Simple Webscraping Example with Beautiful Soup

In [1]:
```
# Import all neccessary libraries

from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
```

In [24]:
```
# Assign the URL to a variable

url = "https://www12.statcan.gc.ca/census-recensement/2021/dp-pd/prof/details/pa


# use the urlopen function to open the webpage
html = urllib.request.urlopen(url)

# show object html
html
```

Out[24]:
```
<http.client.HTTPResponse at 0x17a00c58610>
```

In [25]:
```
# Create the BeautifulSoup object

html_to_parse = BeautifulSoup(html, "html.parser")
```

In [26]:
```
# create a list of tables.  There is only 1 table in this webpage

tables = html_to_parse.find_all("table")
print(f"Number of tables found: {len(tables)}")
```

```
Number of tables found: 1
```

In [34]:
```
# Create list of all the <th> tags in the table that has the title "2021A0011M1C

td = tables[0].find(attrs={"title":"2021A0011M1C - Population, 2021 - Counts - T
# td = tables[0].find(attrs={"title":"2021A0011M1C - Children - Counts - Total"}
```

In [35]:
```
td
```

Out[35]:
```
<td class="text-right text-nowrap" headers="rh1 r1 geo2021A0011M1C geo2021A0011
M1Cstat1 geo2021A0011M1Cstat1gen1" title="2021A0011M1C - Population, 2021 - Cou
nts - Total"> 35,642</td>
```

In [36]:
```
# convert to float
float(td.text.replace(",", ""))
```

Out[36]:
```
35642.0
```

Create a script that will look up from a list of Postal codes

In [8]:
```
import urllib.parse as urlparse
from urllib.parse import urlencode
```

In [ ]:
```
# A list of postal code from the previous part

# postal = ['M3A', 'M4A', 'M5A', 'M6A', 'M7A']

# select all postal codes from Central Toronto
toronto_DF = pd.read_csv('toronto_DF.csv')
postal = list(toronto_DF[toronto_DF['Borough'] == 'Central Toronto']['Postalcode
```

In [46]:
```
# Creating Empty DataFrame and Storing it in variable df

# df = pd.DataFrame(columns = ['postal_code', 'data', 'value'])
df = pd.DataFrame(columns = ['postal_code', 'count_children', 'rate_children', '
```

In [48]:
```
# Loop through each postal code


for i in postal:
    url = "https://www12.statcan.gc.ca/census-recensement/2021/dp-pd/prof/detail
    params = {
        'SearchText': i,
        'DGUIDlist': '2021A0011'+i
    }

    # this part switches up the postal code parameter in the url
    url_parts = list(urlparse.urlparse(url))
    query = dict(urlparse.parse_qsl(url_parts[4]))
    query.update(params)

    url_parts[4] = urlencode(query)
    query = urlparse.urlunparse(url_parts)

    # the following code is similar to the above
    html = urllib.request.urlopen(query)
    html_to_parse = BeautifulSoup(html, "html.parser")
    tables = html_to_parse.find_all("table")
    print(f"Number of tables found: {len(tables)}")

    # change the title to find the data you want
    cnt_children = (f"2021A0011{i} - Children - Counts - Total")
    rate_children = (f"2021A0011{i} - Children - Rates - Total")
    median_income = (f"2021A0011{i} - Median after-tax income in 2020 among reci
    cnt_employed = (f"2021A0011{i} - Employed - Counts - Total")
    rate_un = (f"2021A0011{i} - Unemployment rate - Counts - Total")

    v1 = tables[0].find(attrs={"title":cnt_children})
    v2 = tables[0].find(attrs={"title":rate_children})
    v3 = tables[0].find(attrs={"title":median_income})
    v4 = tables[0].find(attrs={"title":cnt_employed})
    v5 = tables[0].find(attrs={"title":rate_un})

    # print(td)
    df.loc[len(df.index)] = [i, float(v1.text.replace(",", "")), float(v2.text.r
                             float(v3.text.replace(",", "")), float(v4.text.repl
                             float(v5.text.replace(",", ""))]
```

```
Number of tables found: 1
Number of tables found: 1
Number of tables found: 1
Number of tables found: 1
Number of tables found: 1
Number of tables found: 1
Number of tables found: 1
Number of tables found: 1
Number of tables found: 1
```

In [51]:
```
df.sort_values('count_children', ascending = False)
```

Out[51]:

	postal_code	count_children	rate_children	median_income	count_employed	un
3	M4S	6360.0	21.1	47600.0	17140.0	
6	M5N	5350.0	33.6	44400.0	7705.0	
0	M4N	4765.0	31.5	54400.0	6820.0	
7	M5P	4715.0	24.0	48800.0	10280.0	
1	M4P	4505.0	18.1	42400.0	14115.0	
8	M5R	3865.0	15.6	46400.0	13835.0	
5	M4V	3635.0	19.2	52800.0	10320.0	
2	M4R	3365.0	28.3	49600.0	6190.0	
4	M4T	2370.0	23.3	56400.0	4935.0	

In [50]:
```
# Now you can export this to a CSV file for further analysis or visulization
df.to_csv('data.csv')
```