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

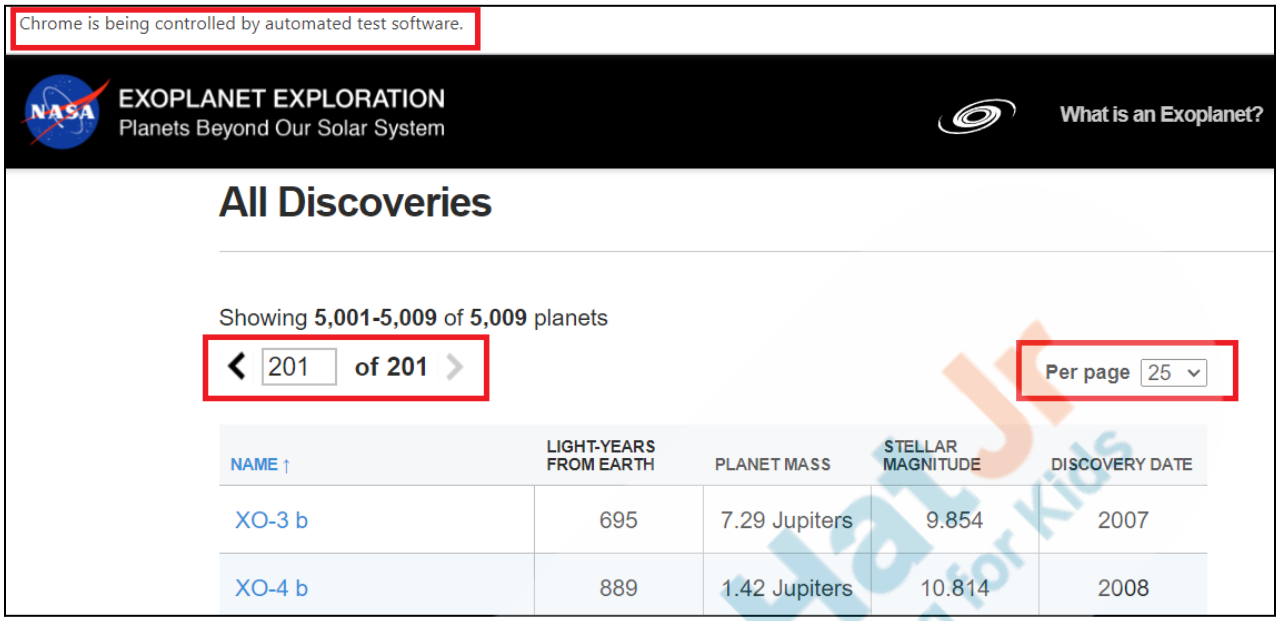
Continue WARM-UP Session 
Slide 5 to 13

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* ***201 Pages*** *with* ***25 Planets per page*** *(except the last page) showing a total of* ***5009 planets data****.*	**ESR:** Yes



Here you can see 5,009 exoplanets are being discovered and data are given in a tabular 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.*

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.

ESR: Varied

ESR: Yes

Are you excited? 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 ** ** tags inside the ul tags, 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
```

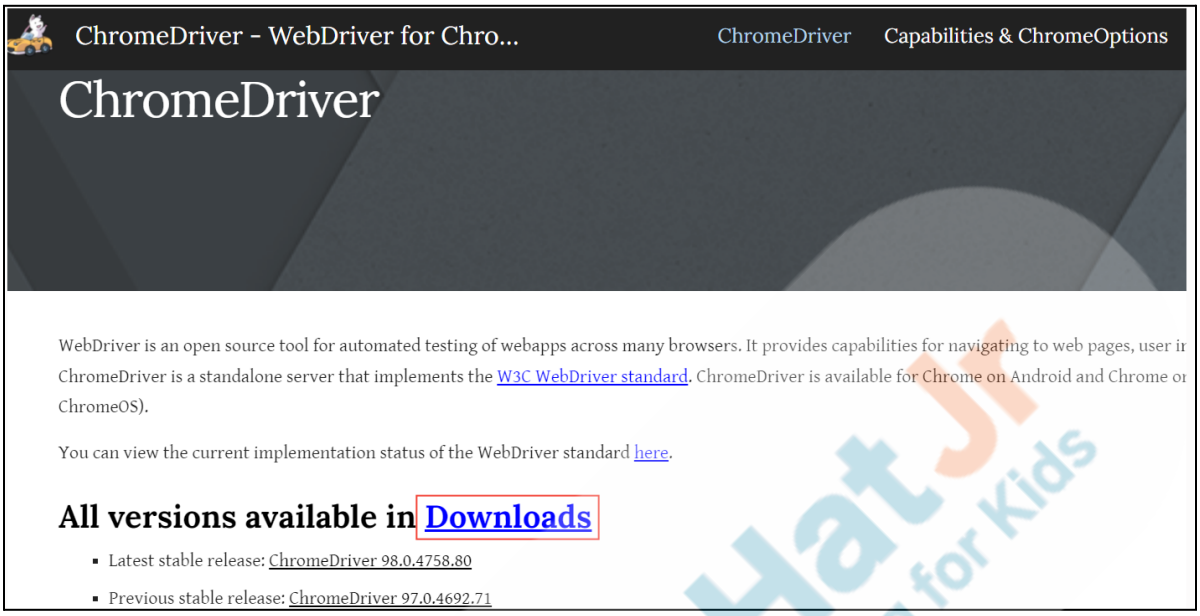
Now, we have to define:

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:

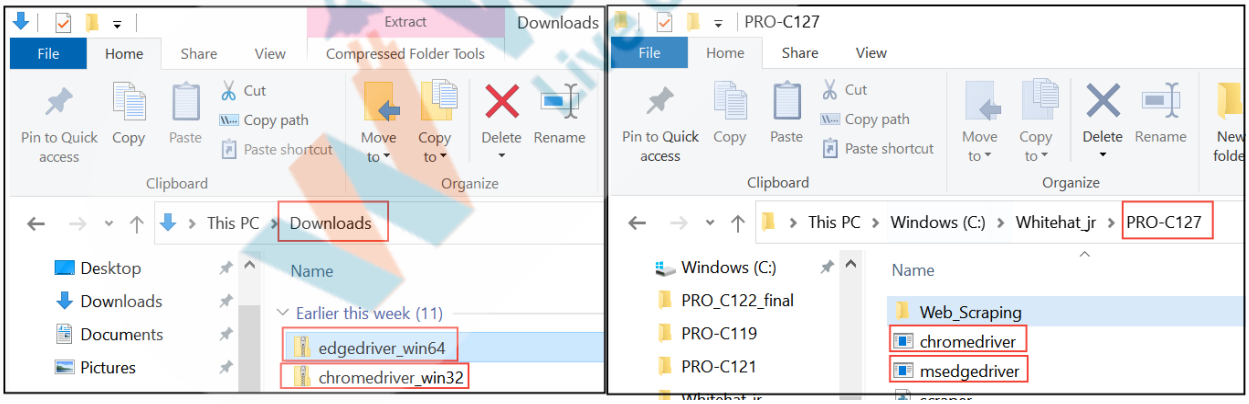
Teacher Activity 4: Webdriver for Selenium

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



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.

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.

```
7   # NASA Exoplanet URL
8   START_URL = "https://exoplanets.nasa.gov/exoplanet-catalog/"
9
10  # Webdriver
11  browser = webdriver.Edge("C:/Whitehat_jr/PRO-127-130/msedgedriver.exe")
12  browser.get(START_URL)
13
14  time.sleep(10)
15
```

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()

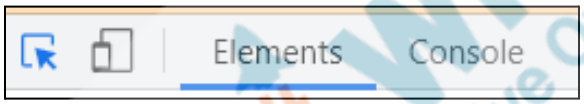
```
16  planets_data = []
17
18  # Define Exoplanet Data Scrapping Method
19  def scrape():
20
```

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

Inspect a webpage:

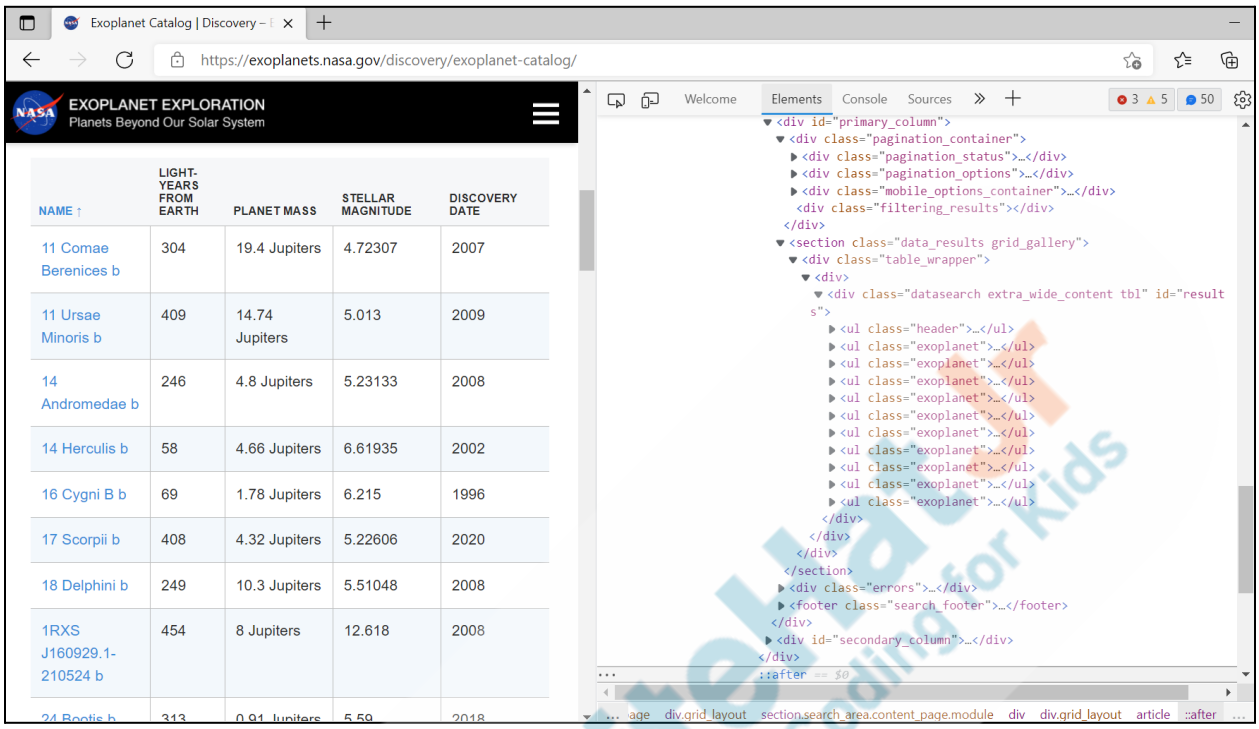
Before we do that, let's inspect the page:

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 all the rows in the table are **<ul>** tags with the **class** as **exoplanet**.



Therefore, we need to **find all the <ul> tags with class=" exoplanet"** in order to scrape the data.

We can do this with the following code:

1. Use the **for** loop to iterate over 10 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**.

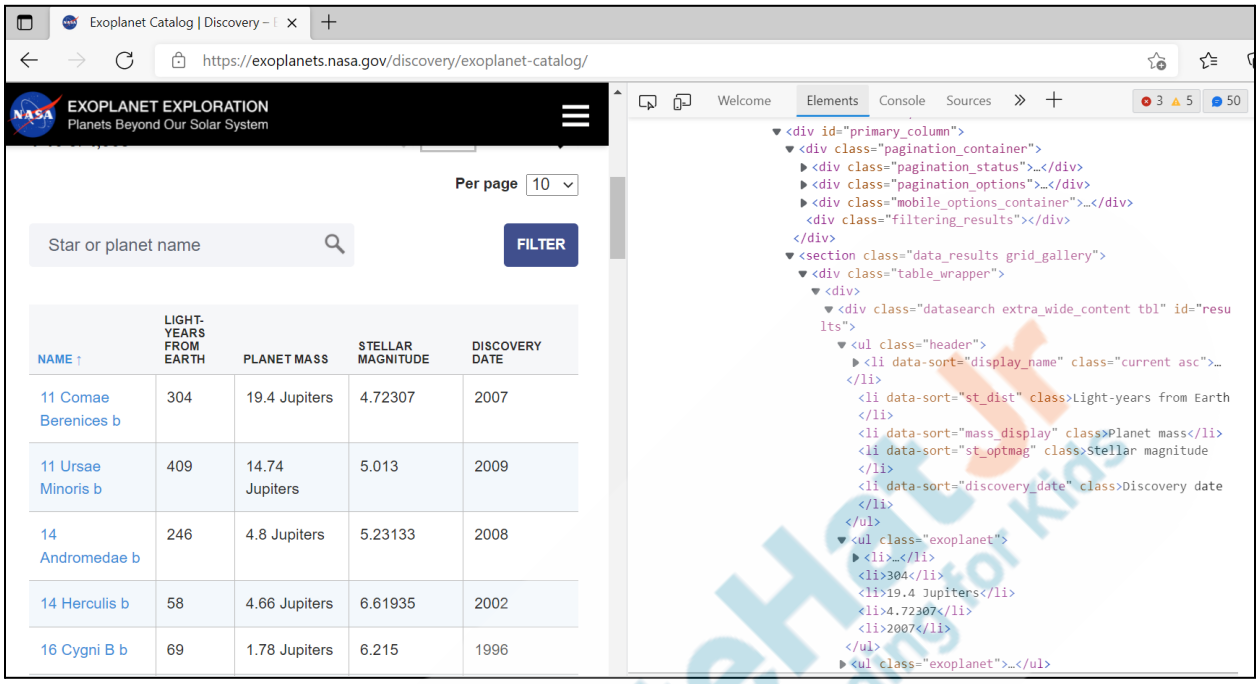
   Now, we are creating soup.
   It is a **BeautifulSoup** object where we are passing the first argument as the browser's page source using the **.page_source** attribute to get the HTML page code, and **html.parser** as the second argument to extract the page content of the **HTML** tags.

```
19  def scrape():
20
21      for i in range(0,10):
22          print(f'Scrapping page {i+1} ...' )
23
24          # BeautifulSoup Object
25          soup = BeautifulSoup(browser.page_source, "html.parser")
26
```

Next, we are creating a **for** loop to iterate over all the **<ul>** tags using variable **ul_tags**.

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 **ul_tags** with a **class** as **"exoplanet".**

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



Here, we can see that it consists of **<li>** tags inside which we can get listed data. Again, we will need to iterate over all the <li> tags. For this, we will find all the **<li>** tags.

Can you tell me how I can find all the **<li>** tags inside the **<ul>** tag?

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

ESR: li_tags = ul_tag.find_all("li")

```python
def scrape():

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

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

        # Loop to find elements inside ul and li tags
        for ul_tag in soup.find_all("ul", attrs={"class", "exoplanet"}):

            li_tags = ul_tag.find_all("li")

```

```html
<ul class="exoplanet">
  <li>
    <a href="/exoplanet-catalog/6988/11-comae-berenices-b/">11 Comae Berenices b</a>
  </li>
  <li>304</li>
  <li>19.4 Jupiters</li>
  <li>4.72307</li>
  <li>2007</li>
</ul>
```

Here, we can see that the ** tags** have the name of the planet inside an **<a>** tag which is the **anchor tag**, and other details directly as HTML.

For this, we need to make sure that **we treat the first <li> tag differently and others differently**.

For this, we will write the following code:

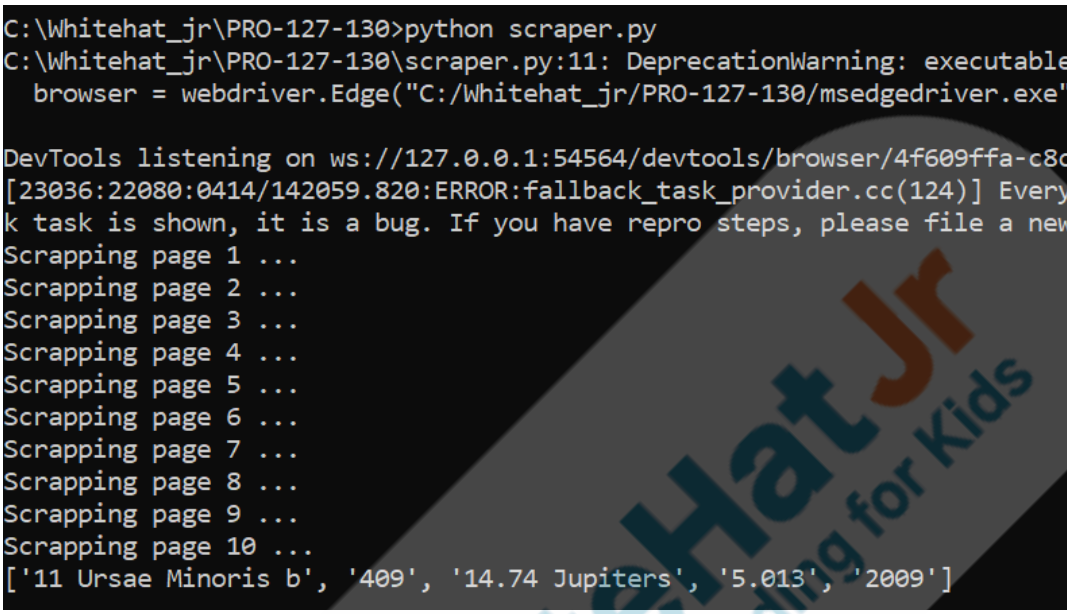
1. Create an empty list called **temp_list.**
2. Get the **li_tags** with **index** using the **enumerate()** function.

Can you tell me the use of **enumerate** function?

ESR: Enumerate is a function that returns the index along with the element.

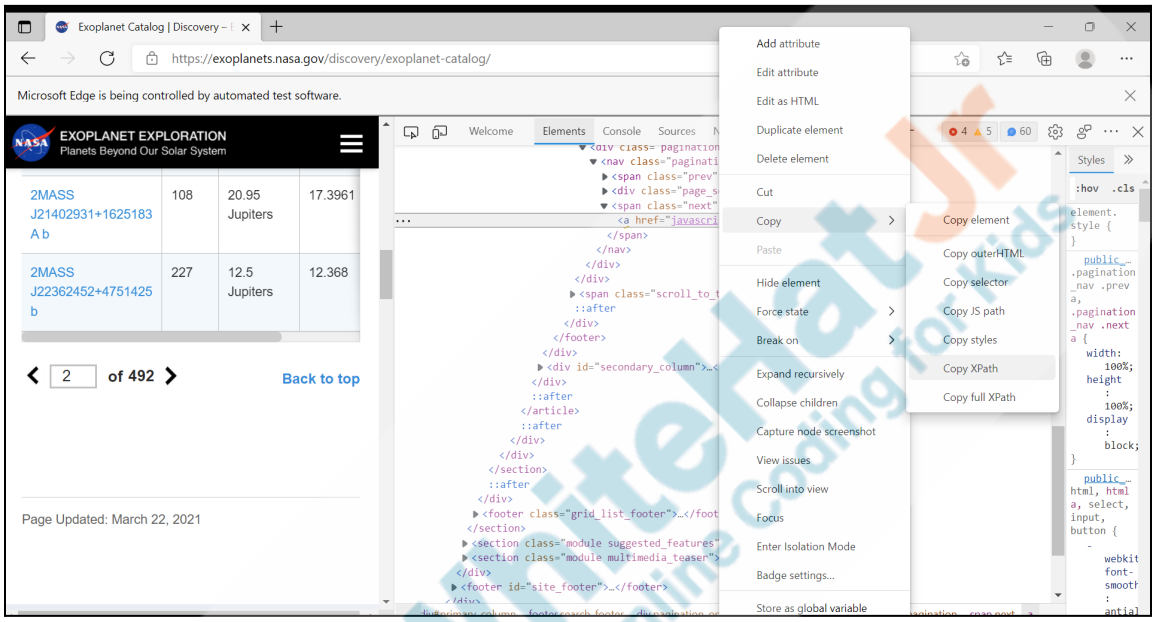
3. Use the **enumerate()** method to get the list of **indexes** and **tags**.
4. If the index is zero, then we are appending the contents of the **<a>** tag in the **temp_list**.
5. Else we'll be appending the contents of all the **<li>** tags. Inside **else**, we are handling an exception with the **try** keyword. If no content is present then append an empty string.
6. Lastly, we will append this **temp_list** into the **planet_data**.
7. Let's try to check the list **palnet_data** that we have created just now. Access any index number to check the data. For example (**planets_data[1]**) will print the planet data at index number 1.

```
def scrape():

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

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

        # Loop to find elements inside ul and li tags
        for ul_tag in soup.find_all("ul", attrs={"class", "exoplanet"}):

            li_tags = ul_tag.find_all("li")

            temp_list = []

            for index, li_tag in enumerate(li_tags):

                if index == 0:
                    temp_list.append(li_tag.find_all("a")[0].contents[0])
                else:
                    try:
                        temp_list.append(li_tag.contents[0])
                    except:
                        temp_list.append("")

            planets_data.append(temp_list)

print(planets_data[1])
```

Run the file using the command prompt.	
	
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 **<a>** tag for the element is given. Right-click this tag and click on **copy XPath**.



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 10 pages, let's keep it inside the **for** loop.

```
def scrape():

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

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

        # Loop to find elements inside ul and li tags
        for ul_tag in soup.find_all("ul", attrs={"class", "exoplanet"}):

            li_tags = ul_tag.find_all("li")

            temp_list = []

            for index, li_tag in enumerate(li_tags):

                if index == 0:
                    temp_list.append(li_tag.find_all("a")[0].contents[0])
                else:
                    try:
                        temp_list.append(li_tag.contents[0])
                    except:
                        temp_list.append("")

            planets_data.append(temp_list)

        # Find all elements on the page and click to move to the next page
        browser.find_element(by=By.XPATH, value='//*[@id="primary_column"]/footer/div/div/div/nav/span[2]/a').click()
```

```
browser.find_element(by=By.XPATH,
value='//*[@id="primary_column"]/footer/div/div/div/nav/span[2]/a'
).click()
```

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?

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

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'**.

```
49  # Calling Method
50  scrape()
51
52  # Define Header
53  headers = ["name", "light_years_from_earth", "planet_mass", "stellar_magnitude", "discovery_date"]
54
55  # Define pandas DataFrame
56  planet_df_1 = pd.DataFrame(planets_data, columns=headers)
57
58  # Convert to CSV
59  planet_df_1.to_csv('scraped_data.csv',index=True, index_label="id")
60
```

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

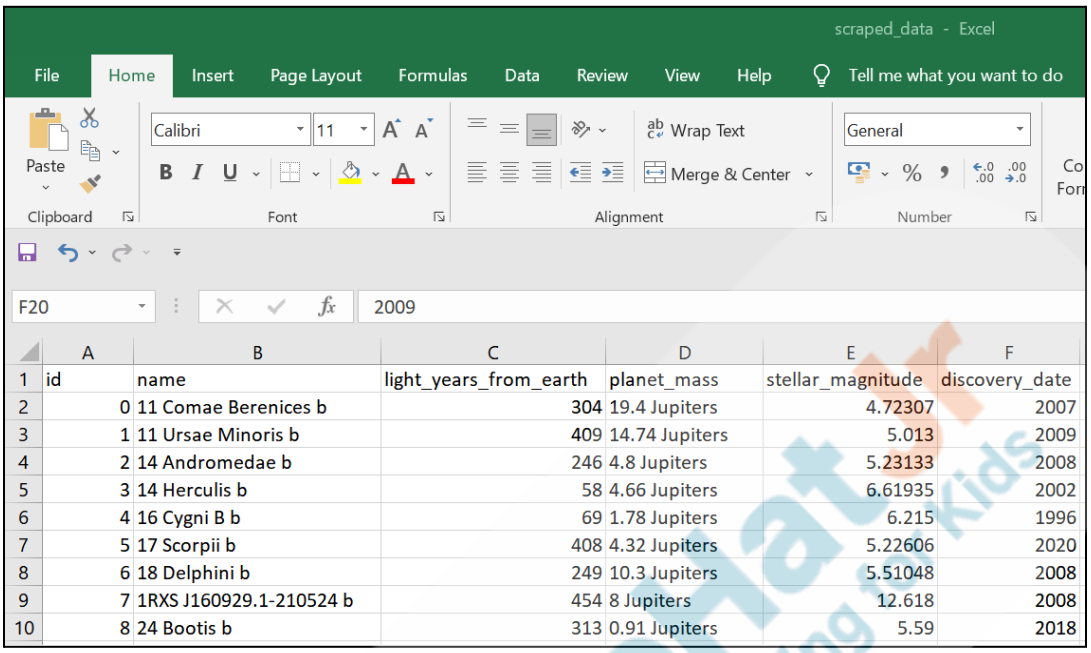
Note: *Guide the student to run the Python file using a virtual environment (scraper.py).*

```
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.

scraped_data.csv

```
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:* 

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-iWKsRJlGW9MEqaOi3_n4HAVqMRkE2Wxbhul4jQKM0/edit?usp=sharing
Teacher Activity 6	Reference code	https://github.com/procodingclass/PRO-C127-Reference-Code
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/procodingclass/PRO-C127-Student-Boilerplate-Code