

```
except Exception as e
    print(e)
    csv_file.close()
    driver.close()
    break
```

BeerMe: A Comedy of Errors

Devon Blumenthal

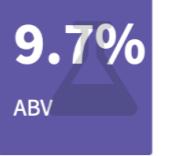
BeerMe

Show 10 entries Search:

Brewer and Beer Name	Style of Beer	Overall Score
Central 28 !Ideal	Imperial Porter	84
Birds Fly South BA Wolves in the Piano (2017)	Imperial Stout	93
Lagunitas 12th of Never Ale	Pale Ale	97
Dry Dock Brewing 2014 Bligh's Barleywine	Barley Wine	96
The Bruery So Happens It's Tuesday 2016	Barrel-Aged Stout	99
Jester King 2016 Spon Mourvedre & Sangiovese	Spontaneous Ale	100
Melvin Brewing 2x4 Double IPA	American Imperial IPA	91
Against the Grain Brewery 35K	Stout	88
South County 851 Helles	Helles	87
Brewery Ommegang Abbey Ale	Dubbel	90

Showing 1 to 10 of 992 entries Previous 1 2 3 4 5 ... 100 Next

Description
Reviews
Picture



9.7%
ABV



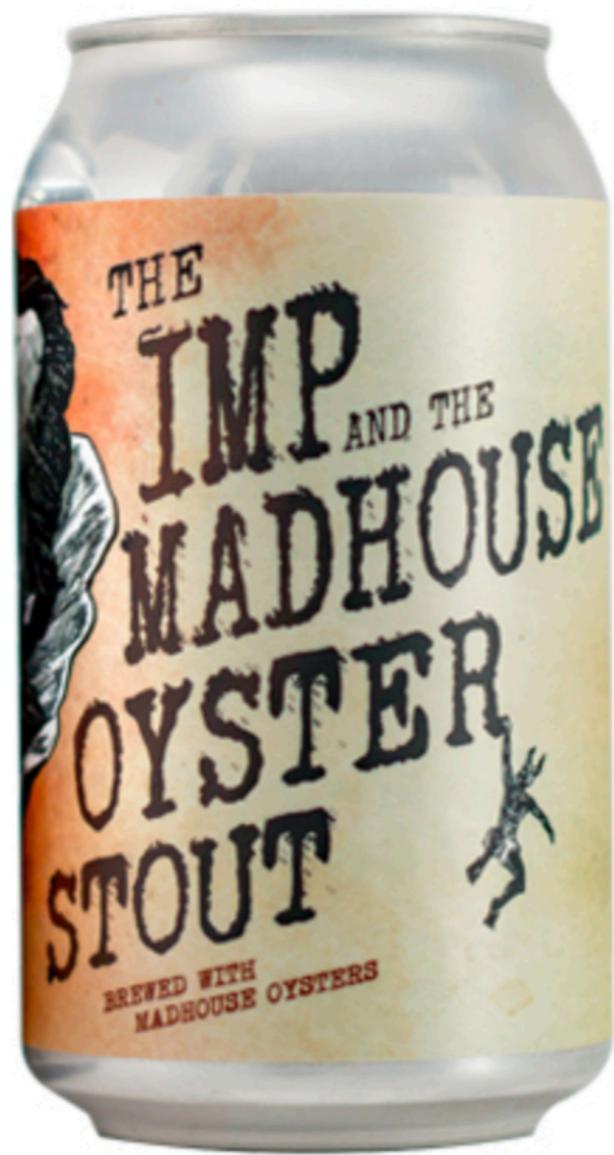
40
IBU

Brewer Description

Imperial Porter brewed with Columbian coffee and vanilla beans.

The App

<https://dmblumen.shinyapps.io/BeerMe2/>



The Imp and the Madhouse Oyster Stout

RavenBeer

Experimental Beer | Year-Round | Maryland | United States

92

Judges Ratings

[How We Score](#)

[Read Full Review](#)

1 Review

Aroma: 22 / 24

Flavor: 36 / 40

Appearance: 6 / 6

Mouthfeel: 10 / 10

Overall Impression: 18 / 20

The First Website

<https://beerconnoisseur.com/search-beer>

```

8     start_urls = ['https://beerconnoisseur.com/search-beer']
9
10    def parse(self, response):
11        result_urls = ['https://beerconnoisseur.com/search-beer?keys=&field_beer_style_tid=All&field_state_value=All&field_country_value=United%20States&sort_by=-rating']
12
13        for url in result_urls:
14            yield Request(url=url, callback=self.parse_result_page)
15
16    def parse_result_page(self, response):
17        condensed_urls = response.xpath('//div[@class ="views-field views-field-title"]//a/@href').extract()
18        full_urls = []
19        for url in condensed_urls:
20            full_urls.append('https://beerconnoisseur.com' + url)
21
22        for url in full_urls:
23            yield Request(url=url, callback=self.parse_detail_page)
24
25    def parse_detail_page(self, response):
26        name = response.xpath('//h1/text()').extract()
27        image = response.xpath('//div[@class="field field-name-field-beer-image field-type-image field-label-hidden"]//img/@src').extract()
28        brewer = response.xpath('//div[@class="field field-name-field-brewery field-type-entityreference field-label-hidden"]//a/text()').extract()
29        beer_type = response.xpath('//div[@class="field field-name-field-beer-style field-type-taxonomy-term-reference field-label-hidden"]//a/text()').extract()
30        release = response.xpath('//div[@class="field field-name-field-availability field-type-taxonomy-term-reference field-label-hidden"]//a/text()').extract()
31
32        ###State needs the second element pulled
33        state = response.xpath('//div[@class="field field-name-field-state field-type-list-text field-label-hidden"]//div/text()').extract()[1]
34
35        description = response.xpath('//div[@class="field field-name-body field-type-text-with-summary field-label-above"]//p/text()').extract()
36
37        ### abv needs the % sign removed
38        abv_p = response.xpath('//div[@class="field field-name-field-abv field-type-text field-label-inline clearfix"]//div[@class ="field-item even"]/text()')
39        abv = list(map(lambda s: s.replace('%',''),abv_p))
40
41        ibu = response.xpath('//div[@class="field field-name-field-ibus field-type-text field-label-inline clearfix"]//div[@class ="field-item even"]/text()')
42        serv_temp = response.xpath('//div[@class="field field-name-field-served-at field-type-text field-label-inline clearfix"]//div[@class ="field-item even"]/text()')
43        hops = response.xpath('//div[@class="field field-name-field-hops field-type-text field-label-inline clearfix"]//div[@class ="field-item even"]/text()')
44        malts = response.xpath('//div[@class="field field-name-field-malts field-type-text field-label-inline clearfix"]//div[@class ="field-item even"]/text()')
45
46        ### Aroma is going to require Regex
47        aroma_p = response.xpath('//div[@class="views-field views-field-field-judge-aroma"]//div[@class ="field-content"]/text()').extract_first()
48        aroma = re.findall('(\d+) /',aroma_p)
49

```

The Scrappy

100 Lines of Code. Took 2 minutes to scrape 2000 websites.



Shares



Style: Mixed Fermentation Farmhouse ale

Wiley Roots Galaxy DH Funk Yo Couch

[Print Shelf Talker](#)[How we review](#)

What the brewers say

"Tart farmhouse saison fermented with wild captured Brettanomyces, dry hopped with more than 1 pound per barrel of Australian Galaxy hops. Winner of a gold medal at the 2017 GABF."

What our panel thought

Aroma: "Very fruity hops and ester aroma with passion fruit, peach, pear, and apple. Behind that's there's tangerine, lemon, and a touch of herbal. Spicy phenols, light Brett funk. Slight leather."

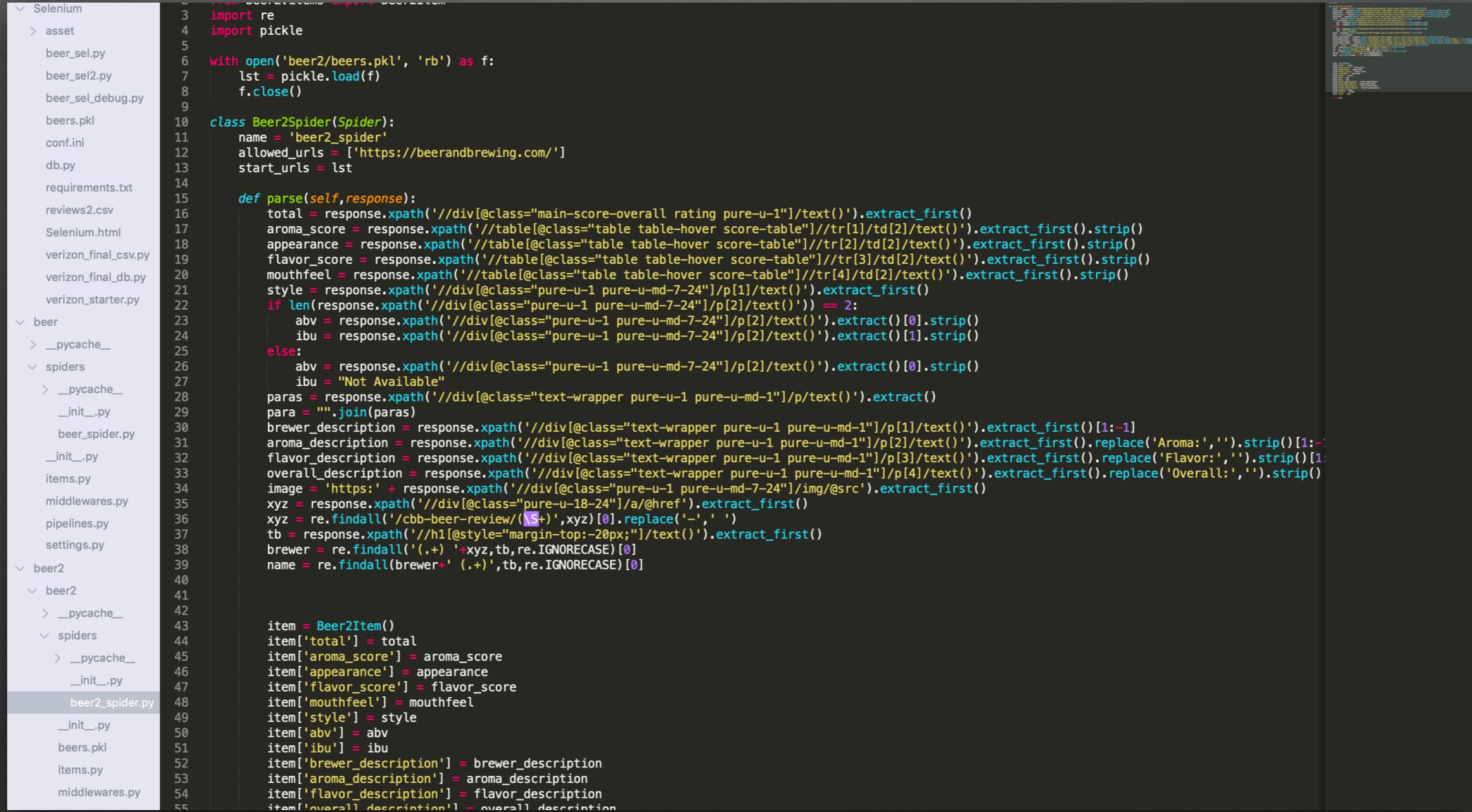
Flavor: "Very juicy hops flavor with a mix of tropical and citrus. Moderately low sweetness with moderately high sour that fades into a dry finish. The body supports the beer with a malt character that seems a bit toothier than others in the category. Light Brett funk and tart. Lingering aftertaste of sweet fruit balanced with light tartness. Acidity is masterfully placed to accentuate the fruit flavors from the hops."

Overall: "Almost a fruit-cocktail blend with some funk, but it all finishes nice and dry and drinkable. A fantastic beer that delivers an ultimately satisfying hops character with a



The Siren: The Second Website

<https://beerandbrewing.com/beer-reviews/>



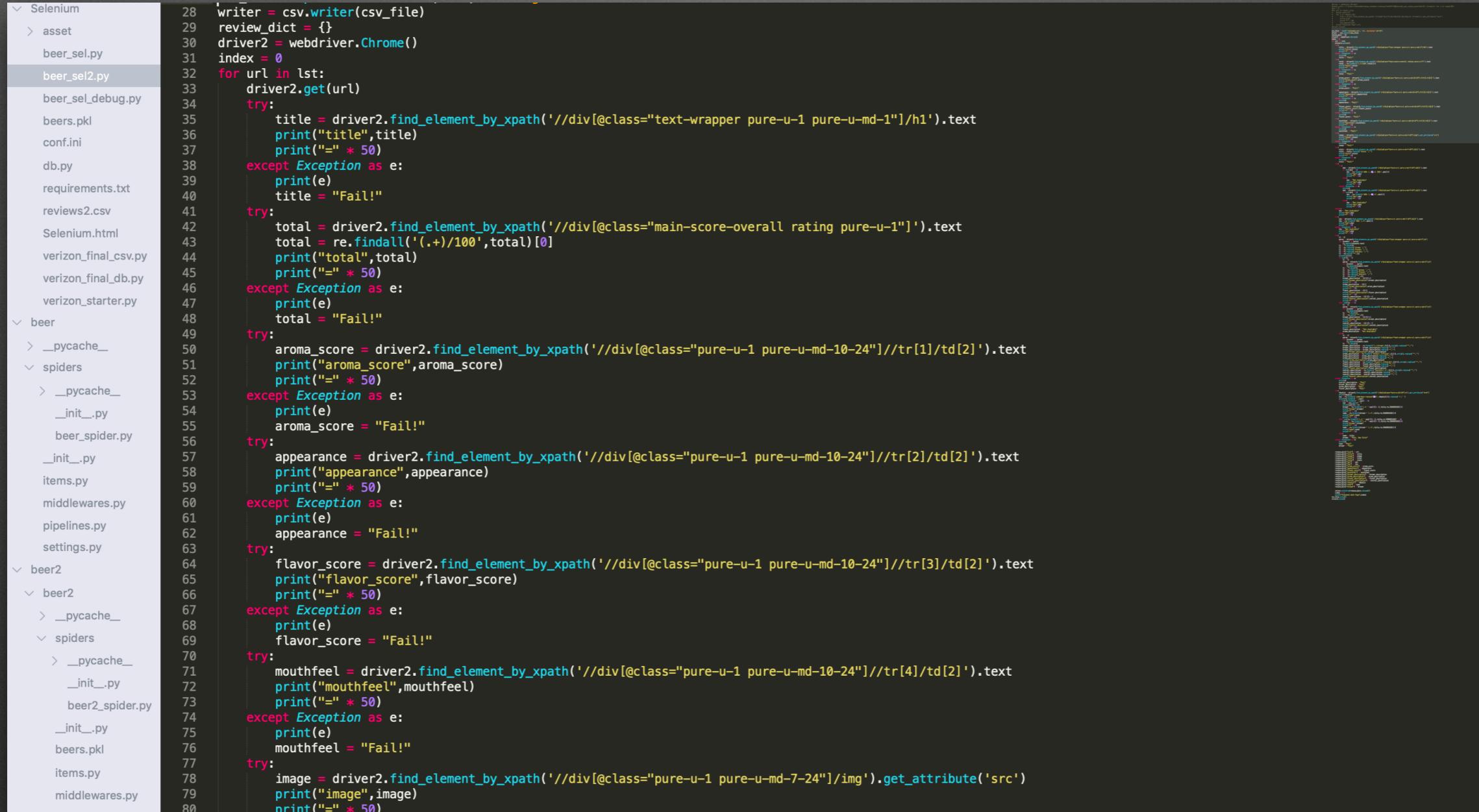
```

1  from bs4 import BeautifulSoup
2  from selenium import webdriver
3  import re
4  import pickle
5
6  with open('beer2/beers.pkl', 'rb') as f:
7      lst = pickle.load(f)
8      f.close()
9
10 class Beer2Spider(Spider):
11     name = 'beer2_spider'
12     allowed_urls = ['https://beerandbrewing.com/']
13     start_urls = lst
14
15     def parse(self, response):
16         total = response.xpath('//div[@class="main-score-overall rating pure-u-1"]/text()').extract_first()
17         aroma_score = response.xpath('//table[@class="table table-hover score-table"]//tr[1]/td[2]/text()').extract_first().strip()
18         appearance = response.xpath('//table[@class="table table-hover score-table"]//tr[2]/td[2]/text()').extract_first().strip()
19         flavor_score = response.xpath('//table[@class="table table-hover score-table"]//tr[3]/td[2]/text()').extract_first().strip()
20         mouthfeel = response.xpath('//table[@class="table table-hover score-table"]//tr[4]/td[2]/text()').extract_first().strip()
21         style = response.xpath('//div[@class="pure-u-1 pure-u-md-7-24"]/p[1]/text()').extract_first()
22         if len(response.xpath('//div[@class="pure-u-1 pure-u-md-7-24"]/p[2]/text()')) == 2:
23             abv = response.xpath('//div[@class="pure-u-1 pure-u-md-7-24"]/p[2]/text()').extract()[0].strip()
24             ibu = response.xpath('//div[@class="pure-u-1 pure-u-md-7-24"]/p[2]/text()').extract()[1].strip()
25         else:
26             abv = response.xpath('//div[@class="pure-u-1 pure-u-md-7-24"]/p[2]/text()').extract()[0].strip()
27             ibu = "Not Available"
28         paras = response.xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/p/text()').extract()
29         para = "".join(paras)
30         brewer_description = response.xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/p[1]/text()').extract_first()[1:-1]
31         aroma_description = response.xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/p[2]/text()').extract_first().replace('Aroma: ','').strip()[1:-1]
32         flavor_description = response.xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/p[3]/text()').extract_first().replace('Flavor: ','').strip()[1:-1]
33         overall_description = response.xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/p[4]/text()').extract_first().replace('Overall: ','').strip()
34         image = 'https:' + response.xpath('//div[@class="pure-u-1 pure-u-md-7-24"]/img@src').extract_first()
35         xyz = response.xpath('//div[@class="pure-u-18-24"]/a/@href').extract_first()
36         xyz = re.findall('/cbb-beer-review/(.+)',xyz)[0].replace('-',' ')
37         tb = response.xpath('//h1[@style="margin-top:-20px;"]/text()').extract_first()
38         brewer = re.findall('(.+)' + xyz,tb,re.IGNORECASE)[0]
39         name = re.findall(brewer + ' (.+)',tb,re.IGNORECASE)[0]
40
41
42         item = Beer2Item()
43         item['total'] = total
44         item['aroma_score'] = aroma_score
45         item['appearance'] = appearance
46         item['flavor_score'] = flavor_score
47         item['mouthfeel'] = mouthfeel
48         item['style'] = style
49         item['abv'] = abv
50         item['ibu'] = ibu
51         item['brewer_description'] = brewer_description
52         item['aroma_description'] = aroma_description
53         item['flavor_description'] = flavor_description
54         item['overall_description'] = overall_description
55

```

The Second Scrapy

Only 60 lines of code! The problem:
Some (but not all) the pages use AJAX



The screenshot shows a code editor with a dark theme, displaying a large Python script. The script is organized into several modules and sub-modules under the package structure shown on the left. The code itself is a Selenium-based scraper, likely for beer reviews, as indicated by the file names like 'beer_sel.py', 'beer2.py', and 'beer_spider.py'. The script uses BeautifulSoup for parsing HTML and various exception handling mechanisms to deal with different page structures. The code is heavily commented, explaining the logic for extracting titles, total scores, aroma scores, appearance scores, flavor scores, mouthfeel scores, and images from the web pages.

```
28 writer = csv.writer(csv_file)
29 review_dict = {}
30 driver2 = webdriver.Chrome()
31 index = 0
32 for url in lst:
33     driver2.get(url)
34     try:
35         title = driver2.find_element_by_xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/h1').text
36         print("title",title)
37         print("=" * 50)
38     except Exception as e:
39         print(e)
40         title = "Fail!"
41     try:
42         total = driver2.find_element_by_xpath('//div[@class="main-score-overall rating pure-u-1"]').text
43         total = re.findall('(.+)/100',total)[0]
44         print("total",total)
45         print("=" * 50)
46     except Exception as e:
47         print(e)
48         total = "Fail!"
49     try:
50         aroma_score = driver2.find_element_by_xpath('//div[@class="pure-u-1 pure-u-md-10-24"]//tr[1]/td[2]').text
51         print("aroma_score",aroma_score)
52         print("=" * 50)
53     except Exception as e:
54         print(e)
55         aroma_score = "Fail!"
56     try:
57         appearance = driver2.find_element_by_xpath('//div[@class="pure-u-1 pure-u-md-10-24"]//tr[2]/td[2]').text
58         print("appearance",appearance)
59         print("=" * 50)
60     except Exception as e:
61         print(e)
62         appearance = "Fail!"
63     try:
64         flavor_score = driver2.find_element_by_xpath('//div[@class="pure-u-1 pure-u-md-10-24"]//tr[3]/td[2]').text
65         print("flavor_score",flavor_score)
66         print("=" * 50)
67     except Exception as e:
68         print(e)
69         flavor_score = "Fail!"
70     try:
71         mouthfeel = driver2.find_element_by_xpath('//div[@class="pure-u-1 pure-u-md-10-24"]//tr[4]/td[2]').text
72         print("mouthfeel",mouthfeel)
73         print("=" * 50)
74     except Exception as e:
75         print(e)
76         mouthfeel = "Fail!"
77     try:
78         image = driver2.find_element_by_xpath('//div[@class="pure-u-1 pure-u-md-7-24"]/img').get_attribute('src')
79         print("image",image)
80         print("=" * 50)
```

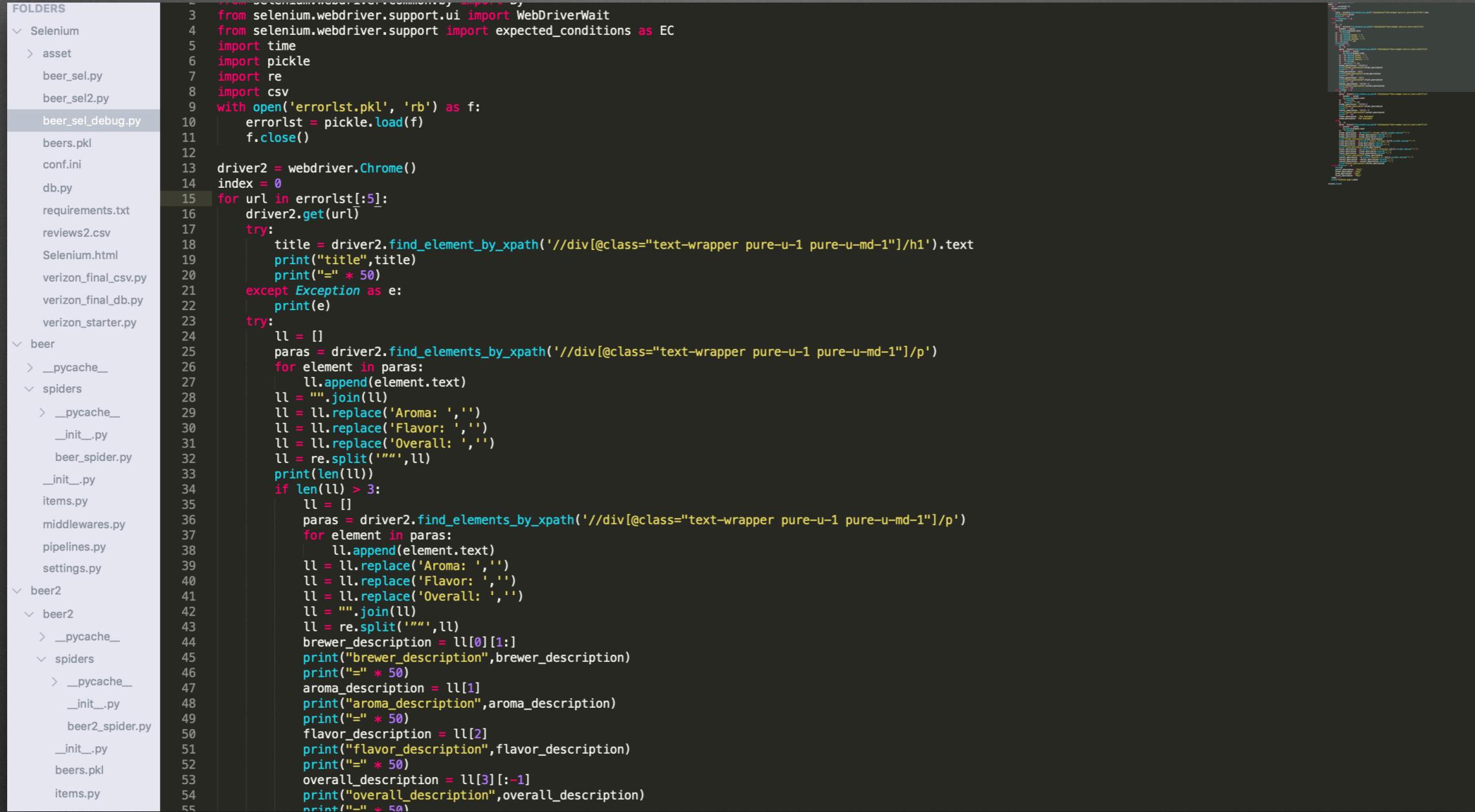
Selenium: The (Almost) Finished Product

250 lines of code for just over 1000 pages. The recurring problem:
Not all pages had the same pattern.

```
except Exception as e  
    print(e)  
    csv_file.close()  
    driver.close()  
    break
```

This was a huge problem:

An exception would occur; Python would tell me what the error was and then just break out of the loop. It wouldn't really tell me the change in the pattern that caused the error to occur.



The screenshot shows a code editor interface with a file tree on the left and a large Python script on the right.

File Tree:

- FOLDERS
 - Selenium
 - > asset
 - beer_sel.py
 - beer_sel2.py
 - beer_sel_debug.py
 - beers.pkl
 - conf.ini
 - db.py
 - requirements.txt
 - reviews2.csv
 - Selenium.html
 - verizon_final_csv.py
 - verizon_final_db.py
 - verizon_starter.py
- beer
 - > __pycache__
 - spiders
 - > __pycache__
 - __init__.py
 - beer_spider.py
 - __init__.py
 - items.py
 - middlewares.py
 - pipelines.py
 - settings.py
- beer2
 - > beer2
 - > __pycache__
 - spiders
 - > __pycache__
 - __init__.py
 - beer2_spider.py
 - __init__.py
 - beers.pkl
 - items.py

Python Script Content:

```

1  from selenium.webdriver.common import *
2  from selenium.webdriver.support.ui import WebDriverWait
3  from selenium.webdriver.support import expected_conditions as EC
4
5  import time
6  import pickle
7  import re
8  import csv
9  with open('errorlst.pkl', 'rb') as f:
10     errorlst = pickle.load(f)
11     f.close()
12
13 driver2 = webdriver.Chrome()
14 index = 0
15 for url in errorlst[:5]:
16     driver2.get(url)
17     try:
18         title = driver2.find_element_by_xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/h1').text
19         print("title", title)
20         print("-" * 50)
21     except Exception as e:
22         print(e)
23     try:
24         ll = []
25         paras = driver2.find_elements_by_xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/p')
26         for element in paras:
27             ll.append(element.text)
28         ll = ''.join(ll)
29         ll = ll.replace('Aroma: ','')
30         ll = ll.replace('Flavor: ','')
31         ll = ll.replace('Overall: ','')
32         ll = re.split('"',ll)
33         print(len(ll))
34         if len(ll) > 3:
35             ll = []
36             paras = driver2.find_elements_by_xpath('//div[@class="text-wrapper pure-u-1 pure-u-md-1"]/p')
37             for element in paras:
38                 ll.append(element.text)
39             ll = ll.replace('Aroma: ','')
40             ll = ll.replace('Flavor: ','')
41             ll = ll.replace('Overall: ','')
42             ll = ''.join(ll)
43             ll = re.split('"',ll)
44             brewer_description = ll[0][1:]
45             print("brewer_description",brewer_description)
46             print("-" * 50)
47             aroma_description = ll[1]
48             print("aroma_description",aroma_description)
49             print("-" * 50)
50             flavor_description = ll[2]
51             print("flavor_description",flavor_description)
52             print("-" * 50)
53             overall_description = ll[3][-1]
54             print("overall_description",overall_description)
55             print("-" * 50)

```

The Solution: A Debugging Selenium

What's happening in the background:

Upload the csv, filter the data based on “Fail!” for a given variable, create a list of the urls that had the issue, use pickle package to save the list and then load said list into the debugger. Then figure out the pattern. Copy and paste the working code for the variable and repeat.

```
image https://images.ccrassets.net/q66655717110/VIEWERAAAMRZoWsb/950C950002512C0ED11C227D007C70/8E57207504E50A7F005fdd7/0ECC880100326_001-CHambyLe.png?w=600
=====
style Witbier
=====
abv 5.0
=====
ibu 10
=====
4
brewer_description Blanche de Chambly is the very first ale brewed by Unibroue and as Belgian tradition dictates for white ales, it is named after the city in which it is brewed. In 1996, the Chicago Beverage Testing institute declared Blanche de Chambly 'The World's Best White Ale.' Since then, it has gone on to win numerous other awards and distinctions.
=====
aroma_description Coriander with a hint of coconut, nice floral notes, and sweet citrus character produce a nose that fades quickly. Light wheat sweetness. The strong coriander character is sharper and less herbal than is typical.
=====
flavor_description Nice level of acidity to carbonic bite. Simple flavors of slightly musty coriander and honey wheat sweetness with very little hop flavor. Finish is a bit cloying instead of crisp.
=====
overall_description Not-too-sweet finish makes this an easy-drinking, refreshing wit. Body does not hold up to strong coriander flavor. The individual pieces are there, but the overall structure doesn't seem coherent. Would make a great beurre blanc.
=====

https://beerandbrewing.com/print/VLWeWikAAMRZoWsb/cbb-beer-review/blanche-de-chamby
[' ', ' ']
blanc
brewer Unibroue
=====
name Blanche de Chambly
=====
Finished with Page 996
title Dogfish Head Craft Brewery 61 Minute IPA
=====
total 75
=====
aroma_score 8
=====
appearance 2
```

Why this is handy:

Allows you to target the specific issues, determine where the changes in pattern are and create code to accommodate changes in patterns.

The Future

- Potentially take the two CSVs and combine them.
- Natural Language Processing With the aroma, flavor, and overall reviews
- Potentially scrape a third website, using the names of the beer to scrape links of where you can buy the beer if it interests you.