ck your progress on the go! Install App

Back

Congrats Daniel! This project has been marked as completed.

Project Rating

★★★★☆

Teacher's Comment

"Good"

-

Was this helpful?

PRO-C127: WEB DATA EXTRACTION - 1 Completed

In Class 127, You Learned About Web Scraping And You Wrote A Scraper To Scrape The Information Of Different Planets From The Nasa Site.

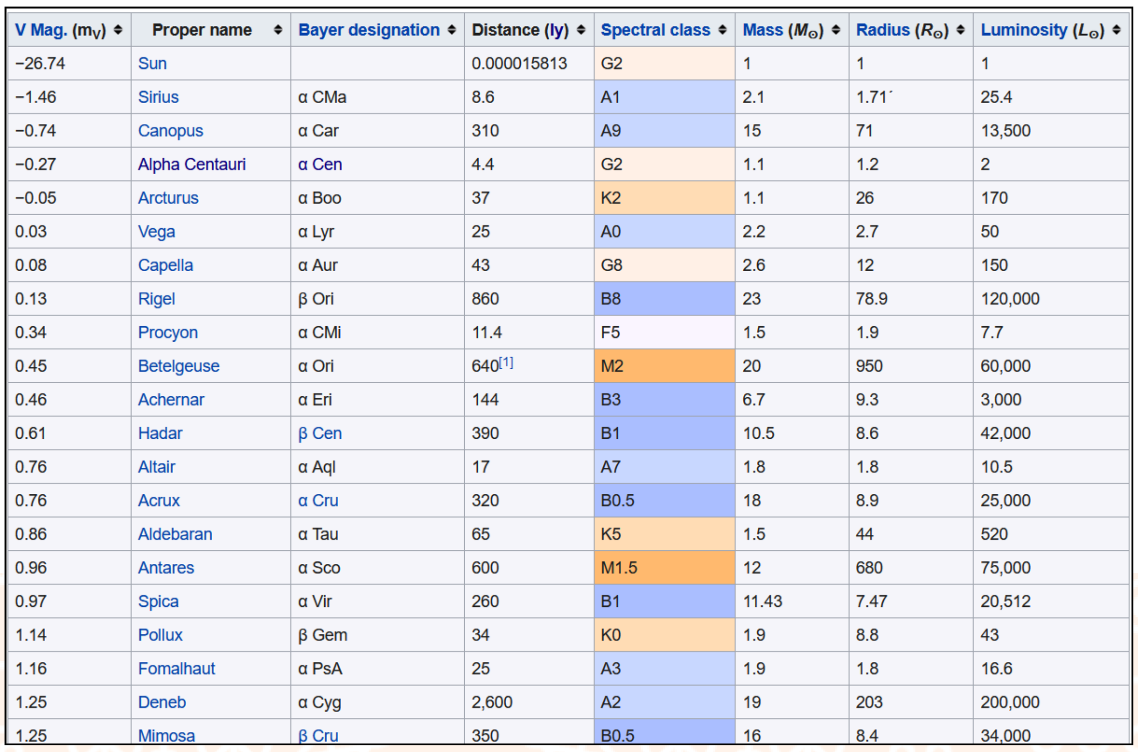
Goal of the Project:

In class 127, you learned about web scraping and you wrote a scraper to scrape the information of different planets from the NASA site.

In this project, you will write a scraper to scrape the data of various stars in our Universe! Stars reveal a lot of information too.

Story:

Our Sun is dying! The world is in an emergency as we are about to lose our star. All groups of scientists around the world have gathered together and created a technology to shift our Earth into another solar system, but which one exactly? Which star out there is safe and welcoming to our Earth? You have been assigned the task to research about stars so that we can choose the best one for us!



V Mag. (m_V)	Proper name	Bayer designation	Distance (ly)	Spectral class	Mass ($M_\odot$)	Radius ($R_\odot$)	Luminosity ($L_\odot$)
−26.74	Sun		0.000015813	G2	1	1	1
−1.46	Sirius	α CMa	8.6	A1	2.1	1.71	25.4
−0.74	Canopus	α Car	310	A9	15	71	13,500
−0.27	Alpha Centauri	α Cen	4.4	G2	1.1	1.2	2
−0.05	Arcturus	α Boo	37	K2	1.1	26	170
0.03	Vega	α Lyr	25	A0	2.2	2.7	50
0.08	Capella	α Aur	43	G8	2.6	12	150
0.13	Rigel	β Ori	860	B8	23	78.9	120,000
0.34	Procyon	α CMi	11.4	F5	1.5	1.9	7.7
0.45	Betelgeuse	α Ori	640[1]	M2	20	950	60,000
0.46	Achernar	α Eri	144	B3	6.7	9.3	3,000
0.61	Hadar	β Cen	390	B1	10.5	8.6	42,000
0.76	Altair	α Aql	17	A7	1.8	1.8	10.5
0.76	Acrux	α Cru	320	B0.5	18	8.9	25,000
0.86	Aldebaran	α Tau	65	K5	1.5	44	520
0.96	Antares	α Sco	600	M1.5	12	680	75,000
0.97	Spica	α Vir	260	B1	11.43	7.47	20,512
1.14	Pollux	β Gem	34	K0	1.9	8.8	43
1.16	Fomalhaut	α PsA	25	A3	1.9	1.8	16.6
1.25	Deneb	α Cyg	2,600	A2	19	203	200,000
1.25	Mimosa	β Cru	350	B0.5	16	8.4	34,000

***This is just for your reference. We expect you to apply your creativity in the project.**

Getting Started:

1. Open your VS Code editor.
2. Create a virtual environment.
3. Source the virtual environment.
4. Install bs4.
   Note: We are not going to use Selenium for this project.
5. Understand the website we want to scrape first: Brightest stars in the universe.
6. Star Data to be scrapped:
   - Star Name
   - Distance
   - Mass
   - Radius
   - Luminosity

Community Link

Publish to Community

Edit Your Project

Last Submitted

Previous Submissions

12th Mar 2024 Open Link

Start Project

Submit Your Project

Learn how to submit your project

Paste your project URL

Submit Project

Class Summary

This project is based on your last class PRO-C127

View Class Summary

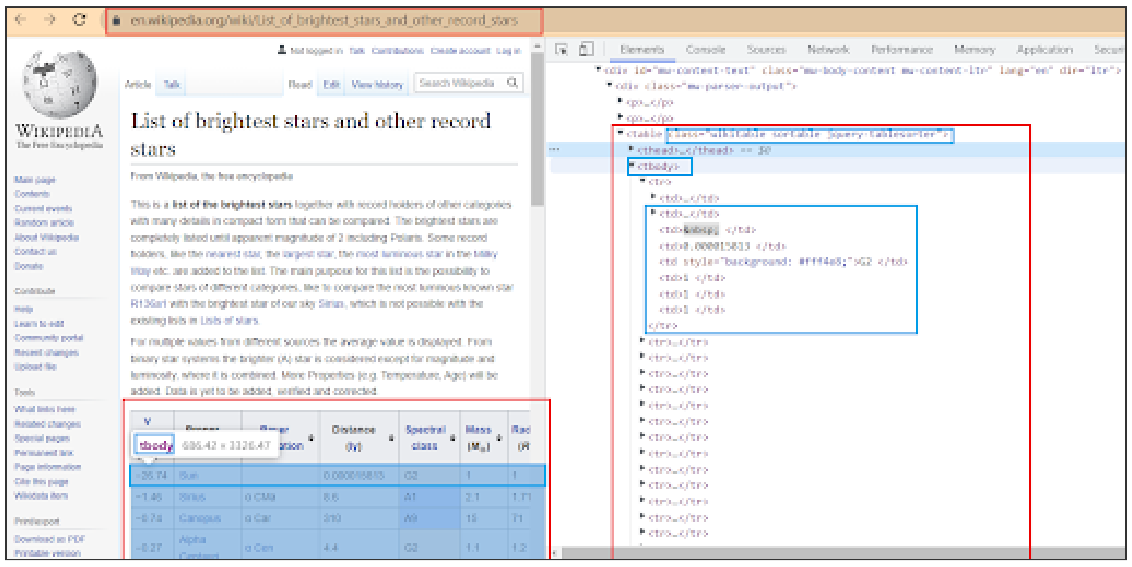
Ask a doubt to your tea

HELP

Specific Tasks to complete the Project:

1. Understand HTML page structure to find required stars data.

   Note: Find the **<tbody>** tag and scrape all **<tr>** for the star data. All the star's data rows are in **<tr>** tags.



List of brightest stars and other record stars

From Wikipedia, the free encyclopedia

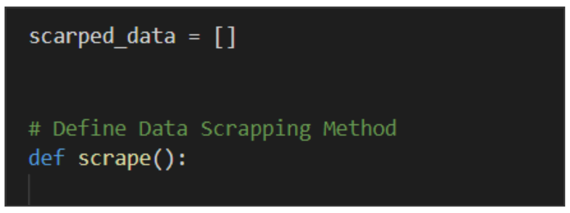
This is a **list of the brightest stars** together with record holders of other categories with many details in compact form that can be compared. The brightest stars are completely listed until apparent magnitude of 2 including Polaris. Some record holders, like the nearest star, the largest star, the most luminous star in the Milky Way etc. are added to the list. The main purpose for this list is the possibility to compare stars of different categories, like to compare the most luminous known star R136a1 with the brightest star of our sky Sirius, which is not possible with the existing lists in Lists of stars.

For multiple values from different sources the average value is displayed. From binary star systems the brighter (A) star is considered except for magnitude and luminosity, where it is combined. More Properties (e.g. Temperature, Age) will be added. Data is yet to be added, verified and corrected.

V	Proper	Bayer ation	Distance (ly)	Spectral class	Mass ($M_☉$)	Rad (R
−26.74	Sun		0.000015813	G2	1	1
−1.46	Sirius	α CMa	8.6	A1	2.1	1.71
−0.74	Canopus	α Car	310	A9	15	71
−0.27	Alpha Centauri	α Cen	4.4	G2	1.1	1.2

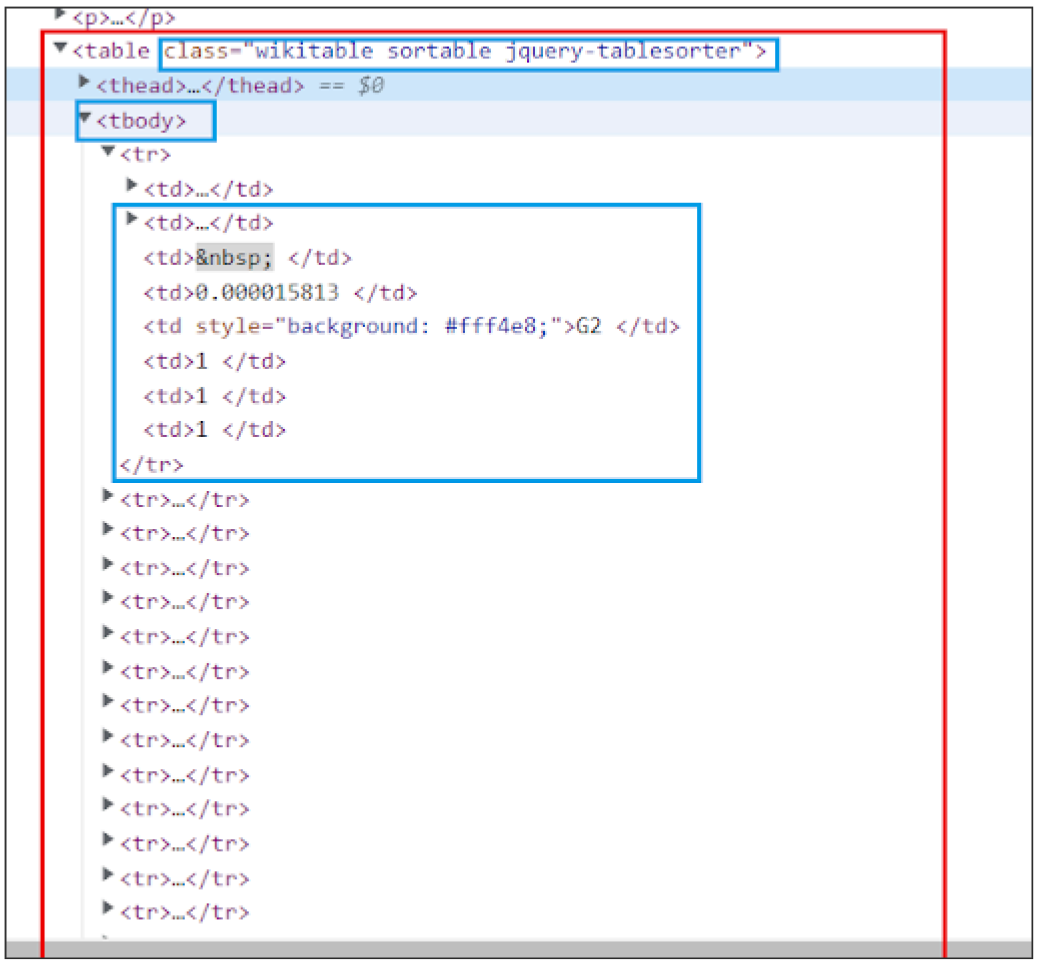
```
▼<div id="mw-content-text" class="mw-body-content mw-content-ltr" lang="en" dir="ltr">
  ▼<div class="mw-parser-output">
    ▶<p>…</p>
    ▶<p>…</p>
    ▼<table class="wikitable sortable jquery-tablesorter">
...   ▶<thead>…</thead> == $0
      ▼<tbody>
        ▼<tr>
          ▶<td>…</td>
          ▶<td>…</td>
           <td>&nbsp; </td>
           <td>0.000015813 </td>
           <td style="background: #fff4e8;">G2 </td>
           <td>1 </td>
           <td>1 </td>
           <td>1 </td>
         </tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
```

2. Import bs4, time and pandas module.
3. Initialise the webdriver
4. Take an empty list **scraped_data** to store scraped data.
5. Define **scrape()** method to scrape all column data.



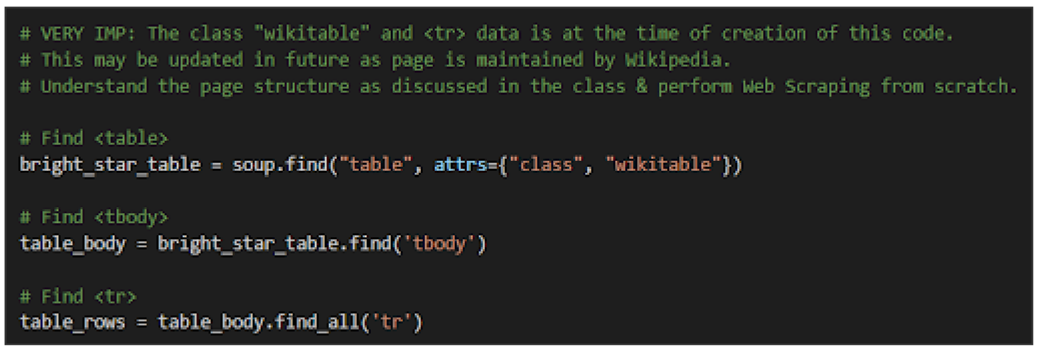
```
scarped_data = []


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

   a. Find the **<table>** using **class="wikitable"**, then find **<tbody>** tag and **<tr>** tag data using **soup**.



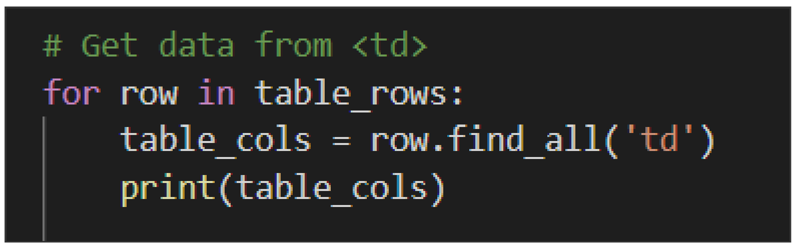
```
  ▶<p>…</p>
  ▼<table class="wikitable sortable jquery-tablesorter">
    ▶<thead>…</thead> == $0
    ▼<tbody>
      ▼<tr>
        ▶<td>…</td>
        ▶<td>…</td>
         <td>&nbsp; </td>
         <td>0.000015813 </td>
         <td style="background: #fff4e8;">G2 </td>
         <td>1 </td>
         <td>1 </td>
         <td>1 </td>
       </tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
      ▶<tr>…</tr>
```

   ***VERY IMP:** The class "wikitable" and <tr> data is at the time of creation of this code. This may be updated in future as the page is maintained by Wikipedia. Understand the page structure as discussed in the class & perform Web Scraping from scratch.*



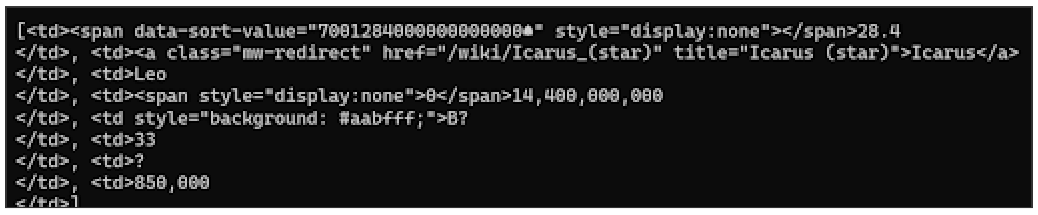
```
# VERY IMP: The class "wikitable" and <tr> data is at the time of creation of this code.
# This may be updated in future as page is maintained by Wikipedia.
# Understand the page structure as discussed in the class & perform Web Scraping from scratch.

# Find <table>
bright_star_table = soup.find("table", attrs={"class", "wikitable"})

# Find <tbody>
table_body = bright_star_table.find('tbody')

# Find <tr>
table_rows = table_body.find_all('tr')
```

   b. Find all <td> tags by looping through all <tr> tags and get all column data.
   ***Note:** Print data to see the output in between.*



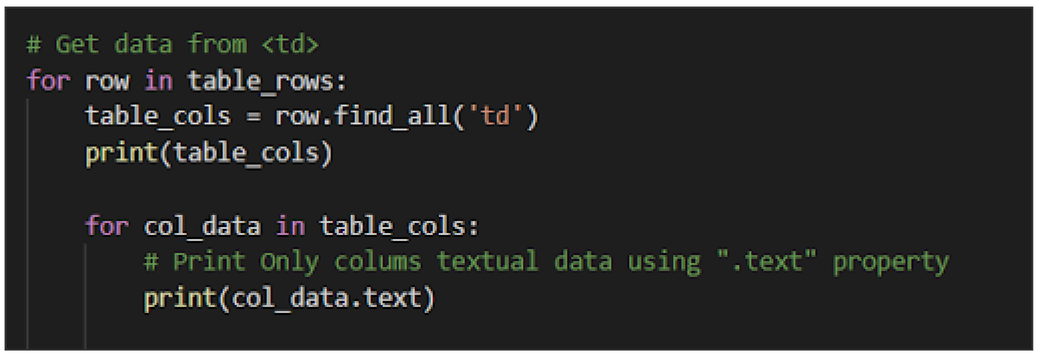
```
# Get data from <td>
for row in table_rows:
    table_cols = row.find_all('td')
    print(table_cols)
```

Output of last <td> tag at the time creation:



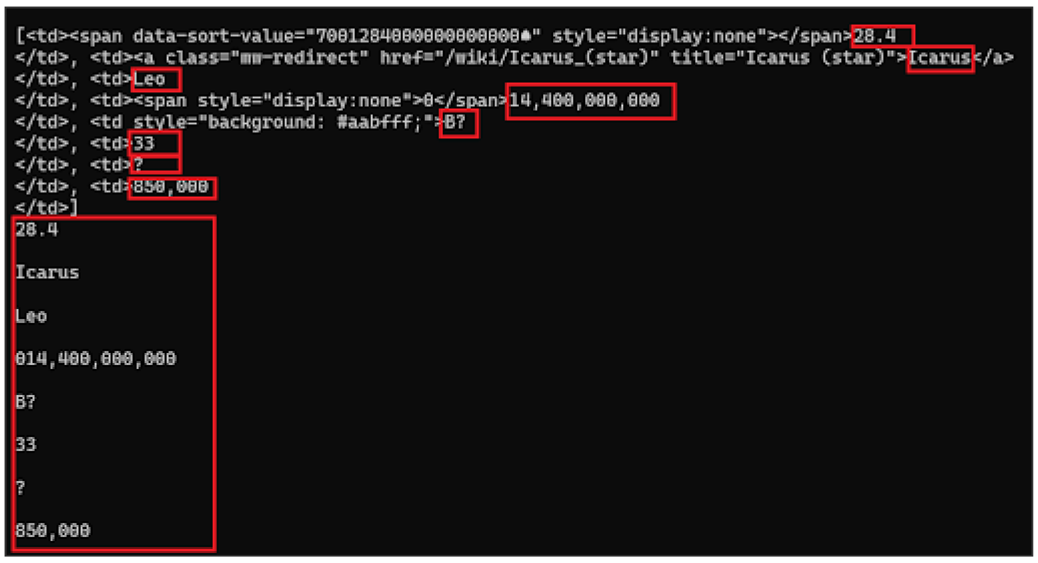
```
[<td><span data-sort-value="7001284000000000000♠" style="display:none"></span>28.4
</td>, <td><a class="mw-redirect" href="/wiki/Icarus_(star)" title="Icarus (star)">Icarus</a>
</td>, <td>Leo
</td>, <td><span style="display:none">0</span>14,400,000,000
</td>, <td style="background: #aabfff;">B?
</td>, <td>33
</td>, <td>?
</td>, <td>850,000
</td>]
```

c. Print only text data of the columns using **.text**



```
# Get data from <td>
for row in table_rows:
    table_cols = row.find_all('td')
    print(table_cols)

    for col_data in table_cols:
        # Print Only colums textual data using ".text" property
        print(col_data.text)
```

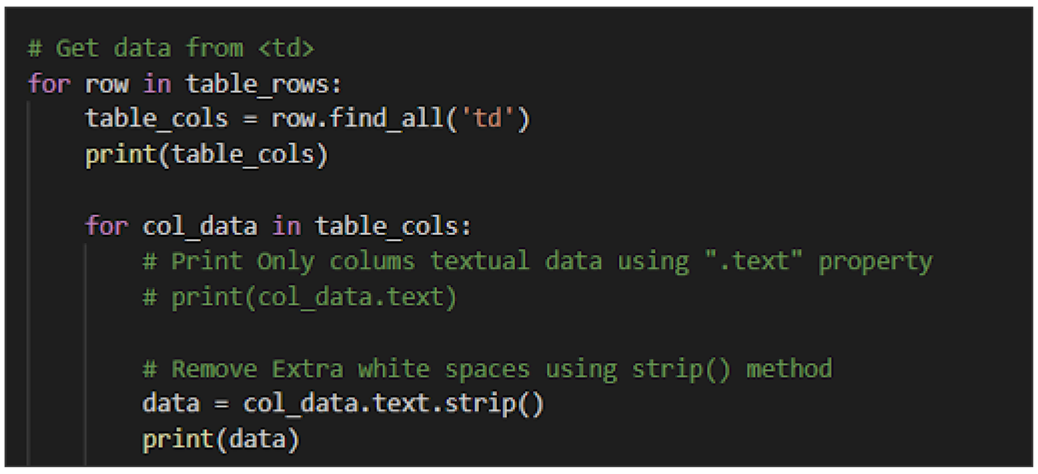
Output of last <td> tag at the time creation:



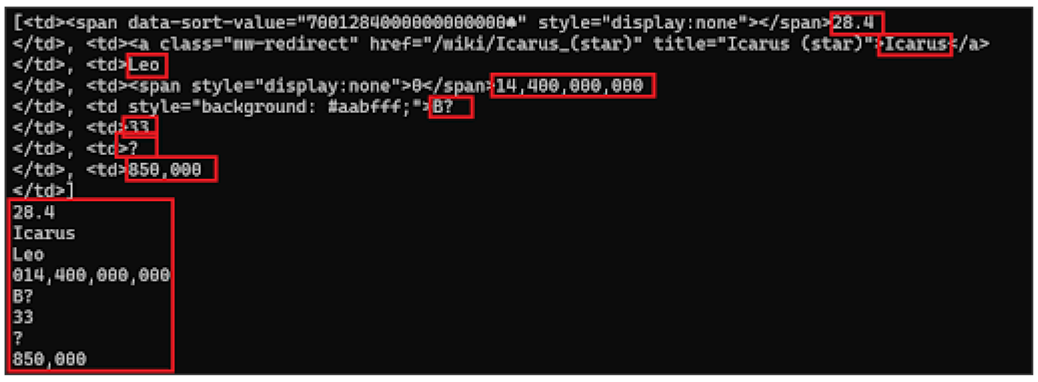
```
[<td><span data-sort-value="7001284000000000000♠" style="display:none"></span>28.4
</td>, <td><a class="mw-redirect" href="/wiki/Icarus_(star)" title="Icarus (star)">Icarus</a>
</td>, <td>Leo
</td>, <td><span style="display:none">0</span>14,400,000,000
</td>, <td style="background: #aabfff;">B?
</td>, <td>33
</td>, <td>?
</td>, <td>850,000
</td>]
28.4

Icarus

Leo

014,400,000,000

B?

33

?

850,000
```

d. Remove extra whitespace in only text data using **strip()** method

Note: The **strip()** method removes characters from both left and right based on the argument specifying the characters to be removed. If no argument is passed whitespaces are removed. Read more here.

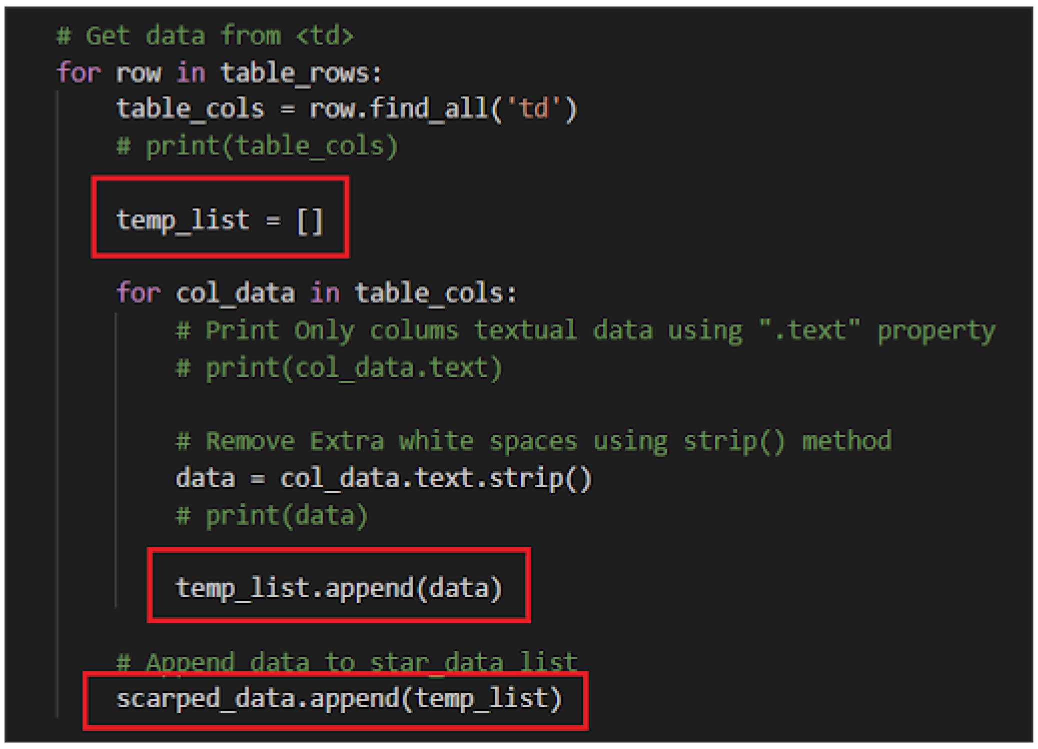


```
# Get data from <td>
for row in table_rows:
    table_cols = row.find_all('td')
    print(table_cols)

    for col_data in table_cols:
        # Print Only colums textual data using ".text" property
        # print(col_data.text)

        # Remove Extra white spaces using strip() method
        data = col_data.text.strip()
        print(data)
```



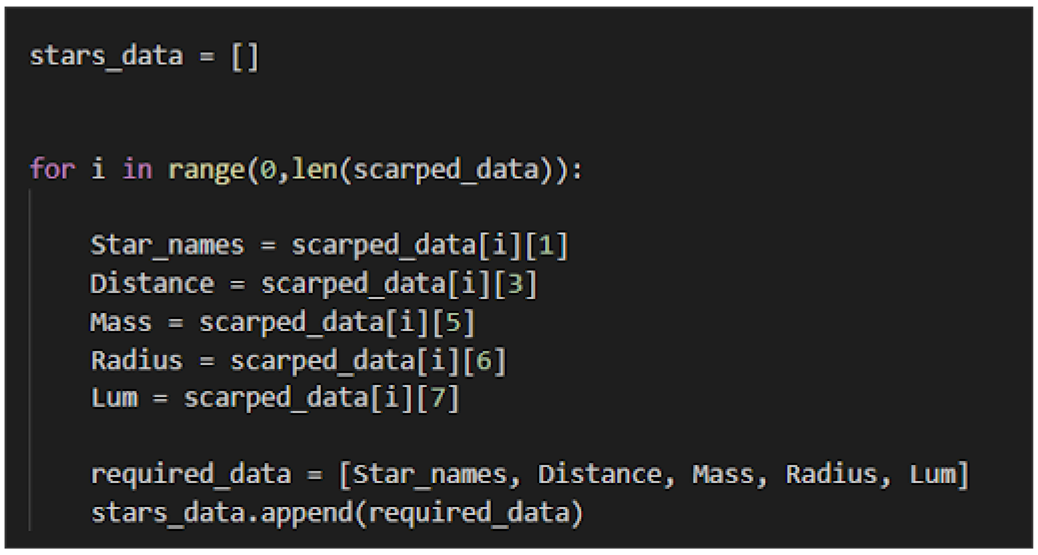
```
[<td><span data-sort-value="7001284000000000000♠" style="display:none"></span>28.4
</td>, <td><a class="mw-redirect" href="/wiki/Icarus_(star)" title="Icarus (star)">Icarus</a>
</td>, <td>Leo
</td>, <td><span style="display:none">0</span>14,400,000,000
</td>, <td style="background: #aabfff;">B?
</td>, <td>33
</td>, <td>?
</td>, <td>850,000
</td>]
28.4
Icarus
Leo
014,400,000,000
B?
33
?
850,000
```

e. Create a **temp_list** to store each column text data and append it to **scraped_data** list



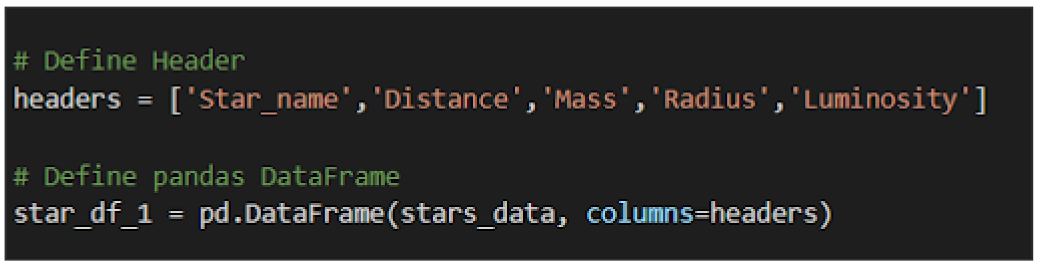
```
# Get data from <td>
for row in table_rows:
    table_cols = row.find_all('td')
    # print(table_cols)

    temp_list = []

    for col_data in table_cols:
        # Print Only colums textual data using ".text" property
        # print(col_data.text)

        # Remove Extra white spaces using strip() method
        data = col_data.text.strip()
        # print(data)

        temp_list.append(data)

    # Append data to star_data list
    scarped_data.append(temp_list)
```

6. Save data to CSV:
   a. Create an empty list **stars_data**
   b. Loop through **star_data** list to append column data for star name, distance, mass, radius and luminosity using index with **Star_names, Distance, Mass, Radius and Lum** as variable names respectively.



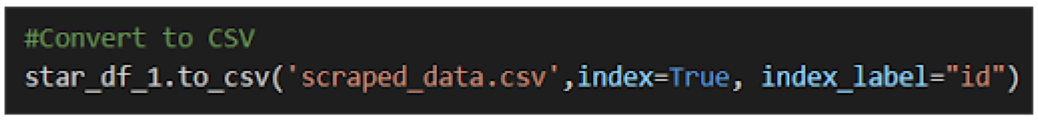
```
stars_data = []


for i in range(0,len(scarped_data)):

    Star_names = scarped_data[i][1]
    Distance = scarped_data[i][3]
    Mass = scarped_data[i][5]
    Radius = scarped_data[i][6]
    Lum = scarped_data[i][7]

    required_data = [Star_names, Distance, Mass, Radius, Lum]
    stars_data.append(required_data)
```

   c. Use **pandas** to create dataframe



```
# Define Header
headers = ['Star_name','Distance','Mass','Radius','Luminosity']

# Define pandas DataFrame
star_df_1 = pd.DataFrame(stars_data, columns=headers)
```

   d. Export dataframe to csv



```
#Convert to CSV
star_df_1.to_csv('scraped_data.csv',index=True, index_label="id")
```

Submitting the Project:

1. Upload your completed project to your GitHub account.
2. Create a new repository named **"Project 127: Web Data Extraction 1"**.
3. **Upload** working code to this GitHub repository.
4. Enable GitHub pages for the repository.
5. Copy the link to the GitHub pages link in the Student Dashboard. link to the GitHub pages link in the Student Dashboard.

Hints:

1. Notice the unit of dimensions of the star data such as radius, distance, and mass. If you are scraping from different sources make sure the units are the same, if not then we have to convert them.

2. There is only one table on the page at the time creation of this document.

HELP

3. HTML table tags:

<table>	HTML table tag
<th>	HTML table heading
<tbody>	HTML table body
<tr>	HTML table row
<td>	HTML table column

HELP