Webscraping Geospatial Data with Python/Beautiful Soup

Hal Mueller, MaptimeSEA, May 7, 2024

Downloads

- https://www.anaconda.com/download
- https://www.google.com/chrome/
- https://github.com/halmueller/SoupTalk

Beautiful Soup

‘Beautiful Soup, so rich and green,
Waiting in a hot tureen!
Who for such dainties would not
stoop?
Soup of the evening, beautiful Soup!
Soup of the evening, beautiful Soup!
Beau--ootiful Soo--oop!
Beau--ootiful Soo--oop!
Soo--oop of the e--e--evening,
Beautiful, beautiful Soup!’

– Lewis Carroll

Beautiful Soup

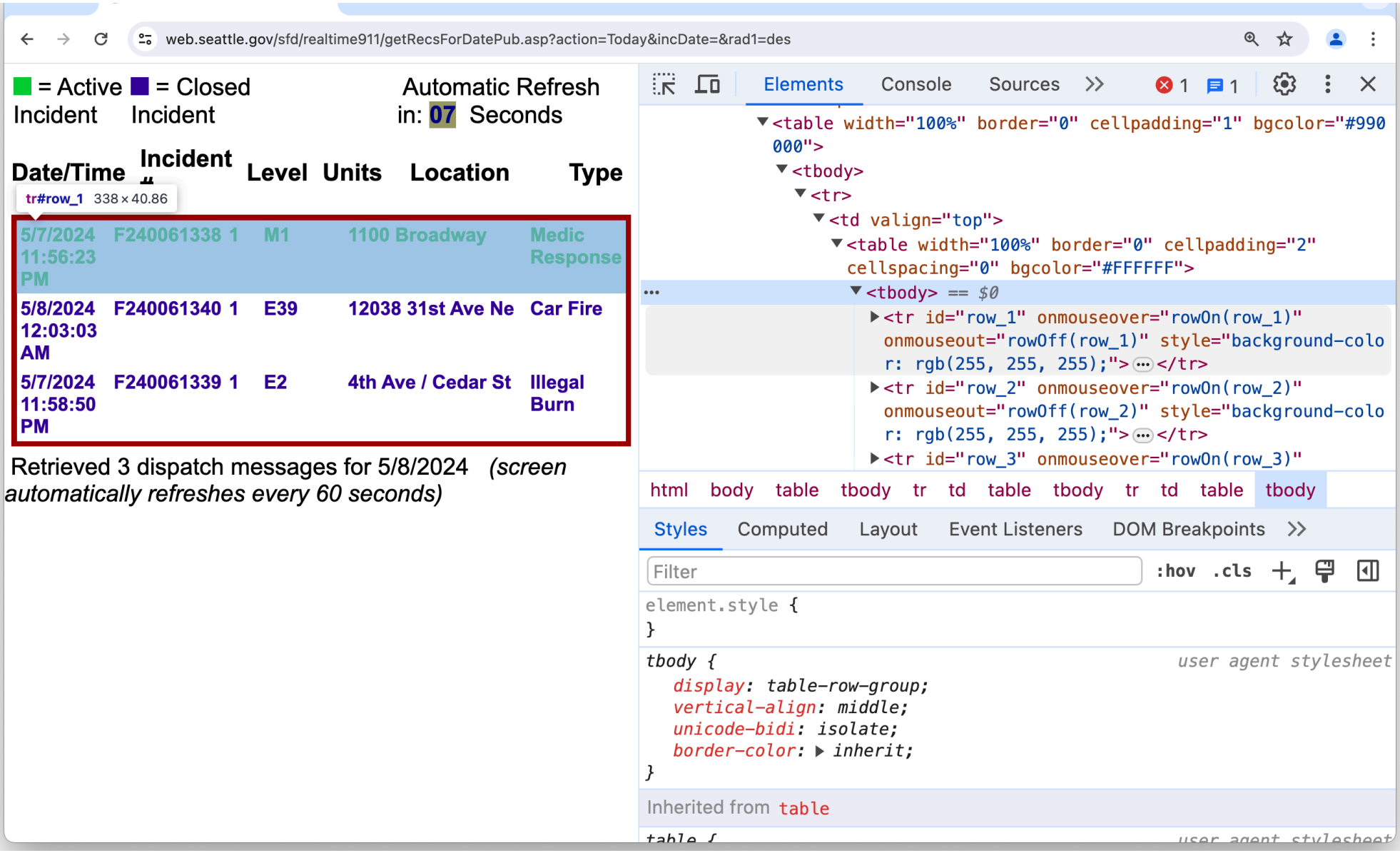
- “Tag soup”: https://en.wikipedia.org/wiki/Tag_soup
- Python Package: “Beautiful Soup”
- https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser)
- Leonard Richardson + Tidelift: https://www.crummy.com/software/BeautifulSoup/
- Python 3, BeautifulSoup4 4.12.3: https://pypi.org/project/beautifulsoup4/
- Alternative: https://scrapy.org/

“Don’t try this at home!”

- Web API if you can: Wikipedia, Seattle City Data
- Be kind to servers: render cost
- Test often
- Sniff URLs with a proxy:
  - Proxyman (my current fave)
  - or Wireshark or Fiddler
  - Capture from mobile device native app, analyze on desktop

Activity

- https://sfdlive.com/
- https://web.seattle.gov/sfd/realtime911/
- https://data.seattle.gov/Public-Safety/Seattle-Real-Time-Fire-911-Calls/kzjm-xkqj/data: export as JSON, custom views
- Google Chrome: Developer Tools (option-command-I) (“eye”)


web.seattle.gov/sfd/realtime911/getRecsForDatePub.asp?action=Today&incDate=&rad1=des
= Active Incident
= Closed Incident
Automatic Refresh in: 07 Seconds
Date/Time
Incident #
Level
Units
Location
Type
tr#row_1 338 × 40.86
5/7/2024 11:56:23 PM F240061338 1 M1 1100 Broadway Medic Response
5/8/2024 12:03:03 AM F240061340 1 E39 12038 31st Ave Ne Car Fire
5/7/2024 11:58:50 PM F240061339 1 E2 4th Ave / Cedar St Illegal Burn
Retrieved 3 dispatch messages for 5/8/2024 (screen automatically refreshes every 60 seconds)
Elements
Console
Sources
<table width="100%" border="0" cellpadding="1" bgcolor="#990000">
<tbody>
<tr>
<td valign="top">
<table width="100%" border="0" cellpadding="2" cellspacing="0" bgcolor="#FFFFFF">
<tbody> == $0
<tr id="row_1" onmouseover="rowOn(row_1)" onmouseout="rowOff(row_1)" style="background-color: rgb(255, 255, 255);">…</tr>
<tr id="row_2" onmouseover="rowOn(row_2)" onmouseout="rowOff(row_2)" style="background-color: rgb(255, 255, 255);">…</tr>
<tr id="row_3" onmouseover="rowOn(row_3)"
html body table tbody tr td table tbody tr td table tbody
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Filter
:hov .cls
element.style {
}
tbody { user agent stylesheet
display: table-row-group;
vertical-align: middle;
unicode-bidi: isolate;
border-color: ▶ inherit;
}
Inherited from table

Python with Anaconda

- Environment manager with Python3 installation
- Package manager

Package and environment setup

- Launch Anaconda
- Add “conda-forge” to Channels
- Create environment for soup course
- Import GeoPy, BS4, requests to the environment
- Launch command line from environment “play” button
- python3 scripts/check_install.py
- Windows: python scripts/check_install.py

Geocoding

- GeoPy: https://geopy.readthedocs.io/en/stable/#module-geopy.geocoders, wraps 32 different services
- Terms of service
- Licensing of data
- Alternative: King County government has local geocoder, with web API

python3 scripts/nominatim_demo.py

```
from geopy.geocoders import Nominatim

geocoder = Nominatim(user_agent="SoupTalk_add_your_name")

# TPC Ballard
# 5101 14th Ave NW
# Suite 201
# Seattle, WA 98103

location1 = geocoder.geocode("5101 14th Avenue NW”)

print(location1.address)

print((location1.latitude, location1.longitude))

print(location1.raw)
```