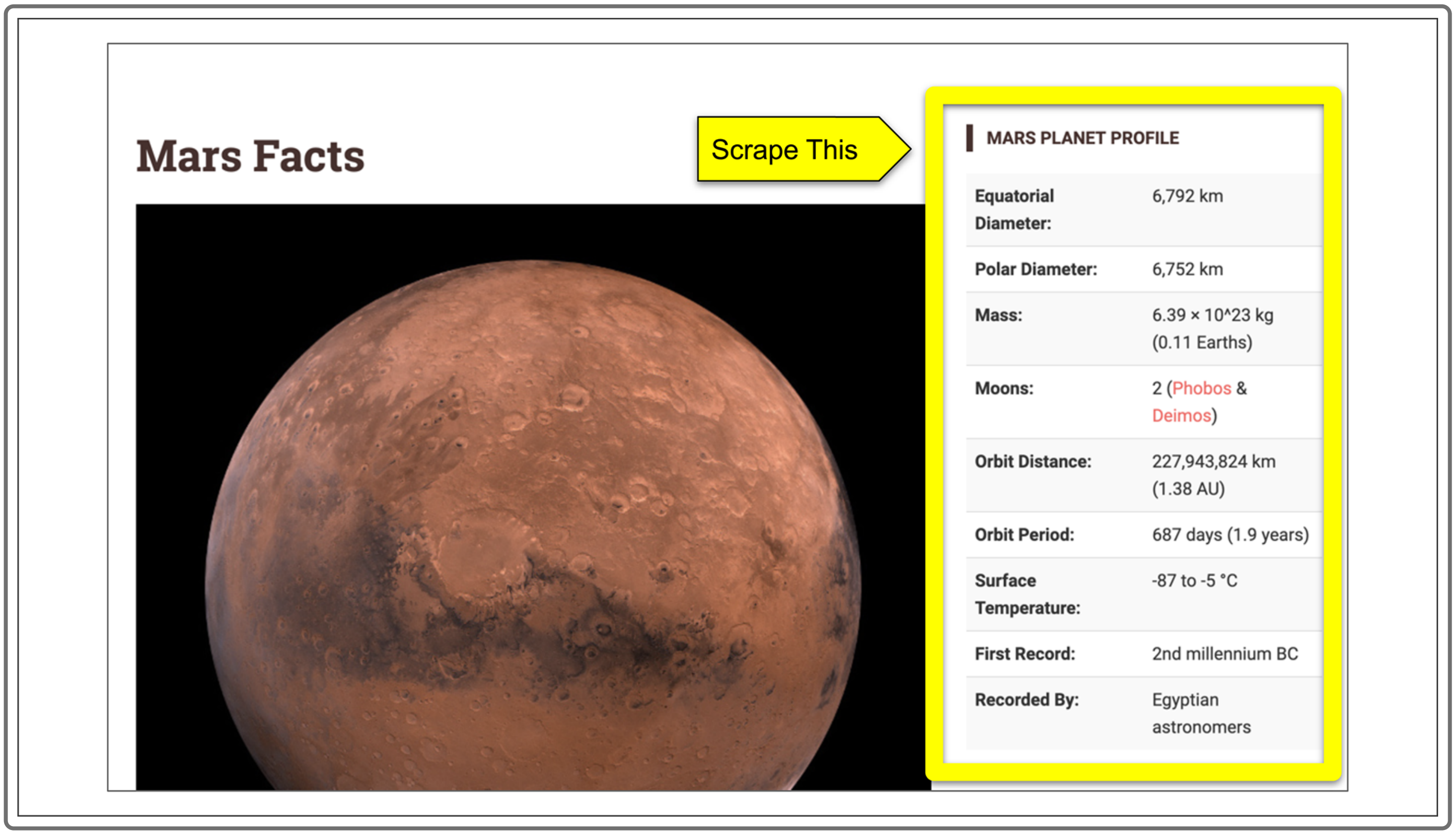
11.6.2

Scraping Mars Facts

Next, Robin wants to try scraping a collection of Mars facts. This information exists in a table. Even though that format differs from what she's encountered before, she can still use DevTools to pinpoint the tag locations. She can then extract the table based on its tag-and-attribute pairing in the HTML code.

Robin has chosen to collect her data from Mars Facts (https://galaxyfacts-mars.com) . So, let's visit that webpage and examine it.



MARS PLANET PROFILE

Equatorial Diameter:	6,792 km
Polar Diameter:	6,752 km
Mass:	6.39 × 10^23 kg (0.11 Earths)
Moons:	2 (Phobos & Deimos)
Orbit Distance:	227,943,824 km (1.38 AU)
Orbit Period:	687 days (1.9 years)
Surface Temperature:	-87 to -5 °C
First Record:	2nd millennium BC
Recorded By:	Egyptian astronomers

Robin wants to scrape the data from the Mars Planet Profile section of the webpage. We'll discover that this data exists in a table.

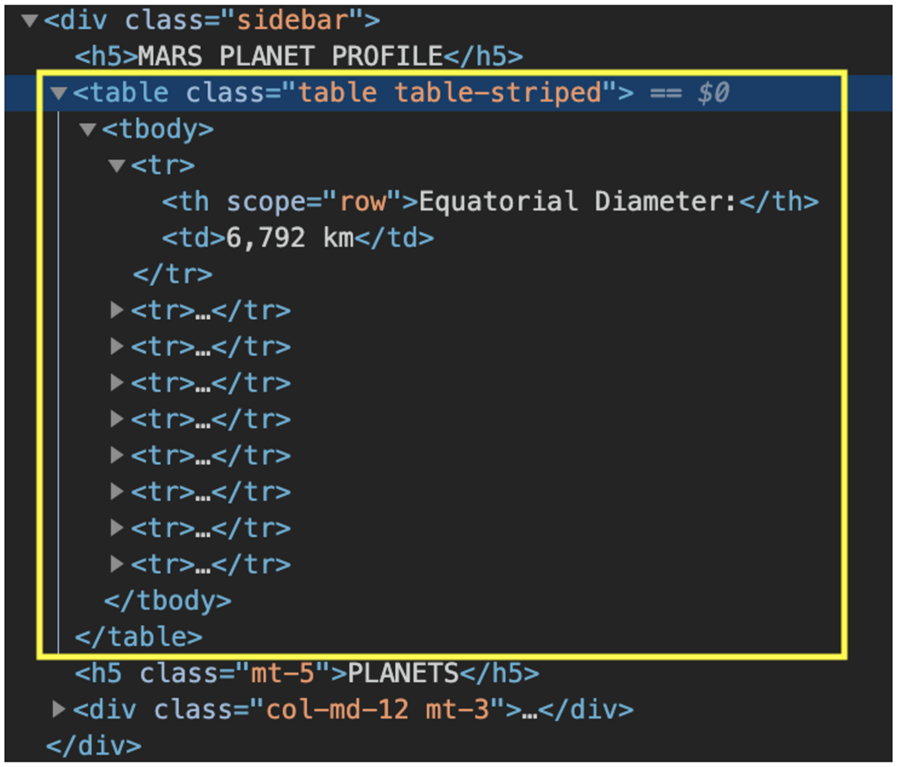
Examine HTML Table Data

Let's examine the webpage again, but this time, we'll use DevTools. A `<div />` tag with a `class` attribute of `sidebar` contains the side panel. But, a child element, which has the `<table />` tag, contains the data that we want. This `<table />` tag, in turn, has `table` and `table-striped` classes.

On websites, a common way of presenting data is through tables. So in this section, we'll examine the HTML table data that contains the Mars facts. In the next two sections, we'll learn two ways to scrape such a table.

The entire block of code that DevTools displays for our table of Mars facts is as follows:

```
<table class = "table table-striped">
   <tbody>
      <tr>
         <th scope="row"> Equatorial Diameter:</th>
         <td>6,792 km</td>
      </tr>
      <tr>...</tr>
      <tr>...</tr>
      <tr>...</tr>
      <tr>...</tr>
      <tr>...</tr>
      <tr>...</tr>
      <tr>...</tr>
      <tr>...</tr>
   </tbody>
</table>
```



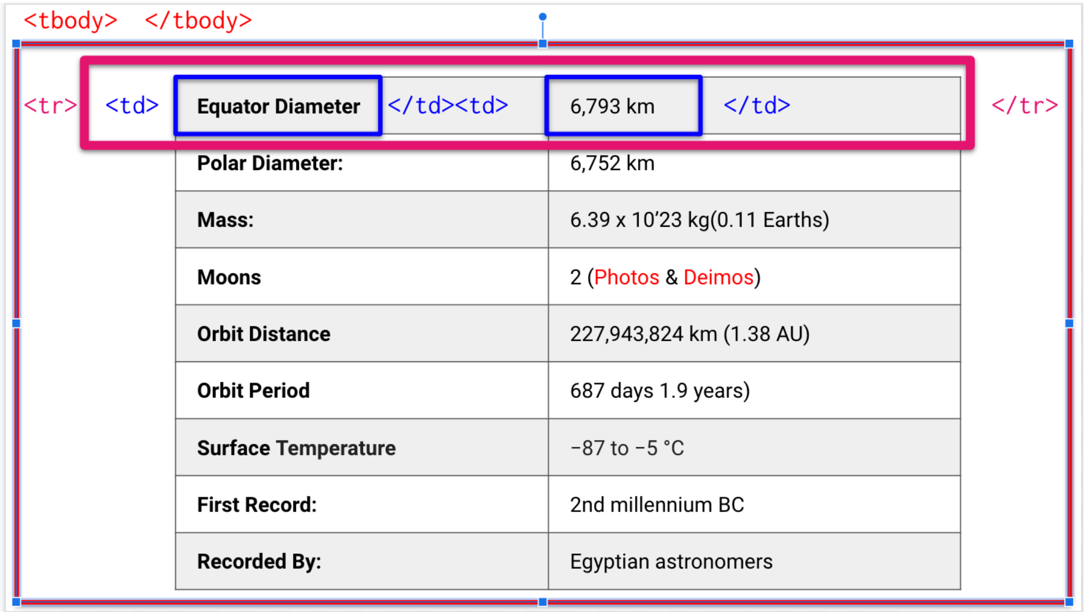
```
▼<div class="sidebar">
   <h5>MARS PLANET PROFILE</h5>
 ▼<table class="table table-striped"> == $0
   ▼<tbody>
     ▼<tr>
         <th scope="row">Equatorial Diameter:</th>
         <td>6,792 km</td>
       </tr>
     ▶<tr>…</tr>
     ▶<tr>…</tr>
     ▶<tr>…</tr>
     ▶<tr>…</tr>
     ▶<tr>…</tr>
     ▶<tr>…</tr>
     ▶<tr>…</tr>
     ▶<tr>…</tr>
     </tbody>
   </table>
   <h5 class="mt-5">PLANETS</h5>
 ▶<div class="col-md-12 mt-3">…</div>
 </div>
```

Although the preceding HTML code that creates our table might seem complex, we just need to break down and name each component.

An HTML table is made up of many smaller containers. The main container is the `<table />` tag. Inside that is the `<tbody />` tag, which is the table body. The table body, in turn, consists of the table headers, columns, and rows.

Each `<tr />` tag refers to a table row. Each row can contain one of two types of table cells: table headers or table data. Table headers are stored in `<th />` tags. These often contain column headings. Table data is stored in `<td />` tags.

In our Mars facts table, the `<tbody />` tag contains all the rows in the Mars Planet Profile section. The first `<tr />` tag contains the "Equator Diameter" and "6,793 km" entries. And, "Equator Diameter" and "6,793 km" are each stored in a `<td />` tag.



Equator Diameter	6,793 km
Polar Diameter:	6,752 km
Mass:	6.39 x 10'23 kg(0.11 Earths)
Moons	2 (Photos & Deimos)
Orbit Distance	227,943,824 km (1.38 AU)
Orbit Period	687 days 1.9 years)
Surface Temperature	-87 to -5 °C
First Record:	2nd millennium BC
Recorded By:	Egyptian astronomers

Scrape a Table with Splinter and Beautiful Soup

With the information that we have about the Mars facts table, let's scrape it.

To begin, create a new Jupyter notebook named `mars_facts.ipynb`. Then set up the scraping entering and running the following code in the first cell:

```
from splinter import Browser
from bs4 import BeautifulSoup as soup
from webdriver_manager.chrome import ChromeDriverManager

# Set up Splinter
executable_path = {'executable_path': ChromeDriverManager().install()}
browser = Browser('chrome', **executable_path, headless=False)

# Visit the Mars Facts site
url = 'https://galaxyfacts-mars.com/'
```

```
browser.visit(url)

html = browser.html
html_soup = soup(html, 'html.parser')
```

Next, we want to find the table. To do so, In the next cell, enter and run the following code:

```
table = html_soup.find('table', class_='table-striped')
```

In the preceding code, we use the Beautiful Soup `find` method to find the table. Because we identified one of the `class` attributes of the sidebar table as `table-striped` earlier, we use it to identify the `table` element that we're searching for. We assign the HTML code for this table to the `table` variable. We can then print the table contents to confirm that we found the correct element.

Next, we want to store the table data in a Python data structure. To do so, in the next cell, enter and run the following code:

```
mars_facts = {}
rows = table.find_all('tr')
for row in rows:
    row_heading = row.find('th').text
    row_data = row.find('td').text.strip()
    mars_facts[row_heading] = row_data
```

Let's go over the preceding code:

1. The `mars_facts` dictionary, empty for now, will eventually hold the table data.
2. We use the `find_all` method to find all the table rows (`tr`). We save the table rows in the `rows` variable.
3. In each `for` loop iteration, the text of the row's table header (`th`) gets saved to the `row_heading` variable.

4. In the same `for` loop iteration, the text of the row's table data (`td`) gets extracted. The `strip` method then strips any white space that this text contains. The text then gets saved to the `row_data` variable.
5. In the same `for` loop iteration, the row becomes an entry in the `mars_facts` dictionary. The table heading becomes the key, and the table data becomes the value.

The preceding code creates the `mars_facts` dictionary as follows:

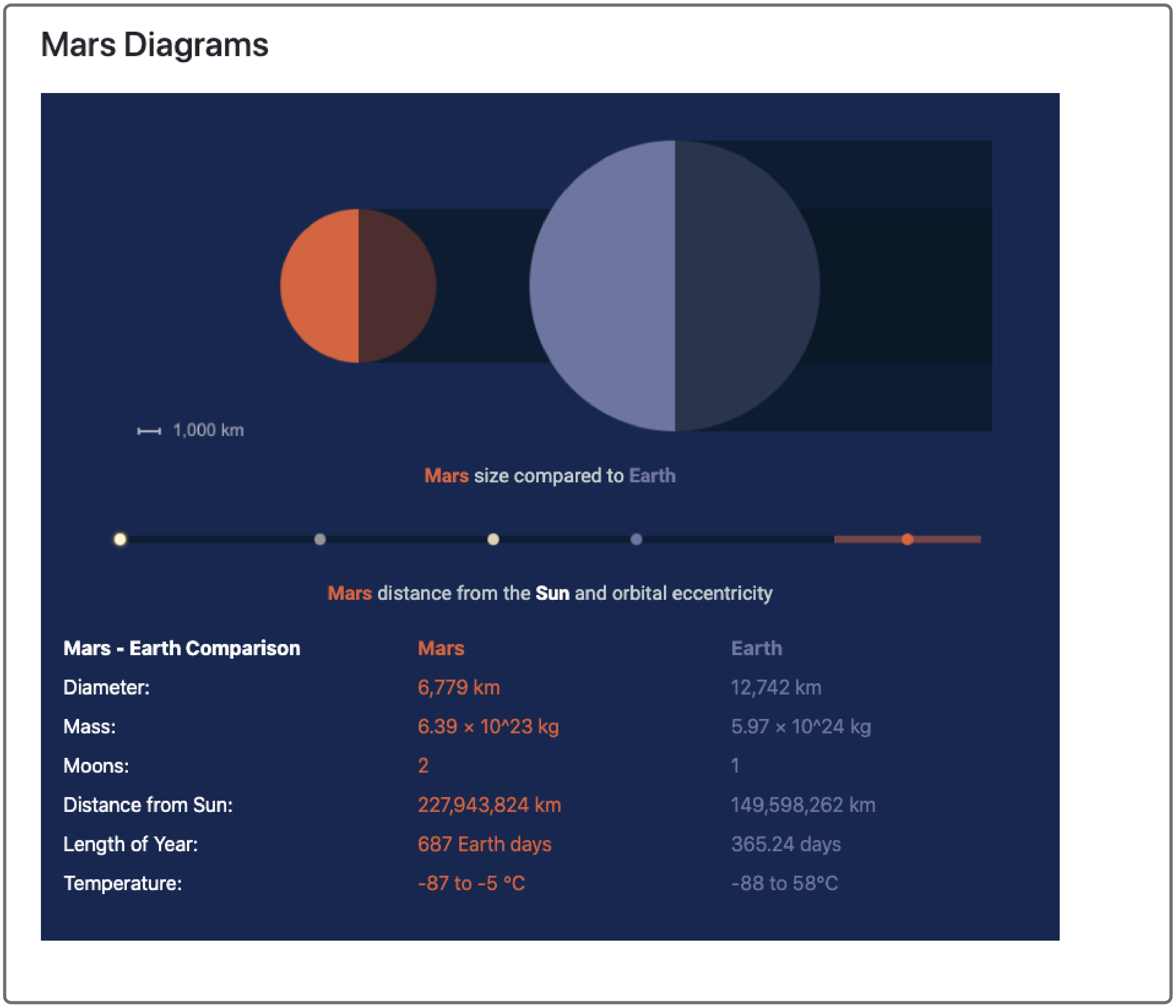
```
{'Equatorial Diameter:': '6,792 km',
 'Polar Diameter:': '6,752 km',
 'Mass:': '6.39 × 10^23 kg (0.10 Earths)',
 'Moons:': '2 ( Phobos  &  Deimos )',
 'Orbit Distance:': '227,943,824 km (1.38 AU)',
 'Orbit Period:': '687 days (1.9 years)',
 'Surface Temperature:': '-87 to -5 °C',
 'First Record:': '2nd millennium BC',
 'Recorded By:': 'Egyptian astronomers'}
```

Finally, we want to quit the browser session. To do so, in the next cell, enter and run `browser.quit()`.

In this section, we scraped an HTML table by using Splinter and Beautiful Soup. But, Pandas also offers a way to scrape HTML tables that's more straightforward to use. So in the next section, we'll scrape another HTML table by using Pandas.

Scrape a Table with Pandas

We can scrape an HTML table by using the Pandas `read_html` function. To explore this, let's try scraping the other table on the webpage—specifically, the "Mars - Earth Comparison" table.


1,000 km
Mars size compared to Earth
Mars distance from the Sun and orbital eccentricity

Mars Diagrams

Mars - Earth Comparison	Mars	Earth
Diameter:	6,779 km	12,742 km
Mass:	6.39 × 10^23 kg	5.97 × 10^24 kg
Moons:	2	1
Distance from Sun:	227,943,824 km	149,598,262 km
Length of Year:	687 Earth days	365.24 days
Temperature:	-87 to -5 °C	-88 to 58°C

To begin, in a new cell in your Jupyter notebook, enter and run `import pandas as pd`. Then in the next blank cell, enter and run the following code:

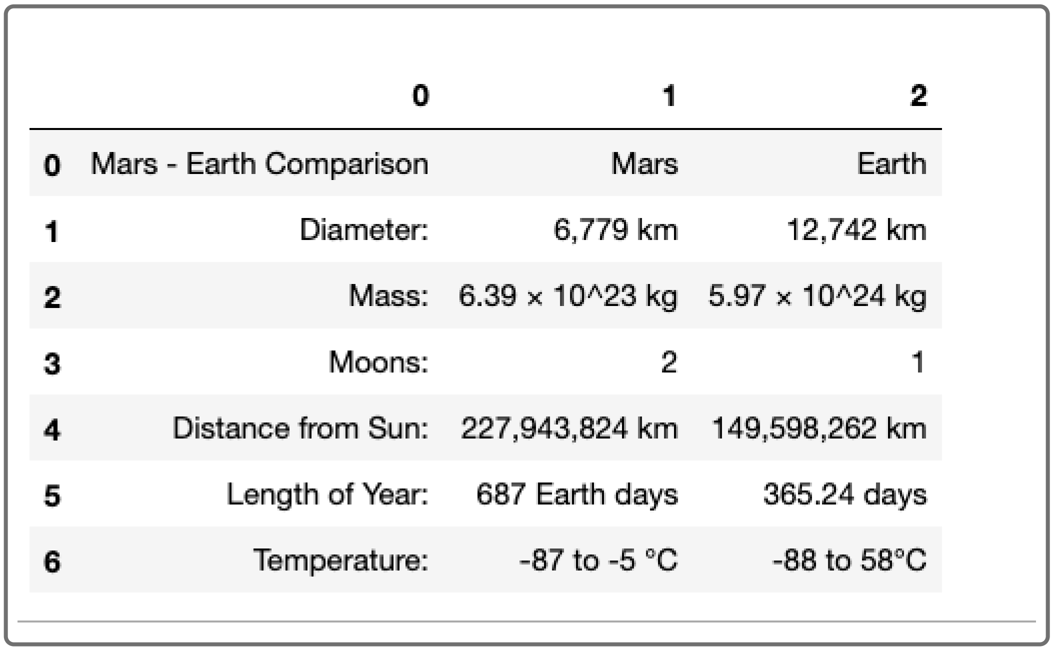
```
df = pd.read_html('https://galaxyfacts-mars.com')
df
```

In the preceding code, the first line creates a new DataFrame from the HTML table. The way that it works is this: The Pandas `read_html` method searches for tables and returns a list of those that exist in the HTML code of the webpage. The `df` variable thus gets assigned the two tables in the form of a list. The first table is the one that we're interested in. The second table is the one that we previously scraped. So, we'll select the first table by using its index of 0.

To do so, in the next cell, enter and run the following code:

```
mars_df = df[0]
mars_df
```

The preceding code selects the first table by using `df[0]`. This table contains three columns of data. The first column contains the "Mars - Earth Comparison" items, like "Diameter" and "Mass." The second column contains the Mars measurements. The third column contains the Earth measurements. But, we still need to clean up this table a bit. For example, we want column names that are more specific than 0, 1, and 2.

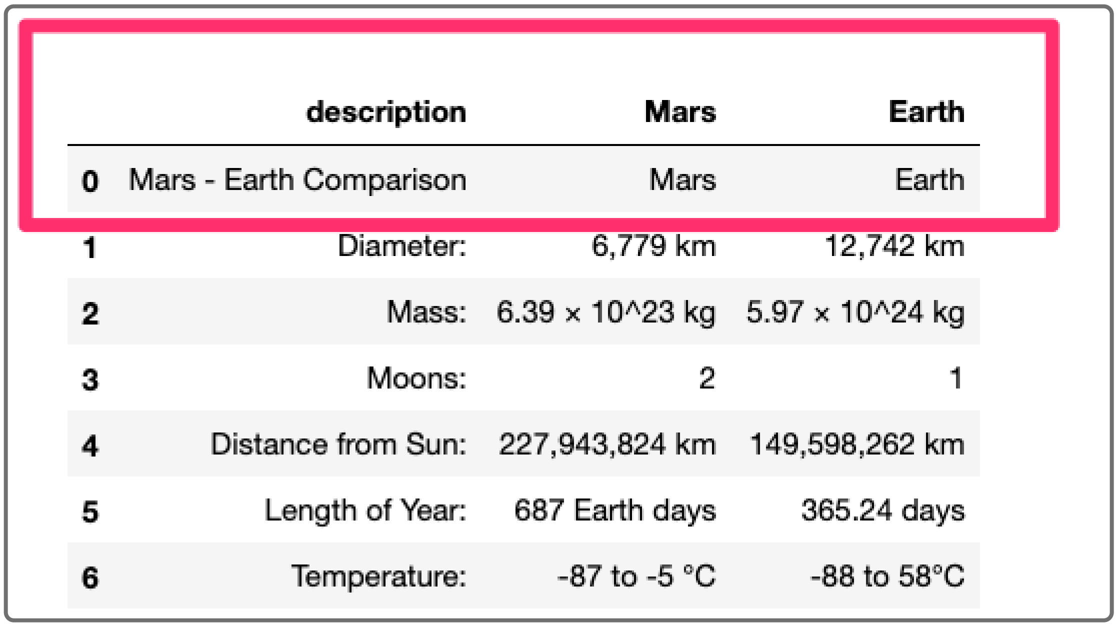


	0	1	2
0	Mars - Earth Comparison	Mars	Earth
1	Diameter:	6,779 km	12,742 km
2	Mass:	6.39 × 10^23 kg	5.97 × 10^24 kg
3	Moons:	2	1
4	Distance from Sun:	227,943,824 km	149,598,262 km
5	Length of Year:	687 Earth days	365.24 days
6	Temperature:	-87 to -5 °C	-88 to 58°C

To rename the columns, in the next cell, enter and run the following code:

```
`mars_df.columns=['description', 'Mars', 'Earth']`.
```

Now, the first row is redundant, because it contains basically the same information as our new column headings.



	description	Mars	Earth
0	Mars - Earth Comparison	Mars	Earth
1	Diameter:	6,779 km	12,742 km
2	Mass:	6.39 × 10^23 kg	5.97 × 10^24 kg
3	Moons:	2	1
4	Distance from Sun:	227,943,824 km	149,598,262 km
5	Length of Year:	687 Earth days	365.24 days
6	Temperature:	-87 to -5 °C	-88 to 58°C

So, we want to eliminate the first row. To do so, in the next cell, enter and run the following code:

```
mars_df = mars_df.iloc[1:]
```

The preceding code eliminates the first row by using `iloc` to assign all the rows after the first row to the `mars_df` variable.

To recap, here's the full block of code that we used to scrape and clean the data:

```
import pandas as pd

# Read in HTML tables into a DataFrame
df = pd.read_html('https://galaxyfacts-mars.com')
# Select the first table
mars_df = df[0]

# Rename columns
mars_df.columns=['description', 'Mars', 'Earth']
```

```
# Remove the first row from the DataFrame
mars_df = mars_df.iloc[1:]
```

The Pandas `read_html` method proves useful for scraping HTML tables. But, you can use Splinter and Beautiful Soup to scrape information that's not neatly contained in tables. And, that's exactly what you'll do in the next section.

Now that you've learned how to scrape data that exists in a table, you'll next put together what you've learned as you scrape Mars news articles.