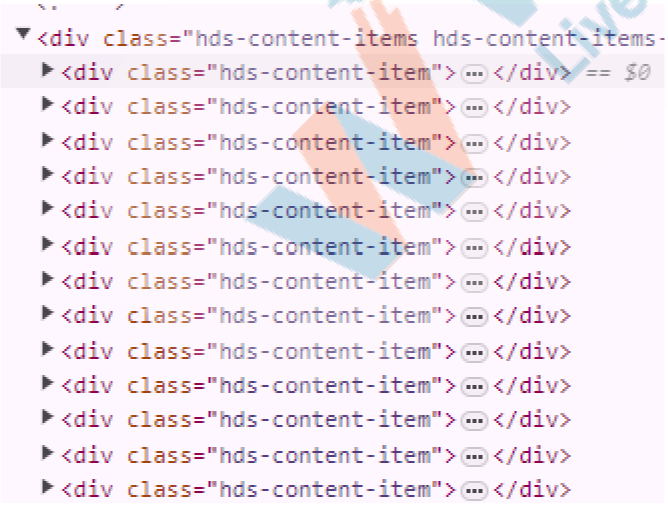
OGLE-2016-BLG-1093L b
Light-Years From Earth: 8120
Planet Mass: 0.71 Jupiters
Stellar Magnitude: Unknown
Discovery Date: 2022

```
▼<div class="desktop:grid-col-8 grid-col-12">
  ▶<div class="grid-row grid-container grid-contai
   <!---->
  ▼<div class="hds-content-items hds-content-items
    ▶<div class="hds-content-item">…</div> == $0
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
    ▶<div class="hds-content-item">…</div>
```

12 div.hds-content-items.hds-content-items-grid div.hds-con

```
element.style {
}
*, :after, :before {
  box-sizing: inherit;
}
div {                                user agent s
  display: block;
  unicode-bidi: isolate;
}
Inherited from body.topic-template-default
```

Therefore, we need to **find all the <div> tags with class=" hds-content-item"** in order to scrape the data.



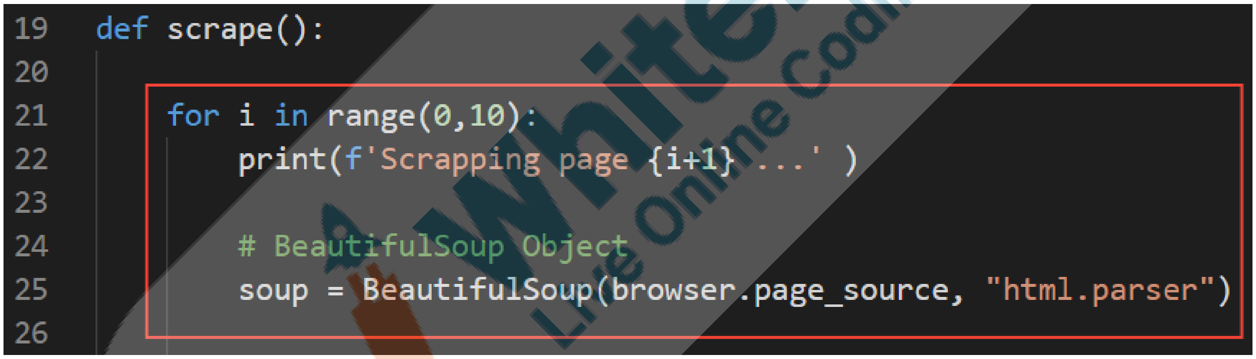
```
▼<div class="hds-content-items hds-content-items-
  ▶<div class="hds-content-item">…</div> == $0
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
  ▶<div class="hds-content-item">…</div>
```

We can do this with the following code:

1. Use the **for** loop to iterate over 5 pages.
2. Print the current page number being scraped.
3. Create a **BeautifulSoup** object called **soup**.

   Earlier, the browser window we opened with Selenium, was named **browser**.

   Now, we are creating soup.
   It is a **BeautifulSoup** object where we are passing the first argument as the browser's page source using the **.page_source** attribute to get the HTML page code, and **html.parser** as the second argument to extract the page content of the **HTML** tags.

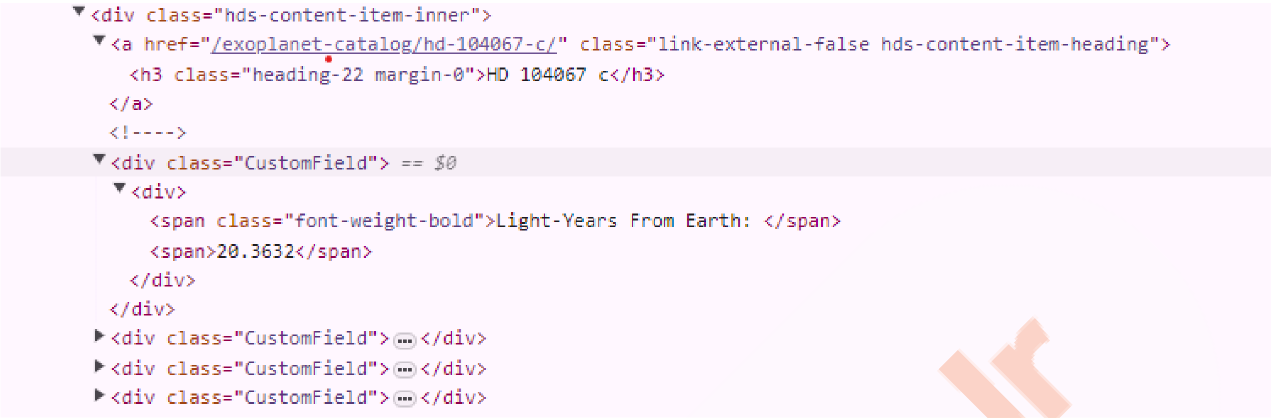


```
def scrape():

    for i in range(0,10):
        print(f'Scrapping page {i+1} ...' )

        # BeautifulSoup Object
        soup = BeautifulSoup(browser.page_source, "html.parser")

```

Next, we are creating a **for** loop to iterate over all the **<divl>** tags with class='hds-content-item'.

Inside it we are using the **soup.find_all()** method.
In this method, we have to mention the tag and its attributes.
It will find all the **div** with a **class** as **"hds-content-item".**

Let's again check the HTML with google inspect to see what's inside the **<div>** tag.



```
▼<div class="hds-content-item-inner">
  ▼<a href="/exoplanet-catalog/hd-104067-c/" class="link-external-false hds-content-item-heading">
      <h3 class="heading-22 margin-0">HD 104067 c</h3>
    </a>
    <!---->
  ▼<div class="CustomField"> == $0
    ▼<div>
        <span class="font-weight-bold">Light-Years From Earth: </span>
        <span>20.3632</span>
      </div>
    </div>
  ▶<div class="CustomField">…</div>
  ▶<div class="CustomField">…</div>
  ▶<div class="CustomField">…</div>
```

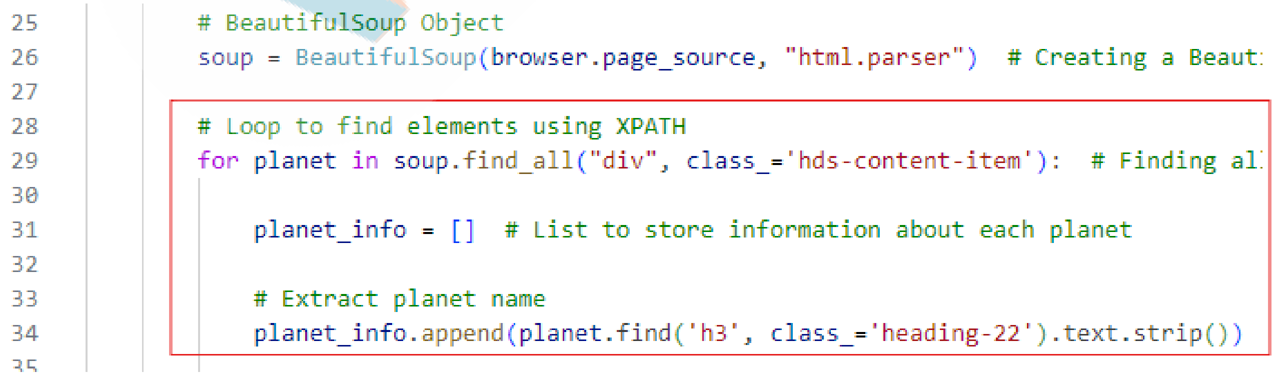
Here, we can see that the name of the planet is inside a <**h3**> with class **heading-22**.
Also there is two span for every other info where first span tells what the info is and second holds the info

Can you tell me how I can find **<h3>** tags with class **heading-22**

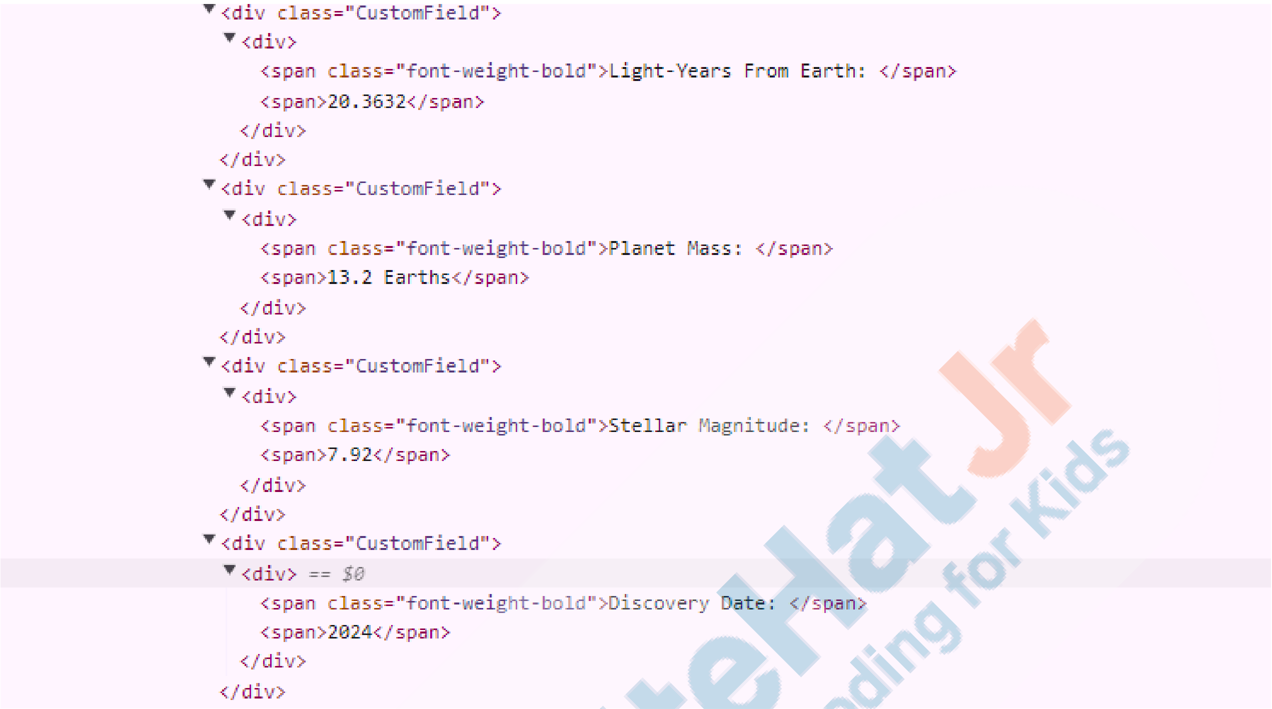
ESR: li_tags = ul_tag.find_all("h3", class=”heading-22”)

Right but we will use **find** and not **find_all** as we only have to find one inside the current iteration of the planet.

Great, now all we have to do is to iterate over these **<div>** tags and fetch the data, create a temporary list and then finally append that list into the **planet_data** list that we created earlier.



```
        # BeautifulSoup Object
        soup = BeautifulSoup(browser.page_source, "html.parser")  # Creating a Beaut

        # Loop to find elements using XPATH
        for planet in soup.find_all("div", class_='hds-content-item'):  # Finding al

            planet_info = []  # List to store information about each planet

            # Extract planet name
            planet_info.append(planet.find('h3', class_='heading-22').text.strip())
```



```
▼<div class="CustomField">
  ▼<div>
     <span class="font-weight-bold">Light-Years From Earth: </span>
     <span>20.3632</span>
   </div>
 </div>
▼<div class="CustomField">
  ▼<div>
     <span class="font-weight-bold">Planet Mass: </span>
     <span>13.2 Earths</span>
   </div>
 </div>
▼<div class="CustomField">
  ▼<div>
     <span class="font-weight-bold">Stellar Magnitude: </span>
     <span>7.92</span>
   </div>
 </div>
▼<div class="CustomField">
  ▼<div> == $0
     <span class="font-weight-bold">Discovery Date: </span>
     <span>2024</span>
   </div>
 </div>
```

Here, we can see that the first **<span> tags** have the properties of planet as `["Light-Years From Earth", "Planet Mass", "Stellar Magnitude", "Discovery Date"]`

And the next **span** has that **value**. We can find a **span** by its **text** and then navigate to the next **span** for the required **value**. And as we have to do it for 4 value will use **for each** loop to loop over each **text** to find a **span** and then go to next **span** to get the **value**

For this, we need to make a list of text **information_to_extract** as these text

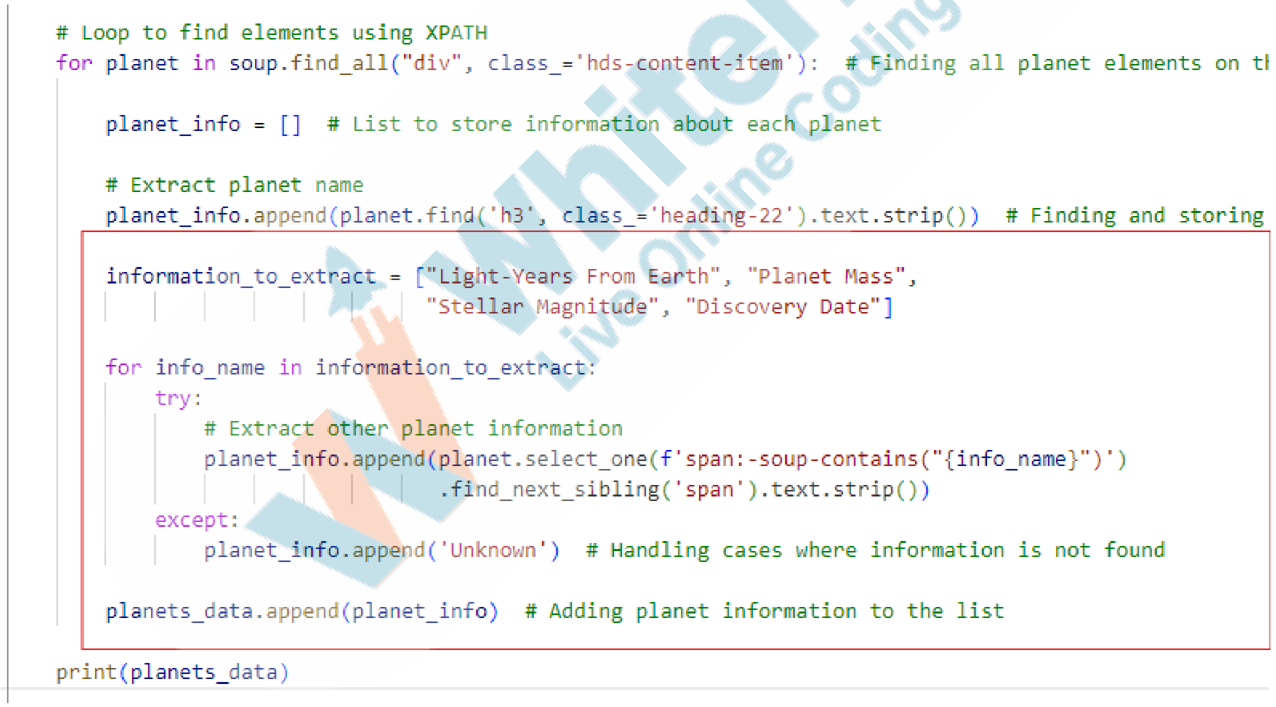
For this, we will write the following code:

1. Create a list called **information_to_extract .**
2. Run a **for each** loop on **information_to_extract** .

ESR: It runs for each value of the list once

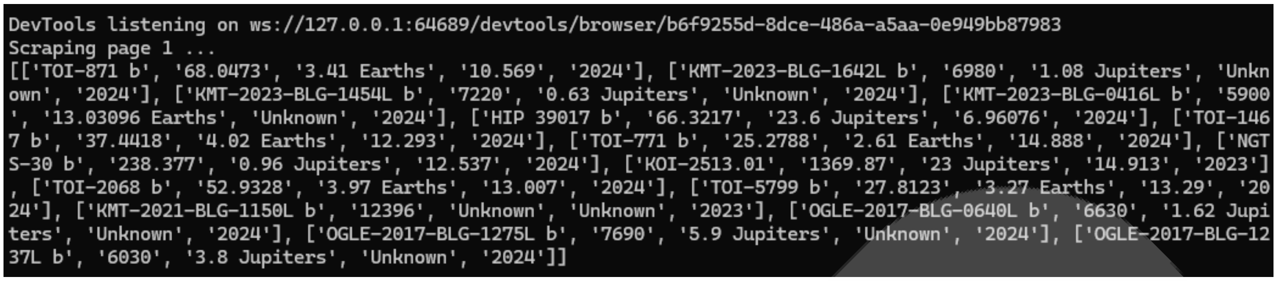
Can you tell me the use of **for-each** loop?

3. Add a **try-except** block so that is anything goes wrong or there is a missing value we can add Undefined in **except** block
4. Find the span using **select_one** method on the planet.
5. Then move to the next span using **find_next_sibling** method.
6. Lastly, strip the text value required using **text.strip()**.
7. Let’s try to check the list **palnet_data** that we have created just now..



```
# Loop to find elements using XPATH
for planet in soup.find_all("div", class_='hds-content-item'):  # Finding all planet elements on th

    planet_info = []  # List to store information about each planet

    # Extract planet name
    planet_info.append(planet.find('h3', class_='heading-22').text.strip())  # Finding and storing

    information_to_extract = ["Light-Years From Earth", "Planet Mass",
                              "Stellar Magnitude", "Discovery Date"]

    for info_name in information_to_extract:
        try:
            # Extract other planet information
            planet_info.append(planet.select_one(f'span:-soup-contains("{info_name}")')
                               .find_next_sibling('span').text.strip())
        except:
            planet_info.append('Unknown')  # Handling cases where information is not found

    planets_data.append(planet_info)  # Adding planet information to the list

print(planets_data)
```

Run the file using the command prompt and also make sure to run the main for loop from 0 to 1.



```
DevTools listening on ws://127.0.0.1:64689/devtools/browser/b6f9255d-8dce-486a-a5aa-0e949bb87983
Scraping page 1 ...
[['TOI-871 b', '68.0473', '3.41 Earths', '10.569', '2024'], ['KMT-2023-BLG-1642L b', '6980', '1.08 Jupiters', 'Unkn
own', '2024'], ['KMT-2023-BLG-1454L b', '7220', '0.63 Jupiters', 'Unknown', '2024'], ['KMT-2023-BLG-0416L b', '5900
', '13.03096 Earths', 'Unknown', '2024'], ['HIP 39017 b', '66.3217', '23.6 Jupiters', '6.96076', '2024'], ['TOI-146
7 b', '37.4418', '4.02 Earths', '12.293', '2024'], ['TOI-771 b', '25.2788', '2.61 Earths', '14.888', '2024'], ['NGT
S-30 b', '238.377', '0.96 Jupiters', '12.537', '2024'], ['KOI-2513.01', '1369.87', '23 Jupiters', '14.913', '2023']
, ['TOI-2068 b', '52.9328', '3.97 Earths', '13.007', '2024'], ['TOI-5799 b', '27.8123', '3.27 Earths', '13.29', '20
24'], ['KMT-2021-BLG-1150L b', '12396', 'Unknown', 'Unknown', '2023'], ['OGLE-2017-BLG-0640L b', '6630', '1.62 Jupi
ters', 'Unknown', '2024'], ['OGLE-2017-BLG-1275L b', '7690', '5.9 Jupiters', 'Unknown', '2024'], ['OGLE-2017-BLG-12
37L b', '6030', '3.8 Jupiters', 'Unknown', '2024']]
```

Thus, you can see that **planet_data** is a **list of lists**.
Now, one final thing that we need to still figure out is, how to change the page by clicking on the next button. Now you have to create the function and then we'll automate the browser to turn the pages and scrape the data.
Moreover, you'll have to create a CSV file for storing the data.

Teacher Stops Screen Share

Please share your screen with me.

Teacher Starts Slideshow
Slide 14 to 16
Refer to speaker notes and follow the instructions on each slide.

We have one more class challenge for you.
Can you solve it?

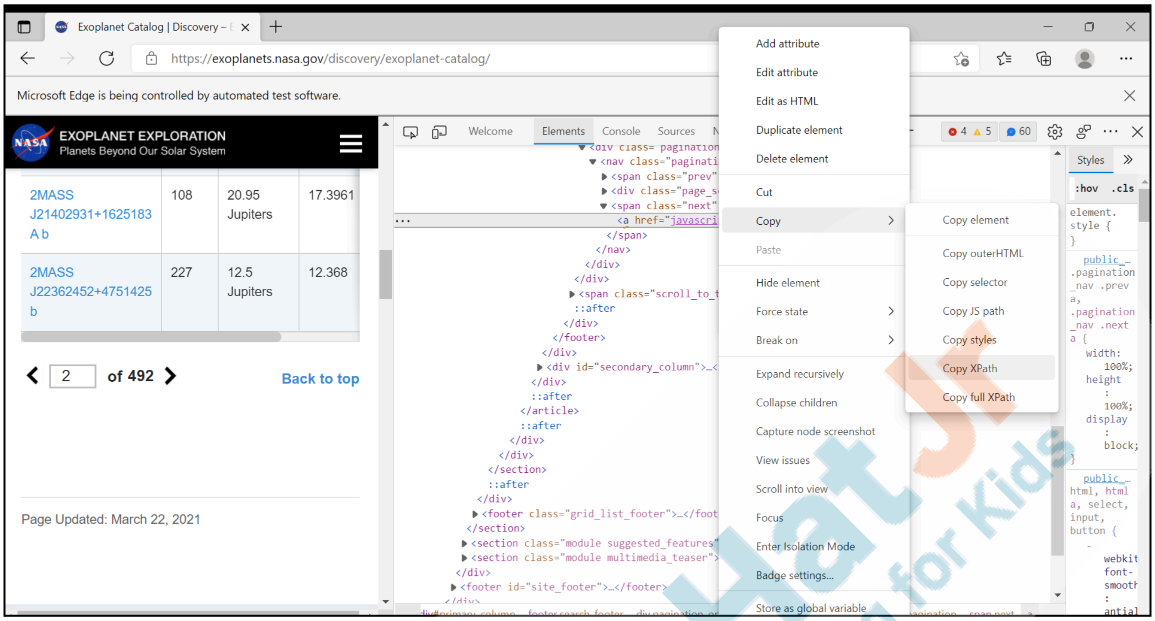
Let's try. I will guide you through it.

Teacher Ends Slideshow

STUDENT-LED ACTIVITY - 20 mins

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Full Screen.**

Student Initiates Screen Share	
● **Create scrape function** ● **Automate the browser to turn the page** ● **Store the data into a CSV file**	
Teacher Action	**Student Action**
Open Student Activity 1 for Boilerplate code. ***Note:*** *The student will write the scrape function as the teacher has written. Guide the student to create the function, run the file using the command prompt. (Create Virtual Environment)*	
Great! So we have to check the browser for turning the pages automatically. We have a button at the **bottom** of the page that is used to go to the next page. We need the **XPath** for this button. Here, we are finding an element with XPath, and then clicking it to turn the page. XPath can be used to navigate through elements and attributes in an XML document. **XML** stands for **eXtensible Markup Language**. XML is a markup language much like HTML. XML was designed to store and transport data. **XPath** is a syntax for defining parts of an XML document. Here, we are using it to define the button. We just need to right-click on the element and click on inspect. An **<button>** tag for the element is given. Right-click this tag and click on **copy XPath**.	



2MASS J21402931+1625183 A b	108	20.95 Jupiters	17.3961
2MASS J22362452+4751425 b	227	12.5 Jupiters	12.368

Page Updated: March 22, 2021

```
▼<div class="pagination
  ▼<nav class="paginati
    ▶<span class="prev"
    ▶<div class="page_s
    ▼<span class="next"
      <a href="javascri
      </span>
    </nav>
  </div>
</div>
▶<span class="scroll_to_t
  ::after
</div>
</footer>
</div>
▶<div id="secondary_column">…<
</div>
::after
</article>
::after
</div>
</div>
</section>
::after
</div>
▶<footer class="grid_list_footer">…</foot
</section>
▶<section class="module suggested_features"
▶<section class="module multimedia_teaser">
</div>
▶<footer id="site_footer">…</footer>
```

```
element.
style {
}
public_…
.pagination
_nav .prev
a,
.pagination
_nav .next
a {
  width:
    100%;
  height
    :
    100%;
  display
    :
    block;
}
public_…
html, html
a, select,
input,
button {
  -
    webkit
    font-
    smooth
    :
    antial
```

Go to **scraper.py** Python file.

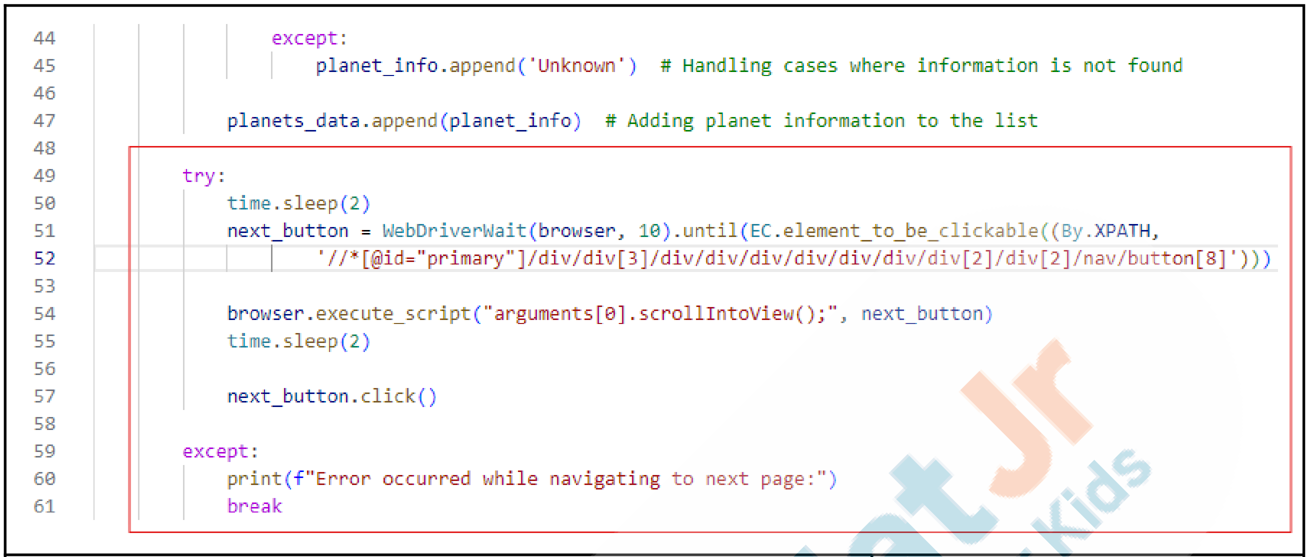
Now we'll be writing the **find_element()** function for the browser to locate the button.

The element or button is located by XPath.

Thus in this function parameter specify **By.XPATH** using the **'by'** variable.

Also, value takes the actual XPath we just copied from the browser. After finding the element it is clicked using the **click()** function.

Since we have to repeat it for 5 pages, let's keep it inside the **for** loop.



```
                except:
                    planet_info.append('Unknown')  # Handling cases where information is not found

            planets_data.append(planet_info)  # Adding planet information to the list

        try:
            time.sleep(2)
            next_button = WebDriverWait(browser, 10).until(EC.element_to_be_clickable((By.XPATH,
                    '//*[@id="primary"]/div/div[3]/div/div/div/div/div/div[2]/div[2]/nav/button[8]')))

            browser.execute_script("arguments[0].scrollIntoView();", next_button)
            time.sleep(2)

            next_button.click()

        except:
            print(f"Error occurred while navigating to next page:")
            break
```

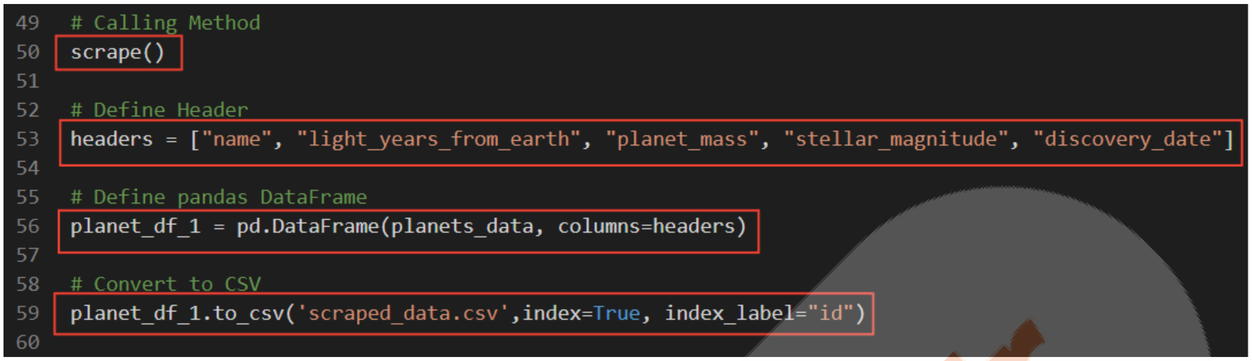
1. Call the **scrape()** function to scrape the data.

Now we have to store the data in a CSV file. For this we will use the Python **pandas** module.

Do you remember what pandas DataFrame is?

2. Create the list of headers that will be used as column names in the CSV file for the data we scraped.
3. Create **pandas** DataFrame to append list **planets_data** with column headers.
4. Use the **to_csv()** method of pandas to convert the DataFrame into a csv file:
    a. Provide the name of the file in this method. This file will be generated automatically in the same directory.
    b. To add the first column with serial numbers, use the **index** attribute with the label **'id'**.

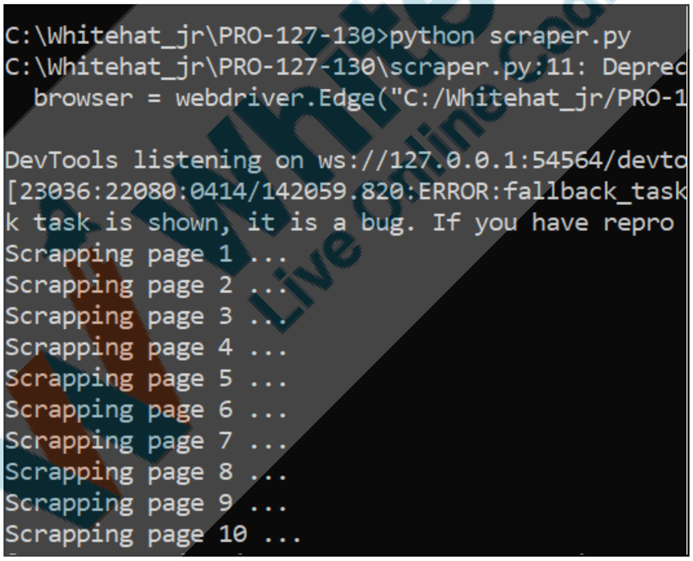
ESR: Yes, The **pandas** DataFrame stores data in tabular format using rows and columns.



```
# Calling Method
scrape()

# Define Header
headers = ["name", "light_years_from_earth", "planet_mass", "stellar_magnitude", "discovery_date"]

# Define pandas DataFrame
planet_df_1 = pd.DataFrame(planets_data, columns=headers)

# Convert to CSV
planet_df_1.to_csv('scraped_data.csv',index=True, index_label="id")

```

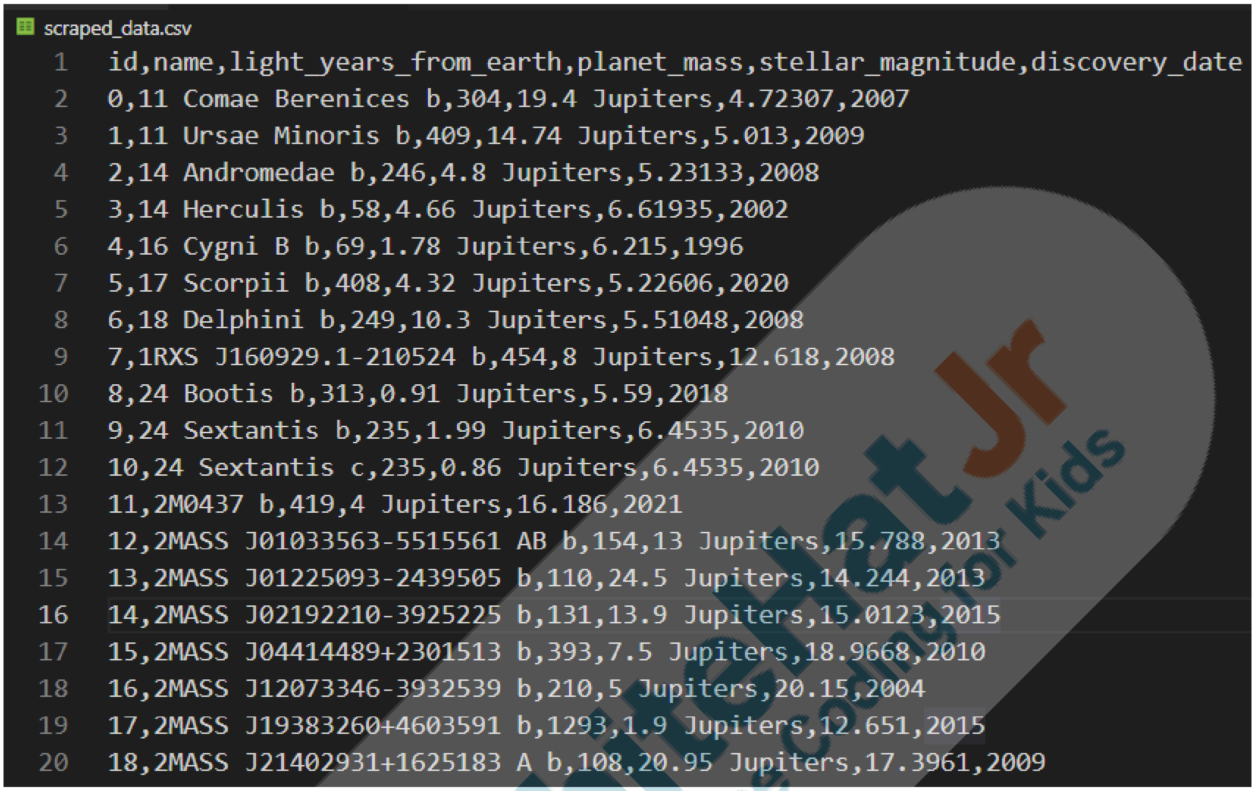
To run the file go to the command prompt. Create a virtual environment. Activate it and install all the necessary libraries.

Note: *Guide the student to run the Python file using a virtual environment (scraper.py).*

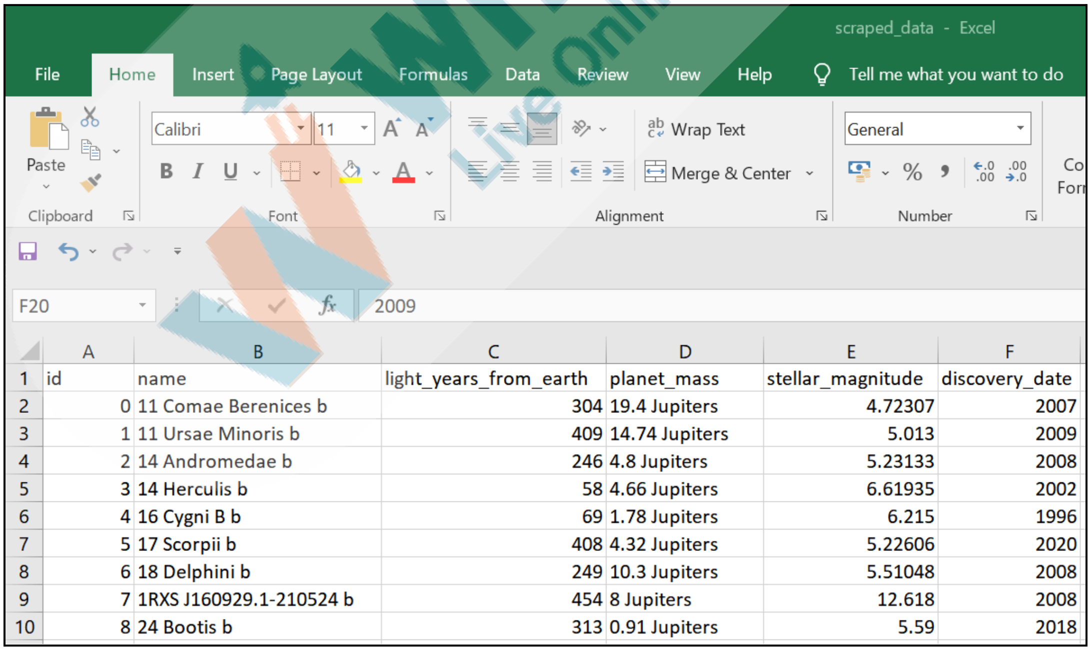


```
C:\Whitehat_jr\PRO-127-130>python scraper.py
C:\Whitehat_jr\PRO-127-130\scraper.py:11: Deprec
  browser = webdriver.Edge("C:/Whitehat_jr/PRO-1

DevTools listening on ws://127.0.0.1:54564/devto
[23036:22080:0414/142059.820:ERROR:fallback_task
k task is shown, it is a bug. If you have repro
Scrapping page 1 ...
Scrapping page 2 ...
Scrapping page 3 ...
Scrapping page 4 ...
Scrapping page 5 ...
Scrapping page 6 ...
Scrapping page 7 ...
Scrapping page 8 ...
Scrapping page 9 ...
Scrapping page 10 ...
```

It opened the browser and started scraping the data into a CSV file. Go back to VS Code to check the **scraped_data.csv** file. Also, check the file in the directory.



```
id,name,light_years_from_earth,planet_mass,stellar_magnitude,discovery_date
0,11 Comae Berenices b,304,19.4 Jupiters,4.72307,2007
1,11 Ursae Minoris b,409,14.74 Jupiters,5.013,2009
2,14 Andromedae b,246,4.8 Jupiters,5.23133,2008
3,14 Herculis b,58,4.66 Jupiters,6.61935,2002
4,16 Cygni B b,69,1.78 Jupiters,6.215,1996
5,17 Scorpii b,408,4.32 Jupiters,5.22606,2020
6,18 Delphini b,249,10.3 Jupiters,5.51048,2008
7,1RXS J160929.1-210524 b,454,8 Jupiters,12.618,2008
8,24 Bootis b,313,0.91 Jupiters,5.59,2018
9,24 Sextantis b,235,1.99 Jupiters,6.4535,2010
10,24 Sextantis c,235,0.86 Jupiters,6.4535,2010
11,2M0437 b,419,4 Jupiters,16.186,2021
12,2MASS J01033563-5515561 AB b,154,13 Jupiters,15.788,2013
13,2MASS J01225093-2439505 b,110,24.5 Jupiters,14.244,2013
14,2MASS J02192210-3925225 b,131,13.9 Jupiters,15.0123,2015
15,2MASS J04414489+2301513 b,393,7.5 Jupiters,18.9668,2010
16,2MASS J12073346-3932539 b,210,5 Jupiters,20.15,2004
17,2MASS J19383260+4603591 b,1293,1.9 Jupiters,12.651,2015
18,2MASS J21402931+1625183 A b,108,20.95 Jupiters,17.3961,2009
```



	A	B	C	D	E	F
1	id	name	light_years_from_earth	planet_mass	stellar_magnitude	discovery_date
2	0	11 Comae Berenices b	304	19.4 Jupiters	4.72307	2007
3	1	11 Ursae Minoris b	409	14.74 Jupiters	5.013	2009
4	2	14 Andromedae b	246	4.8 Jupiters	5.23133	2008
5	3	14 Herculis b	58	4.66 Jupiters	6.61935	2002
6	4	16 Cygni B b	69	1.78 Jupiters	6.215	1996
7	5	17 Scorpii b	408	4.32 Jupiters	5.22606	2020
8	6	18 Delphini b	249	10.3 Jupiters	5.51048	2008
9	7	1RXS J160929.1-210524 b	454	8 Jupiters	12.618	2008
10	8	24 Bootis b	313	0.91 Jupiters	5.59	2018

Great work!!!

So today we were able to access the data of a website. We learned about BeautifulSoup and Selenium to scrape exoplanet data from NASA's website.	

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

Teacher Starts Slideshow
Slide 17 to 22

Activity details

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Continue WRAP-UP Session
Slide 23 to 28

Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	Student Action
You get “hats-off” for your excellent work! In the next class, we’ll be scraping more data and learning more useful techniques.	*Make sure you have given at least 2 hats-off during the class for:* Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections	
Teacher Clicks ✖ End Class	

ACTIVITY LINKS

Activity Name	Description	Links
Teacher Activity 1	Exoplanet Exploration	https://exoplanets.nasa.gov/discovery/exoplanet-catalog/
Teacher Activity 2	Beautiful Soup 4	https://www.crummy.com/software/BeautifulSoup/bs4/doc/
Teacher Activity 3	Selenium	https://www.selenium.dev/documentation/
Teacher Activity 4	Selenium Webdriver	https://www.selenium.dev/downloads/
Teacher Activity 5	Browser installation Version Check	https://docs.google.com/document/d/1O-iWKsRJIGW9MEqaOi3_n4HAVqMRkE2Wxbhul4jQKM0/edit?usp=sharing

Teacher Activity 6	Reference code	https://github.com/React-Native-Frontier/1-1-V3-C127
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/3754887a-1945-4d20-bf34-291c260653e9.pdf
Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-C127-Project-Solution
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/ba9890c9-2f64-4d20-a4a1-9c3b8fd9db26.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/e02d43a0-e53c-461b-be68-1ec7b2f4f4c6.pdf
Student Activity 1	Boilerplate Code	https://github.com/React-Native-Frontier/1-1-V3-C127