Beautiful Soup

DANIELLE ZICKL

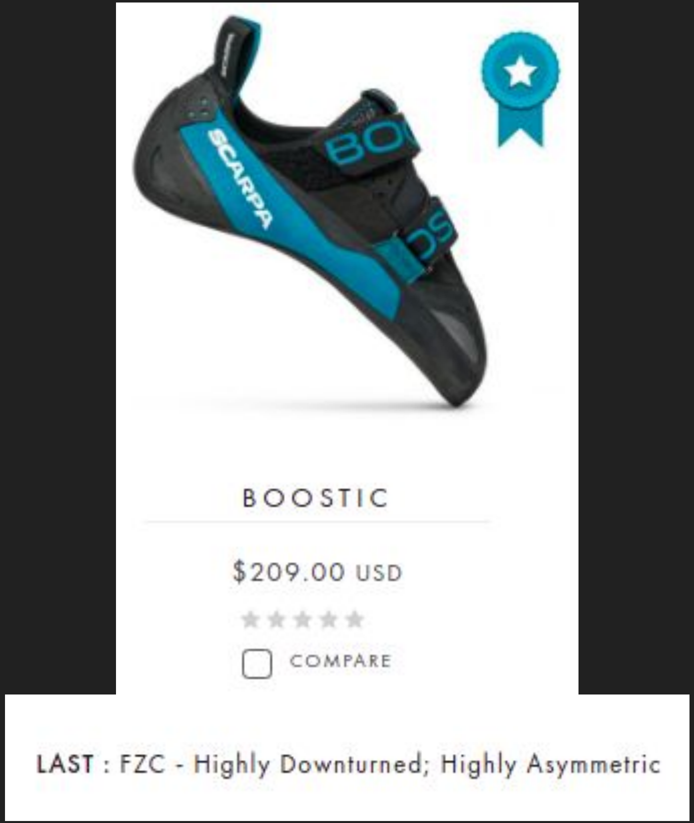
(a python library, not actual soup)

Problem

Let's say we want to buy a pair of climbing shoes that fit well. There are shoe molds (lasts) with the same fit on different models. It'd sure be nice to have a table of these lasts. Luckily, they post that on their website. Unfortunately, you have to visit each shoe's page separately to see this information.

The existing charts (next slide) show how shoe volume is related, but it'd be nice to have something more comprehensive.

Scarpa.com


BOOSTIC

$209.00 USD

COMPARE

LAST : FZC - Highly Downturned; Highly Asymmetric

La Sportiva

SEGMENTATION

NARROW FOOT

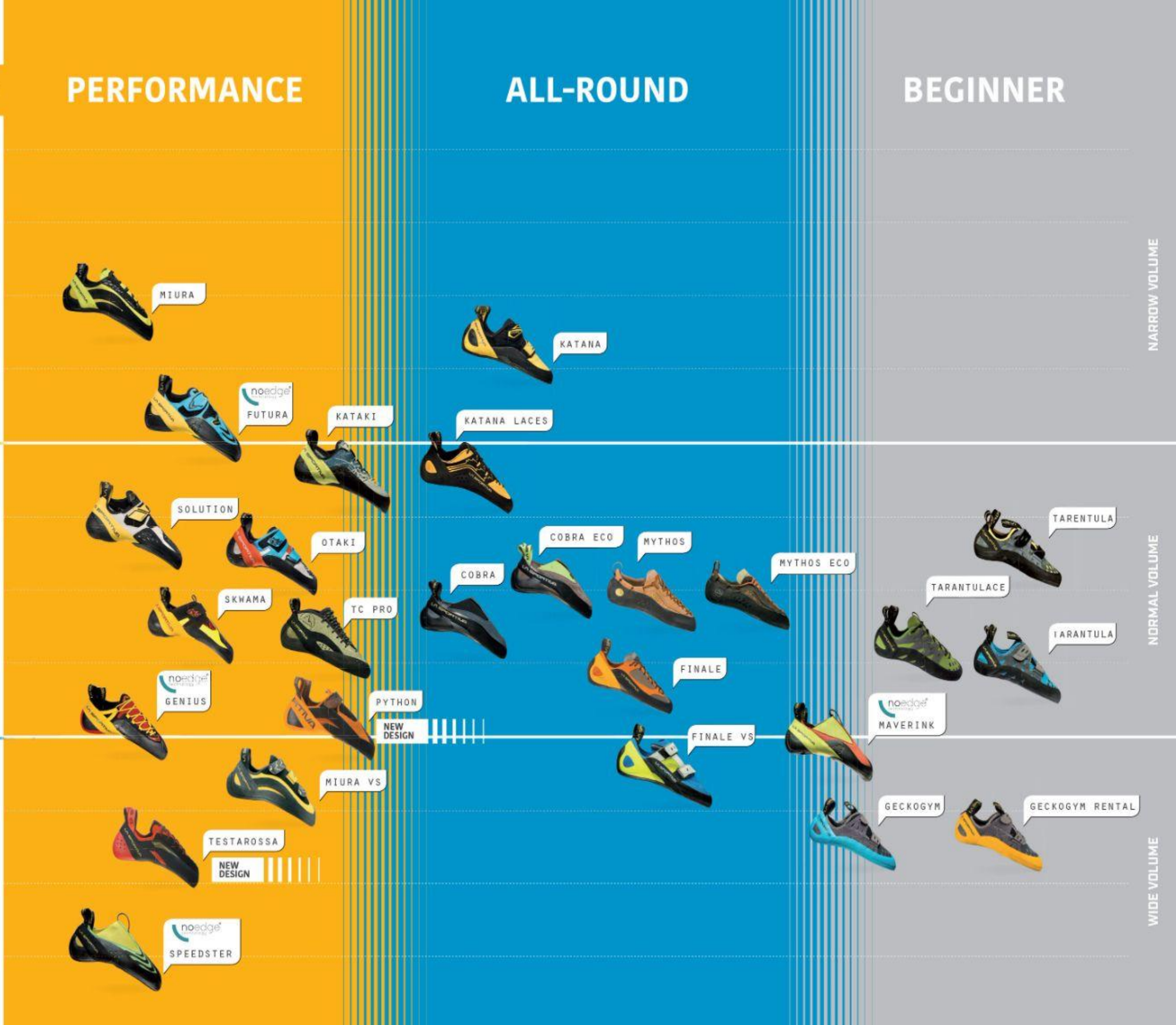
Shoes designed for narrow feet with toes "sloping" from the big to the little toe, giving the foot a pointed shape and a relatively flat instep.

MEDIUM FOOT

Shoes designed for "normal" feet with a neutral instep, neither narrow nor wide. These shoes are ideal both for narrow and wide feet.

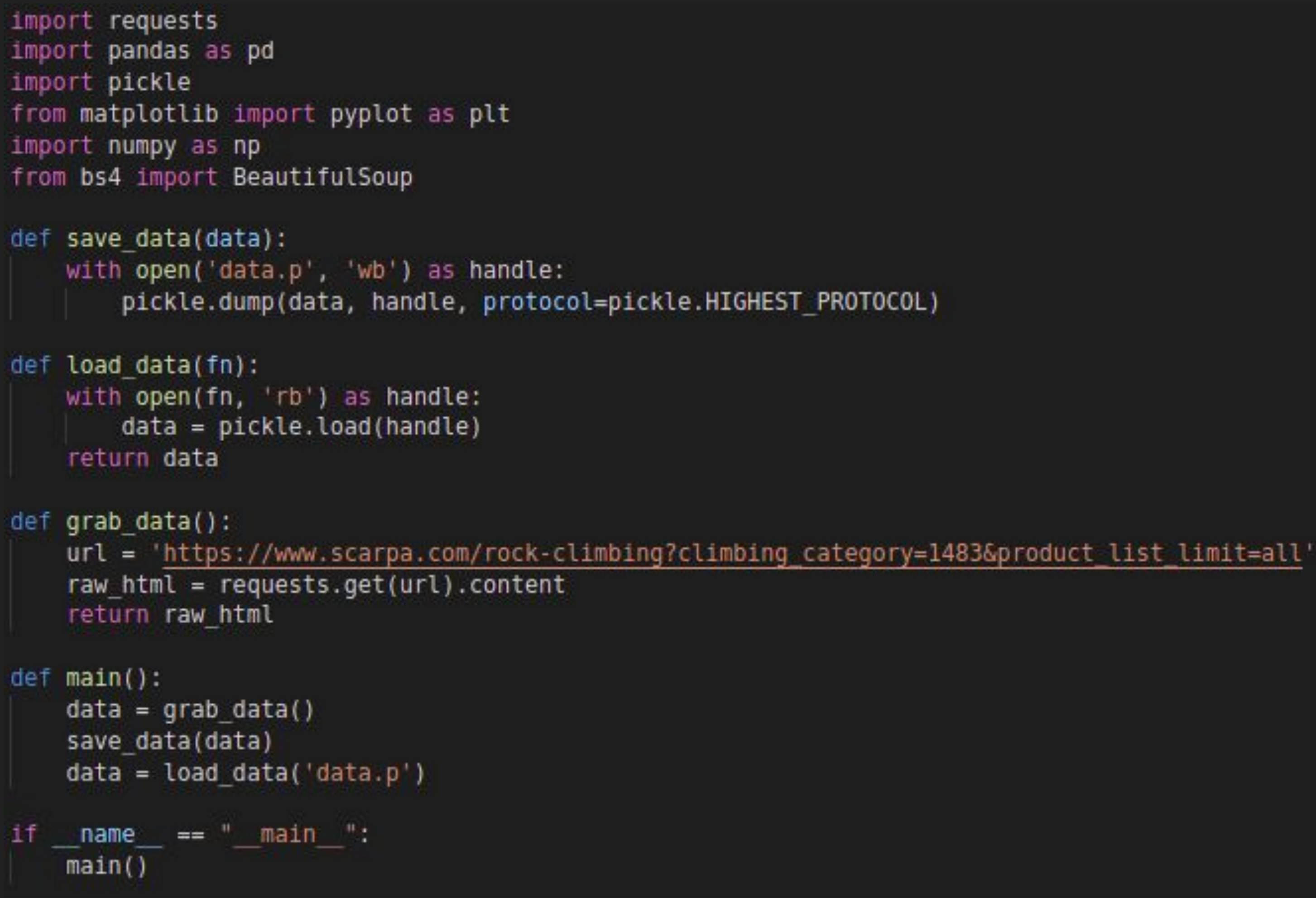
WIDE FOOT

Shoes designed for wider feet, with toes do not "slope", giving a less pointed shape to the foot but with a more pronounced instep.



Solution! 1/6

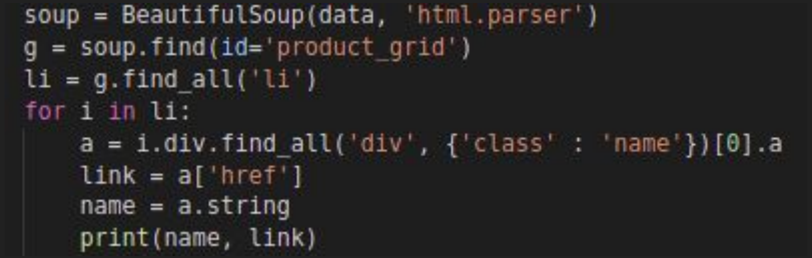
Let’s try developing a web scraper to create this table. First we import relevant libraries and grab the raw html of their shoe listing page then save to a pickle file.



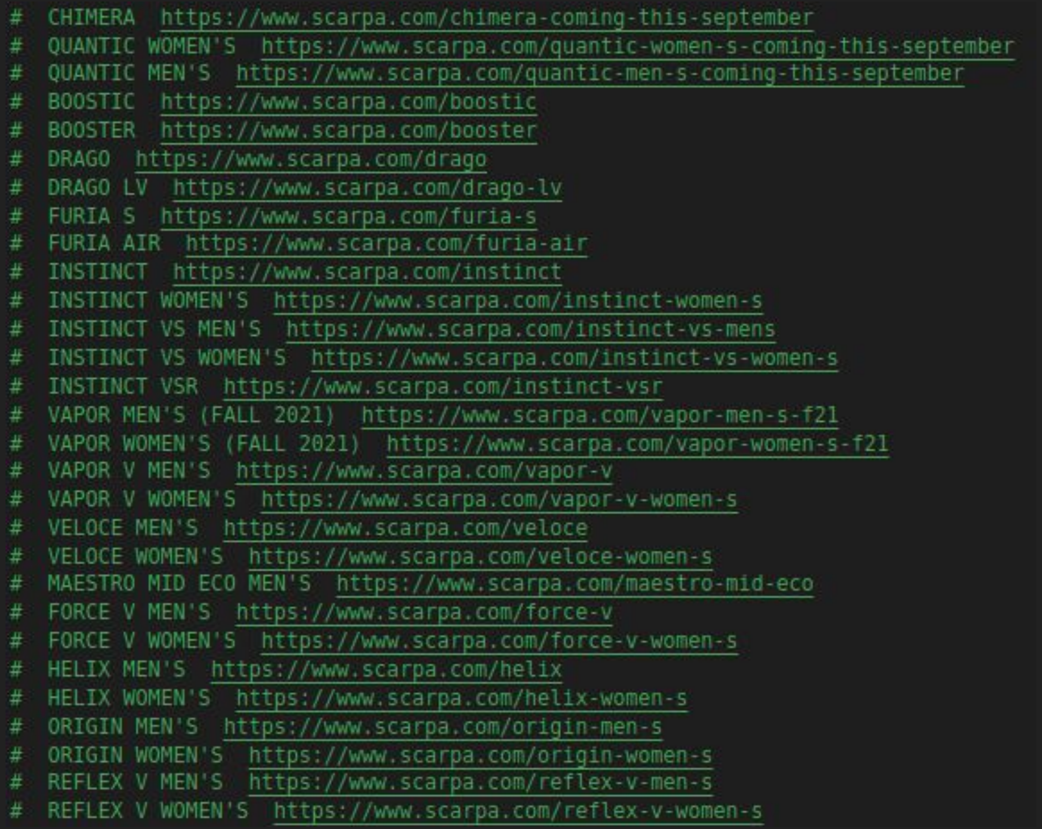
```python
import requests
import pandas as pd
import pickle
from matplotlib import pyplot as plt
import numpy as np
from bs4 import BeautifulSoup

def save_data(data):
    with open('data.p', 'wb') as handle:
        pickle.dump(data, handle, protocol=pickle.HIGHEST_PROTOCOL)

def load_data(fn):
    with open(fn, 'rb') as handle:
        data = pickle.load(handle)
    return data

def grab_data():
    url = 'https://www.scarpa.com/rock-climbing?climbing_category=1483&product_list_limit=all'
    raw_html = requests.get(url).content
    return raw_html

def main():
    data = grab_data()
    save_data(data)
    data = load_data('data.p')

if __name__ == "__main__":
    main()
```

Solution! 2/6

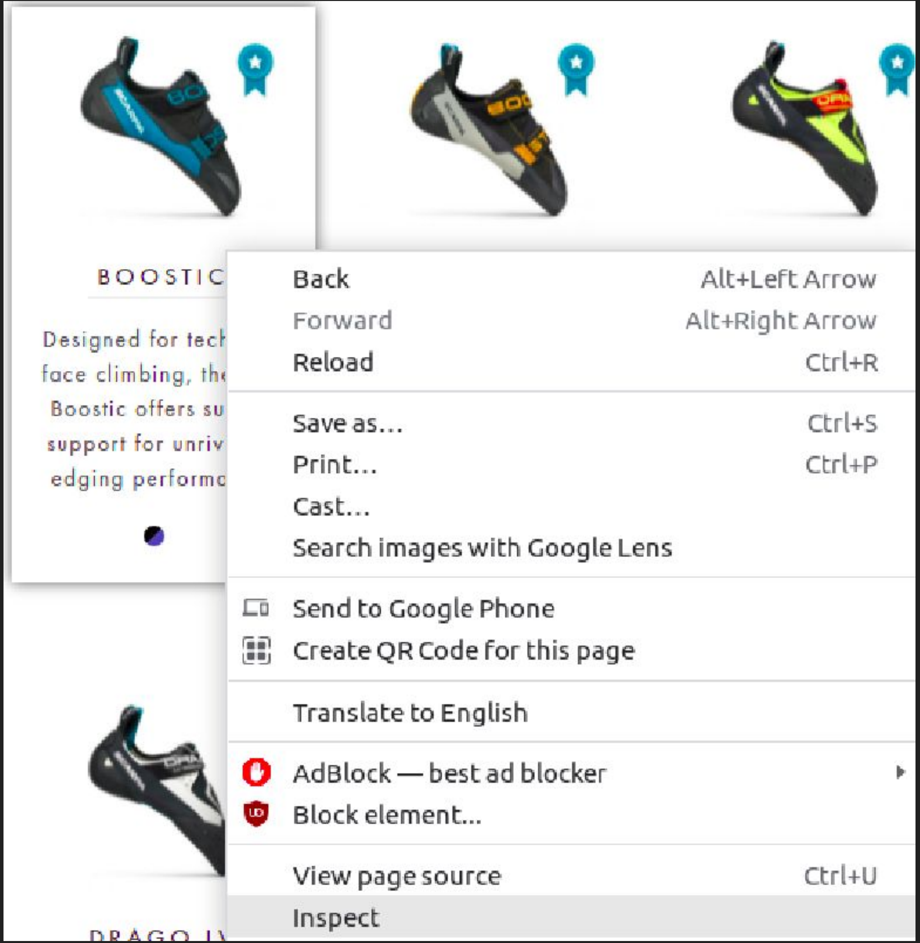
Now, lets process the data to find the names of the shoes and their corresponding shoe page link.

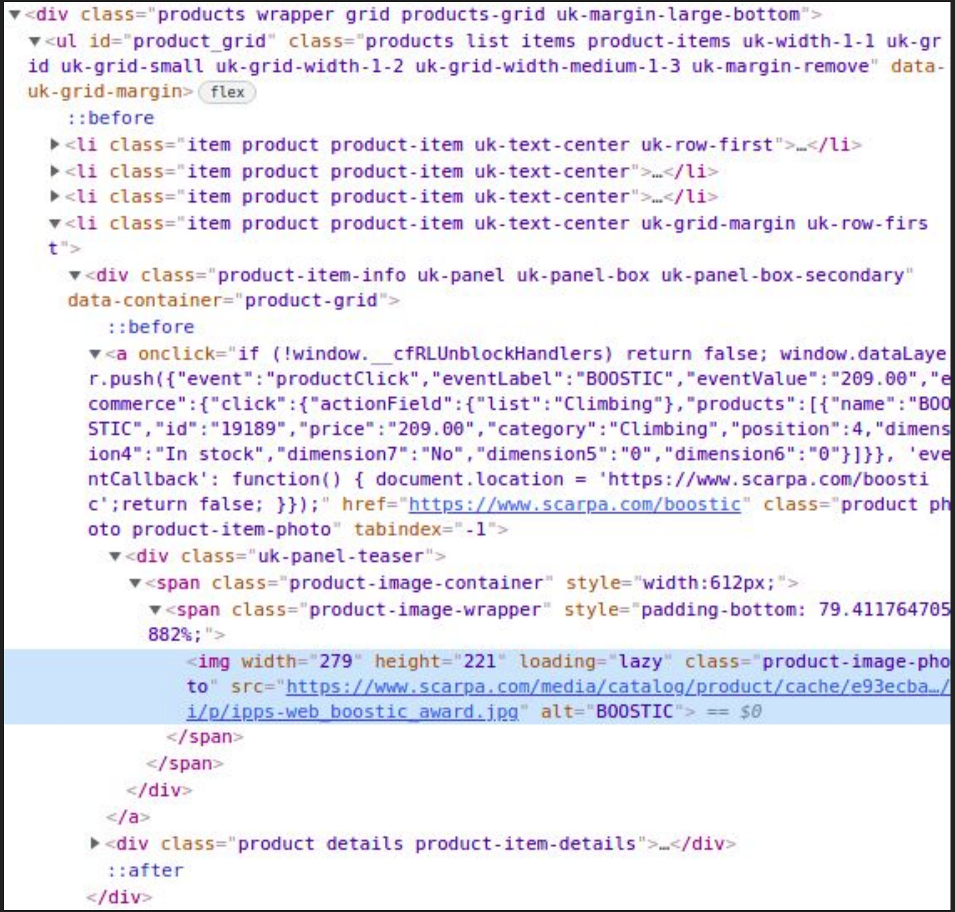


```python
soup = BeautifulSoup(data, 'html.parser')
g = soup.find(id='product_grid')
li = g.find_all('li')
for i in li:
    a = i.div.find_all('div', {'class' : 'name'})[0].a
    link = a['href']
    name = a.string
    print(name, link)
```



```
#  CHIMERA  https://www.scarpa.com/chimera-coming-this-september
#  QUANTIC WOMEN'S  https://www.scarpa.com/quantic-women-s-coming-this-september
#  QUANTIC MEN'S  https://www.scarpa.com/quantic-men-s-coming-this-september
#  BOOSTIC  https://www.scarpa.com/boostic
#  BOOSTER  https://www.scarpa.com/booster
#  DRAGO  https://www.scarpa.com/drago
#  DRAGO LV  https://www.scarpa.com/drago-lv
#  FURIA S  https://www.scarpa.com/furia-s
#  FURIA AIR  https://www.scarpa.com/furia-air
#  INSTINCT  https://www.scarpa.com/instinct
#  INSTINCT WOMEN'S  https://www.scarpa.com/instinct-women-s
#  INSTINCT VS MEN'S  https://www.scarpa.com/instinct-vs-mens
#  INSTINCT VS WOMEN'S  https://www.scarpa.com/instinct-vs-women-s
#  INSTINCT VSR  https://www.scarpa.com/instinct-vsr
#  VAPOR MEN'S (FALL 2021)  https://www.scarpa.com/vapor-men-s-f21
#  VAPOR WOMEN'S (FALL 2021)  https://www.scarpa.com/vapor-women-s-f21
#  VAPOR V MEN'S  https://www.scarpa.com/vapor-v
#  VAPOR V WOMEN'S  https://www.scarpa.com/vapor-v-women-s
#  VELOCE MEN'S  https://www.scarpa.com/veloce
#  VELOCE WOMEN'S  https://www.scarpa.com/veloce-women-s
#  MAESTRO MID ECO MEN'S  https://www.scarpa.com/maestro-mid-eco
#  FORCE V MEN'S  https://www.scarpa.com/force-v
#  FORCE V WOMEN'S  https://www.scarpa.com/force-v-women-s
#  HELIX MEN'S  https://www.scarpa.com/helix
#  HELIX WOMEN'S  https://www.scarpa.com/helix-women-s
#  ORIGIN MEN'S  https://www.scarpa.com/origin-men-s
#  ORIGIN WOMEN'S  https://www.scarpa.com/origin-women-s
#  REFLEX V MEN'S  https://www.scarpa.com/reflex-v-men-s
#  REFLEX V WOMEN'S  https://www.scarpa.com/reflex-v-women-s
```

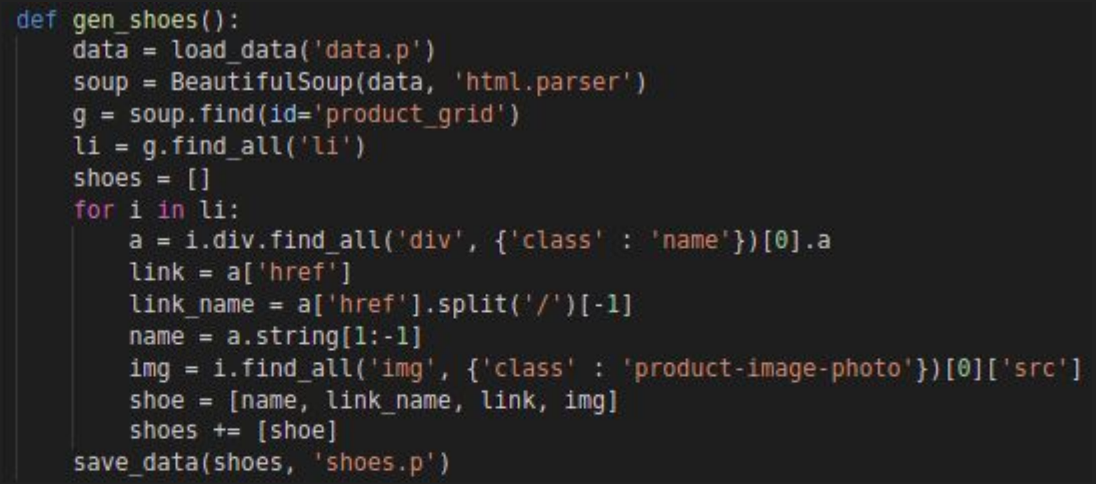




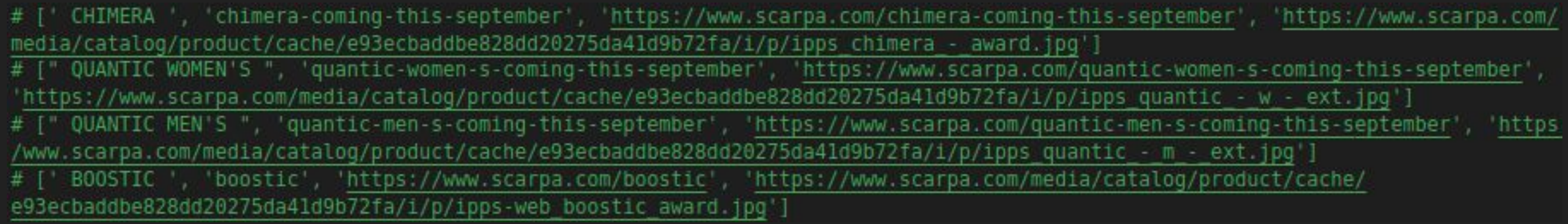
```
▼<div class="products wrapper grid products-grid uk-margin-large-bottom">
  ▼<ul id="product_grid" class="products list items product-items uk-width-1-1 uk-gr
  id uk-grid-small uk-grid-width-1-2 uk-grid-width-medium-1-3 uk-margin-remove" data-
  uk-grid-margin> flex
      ::before
    ▶<li class="item product product-item uk-text-center uk-row-first">…</li>
    ▶<li class="item product product-item uk-text-center">…</li>
    ▶<li class="item product product-item uk-text-center">…</li>
    ▼<li class="item product product-item uk-text-center uk-grid-margin uk-row-firs
      t">
      ▼<div class="product-item-info uk-panel uk-panel-box uk-panel-box-secondary"
       data-container="product-grid">
          ::before
        ▼<a onclick="if (!window.__cfRLUnblockHandlers) return false; window.dataLaye
         r.push({"event":"productClick","eventLabel":"BOOSTIC","eventValue":"209.00","e
         commerce":{"click":{"actionField":{"list":"Climbing"},"products":[{"name":"BOO
         STIC","id":"19189","price":"209.00","category":"Climbing","position":4,"dimens
         ion4":"In stock","dimension7":"No","dimension5":"0","dimension6":"0"}]}}, 'eve
         ntCallback': function() { document.location = 'https://www.scarpa.com/boosti
         c';return false; }});" href="https://www.scarpa.com/boostic" class="product ph
         oto product-item-photo" tabindex="-1">
          ▼<div class="uk-panel-teaser">
            ▼<span class="product-image-container" style="width:612px;">
              ▼<span class="product-image-wrapper" style="padding-bottom: 79.411764705
               882%;">
                  <img width="279" height="221" loading="lazy" class="product-image-pho
                  to" src="https://www.scarpa.com/media/catalog/product/cache/e93ecba…/
                  i/p/ipps-web_boostic_award.jpg" alt="BOOSTIC"> == $0
                </span>
              </span>
            </div>
          </a>
        ▶<div class="product details product-item-details">…</div>
          ::after
        </div>
```

Solution! 3/6

We might need the link name and the image link, saving it as a pickle could come in handy so lets do that for now.



```python
def gen_shoes():
    data = load_data('data.p')
    soup = BeautifulSoup(data, 'html.parser')
    g = soup.find(id='product_grid')
    li = g.find_all('li')
    shoes = []
    for i in li:
        a = i.div.find_all('div', {'class' : 'name'})[0].a
        link = a['href']
        link_name = a['href'].split('/')[-1]
        name = a.string[1:-1]
        img = i.find_all('img', {'class' : 'product-image-photo'})[0]['src']
        shoe = [name, link_name, link, img]
        shoes += [shoe]
    save_data(shoes, 'shoes.p')
```

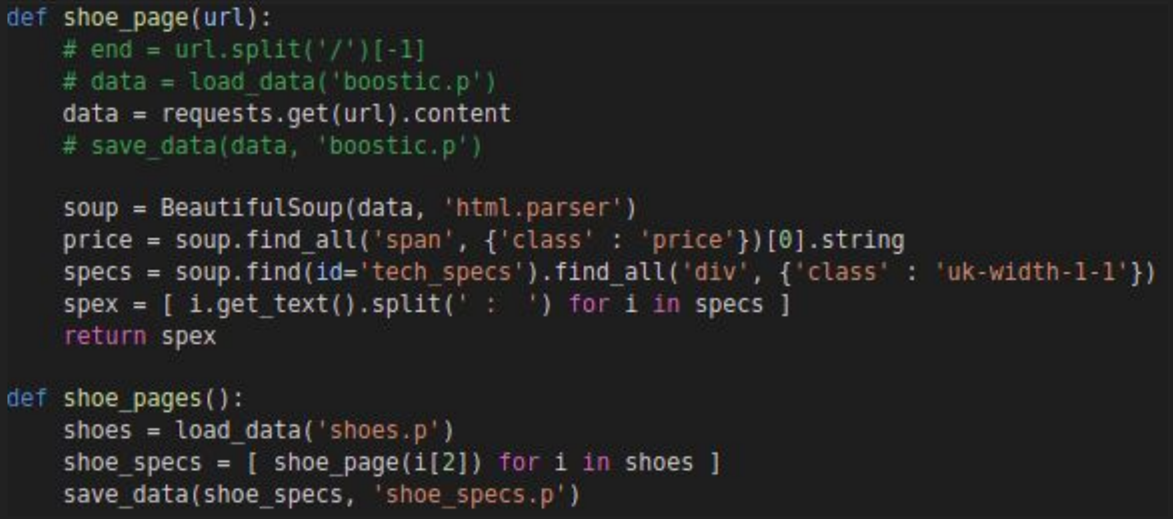


```
# [' CHIMERA ', 'chimera-coming-this-september', 'https://www.scarpa.com/chimera-coming-this-september', 'https://www.scarpa.com/
media/catalog/product/cache/e93ecbaddbe828dd20275da41d9b72fa/i/p/ipps_chimera_-_award.jpg']
# [" QUANTIC WOMEN'S ", 'quantic-women-s-coming-this-september', 'https://www.scarpa.com/quantic-women-s-coming-this-september',
'https://www.scarpa.com/media/catalog/product/cache/e93ecbaddbe828dd20275da41d9b72fa/i/p/ipps_quantic_-_w_-_ext.jpg']
# [" QUANTIC MEN'S ", 'quantic-men-s-coming-this-september', 'https://www.scarpa.com/quantic-men-s-coming-this-september', 'https
/www.scarpa.com/media/catalog/product/cache/e93ecbaddbe828dd20275da41d9b72fa/i/p/ipps_quantic_-_m_-_ext.jpg']
# [' BOOSTIC ', 'boostic', 'https://www.scarpa.com/boostic', 'https://www.scarpa.com/media/catalog/product/cache/
e93ecbaddbe828dd20275da41d9b72fa/i/p/ipps-web_boostic_award.jpg']
```

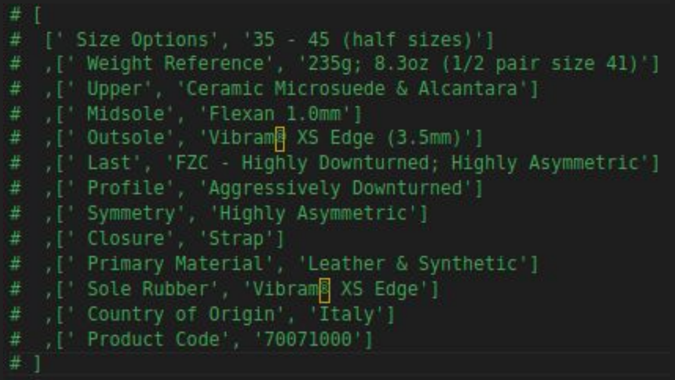
Solution! 4/6

We can now grab the data from one of the URLs and save it as a pickle to load and play with. We can then find the technical specifications we're after. While we're at it, let's just save everything.

Note that price isn't used.



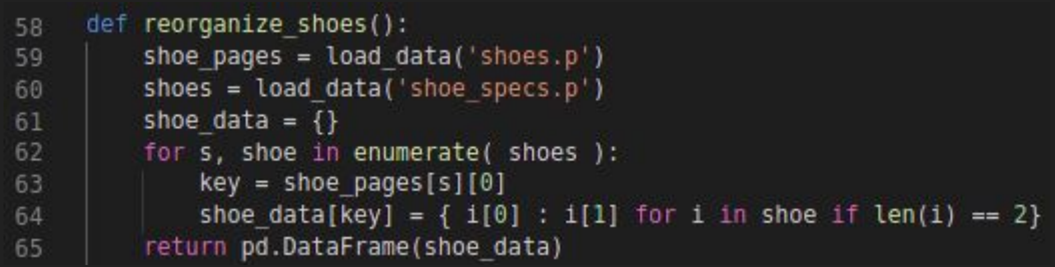
```
def shoe_page(url):
    # end = url.split('/')[-1]
    # data = load_data('boostic.p')
    data = requests.get(url).content
    # save_data(data, 'boostic.p')

    soup = BeautifulSoup(data, 'html.parser')
    price = soup.find_all('span', {'class' : 'price'})[0].string
    specs = soup.find(id='tech_specs').find_all('div', {'class' : 'uk-width-1-1'})
    spex = [ i.get_text().split(' :  ') for i in specs ]
    return spex

def shoe_pages():
    shoes = load_data('shoes.p')
    shoe_specs = [ shoe_page(i[2]) for i in shoes ]
    save_data(shoe_specs, 'shoe_specs.p')
```



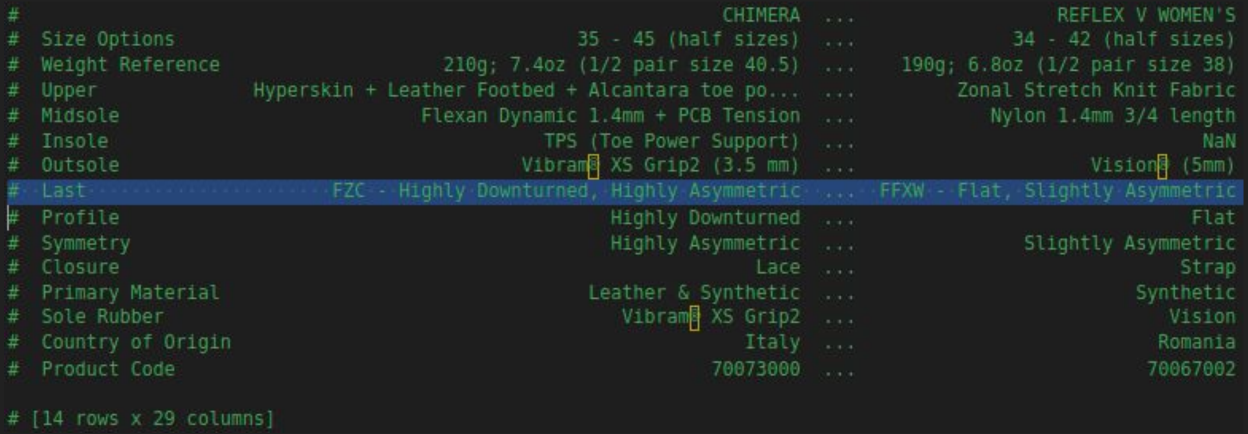
```
# [
#  [' Size Options', '35 - 45 (half sizes)']
#  ,[' Weight Reference', '235g; 8.3oz (1/2 pair size 41)']
#  ,[' Upper', 'Ceramic Microsuede & Alcantara']
#  ,[' Midsole', 'Flexan 1.0mm']
#  ,[' Outsole', 'Vibram▯ XS Edge (3.5mm)']
#  ,[' Last', 'FZC - Highly Downturned; Highly Asymmetric']
#  ,[' Profile', 'Aggressively Downturned']
#  ,[' Symmetry', 'Highly Asymmetric']
#  ,[' Closure', 'Strap']
#  ,[' Primary Material', 'Leather & Synthetic']
#  ,[' Sole Rubber', 'Vibram▯ XS Edge']
#  ,[' Country of Origin', 'Italy']
#  ,[' Product Code', '70071000']
# ]
```

Solution! 5/6

What we saved was a bit of a mess, so let’s clean it up. This converts the array of shoe data into a dictionary of dictionaries. Line 64 is a dictionary comprehension. Then, we use pandas.DataFrame to convert it to the dataframe you see to the right.



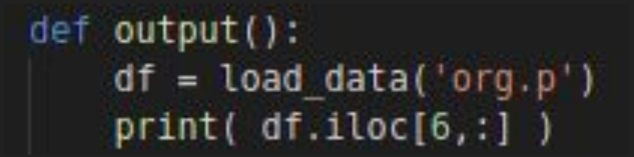
```python
58  def reorganize_shoes():
59      shoe_pages = load_data('shoes.p')
60      shoes = load_data('shoe_specs.p')
61      shoe_data = {}
62      for s, shoe in enumerate( shoes ):
63          key = shoe_pages[s][0]
64          shoe_data[key] = { i[0] : i[1] for i in shoe if len(i) == 2}
65      return pd.DataFrame(shoe_data)
```



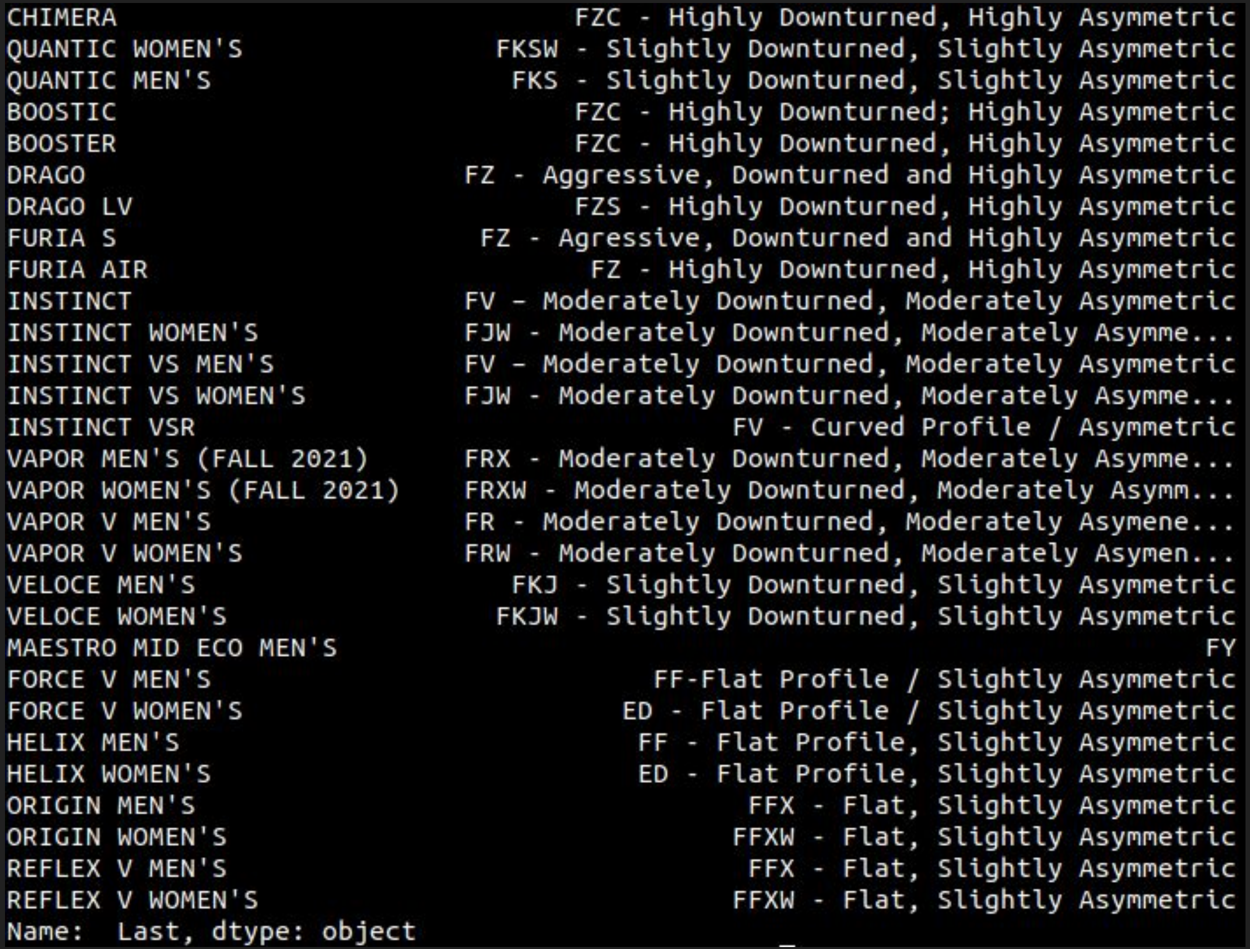
```
#                                                              CHIMERA  ...                 REFLEX V WOMEN'S
#  Size Options                                   35 - 45 (half sizes)  ...             34 - 42 (half sizes)
#  Weight Reference                   210g; 7.4oz (1/2 pair size 40.5)  ...   190g; 6.8oz (1/2 pair size 38)
#  Upper             Hyperskin + Leather Footbed + Alcantara toe po...  ...        Zonal Stretch Knit Fabric
#  Midsole                          Flexan Dynamic 1.4mm + PCB Tension  ...           Nylon 1.4mm 3/4 length
#  Insole                                      TPS (Toe Power Support)  ...                              NaN
#  Outsole                                   Vibram□ XS Grip2 (3.5 mm)  ...                    Vision□ (5mm)
#  Last                     FZC - Highly Downturned, Highly Asymmetric  ...  FFXW - Flat, Slightly Asymmetric
#  Profile                                           Highly Downturned  ...                             Flat
#  Symmetry                                          Highly Asymmetric  ...              Slightly Asymmetric
#  Closure                                                        Lace  ...                            Strap
#  Primary Material                                Leather & Synthetic  ...                        Synthetic
#  Sole Rubber                                        Vibram□ XS Grip2  ...                           Vision
#  Country of Origin                                             Italy  ...                          Romania
#  Product Code                                               70073000  ...                         70067002

# [14 rows x 29 columns]
```

Solution! 6/6

After all that work it’s finally time to print out the table of lasts. We could do other things with our dataset like making graphs.



```
def output():
    df = load_data('org.p')
    print( df.iloc[6,:] )
```



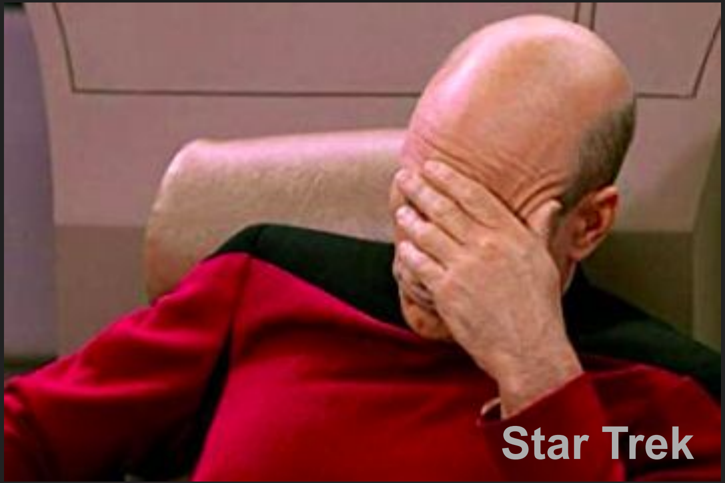
```
CHIMERA                              FZC - Highly Downturned, Highly Asymmetric
QUANTIC WOMEN'S                 FKSW - Slightly Downturned, Slightly Asymmetric
QUANTIC MEN'S                    FKS - Slightly Downturned, Slightly Asymmetric
BOOSTIC                              FZC - Highly Downturned; Highly Asymmetric
BOOSTER                              FZC - Highly Downturned, Highly Asymmetric
DRAGO                         FZ - Aggressive, Downturned and Highly Asymmetric
DRAGO LV                             FZS - Highly Downturned, Highly Asymmetric
FURIA S                        FZ - Agressive, Downturned and Highly Asymmetric
FURIA AIR                             FZ - Highly Downturned, Highly Asymmetric
INSTINCT                      FV – Moderately Downturned, Moderately Asymmetric
INSTINCT WOMEN'S              FJW - Moderately Downturned, Moderately Asymme...
INSTINCT VS MEN'S             FV – Moderately Downturned, Moderately Asymmetric
INSTINCT VS WOMEN'S           FJW - Moderately Downturned, Moderately Asymme...
INSTINCT VSR                                   FV - Curved Profile / Asymmetric
VAPOR MEN'S (FALL 2021)       FRX - Moderately Downturned, Moderately Asymme...
VAPOR WOMEN'S (FALL 2021)     FRXW - Moderately Downturned, Moderately Asymm...
VAPOR V MEN'S                 FR - Moderately Downturned, Moderately Asymene...
VAPOR V WOMEN'S               FRW - Moderately Downturned, Moderately Asymen...
VELOCE MEN'S                     FKJ - Slightly Downturned, Slightly Asymmetric
VELOCE WOMEN'S                  FKJW - Slightly Downturned, Slightly Asymmetric
MAESTRO MID ECO MEN'S                                                        FY
FORCE V MEN'S                             FF-Flat Profile / Slightly Asymmetric
FORCE V WOMEN'S                         ED - Flat Profile / Slightly Asymmetric
HELIX MEN'S                              FF - Flat Profile, Slightly Asymmetric
HELIX WOMEN'S                            ED - Flat Profile, Slightly Asymmetric
ORIGIN MEN'S                                    FFX - Flat, Slightly Asymmetric
ORIGIN WOMEN'S                                 FFXW - Flat, Slightly Asymmetric
REFLEX V MEN'S                                  FFX - Flat, Slightly Asymmetric
REFLEX V WOMEN'S                               FFXW - Flat, Slightly Asymmetric
Name:  Last, dtype: object
```

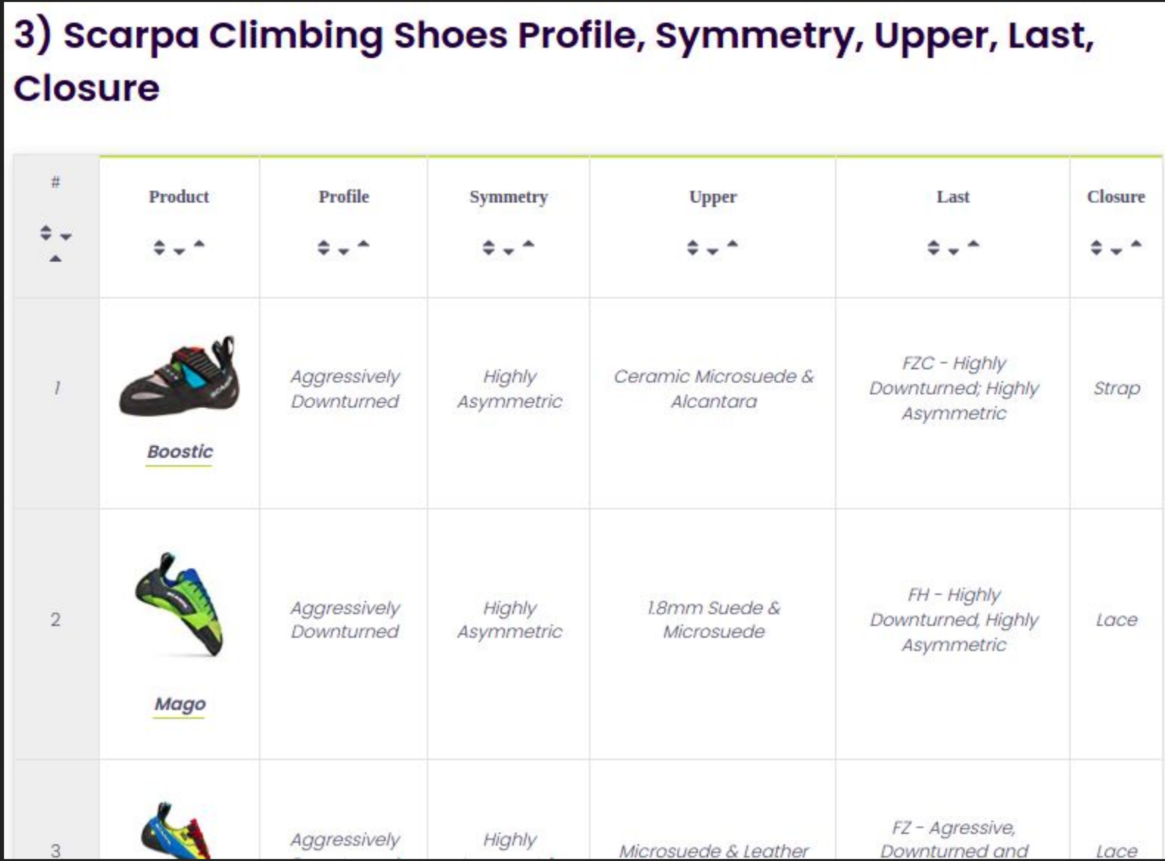
Solution 2!?

Google “Scarpa FZ FF ED”

First link: https://rappellingequipment.com/scarpa-climbing-shoes/

“33 Scarpa Climbing Shoes Compared in 3 Tables”





3) Scarpa Climbing Shoes Profile, Symmetry, Upper, Last, Closure

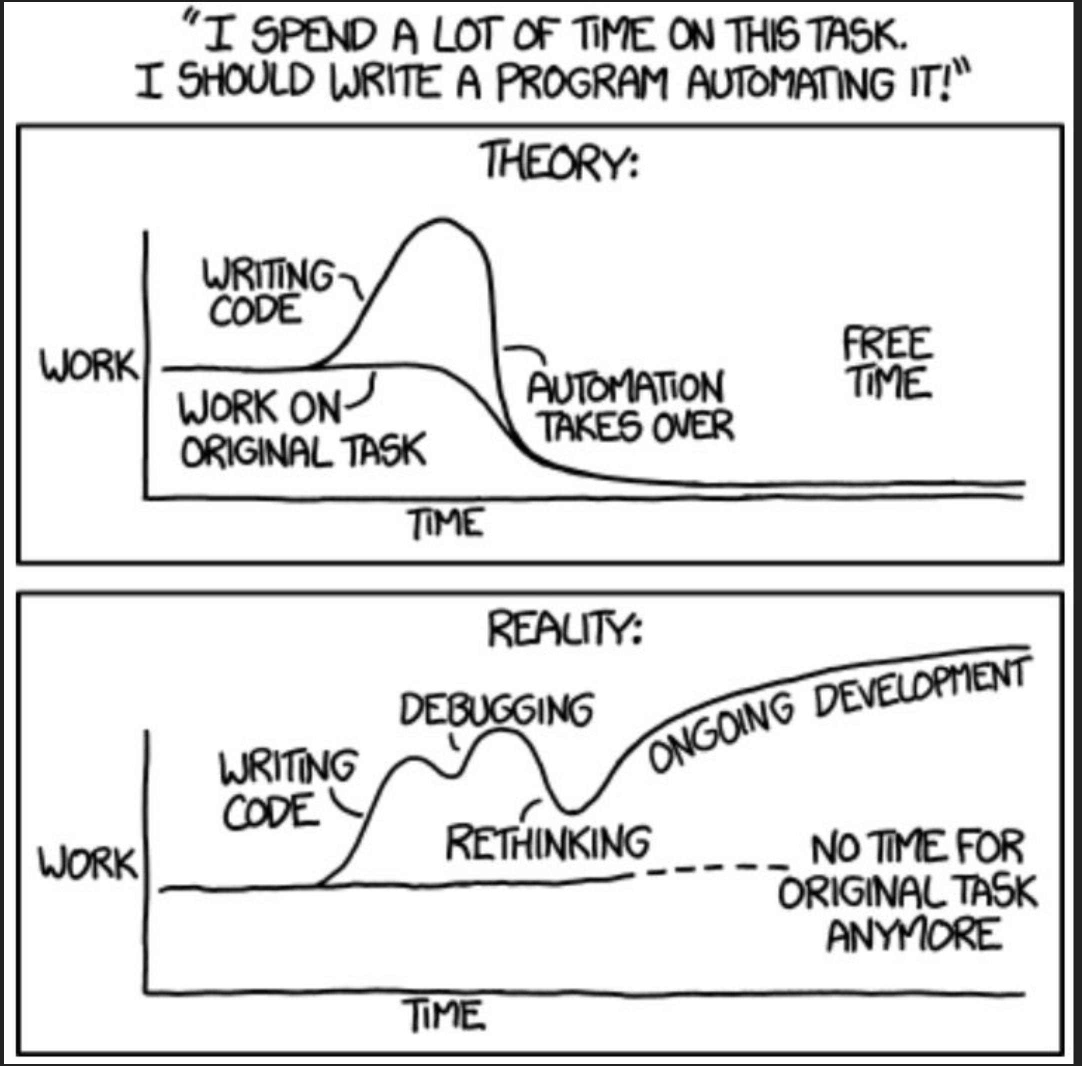
#	Product	Profile	Symmetry	Upper	Last	Closure
1	Boostic	Aggressively Downturned	Highly Asymmetric	Ceramic Microsuede & Alcantara	FZC - Highly Downturned; Highly Asymmetric	Strap
2	Mago	Aggressively Downturned	Highly Asymmetric	1.8mm Suede & Microsuede	FH - Highly Downturned, Highly Asymmetric	Lace
3		Aggressively	Highly	Microsuede & Leather	FZ - Agressive, Downturned and	Lace

Automation

https://xkcd.com/1319/

Mouseover Text: 'Automating' comes from the roots 'auto-' meaning 'self-', and 'mating', meaning 'screwing'.

Why spend 10 minutes doing something that you could spend 4 hours automating? Because, hopefully, you only have to spend that 4 hours once and sometimes that direct work would take much longer than 10 minutes.


"I SPEND A LOT OF TIME ON THIS TASK. I SHOULD WRITE A PROGRAM AUTOMATING IT!"
THEORY:
WORK
TIME
WRITING CODE
WORK ON ORIGINAL TASK
AUTOMATION TAKES OVER
FREE TIME
REALITY:
WORK
TIME
WRITING CODE
DEBUGGING
RETHINKING
ONGOING DEVELOPMENT
NO TIME FOR ORIGINAL TASK ANYMORE