Introduction to Web Scraping in Python:
Comparing the Consistency of Two Disc Golf Manufacturers

By: Campbell DiVenere

The internet is chock full of data. From online shopping, to reading the news, to simply researching a newfound interest, we are exposed to so much data on every webpage we visit, that many of us forget that this data is all fully accessible! However, it is not always to the advantage of a company or a website to simply hand over this data with a convenient click-to-download button. For this reason, we must utilize web scraping.

Most of the data we see online is unstructured, meaning it is not stored in a structured database format. Essentially, this means that we cannot simply copy an entire webpage, and paste all the important bits into a ready to analyze spreadsheet (convenient as that would be). We must actually scrape the necessary data from the site itself. The tutorial below will give you step-by-step instructions for scraping web data in Python, so that you can be analyzing any data the internet has to offer in no time!

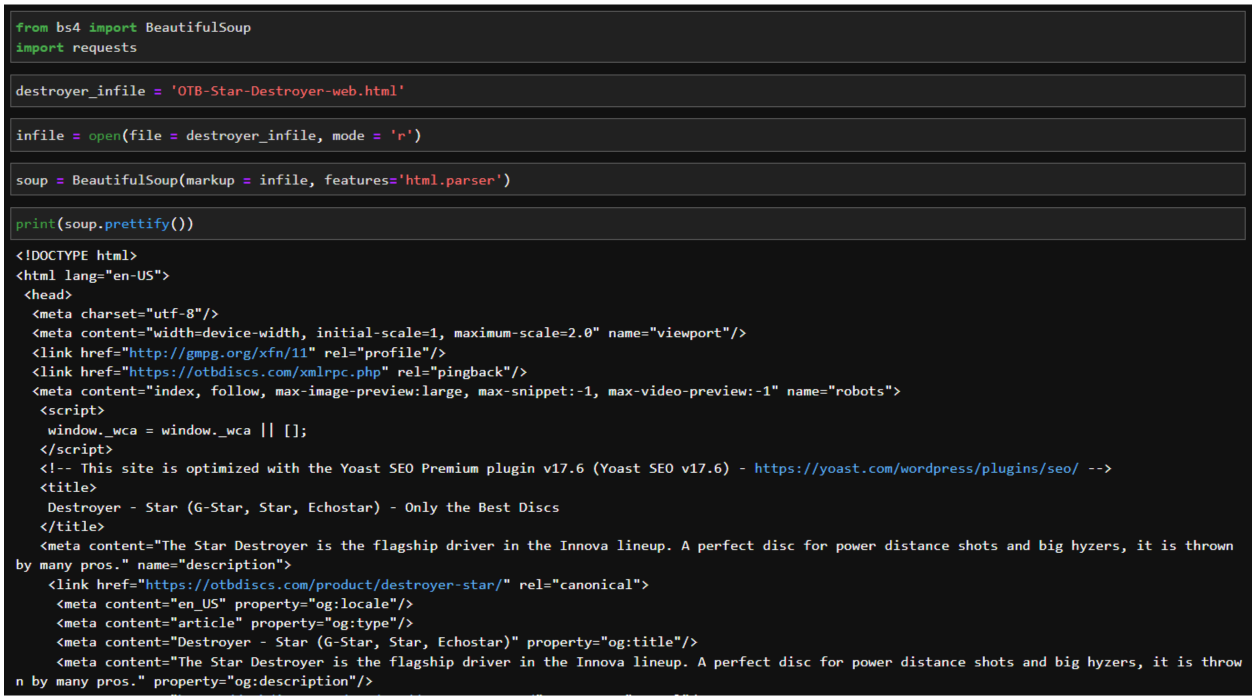
Disc Golf: The Consistency of Manufacturers

For all the disc golf fans out there, you know that consistency is the name of the game. While we all strive to stay in the fairway, it is inevitable that at some point one of your favorite discs will find its way to the bottom of a lake, or simply vanish into a forest. Sad, but no matter, as you should be able to go out and replace it with an exact replica, right? This may not be as easy as it seems. A key component in a disc's flight is the weight of the disc. Even a gram or two difference in weight for the same disc mold can significantly alter the shape of the flight, distance, and how well the disc responds to wind. While disc golf companies do list the weight of their discs, anyone disc golfer with a scale knows that they don't always get it right! This tutorial will put two of the biggest disc golf manufacturers to the test, Innova and MVP, so we can finally settle the debate on who is the most consistent with real world data.

We will use the website Only The Best Discs to scrape the listed and measured weights for all available discs of the flagship mold for each brand, the Star Destroyer from Innova, and the Teleport from MVP. The full code used in this tutorial can be found on github.

The first step will be to install the necessary packages bs4 and requests into your Python environment if you have not already done so (Jupyter Lab will be used in this tutorial). We can then import the packages, and from bs4, we will import BeautifulSoup, an extremely useful and straightforward tool when it comes to web scraping. Now we can call the webpage that has all of the data on the available Star Destroyer discs.

Innova Star Destroyer Consistency Analysis



```python
from bs4 import BeautifulSoup
import requests
```

```python
destroyer_infile = 'OTB-Star-Destroyer-web.html'
```

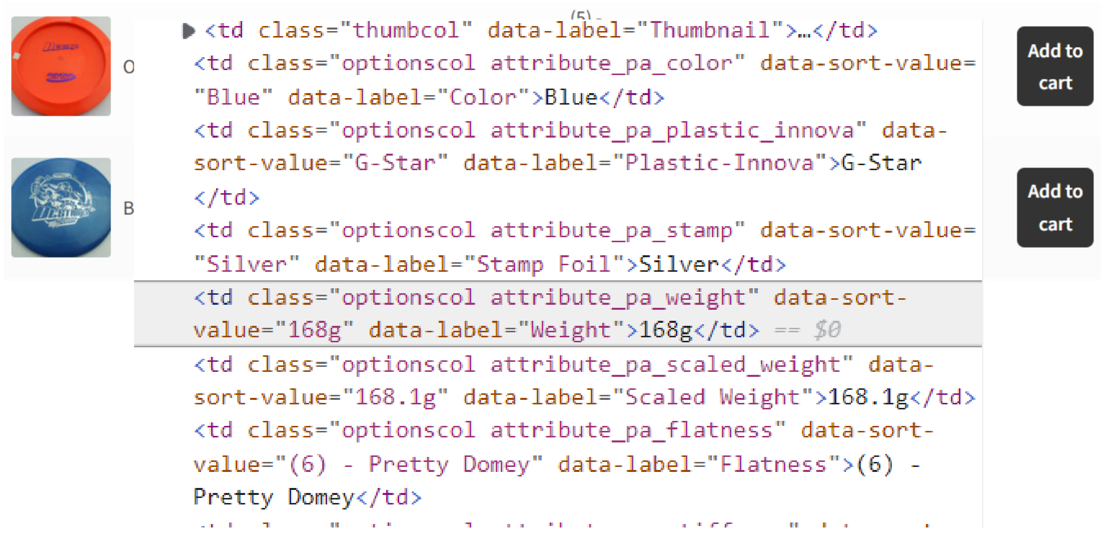
```python
infile = open(file = destroyer_infile, mode = 'r')
```

```python
soup = BeautifulSoup(markup = infile, features='html.parser')
```

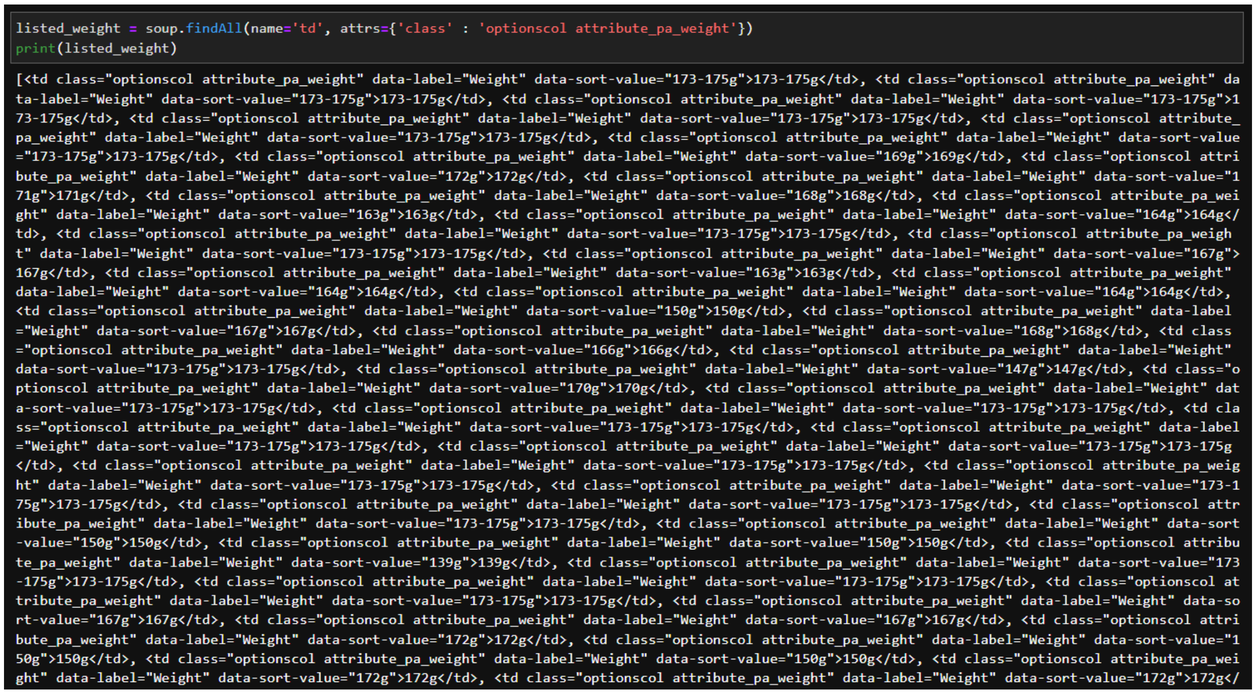
```python
print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en-US">
 <head>
  <meta charset="utf-8"/>
  <meta content="width=device-width, initial-scale=1, maximum-scale=2.0" name="viewport"/>
  <link href="http://gmpg.org/xfn/11" rel="profile"/>
  <link href="https://otbdiscs.com/xmlrpc.php" rel="pingback"/>
  <meta content="index, follow, max-image-preview:large, max-snippet:-1, max-video-preview:-1" name="robots">
   <script>
    window._wca = window._wca || [];
   </script>
   <!-- This site is optimized with the Yoast SEO Premium plugin v17.6 (Yoast SEO v17.6) - https://yoast.com/wordpress/plugins/seo/ -->
   <title>
    Destroyer - Star (G-Star, Star, Echostar) - Only the Best Discs
   </title>
   <meta content="The Star Destroyer is the flagship driver in the Innova lineup. A perfect disc for power distance shots and big hyzers, it is thrown
by many pros." name="description">
    <link href="https://otbdiscs.com/product/destroyer-star/" rel="canonical">
     <meta content="en_US" property="og:locale"/>
     <meta content="article" property="og:type"/>
     <meta content="Destroyer - Star (G-Star, Star, Echostar)" property="og:title"/>
     <meta content="The Star Destroyer is the flagship driver in the Innova lineup. A perfect disc for power distance shots and big hyzers, it is throw
n by many pros." property="og:description"/>
```

We can see that the data in it's current state is not particularly useful nor workable. No problem! Using inspect element on any listed weight from the website, we can find the class of the element, and use this with the ".findAll" command to pull all the other elements with the same class, leaving us with only the elements containing the listed weight we are after!



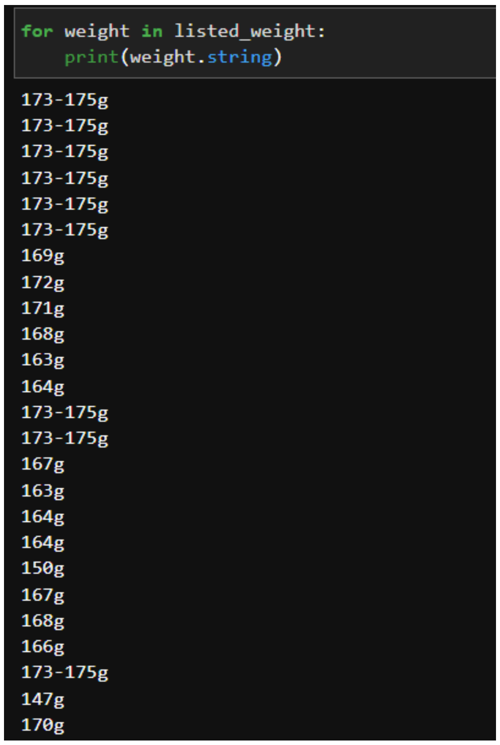
```
<td class="thumbcol" data-label="Thumbnail">…</td>
<td class="optionscol attribute_pa_color" data-sort-value=
"Blue" data-label="Color">Blue</td>
<td class="optionscol attribute_pa_plastic_innova" data-
sort-value="G-Star" data-label="Plastic-Innova">G-Star
</td>
<td class="optionscol attribute_pa_stamp" data-sort-value=
"Silver" data-label="Stamp Foil">Silver</td>
<td class="optionscol attribute_pa_weight" data-sort-
value="168g" data-label="Weight">168g</td> == $0
<td class="optionscol attribute_pa_scaled_weight" data-
sort-value="168.1g" data-label="Scaled Weight">168.1g</td>
<td class="optionscol attribute_pa_flatness" data-sort-
value="(6) - Pretty Domey" data-label="Flatness">(6) -
Pretty Domey</td>
```



```
listed_weight = soup.findAll(name='td', attrs={'class' : 'optionscol attribute_pa_weight'})
print(listed_weight)
```

```
[<td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" da
ta-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">1
73-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_
pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value
="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="169g">169g</td>, <td class="optionscol attri
bute_pa_weight" data-label="Weight" data-sort-value="172g">172g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="1
71g">171g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="168g">168g</td>, <td class="optionscol attribute_pa_wei
ght" data-label="Weight" data-sort-value="163g">163g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="164g">164g</
td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weigh
t" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="167g">
167g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="163g">163g</td>, <td class="optionscol attribute_pa_weight"
data-label="Weight" data-sort-value="164g">164g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="164g">164g</td>,
<td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="150g">150g</td>, <td class="optionscol attribute_pa_weight" data-label
="Weight" data-sort-value="167g">167g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="168g">168g</td>, <td class
="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="166g">166g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight"
data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="147g">147g</td>, <td class="o
ptionscol attribute_pa_weight" data-label="Weight" data-sort-value="170g">170g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" dat
a-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td cla
ss="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label
="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g
</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weig
ht" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-1
75g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attr
ibute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort
-value="150g">150g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="150g">150g</td>, <td class="optionscol attribu
te_pa_weight" data-label="Weight" data-sort-value="139g">139g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173
-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol at
tribute_pa_weight" data-label="Weight" data-sort-value="173-175g">173-175g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-so
rt-value="167g">167g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="167g">167g</td>, <td class="optionscol attri
bute_pa_weight" data-label="Weight" data-sort-value="172g">172g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="1
50g">150g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="150g">150g</td>, <td class="optionscol attribute_pa_wei
ght" data-label="Weight" data-sort-value="172g">172g</td>, <td class="optionscol attribute_pa_weight" data-label="Weight" data-sort-value="172g">172g</
```

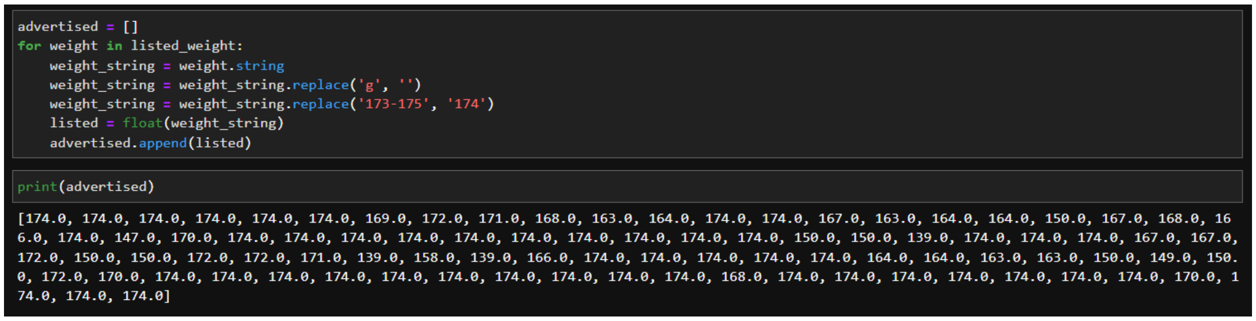
We can see there is still a large amount of clutter with our desired data. We only want the strings that actually contains the weights, and we can view this with a simple for loop, and the ".string" command.



```
for weight in listed_weight:
    print(weight.string)
```

```
173-175g
173-175g
173-175g
173-175g
173-175g
173-175g
169g
172g
171g
168g
163g
164g
173-175g
173-175g
167g
163g
164g
164g
150g
167g
168g
166g
173-175g
147g
170g
```

Voila! We now have all of the advertised weights we are after. However, we would like to be able to analyze this data, so to covert the string to a float, we will first need to remove the unit “g” and replace the interval “173-175” with 174, the average of the interval.

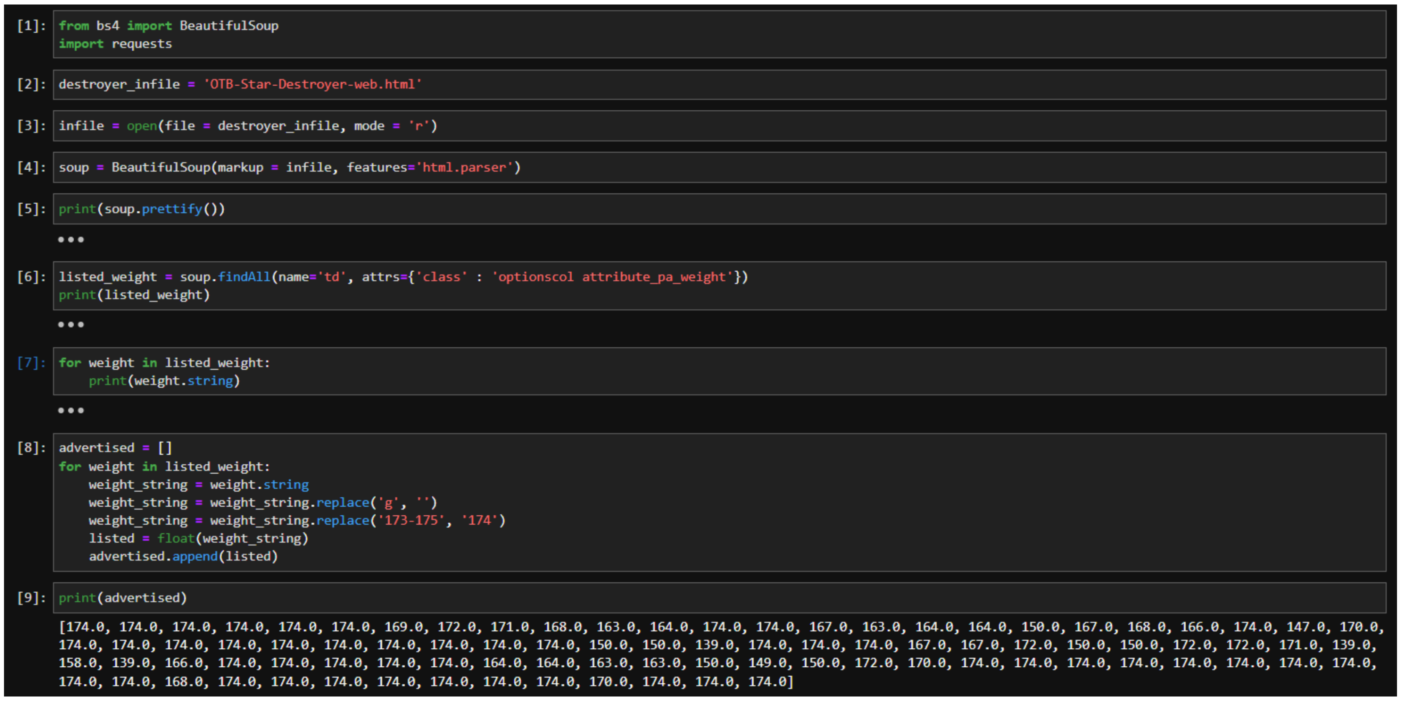


```
advertised = []
for weight in listed_weight:
    weight_string = weight.string
    weight_string = weight_string.replace('g', '')
    weight_string = weight_string.replace('173-175', '174')
    listed = float(weight_string)
    advertised.append(listed)
```

```
print(advertised)
```

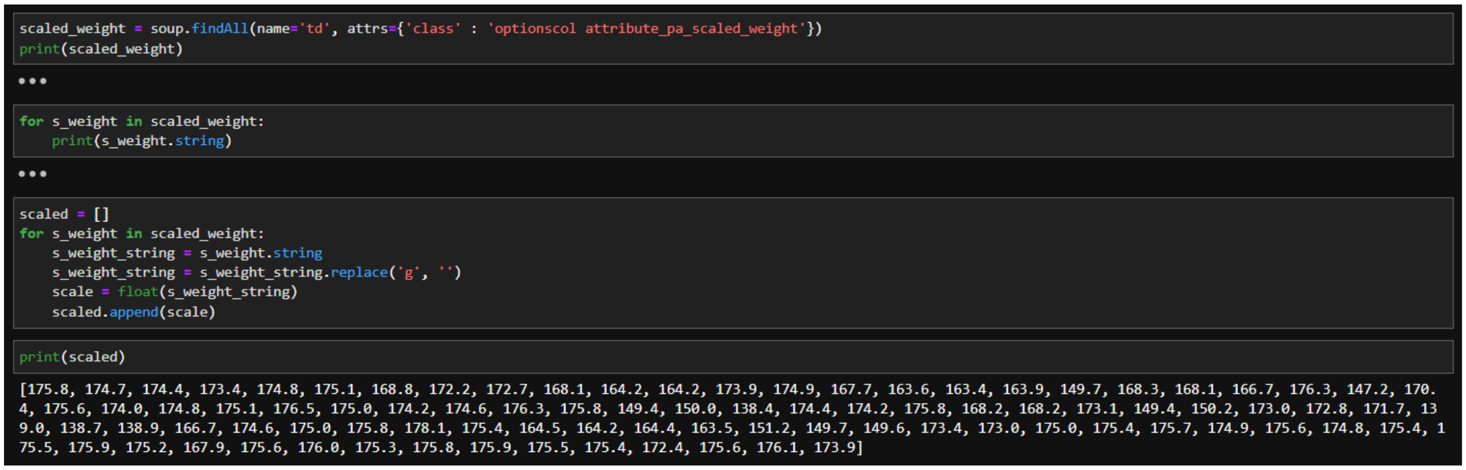
```
[174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 169.0, 172.0, 171.0, 168.0, 163.0, 164.0, 174.0, 174.0, 167.0, 163.0, 164.0, 164.0, 150.0, 167.0, 168.0, 16
6.0, 174.0, 147.0, 170.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 150.0, 150.0, 139.0, 174.0, 174.0, 174.0, 167.0, 167.0,
172.0, 150.0, 150.0, 172.0, 172.0, 171.0, 139.0, 158.0, 139.0, 166.0, 174.0, 174.0, 174.0, 174.0, 174.0, 164.0, 164.0, 163.0, 163.0, 150.0, 149.0, 150.
0, 172.0, 170.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 168.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 170.0, 1
74.0, 174.0, 174.0]
```

Finally, we have a list of the advertised weight for every single Star Destroyer for sale on Only The Best Discs! The entire code is pasted below.



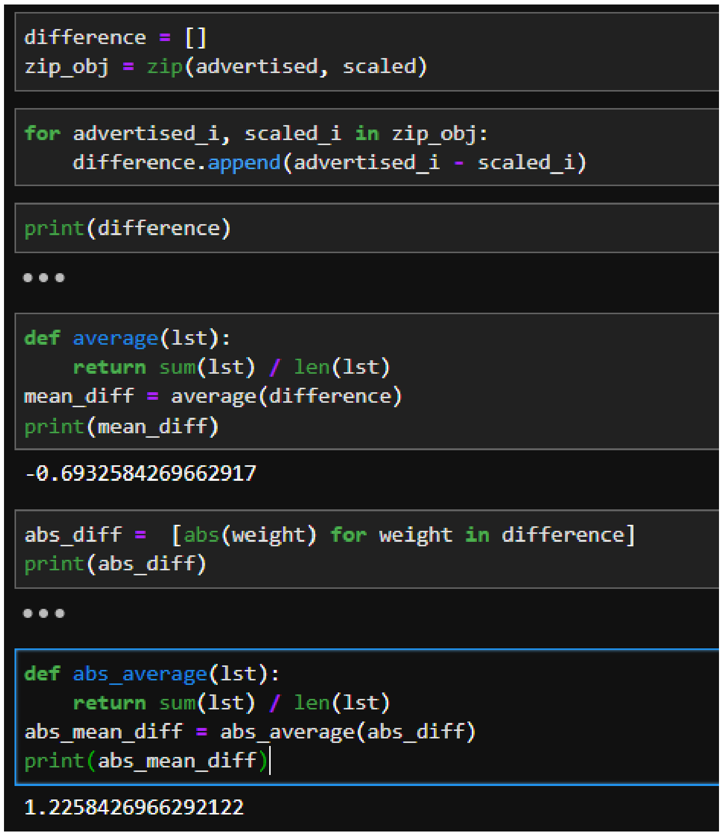
```
[1]: from bs4 import BeautifulSoup
     import requests

[2]: destroyer_infile = 'OTB-Star-Destroyer-web.html'

[3]: infile = open(file = destroyer_infile, mode = 'r')

[4]: soup = BeautifulSoup(markup = infile, features='html.parser')

[5]: print(soup.prettify())
     •••

[6]: listed_weight = soup.findAll(name='td', attrs={'class' : 'optionscol attribute_pa_weight'})
     print(listed_weight)
     •••

[7]: for weight in listed_weight:
         print(weight.string)
     •••

[8]: advertised = []
     for weight in listed_weight:
         weight_string = weight.string
         weight_string = weight_string.replace('g', '')
         weight_string = weight_string.replace('173-175', '174')
         listed = float(weight_string)
         advertised.append(listed)

[9]: print(advertised)
     [174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 169.0, 172.0, 171.0, 168.0, 163.0, 164.0, 174.0, 174.0, 167.0, 163.0, 164.0, 164.0, 150.0, 167.0, 168.0, 166.0, 174.0, 147.0, 170.0,
     174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 150.0, 150.0, 139.0, 174.0, 174.0, 174.0, 167.0, 167.0, 172.0, 150.0, 150.0, 172.0, 172.0, 171.0, 139.0,
     158.0, 139.0, 166.0, 174.0, 174.0, 174.0, 174.0, 174.0, 164.0, 164.0, 163.0, 163.0, 150.0, 149.0, 150.0, 172.0, 170.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0,
     174.0, 174.0, 168.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 174.0, 170.0, 174.0, 174.0, 174.0]
```

We can then use the same process to get a list of the scaled weights from the website, replacing the class of the listed weights with that of the scaled weights, which we can again determine by using inspect element on the webpage.



```
scaled_weight = soup.findAll(name='td', attrs={'class' : 'optionscol attribute_pa_scaled_weight'})
print(scaled_weight)
•••

for s_weight in scaled_weight:
    print(s_weight.string)
•••

scaled = []
for s_weight in scaled_weight:
    s_weight_string = s_weight.string
    s_weight_string = s_weight_string.replace('g', '')
    scale = float(s_weight_string)
    scaled.append(scale)

print(scaled)
[175.8, 174.7, 174.4, 173.4, 174.8, 175.1, 168.8, 172.2, 172.7, 168.1, 164.2, 164.2, 173.9, 174.9, 167.7, 163.6, 163.4, 163.9, 149.7, 168.3, 168.1, 166.7, 176.3, 147.2, 170.
4, 175.6, 174.0, 174.8, 175.1, 176.5, 175.0, 174.2, 174.6, 176.3, 175.8, 149.4, 150.0, 138.4, 174.4, 174.2, 175.8, 168.2, 168.2, 173.1, 149.4, 150.2, 173.0, 172.8, 171.7, 13
9.0, 138.7, 138.9, 166.7, 174.6, 175.0, 175.8, 178.1, 175.4, 164.5, 164.2, 164.4, 163.5, 151.2, 149.7, 149.6, 173.4, 173.0, 175.0, 175.4, 175.7, 174.9, 175.6, 174.8, 175.4, 1
75.5, 175.9, 175.2, 167.9, 175.6, 176.0, 175.3, 175.8, 175.9, 175.5, 175.4, 172.4, 175.6, 176.1, 173.9]
```

Now that we have pulled all the data we are after into our two lists, we can create a list of the differences between these weights. Using this, we can easily calculate the mean difference between the advertised and scaled weights, as well as the average absolute value between these weights.



```
difference = []
zip_obj = zip(advertised, scaled)
```

```
for advertised_i, scaled_i in zip_obj:
    difference.append(advertised_i - scaled_i)
```

```
print(difference)
```

```
•••
```

```
def average(lst):
    return sum(lst) / len(lst)
mean_diff = average(difference)
print(mean_diff)
```

```
-0.6932584269662917
```

```
abs_diff =  [abs(weight) for weight in difference]
print(abs_diff)
```

```
•••
```

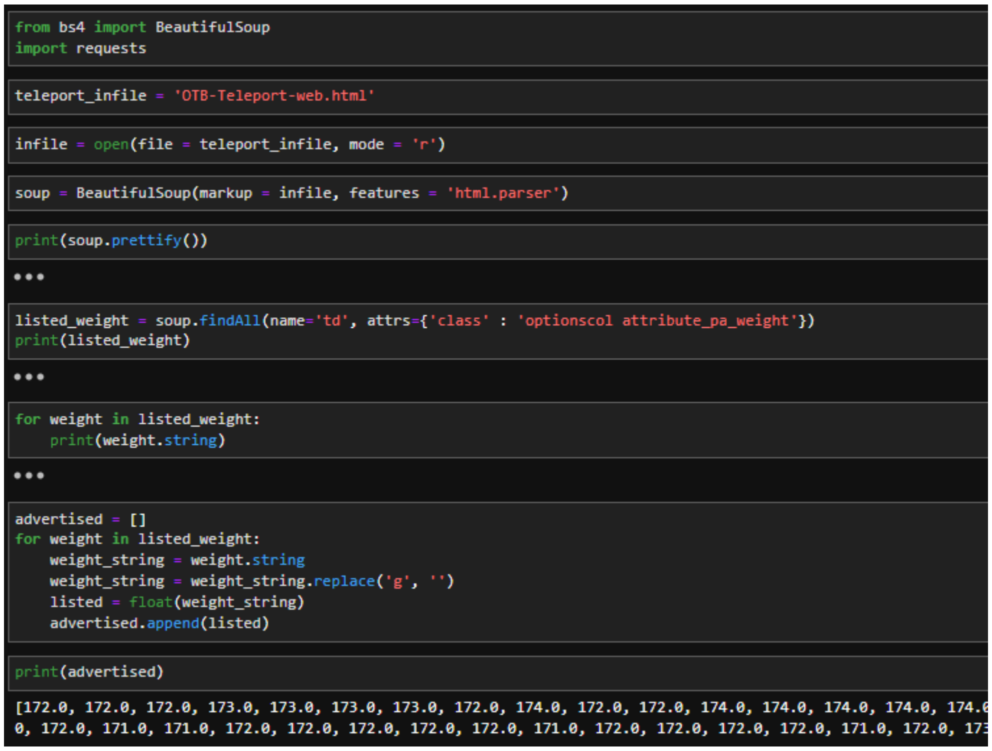
```
def abs_average(lst):
    return sum(lst) / len(lst)
abs_mean_diff = abs_average(abs_diff)
print(abs_mean_diff)
```

```
1.2258426966292122
```

We see that the average difference is about -.69, meaning that the average scaled weight is over 2/3 of a gram heavier than the weight the disc is advertised at! The average absolute value difference in weight is 1.23 gram. This is significant! Being off by over a gram is fairly drastic, as this means that there is almost a 1% difference on average between the weight Innova claims to have manufactured each disc at, and the actual, measured weight.

MVP Teleport Consistency Analysis

MVP prides themselves on being one of the most consistent disc manufacturers out there, and they take rigorous measures to ensure quality control. Using the same methods as before, we will repeat the process to get the average and absolute average difference in advertised versus measured weight. The full code can be seen below.



```
from bs4 import BeautifulSoup
import requests
```

```
teleport_infile = 'OTB-Teleport-web.html'
```

```
infile = open(file = teleport_infile, mode = 'r')
```

```
soup = BeautifulSoup(markup = infile, features = 'html.parser')
```

```
print(soup.prettify())
```

```
•••
```

```
listed_weight = soup.findAll(name='td', attrs={'class' : 'optionscol attribute_pa_weight'})
print(listed_weight)
```

```
•••
```

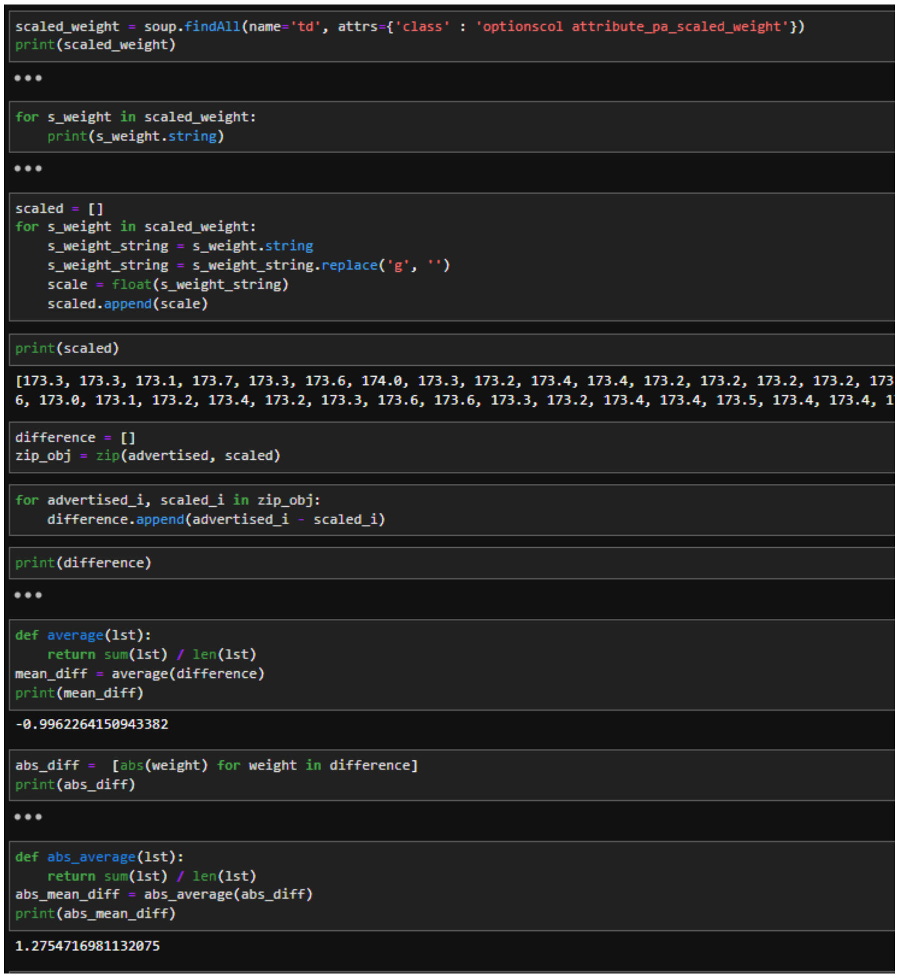
```
for weight in listed_weight:
    print(weight.string)
```

```
•••
```

```
advertised = []
for weight in listed_weight:
    weight_string = weight.string
    weight_string = weight_string.replace('g', '')
    listed = float(weight_string)
    advertised.append(listed)
```

```
print(advertised)
```

```
[172.0, 172.0, 172.0, 173.0, 173.0, 173.0, 173.0, 172.0, 174.0, 172.0, 172.0, 174.0, 174.0, 174.0, 174.0, 174.0
0, 172.0, 171.0, 171.0, 172.0, 172.0, 172.0, 172.0, 172.0, 171.0, 172.0, 172.0, 172.0, 172.0, 171.0, 172.0, 173
```



```
scaled_weight = soup.findAll(name='td', attrs={'class' : 'optionscol attribute_pa_scaled_weight'})
print(scaled_weight)
```

```
•••
```

```
for s_weight in scaled_weight:
    print(s_weight.string)
```

```
•••
```

```
scaled = []
for s_weight in scaled_weight:
    s_weight_string = s_weight.string
    s_weight_string = s_weight_string.replace('g', '')
    scale = float(s_weight_string)
    scaled.append(scale)
```

```
print(scaled)
```

```
[173.3, 173.3, 173.1, 173.7, 173.3, 173.6, 174.0, 173.3, 173.2, 173.4, 173.4, 173.2, 173.2, 173.2, 173.2, 173
6, 173.0, 173.1, 173.2, 173.4, 173.2, 173.3, 173.6, 173.6, 173.3, 173.2, 173.4, 173.4, 173.5, 173.4, 173.4, 1
```

```
difference = []
zip_obj = zip(advertised, scaled)
```

```
for advertised_i, scaled_i in zip_obj:
    difference.append(advertised_i - scaled_i)
```

```
print(difference)
```

```
•••
```

```
def average(lst):
    return sum(lst) / len(lst)
mean_diff = average(difference)
print(mean_diff)
```

```
-0.9962264150943382
```

```
abs_diff =  [abs(weight) for weight in difference]
print(abs_diff)
```

```
•••
```

```
def abs_average(lst):
    return sum(lst) / len(lst)
abs_mean_diff = abs_average(abs_diff)
print(abs_mean_diff)
```

```
1.2754716981132075
```

Looking at the code above, we can really start to appreciate the benefits of web scraping in python. The code we wrote for the Star Destroyer is almost entirely repeatable! The only major difference is the file we are loading in, and we easily swap out the Star Destroyer webpage with that of the MVP Teleport. Since this webpage also comes from Only The Best Discs, we don't even have to worry about changing the class of the objects we are after (though it is quite straightforward to inspect the element and get the necessary class for any website we would like to scrape). The for loop for converting the string elements to floats will be the only other key piece to change if scraping other websites, but the methodology boils down to removing all non-numeric characters from the strings, and then simply changing the type to float (or other necessary type).

Innova vs MVP: King of Consistency

Looking at our scraped MVP Teleport data, we can see that the average difference in listed and scaled weight is even greater than that of the Star Destroyer! The average measured weight of each Teleport disc is almost a full gram (0.996 g) heavier than what it was advertised at! This is significantly larger than the Star Destroyer average difference of 0.69 g. The absolute difference is similar between the two discs, but the Destroyer is again more consistent! The average absolute difference for the MVP Teleport is 1.275 grams, and the Innova Star Destroyer is a touch better at 1.226 grams.

This is a rather surprising result, as mentioned previously, MVP takes great pride in their superior production methods, and their scientific approach to ensuring consistent quality. But, while anyone can claim to have the best or most consistent, we know it takes data to back it up! Scraping all the available data from the flagship mold of each brand gives us a good basis to compare the brands as a whole. From the data scraped in this tutorial, we can claim that not only is Innova on par with the consistency of MVP's "superior" production methods, they may actually be even better! This is almost reassuring to discover. Innova is currently the largest disc manufacturer in the world, and the Star Destroyer is the most popular disc of all time. You would hope that Innova shouldn't make it to the top with sub-par quality!

In this tutorial, we scraped web data to make an argument for who the most consistent disc manufacturer is, with Innova coming out ahead of MVP. This is a very specific instance of web scraping, in the hope that following along with this example will give a solid base understanding of scraping and manipulating any data the vast world wide web has to offer. With the rate that the internet is growing, the importance of acquiring meaningful data from it will only increase.