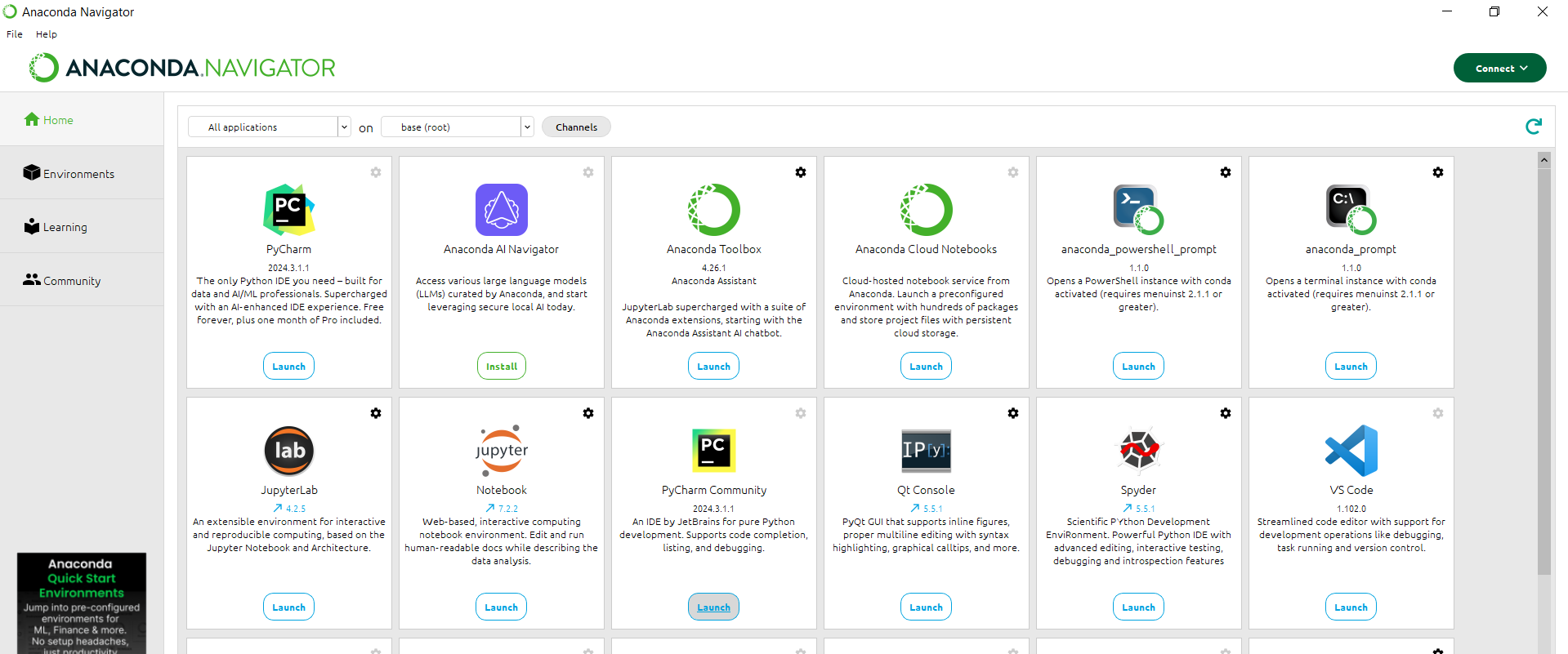
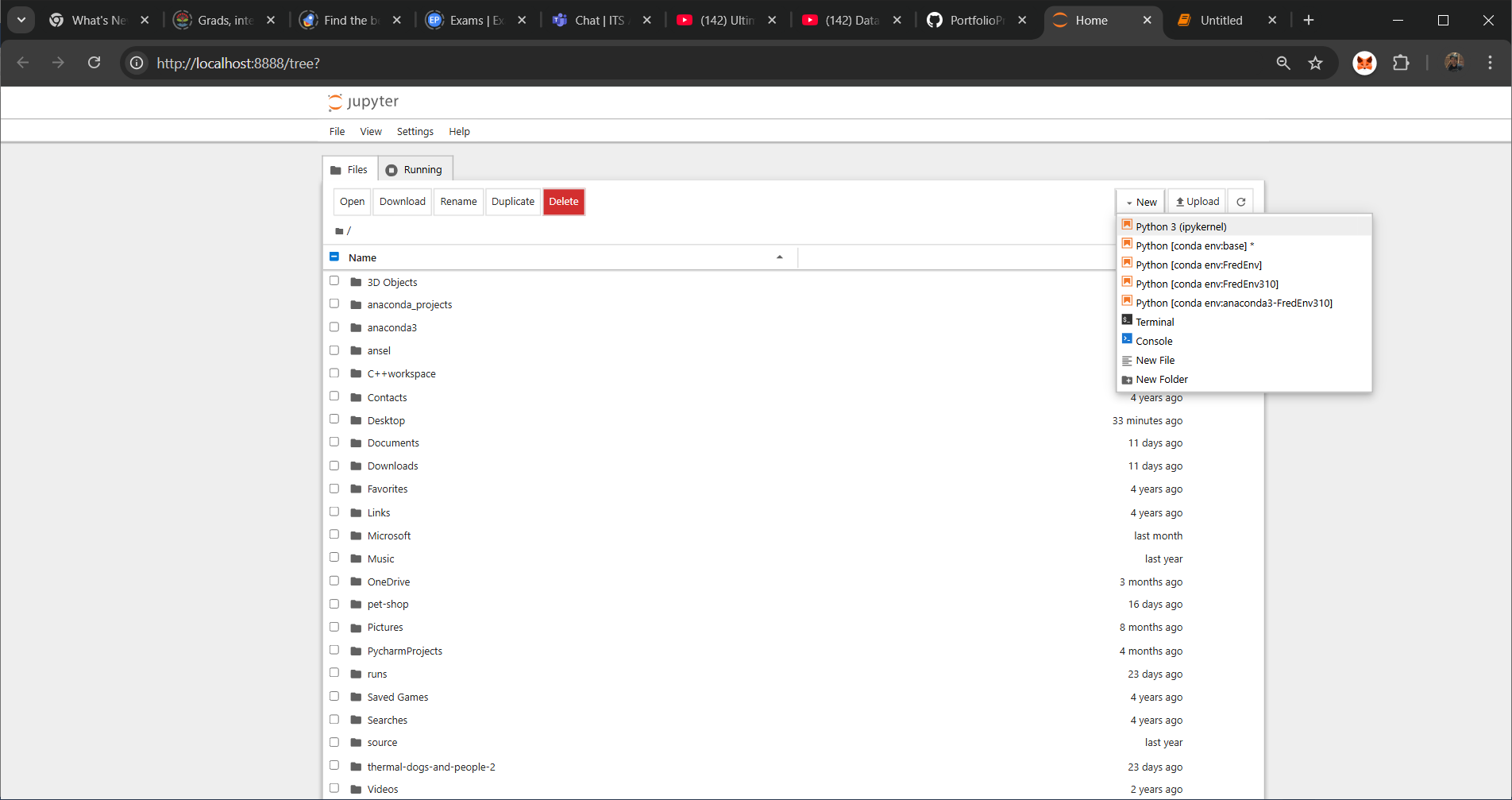
IDE

Install conda then in conda navigator click ‘launch’ on ‘jupyter notebook’

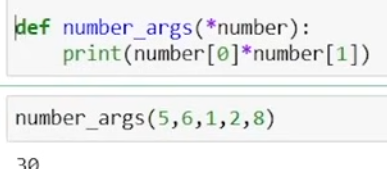


A new server will open. Click ‘New” then “python 3 kernel”. A new notebook will open

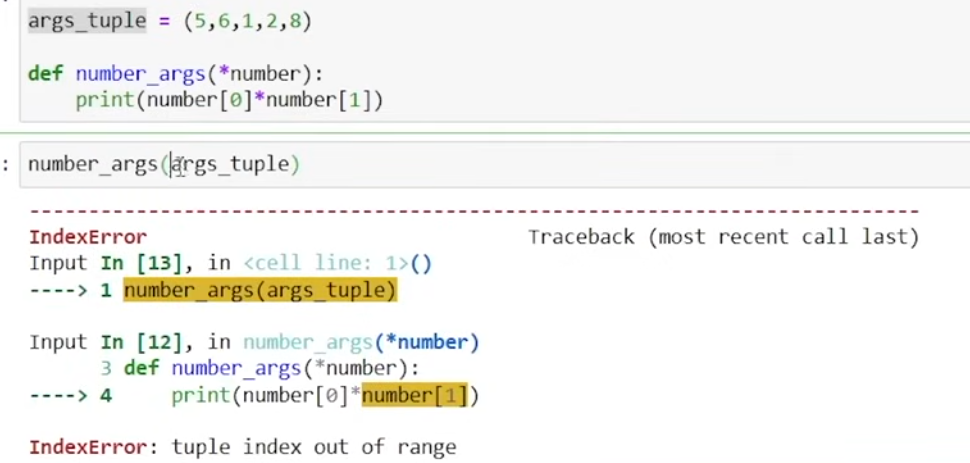


BASIC Python

1. Simple Args



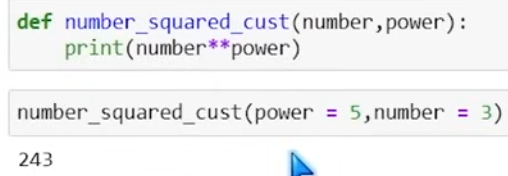
2. Error with args



Corrected error



4. Basic kwarg- is an arbitrary keyword argument. Arbitrary means we don’t know how many arguments will be passed

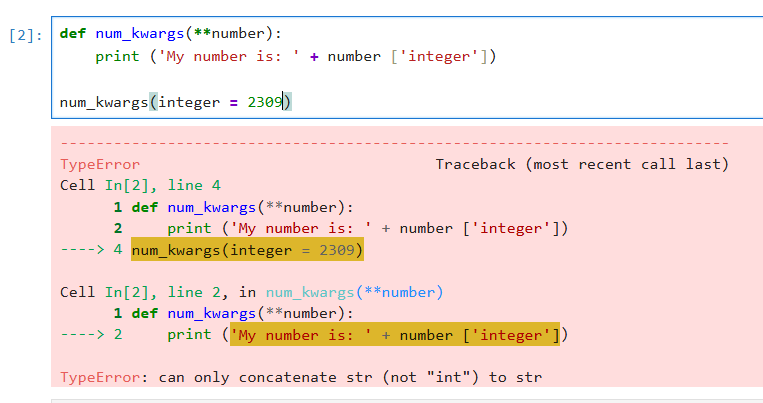


**Explanation of error- you cannot concatenate a string and integer**

**Explanation:**

**\*\*number:**

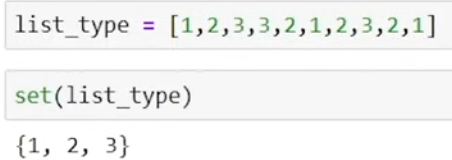
* This syntax means the function accepts **arbitrary keyword arguments**.
* All keyword arguments passed to the function will be collected into a **dictionary** called number.

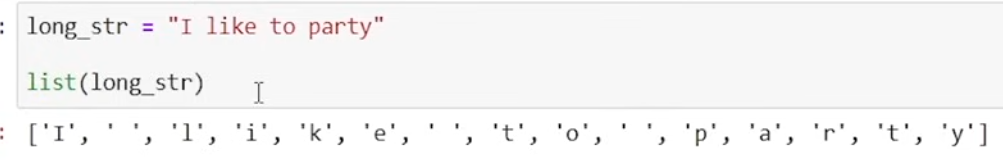


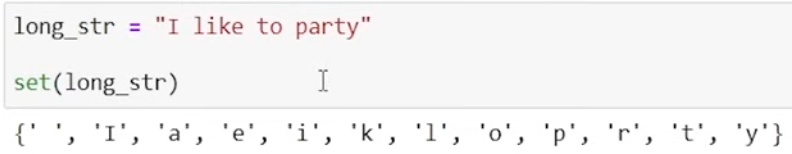
**Corrected error**



**5. Converting data type- int(), str(), set(), list(),float() etc.**







**6. BASIC Dict methods**



**BASIC BMI Calculator**

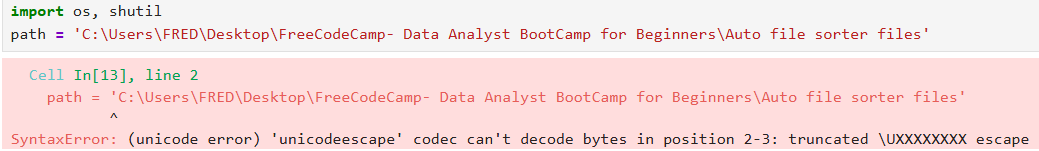


**AUTOMATIC FILE SORTER**

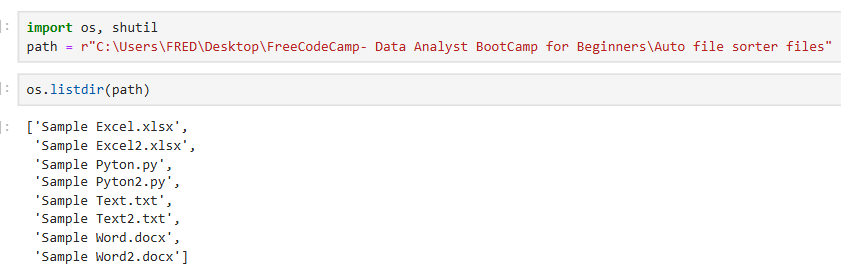
* Depending on the file extension (.csv , .pdf, .docx etc.) the files will automatically be sorted and put into a folder

1. “r” in path will read the path as raw string meaning it will not read on the backslashes

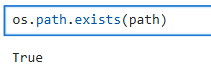


Without ‘r’

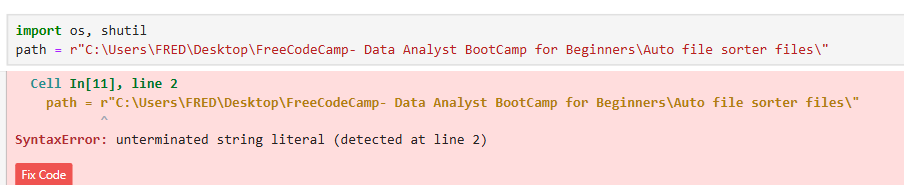
2.. “.listdir” method – shows all files in the directory



3. Checking if path exist



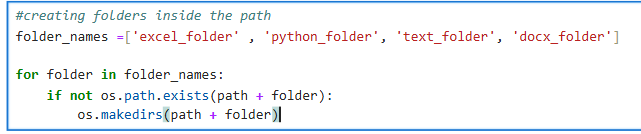
4. Adding backslash makes it available for the path to be used to add a folder in it but this gives error. WHY?



This is because it needs a forward slash



5. Creating new directory- because there is already a forward slash in the path, you can just concatenate the path with the folder name



NOTE: if you are using backslash in your path, there will be no error but also there will be no folder created

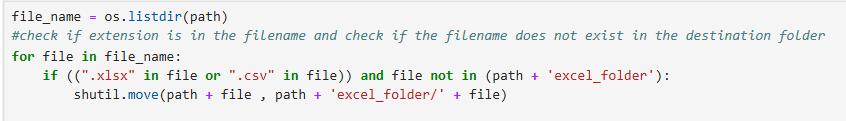


5. Moving file to the folder

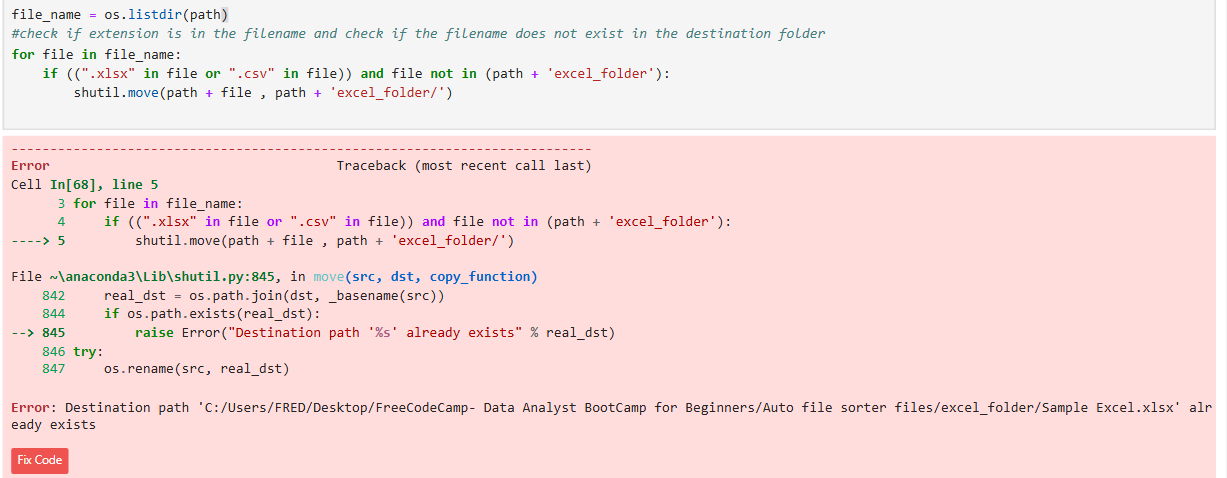
NOTE: “in” used in the “if statement” is comparing a string to another string. It is not checking if the filename is inside the folder. So this line below, the second condition is always FALSE because the file is always in the path (string type)



A. TRANSFER AND OVERWRITE: This code was able to transfer all the file with the extension BUT it overwrites any file with the same name. This is because the 2nd “If” condition is always TRUE. “not in” only check if the file (string) is in the path (string)

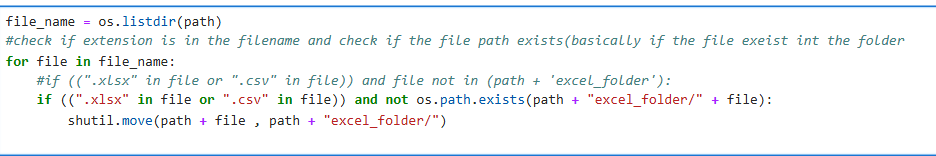


B. TRANSFER ONLY AND GIVE ERROR FOR DUPLICATE TRANSFER: In this code, all files was able to transfer to the folder, however when there is a new file to transfer with the same name the “shutil.move….” code will give error, this is because the second parameter of the method move(), actually checks if the source(path +file) is in the destination.



C. TRANSFER FILES AND LEAVE DUPLICATE

To rectify the problem in all the code above, it is better to use the methods from os library. Using “os.path.exists” it will check if the path already exists and returns TRUE or FALSE. While using “in” only checks if the strings exists in a lists of strings, returns TRUE or FALSE also

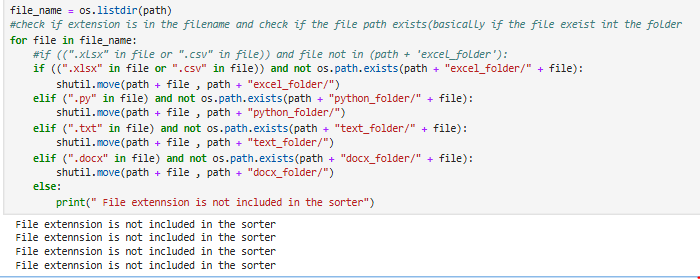




NOTE: the “os.path.exists” checks for location. It does not check if a filename is in the path but rather it checks the actual path exists. So the second condition always TRUE and it is reversed by the keyword ‘not”



This is the whole sorter file code



Alex code has additional “file” in the move method which also works but maybe redundant



WEB SCRAPING

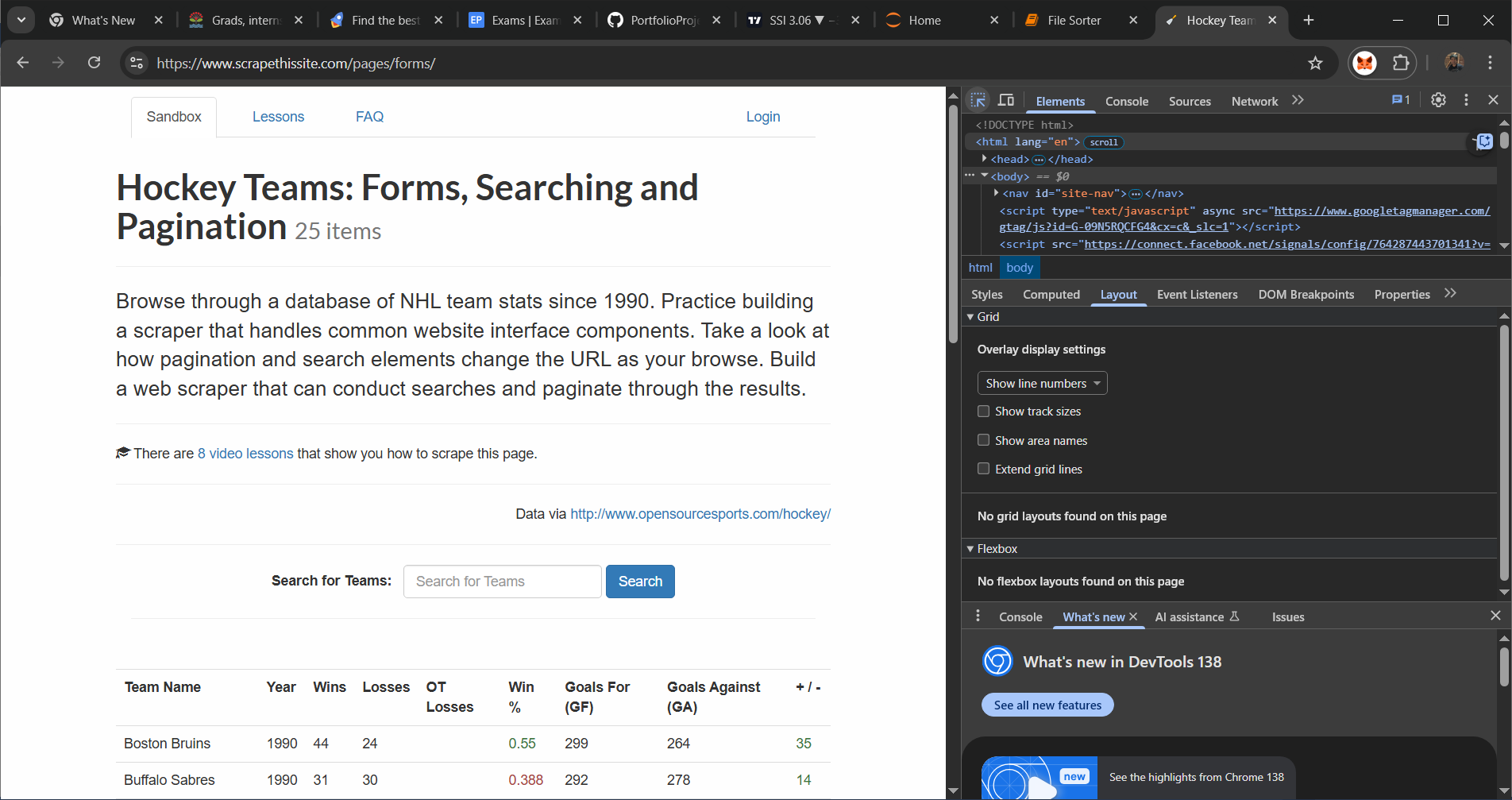
Basic webpage structure

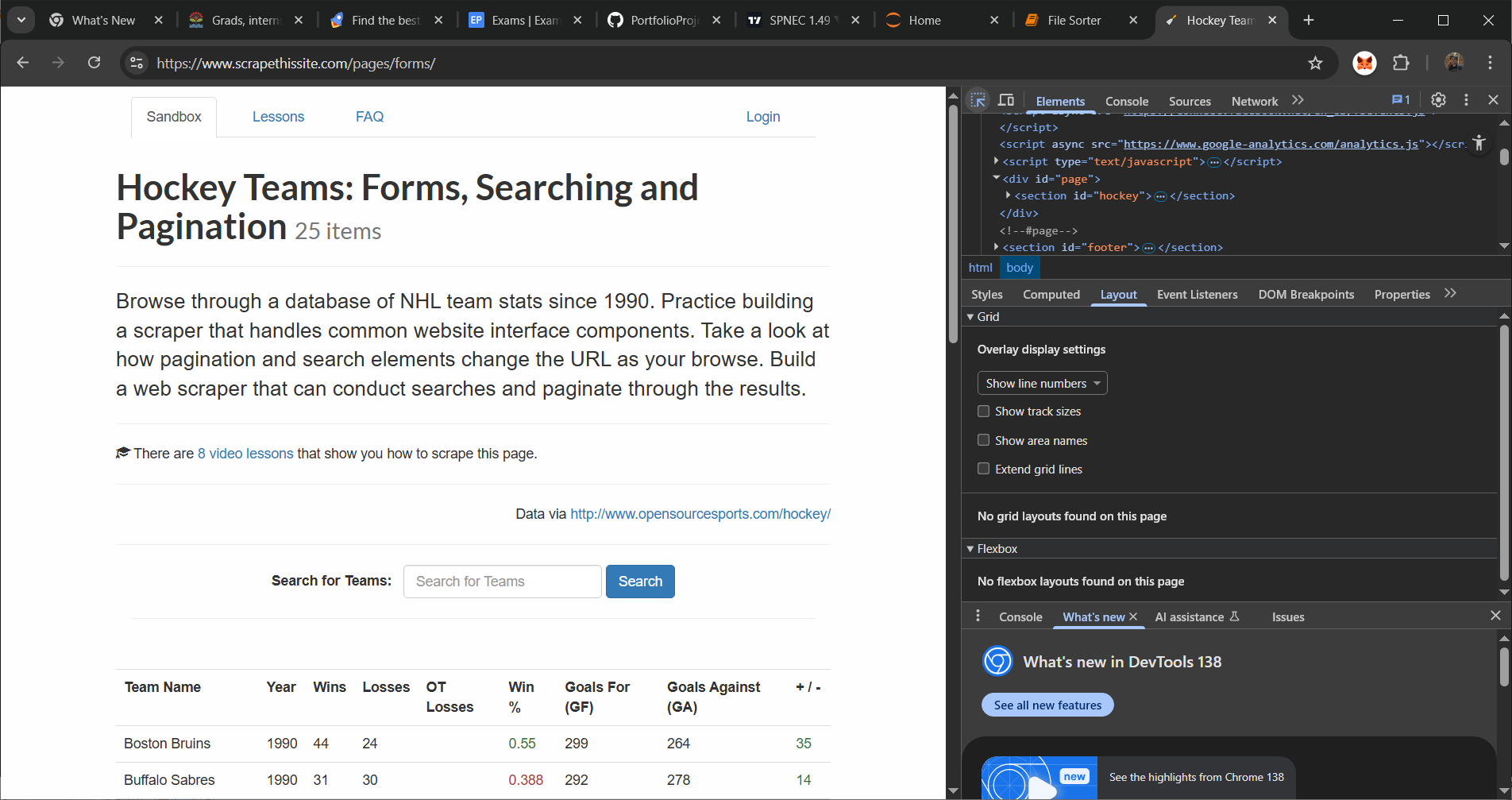
<p></p> tags denotes paragaphs



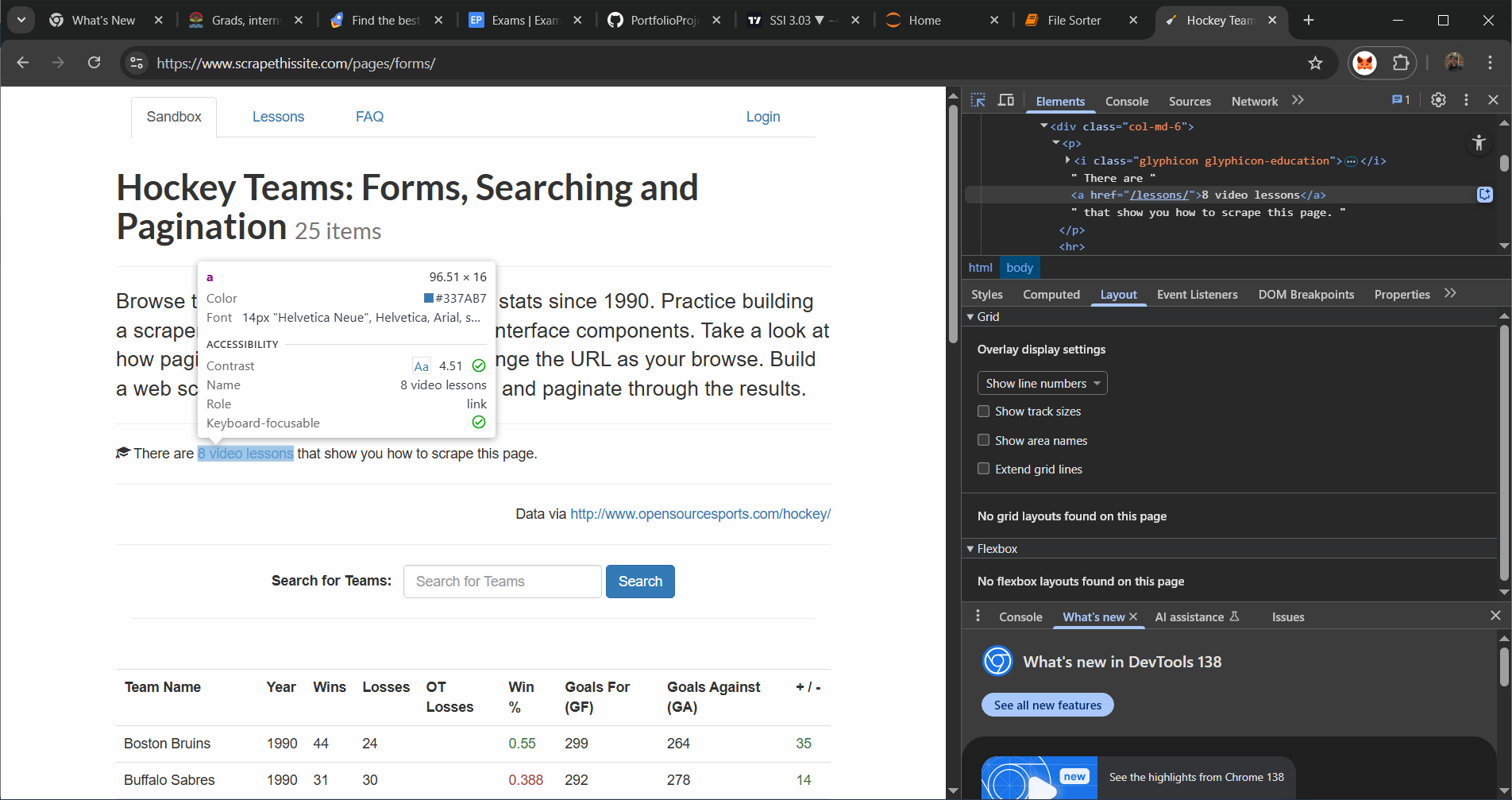
INSPECTING WEB PAGE

Ctrl +shift + I = open the inspection window

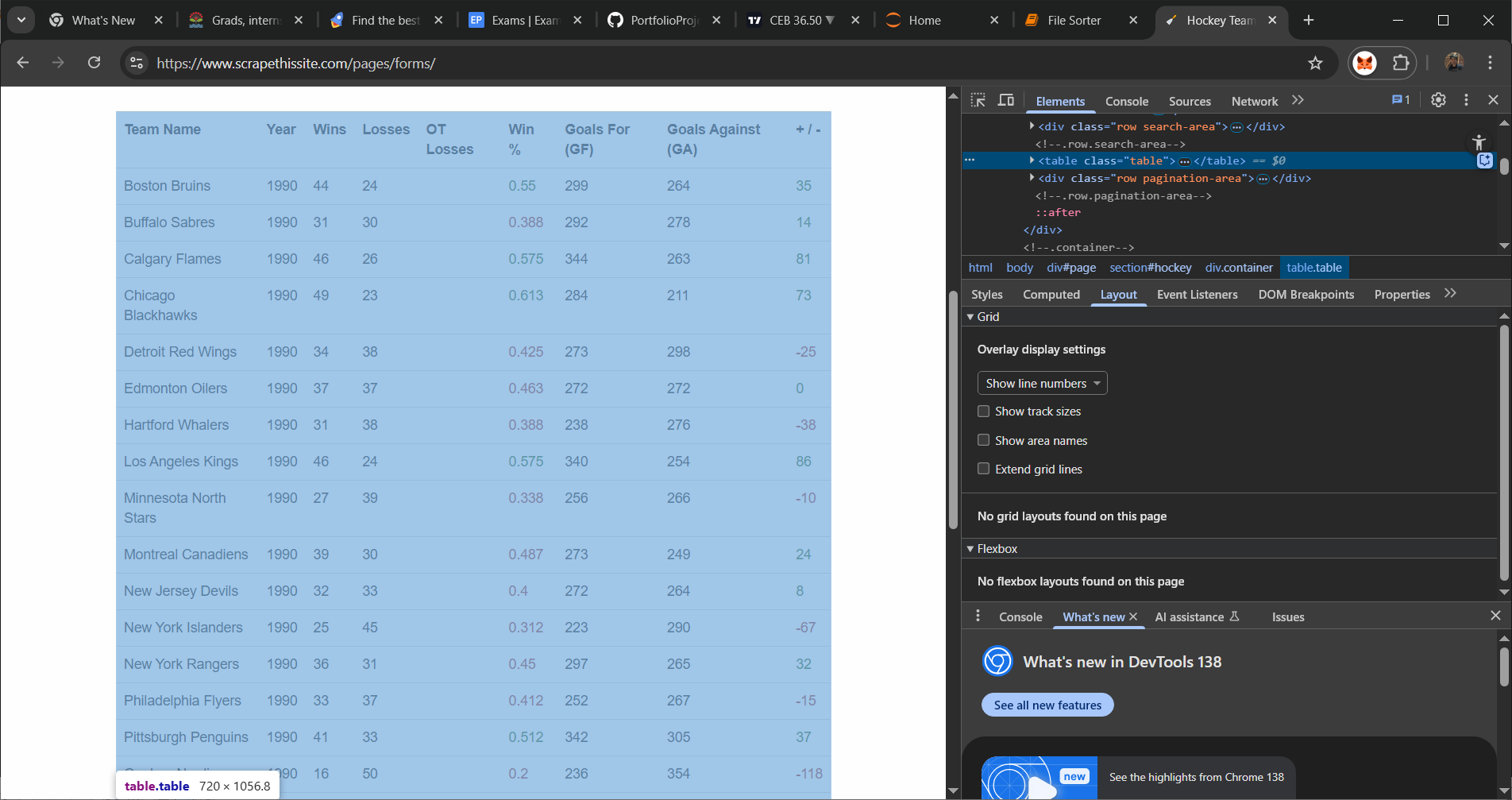
 this button or ctr +shift + C = select an element in the page to inspect it.



href tags are hyperlinks



<table></table> tags are the tables in the web page

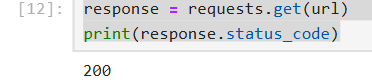


BeautifulSoup and Request

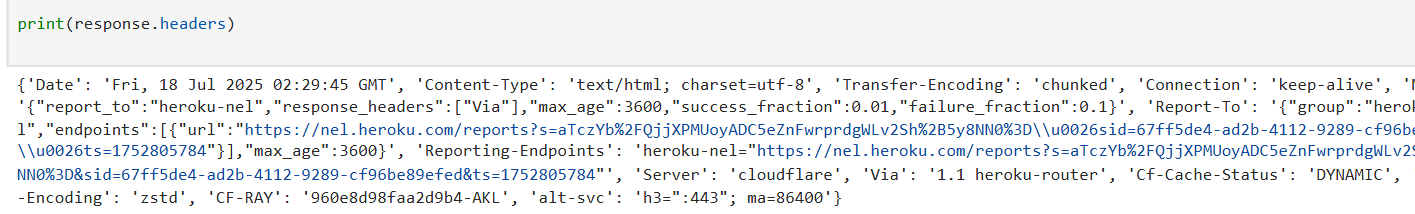
A. Getting response from web

1. To perform web scraping, you should import certain libraries. The urllib. request module is used to open URLs. The BeautifulSoup package is used to extract data from html files. The BeautifulSoup library's name is bs4 , which stands for BeautifulSoup, version 4.

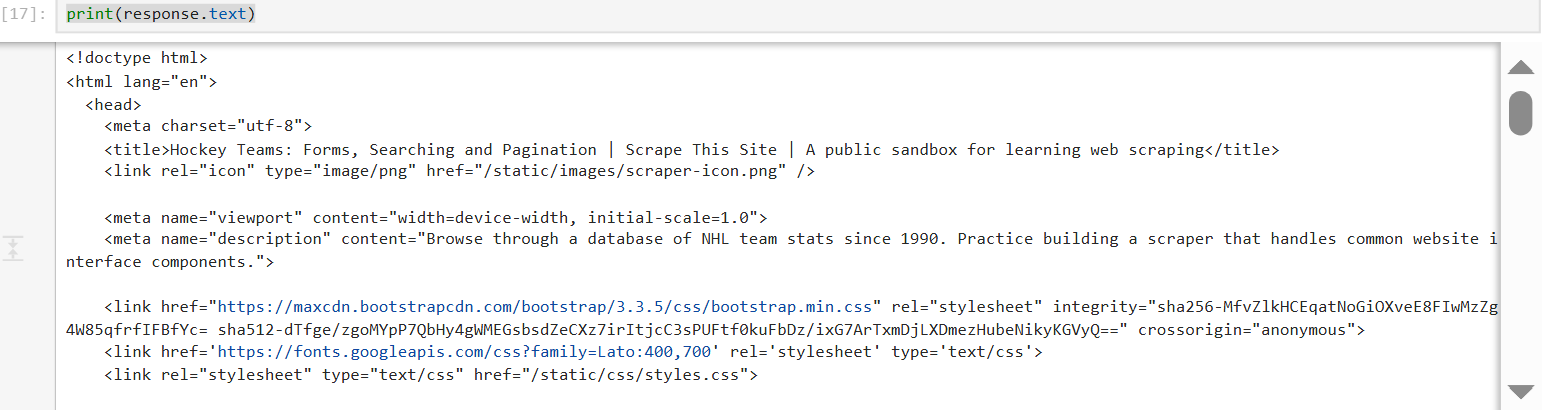
2. response.status\_code: This attribute provides the HTTP status code returned by the server, such as 200 for success, 404 for not found, or 500 for a server error.

 or 

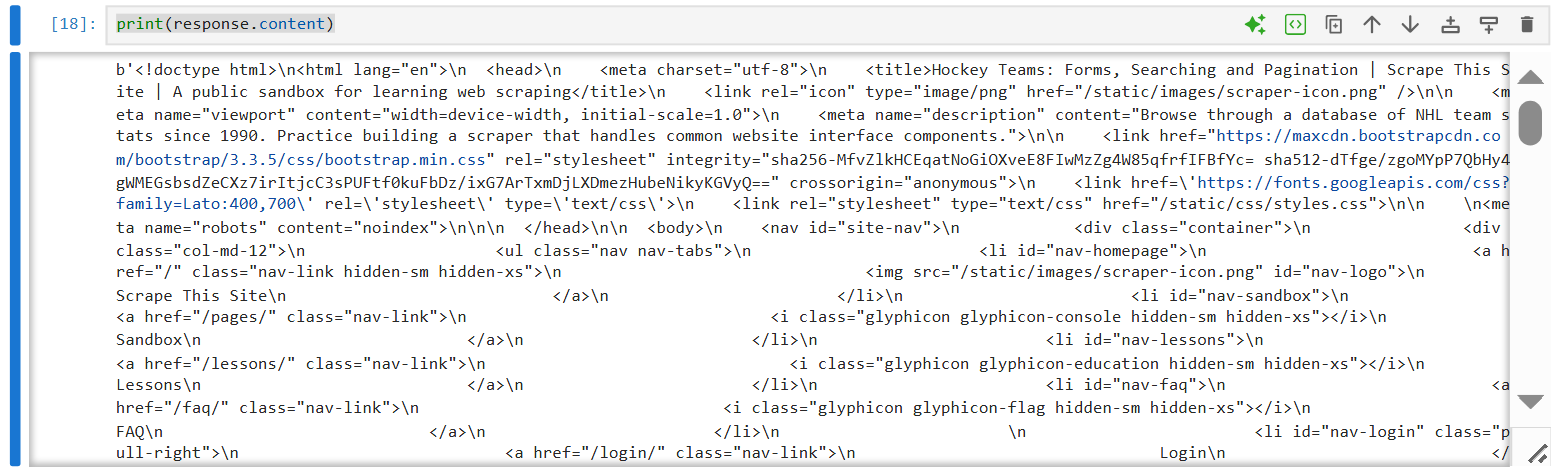
3. response.headers: This attribute provides a dictionary-like object containing the HTTP response headers sent by the server. You can access individual headers like 'Content-Type' or 'Server'.



4. response.text: This attribute provides the content of the response body as a Unicode string. It is useful for retrieving HTML content from webpages or plain text responses.



5. response.content: This attribute provides the content of the response body as raw bytes. This is useful when dealing with binary data like images or when precise encoding control is needed.

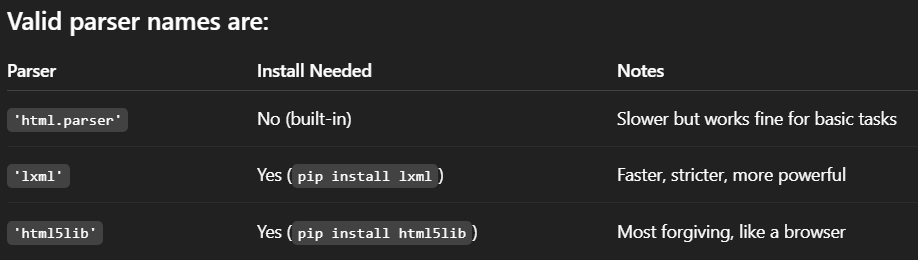


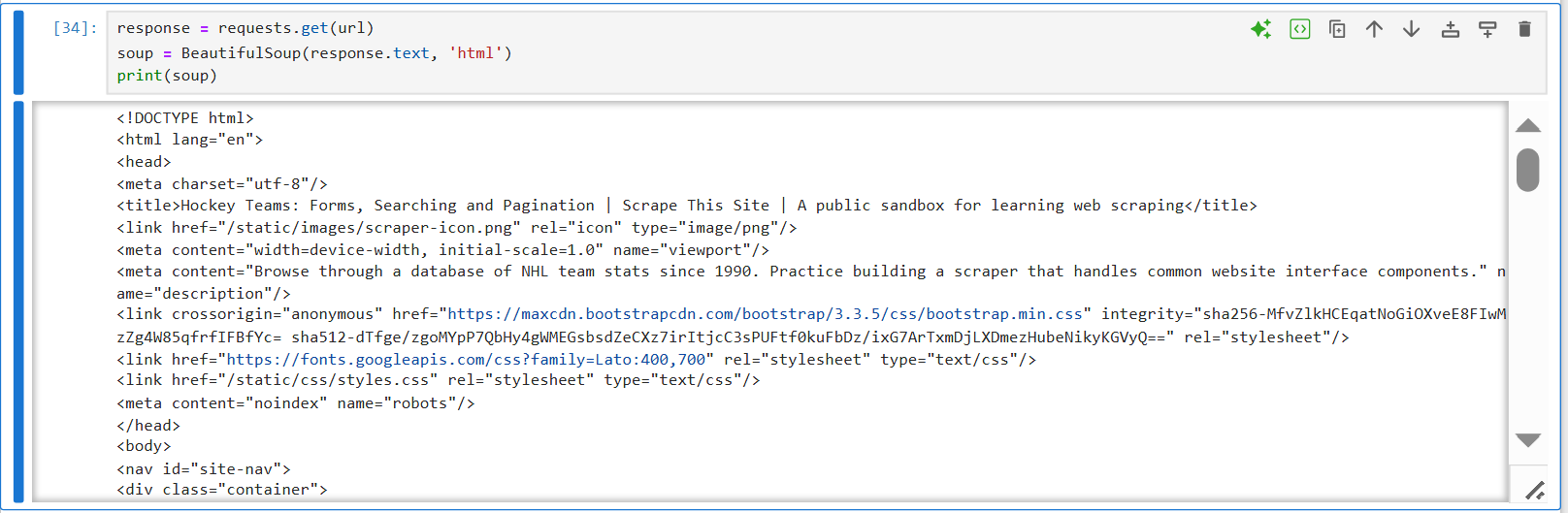
6. response.raise\_for\_status(): This method checks if the request was successful (i.e., status code is in the 2xx range). If not, it raises an HTTPError exception. This is a convenient way to handle potential errors in your code.



B. Using bs4 to parse html information

This takes the raw HTML and turns it into a **BeautifulSoup object**, which is easier to search and navigate using Python.

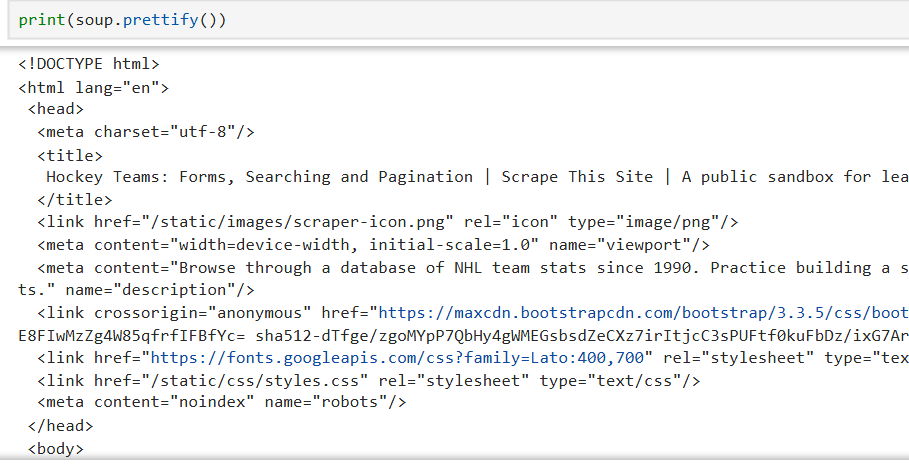




Note: BeautifulSoup needs to know which parser to use, and 'html' isn't a valid parser name. The string you pass is the parser’s name, not a content type. If you passed 'html' to BeautifulSoup and didn’t get an error, there are a couple of possibilities. it might interpret 'html' as 'html.parser' if nothing else matches. It shouldn't, strictly speaking, but depending on your environment or version, it may be silently forgiving.

That said, this behavior isn't guaranteed and may not work consistently across systems or versions.

prettify() method arranges and gives indention hierarchy

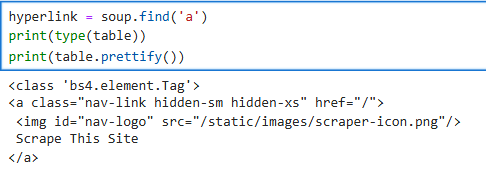


C. find and find\_all- bs4 method that find specific tags

find () – only finds the FIRST occurrence of the tag. Returns single tag

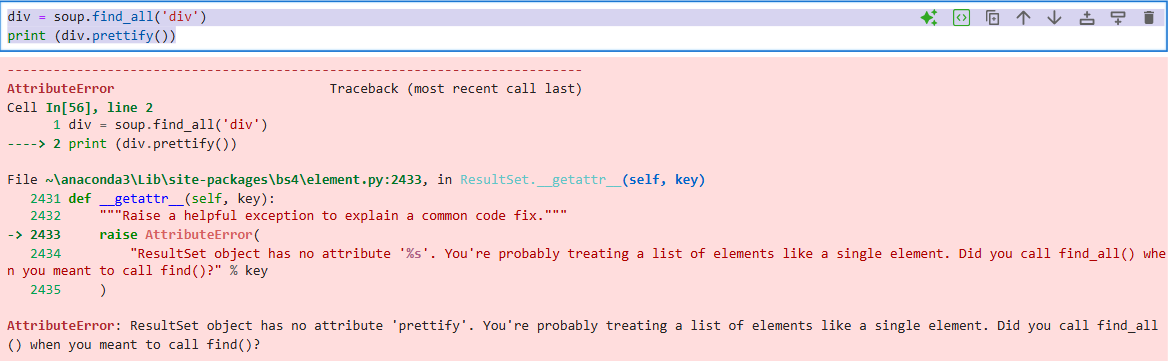
find\_all()- returns all occurrence of the tag. Returns a python list

1. Finding href using find. Remember the data type

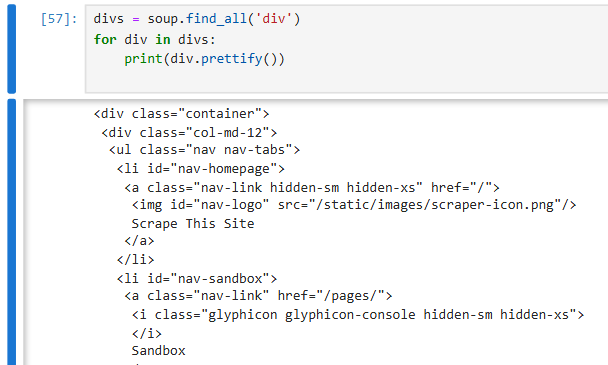


2. Finding all divs

Note : soup.find\_all('div') returns a list of all <div> elements. But prettify() is a method for a BeautifulSoup Tag object, not a Python list. So you're trying to call .prettify() on a list, which raises an AttributeError.

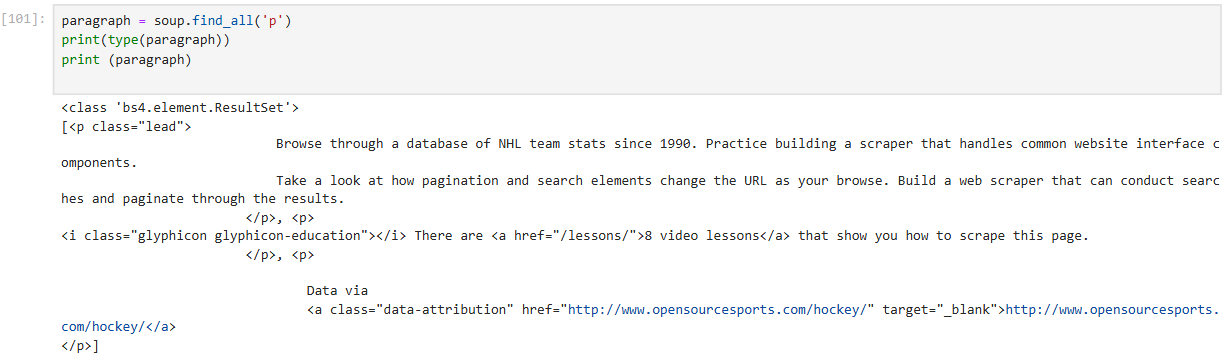


Solution: If you want to prettify **all** divs, you need to loop through them or select one:



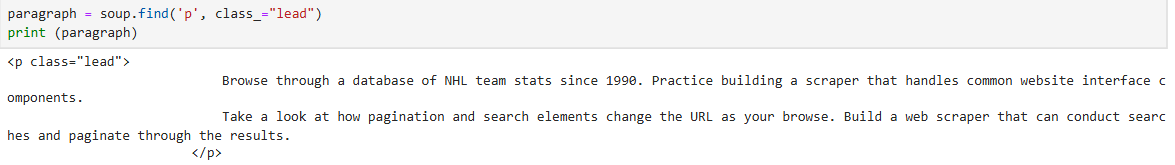
D. scraping specific data

1. display all the tags available on the html. Remember the data type of find\_all

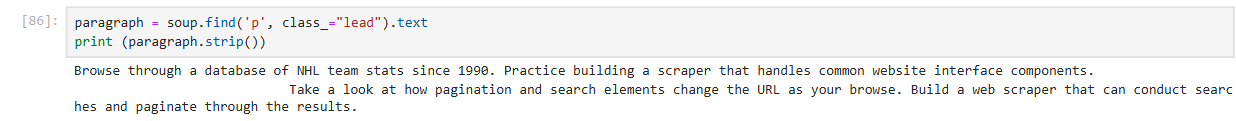


2. or if you have inspected the site you can use the 2nd parameter class directly

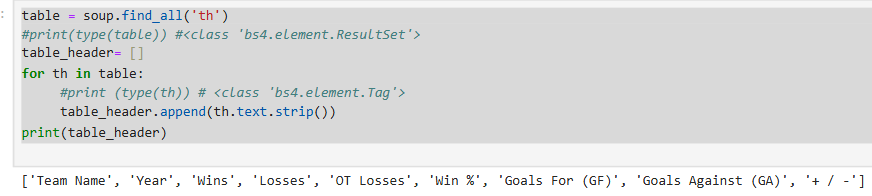
NOTE: The reason the class attribute is written with an underscore (as class\_) in BeautifulSoup's .find() or .find\_all() methods is due to a naming conflict with Python's reserved keyword class. Because of that, you can't use class as a parameter name in functions. So, BeautifulSoup uses class\_ (with an underscore) to avoid this conflict.



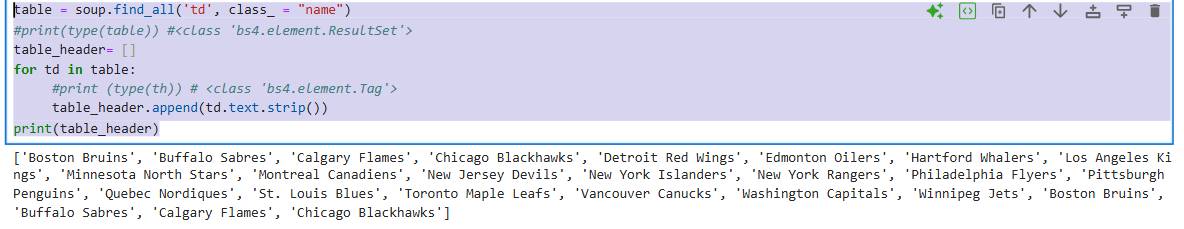
3. convert the html to text and strip the spaces. You can now perform string manipulations on this part



4. Extracting all headers



5. Extracting all the column values (on the current page only)

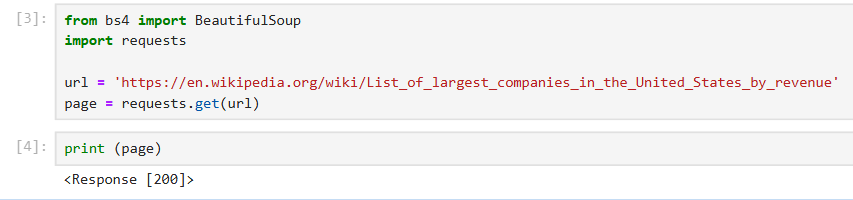


6. more compact code



Extracting data using Pandas

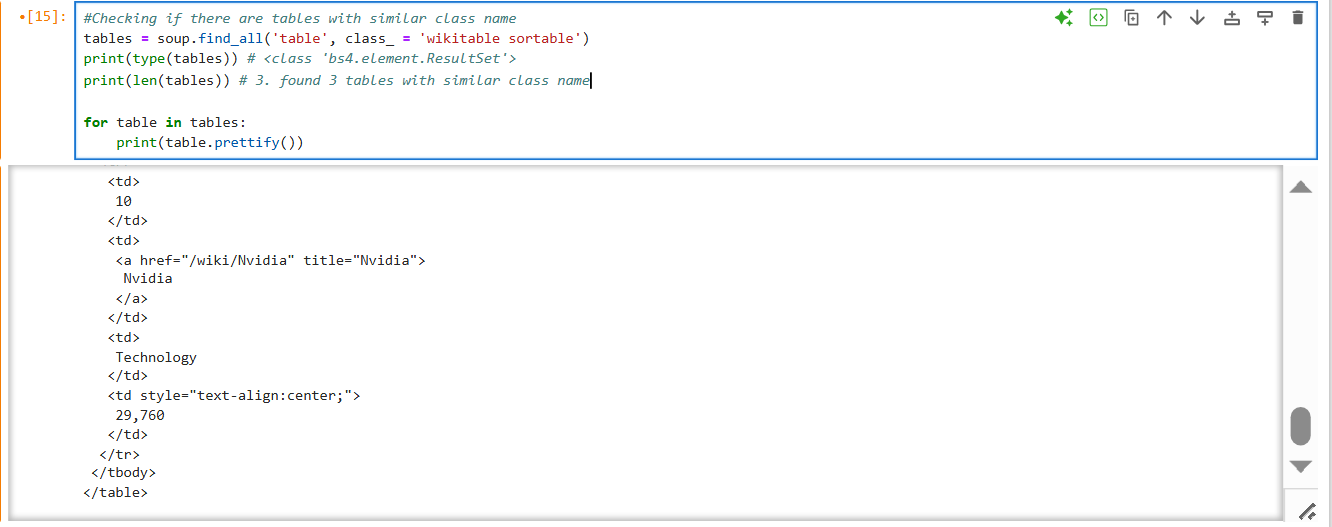
1. Checking if website has good response



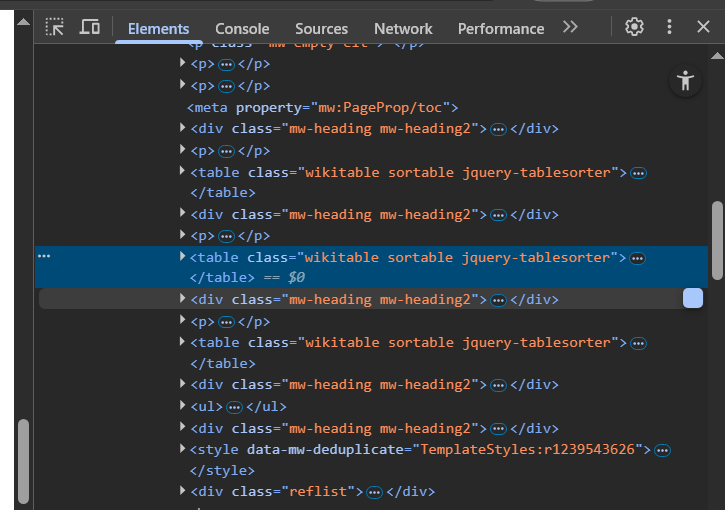
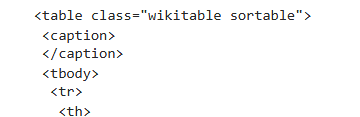
2. Upon inspecting the site, you found that there are 3 tables that have the same HTML class name. You can easily check by using len() function that there a3 table with class name ‘wikitable sortable’

<https://en.wikipedia.org/wiki/List_of_largest_companies_in_the_United_States_by_revenue>

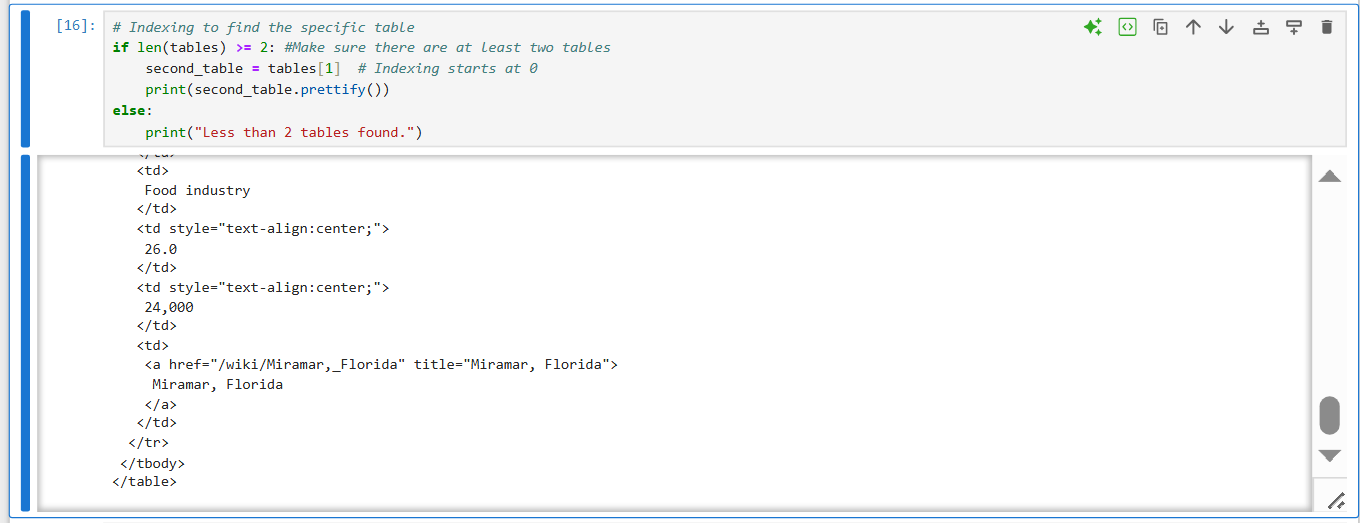
You can confirm the last table by iterating to all the tables with the same class name, confirm that the last title is “Nvidia” which is also the last item ion the last table



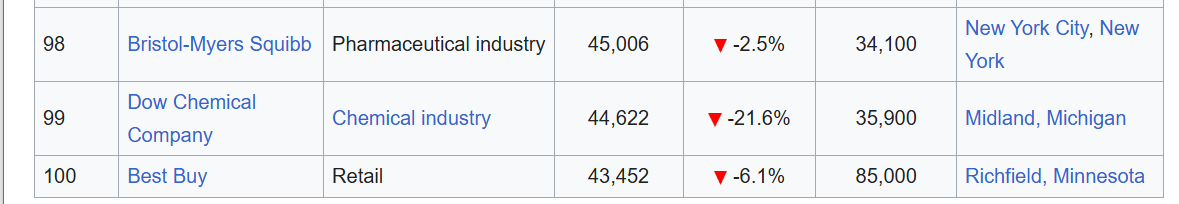
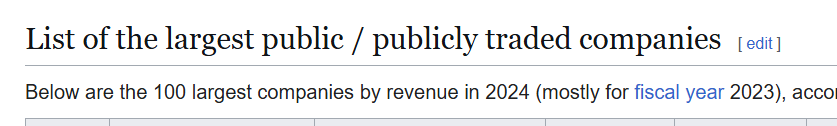
BELOW: Inspection in the site reveals the actual class name of the table is different from the table class name parsed by BeautifulSoup

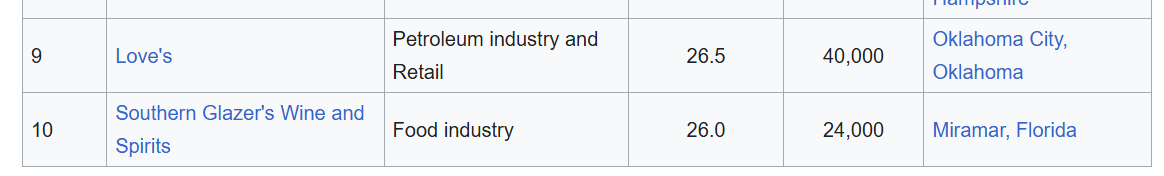
 

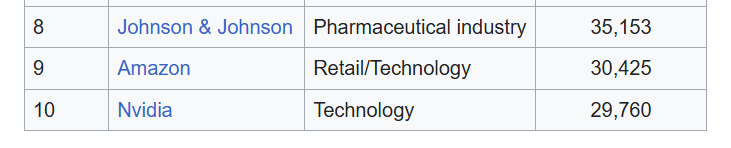
4. To select the specific table, indexing can done BECAUSE FINDALL () returns A PYTHON LIST



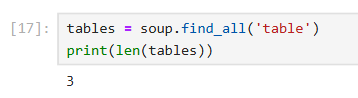
Check the end of the html result like ‘title’ and compare it to the last item of the table in the site

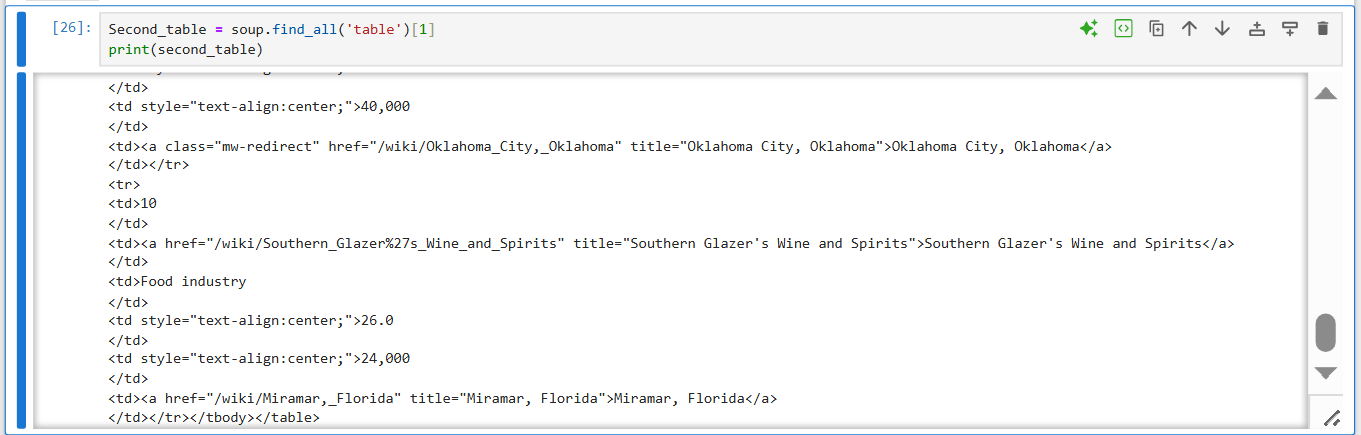






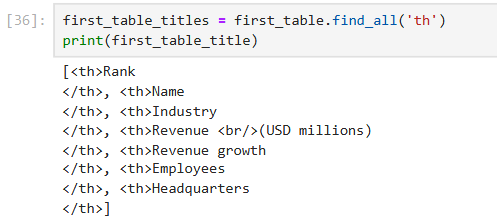
5. You can do indexing like this. Even though there is no class name, you can see that the len() function returns 3 tables. You can just directly index query to see what table will appear.



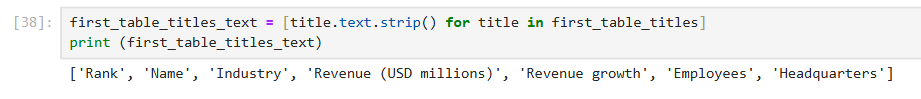




6. Save only the header of the table in the variable

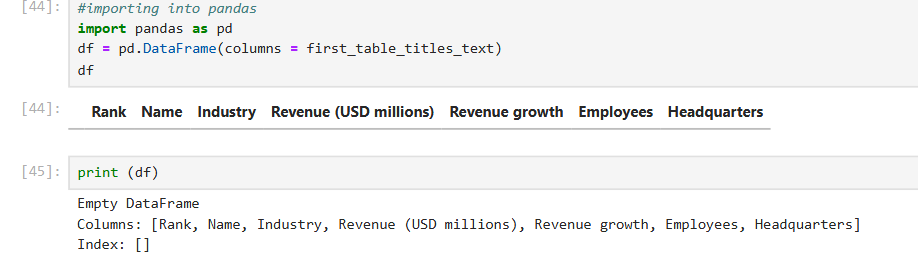


7. Convert the bs4 object into text, strip the spaces in front and back



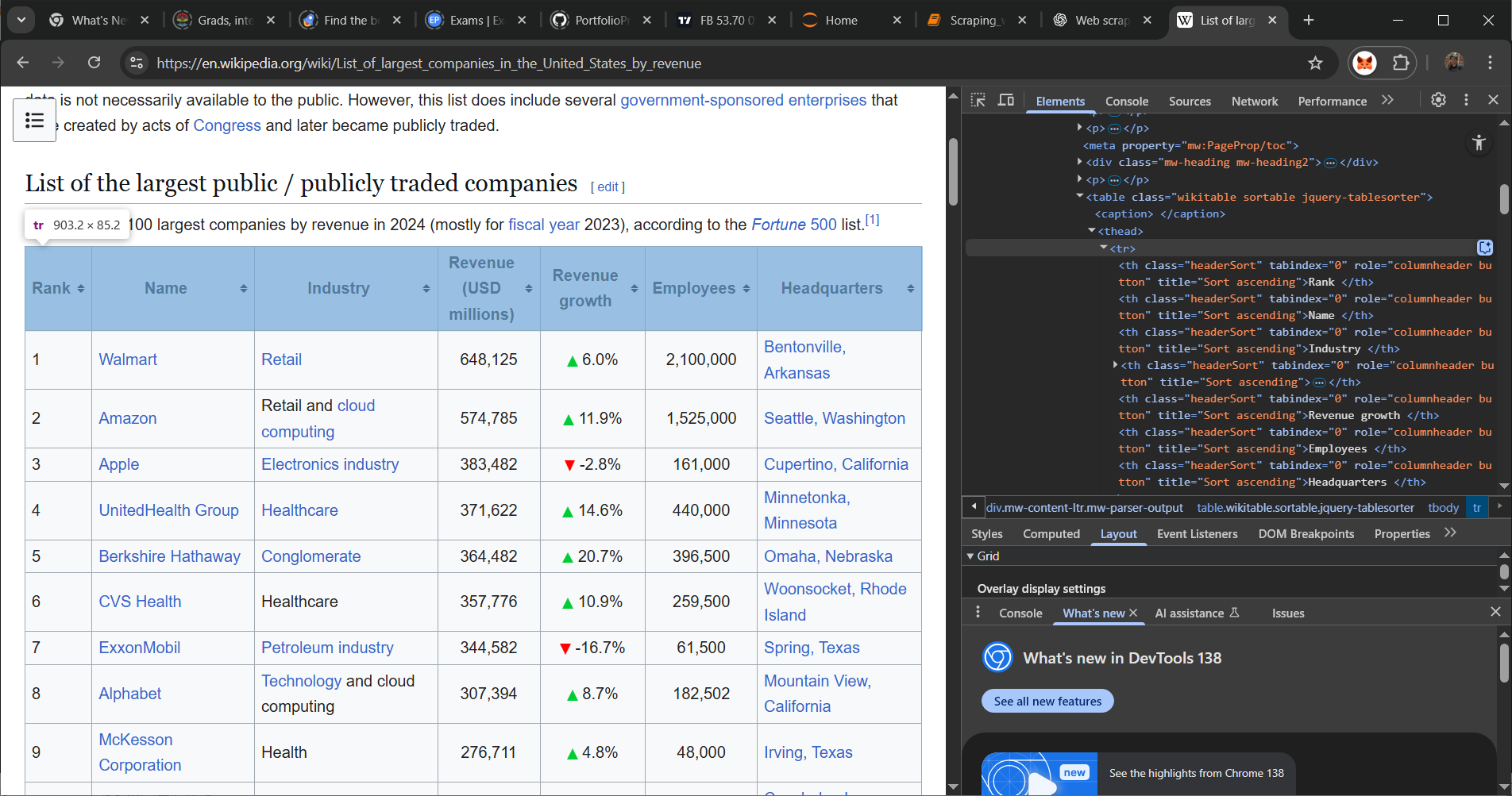
INSERTING INTO PANDAS

1. importing the variable containing the stripped headers into pandas

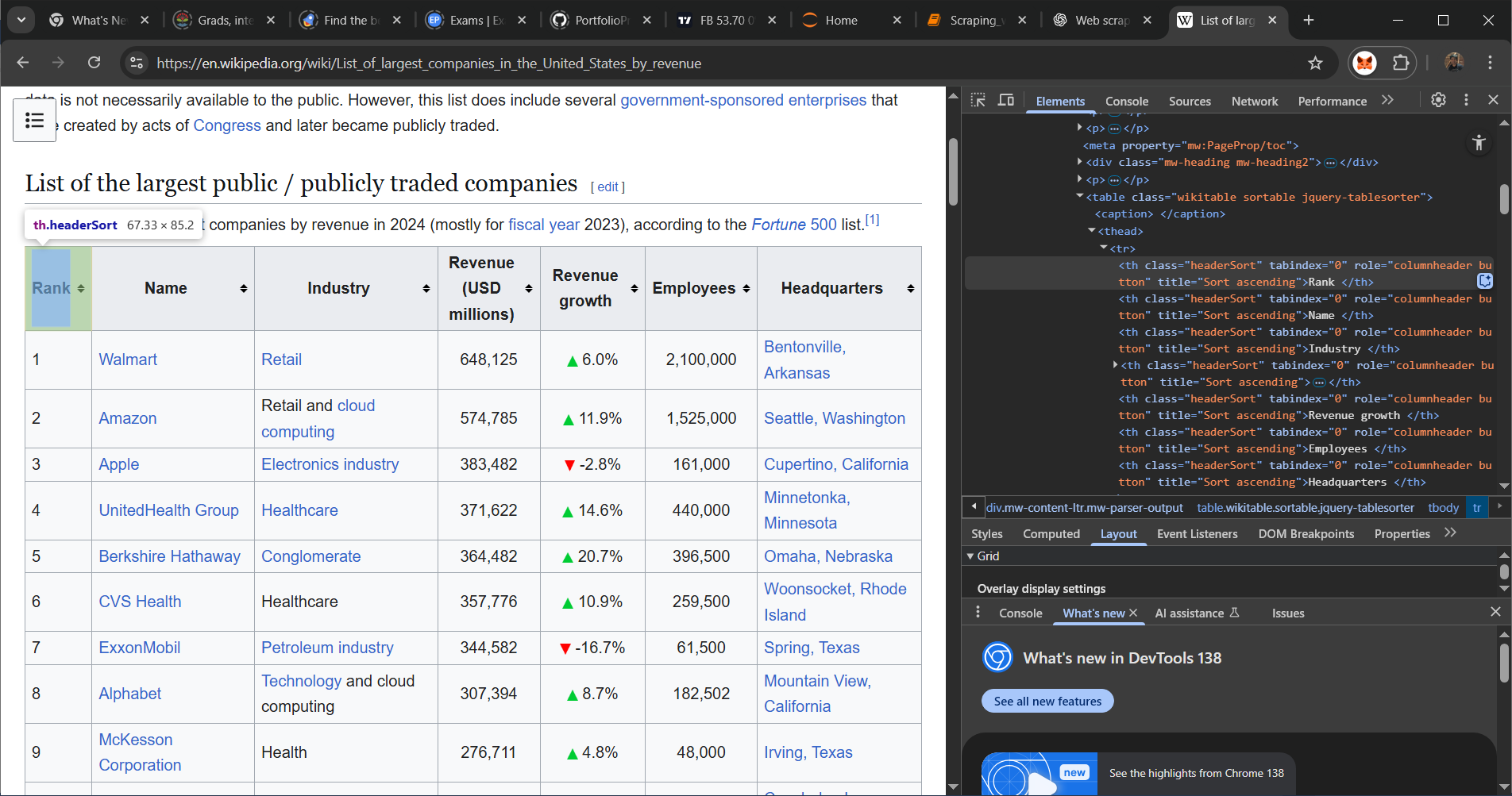


2. Inspecting the rows of each data

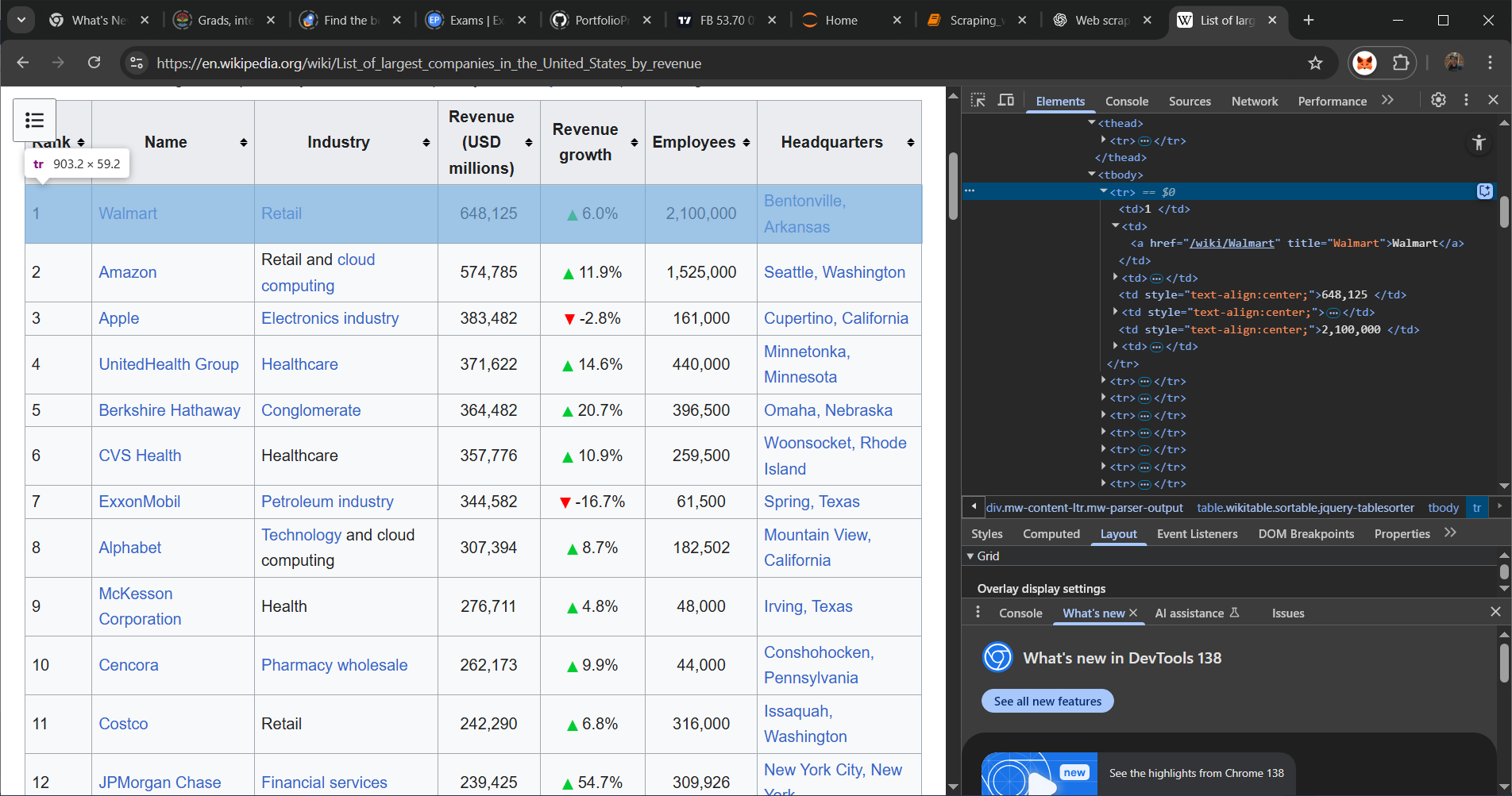
‘tr’ represents each rows in a table



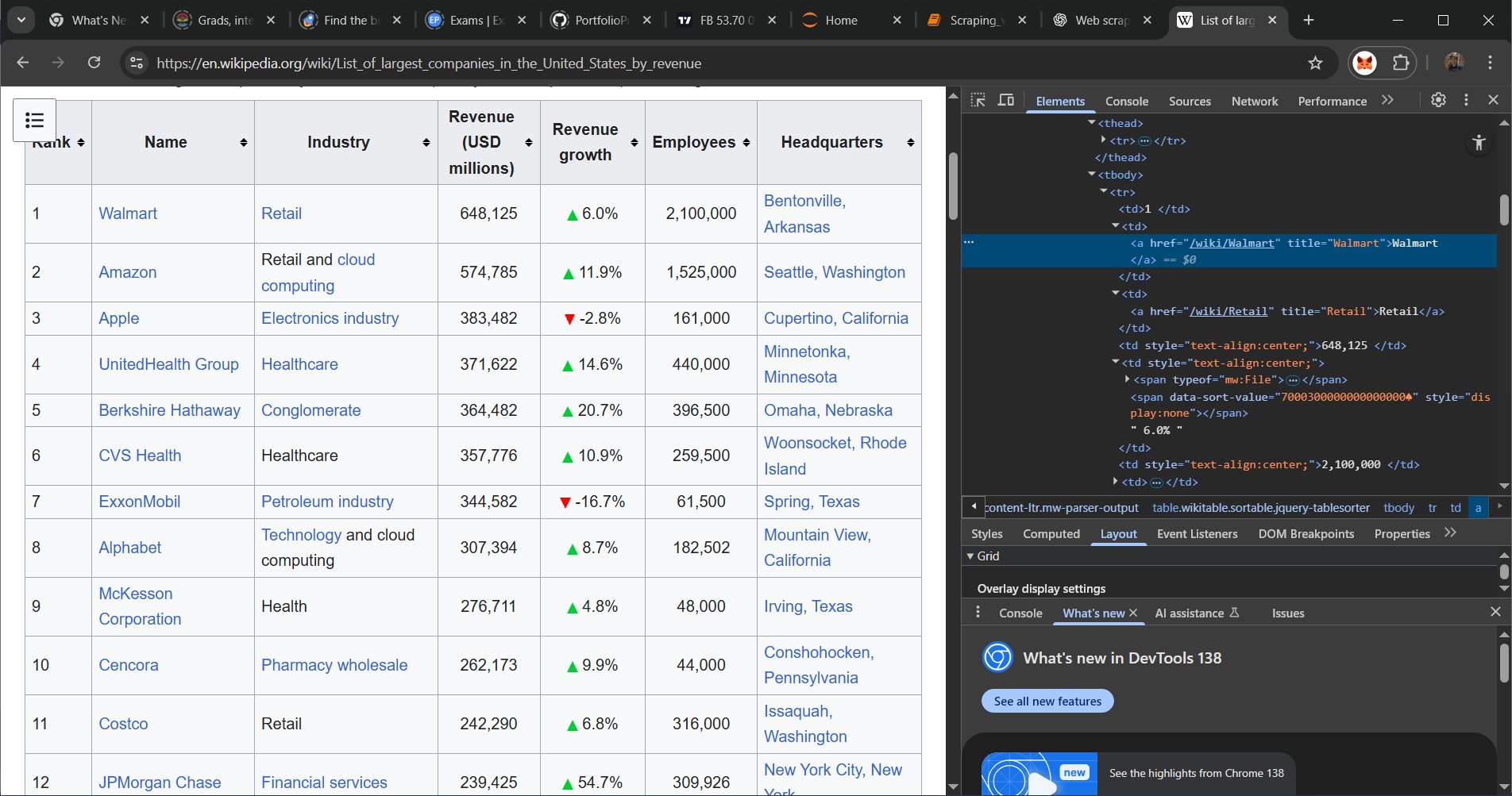
‘th’ (is under ‘tr’) contains header row data



Another tr under body represents each row



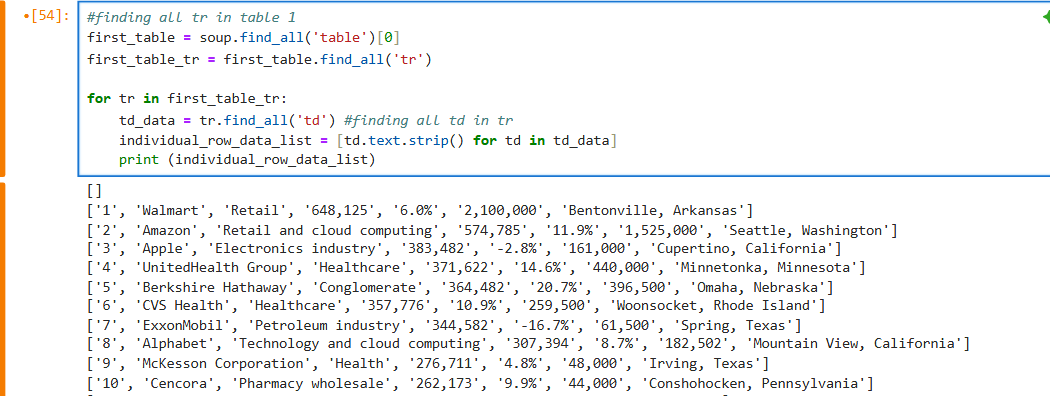
‘td’ (is also under ‘tr’) the contains each row of the table body



3. Now that you know the hierarchy of each container in HTML, you can now start looping through each container to get the data

As you can see below, the second line of code saves all the ‘tr’ in a variable object. And each ‘tr’ is loop through to get all the ‘td’ in it if it has <td></td> in it. The ‘td\_data’ is now an object that contains all ‘td’ in only one row (‘tr’). Then ‘td\_data’ is being loop through again to get individual data, convert it to text, strip of all space and save in a list

The first ‘tr’ was the header that contains <th></th> tags so the list is empty



4. To be able to insert each row in the df we need to loop it and each row is inserted using the ‘loc’ function

loc in a Pandas DataFrame is a label-based data selecting method used to access a group of rows and columns by their labels (index labels and column names) or by a boolean array.

Key characteristics of loc:

1. Label-based indexing:

It uses the actual labels of the index and columns for selection, rather than integer positions.

2. Inclusive slicing:

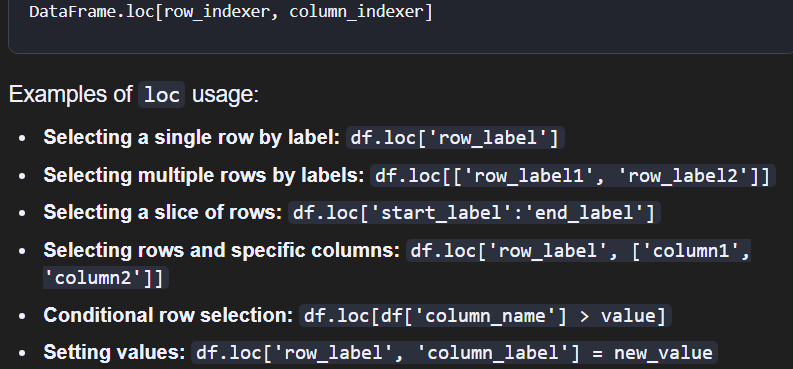
When using slicing (e.g., df.loc[start\_label:end\_label]), both the start\_label and end\_label are included in the selection, unlike standard Python slicing which excludes the end element.

3. Boolean indexing:

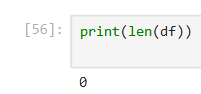
It can accept a boolean array for selecting rows based on conditions. This is a powerful feature for filtering data.

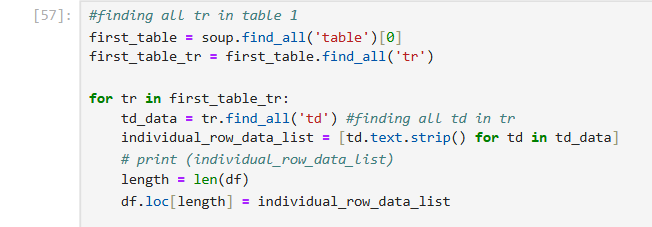
4. Setting values:

loc can also be used to set values for specific cells, rows, or columns based on their labels.



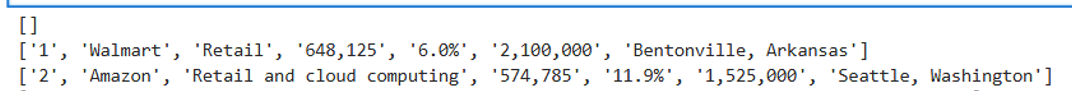
We will use loc to search for the length of rows in the table and use that as the index so that each added row will change the row\_indexer of ‘loc’





Error: 

This is because of the first empty row



To correct that, we will need to use LIST INDEXING



5. Do not put ‘print()’, just use ‘df’ while in jupyter notebook and it will display like below



6. Exporting to csv. Remember to make the ‘index’ parameter ‘False’ so that the index 0-100 will not be exported. Also notice that the name of the file and extension is included in the last section of the path

